Міністерство освіти і науки України

Національний університет «Львівська політехніка»

АВТОМАТИЗОВАНЕ ПРОЕКТУВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ

Лабораторна робота № 3

на тему: «Хрестики-нулики на клієнтській та серверній частинах.»

Виконав:

ст. гр. КІ-404

Давида В.Р.

Прийняв:

Федак П.Р.

Львів – 2024

Task 3.

Implement Server (HW) and Client (SW) parts of game (FEF)	**Create SW build system(EEF)**
1. Develop Server and Client. 2. Required steps.	1. Create **YML file** with next features: a. build all binaries (create scripts in folder **ci/** ifneed); b. run tests; c. create artifacts with binaries and test reports; 2. Required steps.

Варіант 5:

Student number	**Game**	**config format**
5	tik-tac-toe 3x3	JSON

Теоретичні відомості

UART (Universal Asynchronous Receiver/Transmitter) представляє собою стандартний протокол для обміну даними між пристроями через послідовний інтерфейс. Цей інтерфейс дозволяє пристроям передавати та приймати інформацію біт за бітом і знаходить широке застосування в мікроконтролерах, мікропроцесорах, сенсорах, модемах та інших пристроях.

Основні характеристики UART включають асинхронний режим, де дані передаються через два незалежні сигнали - TX (передача) та RX (прийом), структуру кадру, яка включає стартовий біт, біти даних, біти парності (опціонально) та стоповий біт. Ця структура дозволяє правильно ідентифікувати початок та кінець кожного байту.

Швидкість передачі (Baud Rate) грає ключову роль у визначенні того, скільки бітів передається за одну секунду, і ця швидкість повинна бути налаштована на обох пристроях для успішного обміну даними.

Парність (Parity) є опціональною функцією, яка дозволяє визначити четність чи непарність бітів даних для виявлення помилок передачі. Керівництво лінією (Flow Control) може використовуватися для управління потоком даних, особливо при великій швидкості передачі або різних швидкостях пристроїв.

Також іноді використовується ізоляція гальванічна для електричної ізоляції між пристроями та запобігання електричним помилкам.

Дистанція передачі за допомогою UART обмежена, хоча цей інтерфейс дозволяє передавати дані на значні відстані. Зазвичай це кілька метрів без спеціальних заходів.

Хрестики-нулики — це гра для двох осіб, яка грається на квадратному полі розміром 3 на 3 або більшим. Один гравець грає хрестиками, а інший - нуликами, по черзі вони розташовують свої символи на вільних клітинках поля.

Мета гравця — створити лінію із трьох своїх символів горизонтально, вертикально або діагонально.

Хід роботи

1. Написав код для клієнтської частини.

Файл **main.cpp:**

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char* argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.setWindowTitle("TicTacToe");
    w.show();
    return a.exec();
}
```

Файл **mainwindow.cpp:**

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QJsonObject>
#include <QJsonDocument>
#include <QFile>

HANDLE hSerial;
QString portArduino;

QString connect_arduino, game_started, game_mode, ai_strategy, message, next_turn;
QString board[3][3];


void resetValues() {
    // connect_arduino = "0";
    game_started = "0";
    // game_mode = "mva";
    // ai_strategy = "rand";
    message = "";

    // Заповнюємо масив board значенням "-"
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            board[i][j] = "-";
        }
    }
}


QString buildJSON() {
    QJsonObject json;
    json["con"] = connect_arduino;
    json["gs"] = game_started;
    json["gm"] = game_mode;
    json["ais"] = ai_strategy;
    json["msg"] = message;
    json["nt"] = next_turn;

    QJsonObject boardJson;
    boardJson["c11"] = board[0][0];
```

```
    boardJson["c12"] = board[0][1];
    boardJson["c13"] = board[0][2];
    boardJson["c21"] = board[1][0];
    boardJson["c22"] = board[1][1];
    boardJson["c23"] = board[1][2];
    boardJson["c31"] = board[2][0];
    boardJson["c32"] = board[2][1];
    boardJson["c33"] = board[2][2];

    json["brd"] = boardJson;

    QJsonDocument doc(json);
    return doc.toJson(QJsonDocument::Indented);
}

QString getJsonValue(const QJsonObject& json, const QString& key) {
    return json.value(key).toString();
}

void MainWindow::parseJSON(QString input) {
    QJsonDocument doc = QJsonDocument::fromJson(input.toUtf8());
    if (!doc.isObject()) return;

    QJsonObject json = doc.object();

    connect_arduino = getJsonValue(json, "con");
    game_started = getJsonValue(json, "gs");
    game_mode = getJsonValue(json, "gm");
    ai_strategy = getJsonValue(json, "ais");
    message = getJsonValue(json, "msg");
    ui->label_arduino_msg->setText(message);
    next_turn = getJsonValue(json, "nt");

    QJsonObject boardJson = json["brd"].toObject();
    board[0][0] = getJsonValue(boardJson, "c11");
    board[0][1] = getJsonValue(boardJson, "c12");
    board[0][2] = getJsonValue(boardJson, "c13");
    board[1][0] = getJsonValue(boardJson, "c21");
    board[1][1] = getJsonValue(boardJson, "c22");
    board[1][2] = getJsonValue(boardJson, "c23");
    board[2][0] = getJsonValue(boardJson, "c31");
    board[2][1] = getJsonValue(boardJson, "c32");
    board[2][2] = getJsonValue(boardJson, "c33");
}


void MainWindow::saveGameState(const QString& filePath) {
    QFile file(filePath);

    if (!file.open(QIODevice::WriteOnly | QIODevice::Text)) {
        ui->label_pc_msg->setText("cant write file for write");
        return;
    }

    QTextStream out(&file);
    QString JSONData = buildJSON();  // Викликаємо функцію, яка формує JSON
    out << JSONData;

    file.close();

    ui->label_pc_msg->setText("Game Saved");
```

```
}

void MainWindow::loadGameState(const QString& filePath) {
    QFile file(filePath);

    if (!file.open(QIODevice::ReadOnly | QIODevice::Text)) {
        ui->label_pc_msg->setText("cant open file for read");
        return;
    }

    QTextStream in(&file);
    QString JSONData = in.readAll();  // Читаємо весь файл у строку
    file.close();

    parseJSON(JSONData);  // Парсимо JSON та відновлюємо стан гри
    ui->label_pc_msg->setText("Load Complete");
}



MainWindow::MainWindow(QWidget* parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    loadComPorts();

    connect(ui->comboBoxPorts, SIGNAL(currentTextChanged(const QString&)),
        this, SLOT(onComboBoxPortChanged(const QString&)));


    QString JSONData = buildJSON();
    updateGameBoard();


}

MainWindow::~MainWindow()
{
    delete ui;
}



void MainWindow::updateGameBoard() {
    updateButtonIcon(ui->button_11, board[0][0]);
    updateButtonIcon(ui->button_12, board[0][1]);
    updateButtonIcon(ui->button_13, board[0][2]);
    updateButtonIcon(ui->button_21, board[1][0]);
    updateButtonIcon(ui->button_22, board[1][1]);
    updateButtonIcon(ui->button_23, board[1][2]);
    updateButtonIcon(ui->button_31, board[2][0]);
    updateButtonIcon(ui->button_32, board[2][1]);
    updateButtonIcon(ui->button_33, board[2][2]);
}

void MainWindow::updateButtonIcon(QPushButton* button, const QString& value) {
    if (value == "x") {
        button->setIcon(QIcon(":/resources/assets/x2c.png"));  // Іконка хрестика
    }
```

```
    else if (value == "o") {
        button->setIcon(QIcon(":/resources/assets/o2c.png"));  // Іконка круга
    }
    else {
        button->setIcon(QIcon());  // Очищаємо іконку, якщо кнопка порожня
    }
    // button->setEnabled(false);  // Робимо кнопку недоступною для натискання
}

void MainWindow::on_newButton_clicked()
{
    resetValues();
    game_started = "1";

    if (game_mode == "ava")
    {
        ui->newButton->setEnabled(false);
        ui->loadButton->setEnabled(false);
        ui->saveButton->setEnabled(false);

        ui->label_pc_msg->setText("AI vs AI");

        while (game_started == "1")
        {
            // QString receivedJSON = sendArduino();
            // parseJSON(receivedJSON);
            parseJSON(sendArduino());

            updateGameBoard();

            QCoreApplication::processEvents(); // Дозволяємо Qt оновити інтерфейс

            //Sleep(150);
        }

        ui->newButton->setEnabled(true);
        ui->loadButton->setEnabled(true);
        ui->saveButton->setEnabled(true);
    }
    else
    {
        // QString receivedJSON = sendArduino();
        // parseJSON(receivedJSON);
        parseJSON(sendArduino());

        updateGameBoard();
    }


}
void MainWindow::on_loadButton_clicked()
{
    loadGameState("game_state.JSON");
    updateGameBoard();

    if (game_mode == "mva") ui->radioButton_mai->setChecked(true);
    if (game_mode == "mvm") ui->radioButton_mvm->setChecked(true);
    if (game_mode == "ava") ui->radioButton_ava->setChecked(true);

    if (ai_strategy == "rand") ui->radioButton_rm->setChecked(true);
    if (ai_strategy == "win") ui->radioButton_ws->setChecked(true);
```

```
}
void MainWindow::on_saveButton_clicked() {
    saveGameState("game_state.JSON");
}


void MainWindow::loadComPorts()
{
    // Список для збереження імен портів
    QStringList comPortList;

    // Буфер для збереження імен пристроїв
    char lpTargetPath[5000]; // Буфер для збереження шляху до пристрою
    DWORD result;             // Результат виклику функції

    for (int i = 1; i <= 255; ++i) {
        // Формуємо ім'я порту: COM1, COM2, ..., COM255
        QString portName = QString("COM%1").arg(i);

        // Отримуємо інформацію про порт
        result = QueryDosDeviceA(portName.toStdString().c_str(), lpTargetPath, 5000);

        // Якщо порт існує, додаємо його до списку
        if (result != 0) {
            comPortList.append(portName);
        }
    }

    // Додаємо отримані порти в comboBox
    ui->comboBoxPorts->addItems(comPortList);
}

// Слот, який викликається при зміні вибраного порту
void MainWindow::onComboBoxPortChanged(const QString& port)
{
    ui->labelPort->setText(QString("Trying to connect via %1").arg(port));
    ui->labelPort->setStyleSheet("QLabel { color : lightblue; }");
    QCoreApplication::processEvents(); // Дозволяємо Qt оновити інтерфейс

    if (connectArduino(port)) {
        // Якщо відповідь від Arduino коректна
        ui->labelPort->setText("Arduino is connected");
        ui->labelPort->setStyleSheet("QLabel { color : lightgreen; }");

        // Активуємо кнопки
        ui->newButton->setEnabled(true);
        ui->loadButton->setEnabled(true);
        ui->saveButton->setEnabled(true);

        portArduino = port;
    }
    else {
        // Якщо помилка при зв'язку з Arduino
        ui->labelPort->setText(QString("Failed to connect via %1").arg(port));
        ui->labelPort->setStyleSheet("QLabel { color : red; }");

        // Деактивуємо кнопки
        ui->newButton->setEnabled(false);
        ui->loadButton->setEnabled(false);
        ui->saveButton->setEnabled(false);
```

```
    }
}


QString MainWindow::sendArduino() {

    QString JSONData = buildJSON();

    QString JSONData_trim = JSONData.remove(QChar(' '));
    JSONData_trim.remove(QChar('\n'));
    JSONData_trim.remove(QChar('\t'));

    std::string JSONString = JSONData_trim.toStdString();
    const char* dataToSend = JSONString.c_str();

    // qDebug() << "Sended  :" << dataToSend;

    DWORD bytesWritten;
    if (!WriteFile(hSerial, dataToSend, strlen(dataToSend), &bytesWritten, nullptr)) {
        CloseHandle(hSerial);
        return "";  // Помилка при відправці даних
    }

    // Чекаємо на відповідь
    char incomingData[256] = { 0 };
    DWORD bytesRead;
    if (!ReadFile(hSerial, incomingData, sizeof(incomingData), &bytesRead, nullptr)) {
        CloseHandle(hSerial);
        return "";  // Помилка при отриманні даних
    }

    // Закриваємо порт
    // CloseHandle(hSerial);

    // Перевіряємо, чи отримали правильну відповідь
    QString response(incomingData);
    // qDebug() << "Response:" << response;

    return response;
}


// Функція для тестування зв'язку з Arduino
bool MainWindow::connectArduino(const QString& portName)
{
    CloseHandle(hSerial);

    // Відкриваємо COM-порт
    hSerial = CreateFileA(portName.toStdString().c_str(),
        GENERIC_READ | GENERIC_WRITE,
        0, nullptr, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, nullptr);

    if (hSerial == INVALID_HANDLE_VALUE) {
        return false;  // Не вдалося відкрити порт
    }

    // Налаштування порту
    DCB dcbSerialParams = { 0 };
    dcbSerialParams.DCBlength = sizeof(dcbSerialParams);
    if (!GetCommState(hSerial, &dcbSerialParams)) {
        CloseHandle(hSerial);
```

```
        return false;  // Помилка отримання стану порту
    }

    dcbSerialParams.BaudRate = CBR_9600;  // Налаштування швидкості
    dcbSerialParams.ByteSize = 8;
    dcbSerialParams.StopBits = ONESTOPBIT;
    dcbSerialParams.Parity = NOPARITY;

    if (!SetCommState(hSerial, &dcbSerialParams)) {
        CloseHandle(hSerial);
        return false;  // Помилка налаштування порту
    }

    // Налаштування тайм-аутів
    COMMTIMEOUTS timeouts = { 0 };
    timeouts.ReadIntervalTimeout = 50;
    timeouts.ReadTotalTimeoutConstant = 50;
    timeouts.ReadTotalTimeoutMultiplier = 10;
    timeouts.WriteTotalTimeoutConstant = 50;
    timeouts.WriteTotalTimeoutMultiplier = 10;

    if (!SetCommTimeouts(hSerial, &timeouts)) {
        CloseHandle(hSerial);
        return false;  // Помилка налаштування тайм-аутів
    }

    Sleep(2000);

    resetValues();
    connect_arduino = "0";
    QString receivedJSON = sendArduino();

    // QString conValue = getTagValue(receivedJSON, "con");
    // qDebug() << "conValue:" << conValue;

    parseJSON(receivedJSON);

    return connect_arduino.trimmed() == "1";  // Порівнюємо з відповіддю Arduino
}

void MainWindow::add_player_turn(int row, int col)
{
    // connect_arduino, game_started, game_mode, ai_strategy, message, next_turn;
    if (connect_arduino != "1" || game_started != "1" || game_mode == "ava") {
        return;
    }


    if (board[row][col] != "x" && board[row][col] != "o")
    {
        if (next_turn == "x") { board[row][col] = "x"; }
        else { board[row][col] = "o"; }

        ui->label_pc_msg->setText("");
    }
    else {
        ui->label_pc_msg->setText("Select empty cell");
        return;
    }

    updateGameBoard();
```

```
    QCoreApplication::processEvents(); // Дозволяємо Qt оновити інтерфейс

    QString receivedJSON = sendArduino();
    parseJSON(receivedJSON);

    updateGameBoard();
}

void MainWindow::on_button_11_clicked() { add_player_turn(0, 0); }
void MainWindow::on_button_12_clicked() { add_player_turn(0, 1); }
void MainWindow::on_button_13_clicked() { add_player_turn(0, 2); }
void MainWindow::on_button_21_clicked() { add_player_turn(1, 0); }
void MainWindow::on_button_22_clicked() { add_player_turn(1, 1); }
void MainWindow::on_button_23_clicked() { add_player_turn(1, 2); }
void MainWindow::on_button_31_clicked() { add_player_turn(2, 0); }
void MainWindow::on_button_32_clicked() { add_player_turn(2, 1); }
void MainWindow::on_button_33_clicked() { add_player_turn(2, 2); }



void MainWindow::on_radioButton_mai_clicked() {
    game_mode = "mva";
    // qDebug() << "game_mode:" << game_mode;
}
void MainWindow::on_radioButton_mvm_clicked() {
    game_mode = "mvm";
    // qDebug() << "game_mode:" << game_mode;
}
void MainWindow::on_radioButton_ava_clicked() {
    game_mode = "ava";
    // qDebug() << "game_mode:" << game_mode;
}

void MainWindow::on_radioButton_rm_clicked() {
    ai_strategy = "rand";
    // qDebug() << "ai_strategy:" << ai_strategy;
}
void MainWindow::on_radioButton_ws_clicked() {
    ai_strategy = "win";
    // qDebug() << "ai_strategy:" << ai_strategy;
}
```

Файл **mainwindow.ui:**

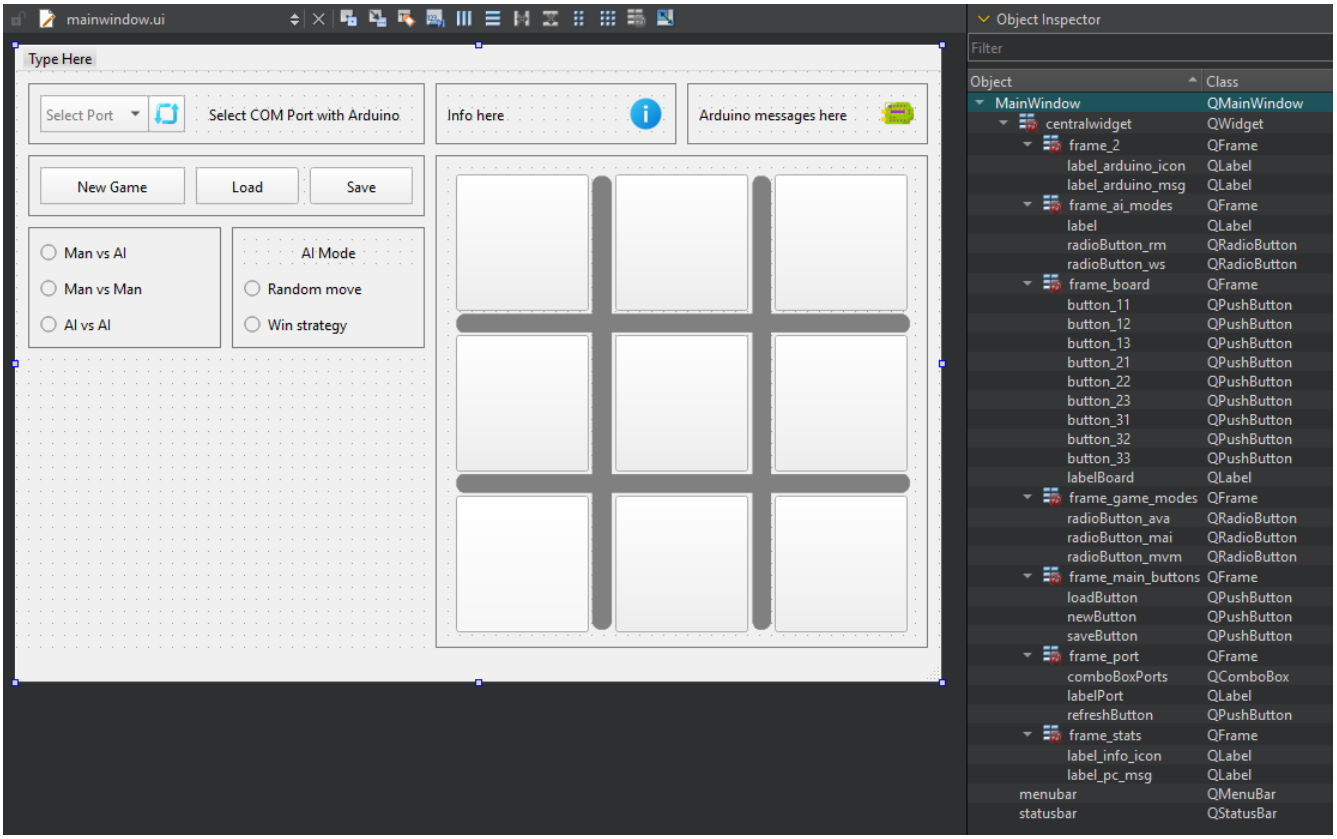

Рис. 1. mainwindow.ui в середовищі qt creator

2. Модифікував код для серверної частини з попередньої лабораторної роботи.

```
String connect_arduino, game_started, game_mode, ai_strategy, message, next_turn;
String board[3][3];

void setup() {

  Serial.begin(9600);  // Налаштування серійного зв'язку зі швидкістю 9600 біт/с
  randomSeed(analogRead(0));  // Ініціалізація генератора випадкових чисел
}

void loop() {
  if (Serial.available() > 0) {

    String receivedMessage = Serial.readString();  // Читаємо дані з серійного порту
    parseJSON(receivedMessage);

    if (connect_arduino == "0"){
```

```
      connect_arduino = "1";
    }

    if (game_started == "1" && game_mode == "mva" && ai_strategy == "rand") { makeRandomMove(); }

    if (game_started == "1" && game_mode == "ava" && ai_strategy == "rand") { avaMove(); }

    if (game_started == "1" && game_mode == "mva" && ai_strategy == "win") { stratMove("o", true); }

    if (game_started == "1" && game_mode == "ava" && ai_strategy == "win"){
      if (next_turn == "o") { stratMove("o", false); }
      else { stratMove("x", false); }
    }

    if (game_started == "1" && game_mode == "mvm") { check_mvm(); }

    String output = buildJSON();
    output.replace("\n", ""); // Заміна /n на пустий рядок
    Serial.print(output);

  }
}

bool checkBoardWin(String board[3][3], String player) {
  // Перевірка рядків (горизонталі)
  for (int i = 0; i < 3; i++) {
    if (board[i][0] == player && board[i][1] == player && board[i][2] == player) {
      return true;
    }
  }

  // Перевірка колонок (вертикалі)
  for (int i = 0; i < 3; i++) {
    if (board[0][i] == player && board[1][i] == player && board[2][i] == player) {
      return true;
    }
  }

  // Перевірка діагоналей
  if (board[0][0] == player && board[1][1] == player && board[2][2] == player) {
    return true;
  }
```

```
61.   if (board[0][2] == player && board[1][1] == player && board[2][0] == player) {
62.     return true;
63.   }
64.
65.   return false;
66. }
67.
68. bool allFieldsOccupied(String board[3][3]) {
69.   for (int i = 0; i < 3; i++) {
70.     for (int j = 0; j < 3; j++) {
71.       if (board[i][j] != "x" && board[i][j] != "o") {
72.         return false; // Знайшли пусте поле, повертаємо false
73.       }
74.     }
75.   }
76.   return true; // Якщо всі поля заповнені
77. }
78.
79. bool checkForWinner() {
80.
81.   if (checkBoardWin(board, "x")) {
82.     message = "Winner_X";
83.     game_started = "0";
84.     return true;
85.   }
86.   else if (checkBoardWin(board, "o")) {
87.     message = "Winner_O";
88.     game_started = "0";
89.     return true;
90.   }
91.   else if (allFieldsOccupied(board)) {
92.     message = "Draw";
93.     game_started = "0";
94.     return true;
95.   }
96.   else {
97.     return false;
98.   }
99. }
100.

101.      void avaMove() {
102.        int emptyCells[9];
103.        int emptyCount = 0;
104.        int x_cells = 0;
105.        int o_cells = 0;
106.
```

```
107.       if (checkForWinner()) { return; }
108.
109.       // Проходимо через поле та знаходимо всі порожні клітинки
110.       for (int row = 0; row < 3; row++) {
111.         for (int col = 0; col < 3; col++) {
112.
113.           if (board[row][col] == "x") {
114.             x_cells++;
115.           }
116.           else if (board[row][col] == "o") {
117.             o_cells++;
118.           }
119.           else {
120.             emptyCells[emptyCount] = row * 3 + col;  // Зберігаємо індекс
    клітинки у вигляді одного числа
121.             emptyCount++;
122.           }
123.
124.         }
125.       }
126.
127.       // Якщо є порожні клітинки, вибираємо випадкову
128.       if (emptyCount > 0) {
129.         int randomIndex = random(0, emptyCount);  // Випадковий індекс від 0
    до emptyCount-1
130.         int rand_cell = emptyCells[randomIndex];  // Індекс випадкової
    клітинки
131.
132.         // Перетворюємо індекс назад у координати (row, col)
133.         int row = rand_cell / 3;
134.         int col = rand_cell % 3;
135.
136.         if(next_turn == "o"){
137.           board[row][col] = "o";
138.           next_turn = "x";
139.         }
140.         else {
141.           board[row][col] = "x";
142.           next_turn = "o";
143.         }
144.       }
145.       else {
146.         message = "Draw";
147.         game_started = "0";
148.       }
149.     }
150.
```

```
151.     void check_mvm() {
152.       int emptyCells[9];
153.       int emptyCount = 0;
154.       int x_cells = 0;
155.       int o_cells = 0;
156.
157.       if (checkForWinner()) { return; }
158.
159.       // Проходимо через поле та знаходимо всі порожні клітинки
160.       for (int row = 0; row < 3; row++) {
161.         for (int col = 0; col < 3; col++) {
162.
163.           if (board[row][col] == "x") {
164.             x_cells++;
165.           }
166.           else if (board[row][col] == "o") {
167.             o_cells++;
168.           }
169.           else {
170.             emptyCells[emptyCount] = row * 3 + col;  // Зберігаємо індекс
    клітинки у вигляді одного числа
171.             emptyCount++;
172.           }
173.         }
174.       }
175.
176.       // Вибір гравця для першого ходу
177.       if (emptyCount == 9) {
178.         int randomChoice = random(0, 2);  // Випадковий вибір: 0 або 1
179.         if (randomChoice == 0) {
180.           message = "Player_X_turn";
181.           next_turn = "x";
182.           return;
183.         }
184.         else {
185.           message = "Player_O_turn";
186.           next_turn = "o";
187.           return;
188.         }
189.       }
190.
191.       if(next_turn == "x") {
192.         message = "Player_O_turn";
193.         next_turn = "o";
194.       }
195.       else {
196.         message = "Player_X_turn";
```

```
197.         next_turn = "x";
198.       }
199.     }
200.
201.     void makeRandomMove() {
202.       int emptyCells[9];
203.       int emptyCount = 0;
204.       int x_cells = 0;
205.       int o_cells = 0;
206.
207.       if (checkForWinner()) { return; }
208.
209.       // Проходимо через поле та знаходимо всі порожні клітинки
210.       for (int row = 0; row < 3; row++) {
211.         for (int col = 0; col < 3; col++) {
212.
213.           if (board[row][col] == "x") {
214.             x_cells++;
215.           }
216.           else if (board[row][col] == "o") {
217.             o_cells++;
218.           }
219.           else {
220.             emptyCells[emptyCount] = row * 3 + col;  // Зберігаємо індекс
    клітинки у вигляді одного числа
221.             emptyCount++;
222.           }
223.         }
224.       }
225.
226.       next_turn = "x";
227.       message = "Player_x_turn";
228.
229.       if (emptyCount == 9) {
230.         int randomChoice = random(0, 2);  // Випадковий вибір: 0 або 1
231.         if (randomChoice == 0) {
232.           return;
233.         }
234.       }
235.
236.       // Якщо є порожні клітинки, вибираємо випадкову
237.       if (emptyCount > 0) {
238.         int randomIndex = random(0, emptyCount);  // Випадковий індекс від 0
    до emptyCount-1
239.         int rand_cell = emptyCells[randomIndex];  // Індекс випадкової
    клітинки
240.
```

```
241.         // Перетворюємо індекс назад у координати (row, col)
242.         int row = rand_cell / 3;
243.         int col = rand_cell % 3;
244.
245.         // Записуємо "o" в обрану клітинку
246.         board[row][col] = "o";
247.       }
248.
249.       checkForWinner();
250.     }
251.

252.     bool checkWin(String player, String currentPlayer) {
253.       for (int i = 0; i < 3; i++) {
254.         // Перевірка рядків і стовпців
255.         if ((board[i][0] == player && board[i][1] == player && board[i][2] ==
    "-") ||
256.             (board[i][0] == player && board[i][2] == player && board[i][1] ==
    "-") ||
257.             (board[i][1] == player && board[i][2] == player && board[i][0] ==
    "-")) {
258.           board[i][0] == "-" ? board[i][0] = currentPlayer :
259.           board[i][1] == "-" ? board[i][1] = currentPlayer : board[i][2] =
    currentPlayer;
260.           return true;
261.         }
262.
263.         if ((board[0][i] == player && board[1][i] == player && board[2][i] ==
    "-") ||
264.             (board[0][i] == player && board[2][i] == player && board[1][i] ==
    "-") ||
265.             (board[1][i] == player && board[2][i] == player && board[0][i] ==
    "-")) {
266.           board[0][i] == "-" ? board[0][i] = currentPlayer :
267.           board[1][i] == "-" ? board[1][i] = currentPlayer : board[2][i] =
    currentPlayer;
268.           return true;
269.         }
270.       }
271.
272.       // Перевірка діагоналей
273.       if ((board[0][0] == player && board[1][1] == player && board[2][2] ==
    "-") ||
274.           (board[0][0] == player && board[2][2] == player && board[1][1] ==
    "-") ||
275.           (board[1][1] == player && board[2][2] == player && board[0][0] ==
    "-")) {
```

```
276.          board[0][0] == "-" ? board[0][0] = currentPlayer :
277.          board[1][1] == "-" ? board[1][1] = currentPlayer : board[2][2] =
   currentPlayer;
278.          return true;
279.        }
280.
281.        if ((board[0][2] == player && board[1][1] == player && board[2][0] ==
   "-") ||
282.            (board[0][2] == player && board[2][0] == player && board[1][1] ==
   "-") ||
283.            (board[1][1] == player && board[2][0] == player && board[0][2] ==
   "-")) {
284.          board[0][2] == "-" ? board[0][2] = currentPlayer :
285.          board[1][1] == "-" ? board[1][1] = currentPlayer : board[2][0] =
   currentPlayer;
286.          return true;
287.        }
288.
289.        return false;
290.      }
291.
292.      void stratMove(String currentPlayer, bool randomPlayer) {
293.
294.        if (checkForWinner()) { return; }
295.
296.        bool made_move = false;
297.
298.        String opponent = (currentPlayer == "x") ? "o" : "x";
299.
300.        next_turn = opponent;
301.        message = "Player_"+ opponent +"_turn";
302.
303.        int emptyCount = 0;
304.
305.        for (int row = 0; row < 3; row++) {
306.          for (int col = 0; col < 3; col++) {
307.
308.            if (board[row][col] != "x" && board[row][col] != "0") {
309.              emptyCount++;
310.            }
311.          }
312.        }
313.
314.        if (randomPlayer){
315.          if (emptyCount == 9) {
316.            int randomChoice = random(0, 2);  // Випадковий вибір: 0 або 1
317.            if (randomChoice == 0) {
```

```
318.             return;
319.           }
320.         }
321.       }
322.
323.       // 1. Спробуємо виграти
324.       if (!made_move && checkWin(currentPlayer, currentPlayer)) {
325.         made_move = true;
326.       }
327.
328.       // 2. Заблокуємо "x", якщо він може виграти
329.       if (!made_move && checkWin(opponent, currentPlayer)) {
330.         made_move = true;
331.       }
332.
333.       // 3. Займаємо центр, якщо він вільний
334.       if (!made_move && board[1][1] == "-") {
335.         board[1][1] = currentPlayer;
336.         made_move = true;
337.       }
338.
339.       // 4. Займаємо кути, якщо вони вільні
340.       if (!made_move && board[0][0] == "-") {
341.         board[0][0] = currentPlayer;
342.         made_move = true;
343.       }
344.       if (!made_move && board[0][2] == "-") {
345.         board[0][2] = currentPlayer;
346.         made_move = true;
347.       }
348.       if (!made_move && board[2][0] == "-") {
349.         board[2][0] = currentPlayer;
350.         made_move = true;
351.       }
352.       if (!made_move && board[2][2] == "-") {
353.         board[2][2] = currentPlayer;
354.         made_move = true;
355.       }
356.
357.       // 5. Займаємо будь-яке інше місце
358.       for (int i = 0; i < 3; i++) {
359.         for (int j = 0; j < 3; j++) {
360.
361.           if (!made_move && board[i][j] == "-") {
362.             board[i][j] = currentPlayer;
363.             made_move = true;
364.           }
```

```
365.          }
366.        }
367.
368.        if (checkForWinner()) { return; }
369.
370.      }
371.
372.      String buildJSON() {
373.        String output = "{";
374.        output += "\"con\":\"" + connect_arduino + "\",";
375.        output += "\"gs\":\"" + game_started + "\",";
376.        output += "\"gm\":\"" + game_mode + "\",";
377.        output += "\"ais\":\"" + ai_strategy + "\",";
378.        output += "\"msg\":\"" + message + "\",";
379.        output += "\"nt\":\"" + next_turn + "\",";
380.        output += "\"brd\":{";
381.        output += "\"c11\":\"" + board[0][0] + "\",";
382.        output += "\"c12\":\"" + board[0][1] + "\",";
383.        output += "\"c13\":\"" + board[0][2] + "\",";
384.        output += "\"c21\":\"" + board[1][0] + "\",";
385.        output += "\"c22\":\"" + board[1][1] + "\",";
386.        output += "\"c23\":\"" + board[1][2] + "\",";
387.        output += "\"c31\":\"" + board[2][0] + "\",";
388.        output += "\"c32\":\"" + board[2][1] + "\",";
389.        output += "\"c33\":\"" + board[2][2] + "\"";
390.        output += "}";
391.        output += "}";
392.        return output;
393.      }
394.
395.      String getJSONValue(String input, String key) {
396.        int start = input.indexOf("\"" + key + "\":\"") + key.length() + 4;
397.        int end = input.indexOf("\"", start);
398.
399.        if (start == -1 || end == -1) {
400.          return "-";
401.        }
402.
403.        return input.substring(start, end);
404.      }
405.
406.      void parseJSON(String input) {
407.        connect_arduino = getJSONValue(input, "con");
408.        game_started = getJSONValue(input, "gs");
409.        game_mode = getJSONValue(input, "gm");
410.        ai_strategy = getJSONValue(input, "ais");
```

```
411.         message = getJSONValue(input, "msg");
412.
413.         board[0][0] = getJSONValue(input, "c11");
414.         board[0][1] = getJSONValue(input, "c12");
415.         board[0][2] = getJSONValue(input, "c13");
416.         board[1][0] = getJSONValue(input, "c21");
417.         board[1][1] = getJSONValue(input, "c22");
418.         board[1][2] = getJSONValue(input, "c23");
419.         board[2][0] = getJSONValue(input, "c31");
420.         board[2][1] = getJSONValue(input, "c32");
421.         board[2][2] = getJSONValue(input, "c33");
422.     }
```

3. Провів перевірку на працездатність.

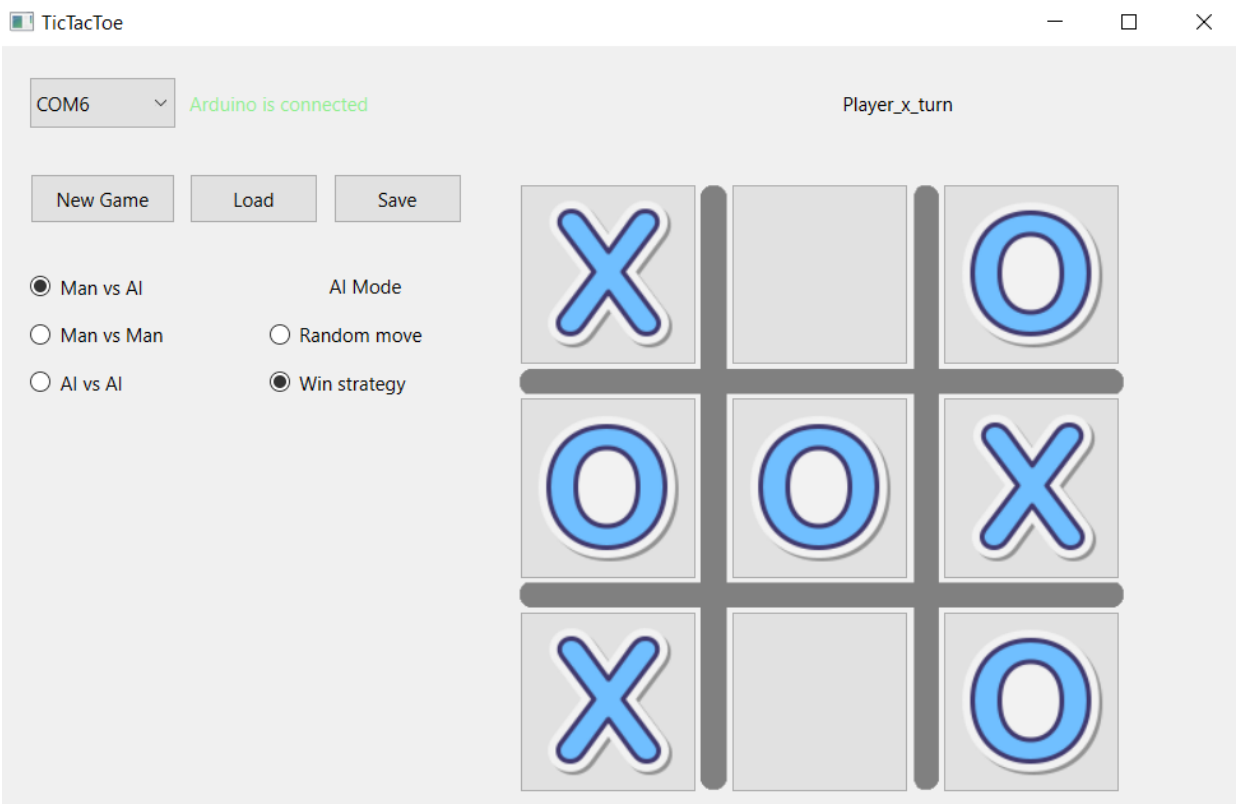

Рис. 2. Вікно програми

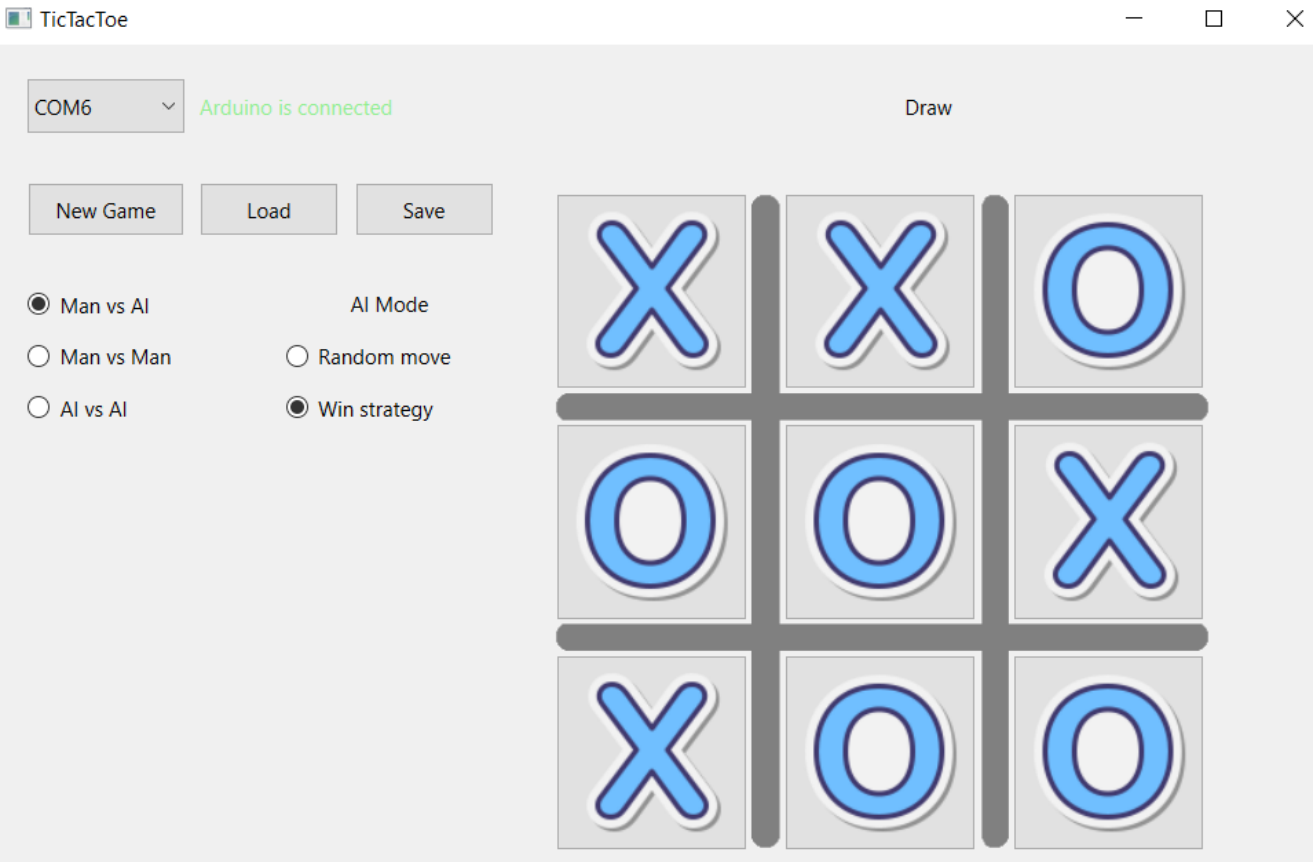

Рис. 3. Гра завершилась нічиєю

4. В результаті завантажив роботу на GitHub :

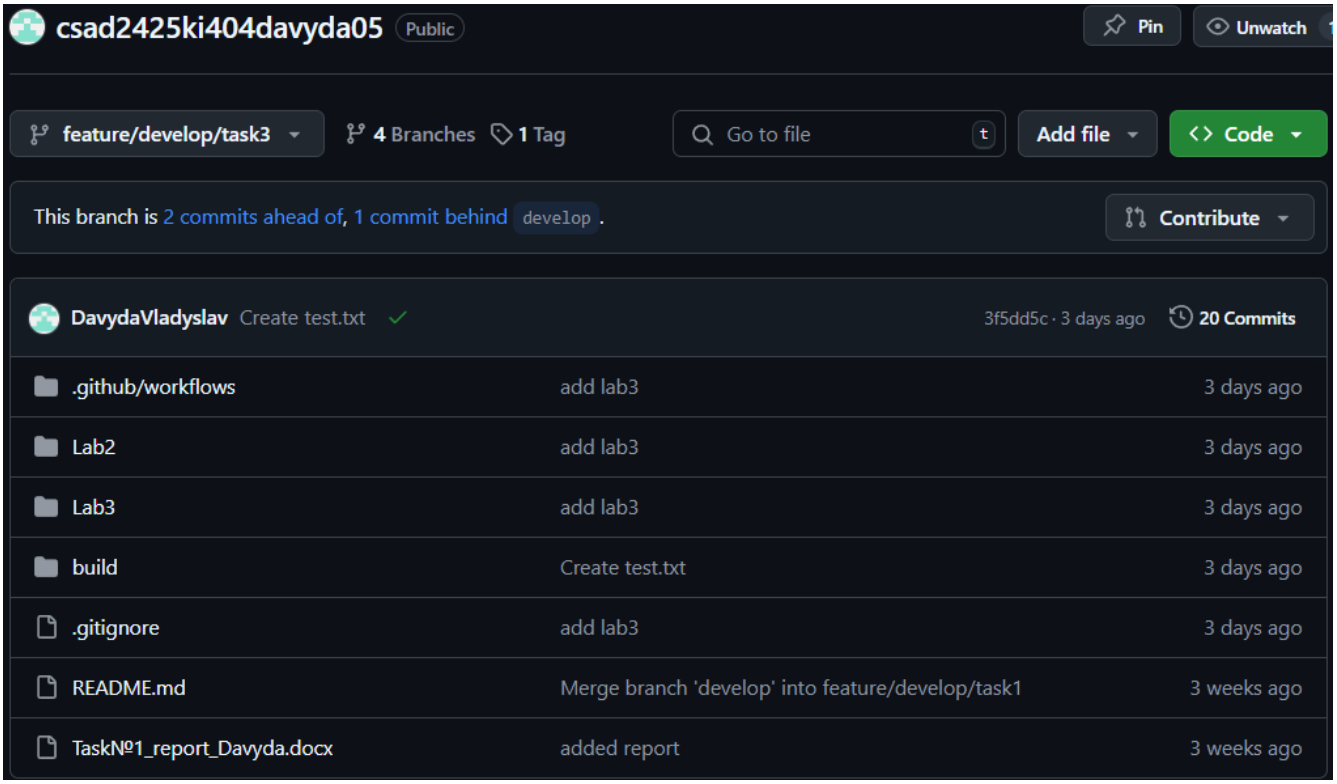

Рис. 4.

Висновок:

В ході виконання лабораторної роботи було розроблено клієнт-серверну гру "хрестики-нулики", яка використовує комунікацію між клієнтською програмою та апаратним забезпеченням через UART інтерфейс. Цей підхід дозволяє взаємодіяти з грою за допомогою апаратного забезпечення.