Tic-Tac-Toe

1. Introduction

Why this project?

This project represents something that I would have wanted to play with when I was a child. Instead of using a paper and a pen, now I can play Tic-Tac-Toe using a device created by me.

What is this project?

The game includes a simple user interface with LEDs and, using the Serial Port, it takes the input from the users and lets two players compete.

2. Design

This game board will be a 3x3 grid of LEDs and all user's input will be taken from the Serial Port. The Arduino will handle the game logic and the LEDs.

Another interesting thing about the design, each player has its own way of turning on the LEDs: for the first player, the LEDs will always be on and for the second player, the LEDs will always blink. It makes things easier for the players and also gives the game board a special look.

3. Required Components

- **Hardware**

  In order to build this project, you will need some hardware components:

➔ Arduino Uno
➔ 9 LEDs (for the 3x3 grid)
➔ 3 Resistors
➔ Breadboard
➔ Jumper wires

Besides the hardware components, some software components are also required:

- **Software**
  ➔ Arduino IDE

4. Assembling the Hardware

- **Grid Setup**
- **Power Connections**

Ensure that the Arduino is properly powered using a USB cable.

- **Photos&Diagrams**

5. Game Logic

The code uses a 2D array to represent the game board. It checks for valid moves, switches turns between players, and evaluates the board for a win or draw after each move. It also provides the option to start a new round.

6. Code Structure

For this project we will use a couple of variables:

```
int current_player = 1;
int p11, p12, p13 = 0;
int p21, p22, p23 = 0;
int p31, p32, p33 = 0;
int blink_position[20] = {0};
int contor = 0;
unsigned long lastMillis = 0;
int blinking = 0;
int player_1_matrix[10] = {0};
int player_2_matrix[10] = {0};
int m1 = 0;
int m2 = 0;
bool gameOver = false;

int player_1_score = 0;
int player_2_score = 0;
```

,

where *p11-p33* will be used for tracking if an user took a certain position in the LEDs matrix. The *blink_position* array is used to keep the moves that player2 took. Also, *blinking* variable is used to synchronize the LEDs that are blinking.

```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(5, OUTPUT);
  Serial.begin(9600);
  Serial.println("Scrie pozitia din matrice unde vrei sa pozitionezi piesa. Ex: 11 pentru linia 1 si coloana 1.");
}
```

The **setup()** function prepares the pins that will be used in this project.

```
42  void loop() {
43
44    if (gameOver) {
45      if (Serial.available() > 0) {
46        int input = Serial.parseInt();
47        if (input == 100) {
48          resetGame();
49        } else {
50          Serial.println("Scrie '100' pentru a incepe un nou joc.");
51        }
52      }
53      return;
54    }
```

The **loop()** function contains all the code responsible for the logic of the game. First of all, we check if the game is over and if *gameOver* is true, then we ask the player to write '100' as a confirmation of giving the consent of starting a new round. If the users input is '100', we start a new round by calling the *resetGame()* function.

```
56  if (Serial.available() > 0) {
57    int input = Serial.parseInt();
58    int c = 0;
59
60    if (castigator_pe_diagonala(player_2_matrix, m2) || castigator_pe_linie(player_2_matrix, m2) || castigator_pe_coloana(player_2_matrix, m2)) {
61      Serial.println("\nJucatorul 2 a castigat!");
62
63      player_2_score++;
64
65      Serial.print("Scor: Player 1 - ");
66      Serial.print(player_1_score);
67      Serial.print(" | Player 2 - ");
68      Serial.println(player_2_score);
69
70      gameOver = true;
71      Serial.println("Scrie '100' pentru a incepe un nou joc.");
72      return;
73    }else if (castigator_pe_diagonala(player_1_matrix, m1) || castigator_pe_linie(player_1_matrix, m1) || castigator_pe_coloana(player_1_matrix, m1)) {
74      Serial.println("\nJucatorul 1 a castigat!");
75
76      player_1_score++;
77
78      Serial.print("Scor: Player 1 - ");
79      Serial.print(player_1_score);
80      Serial.print(" | Player 2 - ");
81      Serial.println(player_2_score);
82
83      gameOver = true;
84      Serial.println("Scrie '100' pentru a incepe un nou joc.");
85      return;
```

If the game is not over, we check if a player won the game and we print the score accordingly. At this step we also ask them if they want to continue with another round.

```
86      }else if (m1 + m2 == 9) {
87        Serial.println("\nRemiza!");
88
89        Serial.print("Scor: Player 1 - ");
90        Serial.print(player_1_score);
91        Serial.print(" | Player 2 - ");
92        Serial.println(player_2_score);
93
94        gameOver = true;
95        Serial.println("Scrie '100' pentru a incepe un nou joc.");
96        return;
97
98      }
```

Also do not forget to check for *draw*.

```
118     if (input == 11) {
119       if (p11 == 0) {
120         digitalWrite(13, HIGH);
121         Serial.print("Jucatorul ");
122         Serial.print(c);
123         Serial.print(" a mutat la pozitia ");
124         Serial.print(input);
125         change_player(current_player);
126         p11 = 1;
127 
128         if (c == 2) {
129           blink_position[P11] = 1;
130           player_2_matrix[m2++] = 13;
131 
132         }else {
133           player_1_matrix[m1++] = 13;
134         }
135 
136       }else if (p11 == 1) {
137         Serial.print("\nPozitia ");
138         Serial.print(input);
139         Serial.print(" este ocupata.");
140       }
```

In this snippet of code we check if the player's option is valid. Otherwise, we print a message and wait for a valid option.

```
333     if (blinking == 1) {
334       unsigned long currentMillis = millis();
335 
336       if (currentMillis - lastMillis >= 200) {
337           lastMillis = currentMillis;
338 
339           for (int i = 0; i < 20; i++) {
340             if (blink_position[i] == 1)
341               digitalWrite(i, !digitalRead(i));
342           }
343       }
344 
345     }
346 }
```

Here we make sure that all the player2 LEDs are blinking synchronized.

```
348 void change_player(int &current_player){
349   if (current_player == 1) {
350       current_player = 2;
351   }else if (current_player == 2) {
352       current_player = 1;
353   }
354 }
```

Another important aspect is to change the player after every move. If we do not do this, we would not be able to print the correct winner.

```
356  bool castigator_pe_diagonala(int matrix[], int m){
357    bool gasit_5 = false;
358    bool gasit_9 = false;
359    bool gasit_13 = false;
360    bool gasit_7 = false;
361    bool gasit_11 = false;
362
363    for (int i = 0; i < m; i++) {
364      if (matrix[i] == 5) {
365        gasit_5 = true;
366      }else if (matrix[i] == 7) {
367        gasit_7 = true;
368      }else if (matrix[i] == 9) {
369        gasit_9 = true;
370      }else if (matrix[i] == 11) {
371        gasit_11 = true;
372      }else if (matrix[i] == 13) {
373        gasit_13 = true;
374      }
375    }
376
377    if (gasit_5 == true && gasit_9 == true && gasit_13 == true) {
378      Serial.println("\nCombinatia castigatoare este: 11, 22, 33");
379      allLedsUp(5, 9, 13);
380      return true;
381    }else if (gasit_7 == true && gasit_9 == true && gasit_11 == true) {
382      Serial.println("\nCombinatia castigatoare este: 13, 22, 31");
383       allLedsUp(7, 9, 11);
384      return true;
385    }else return false;
386  }
387
```

In order to find out if a player is winning the game, we constantly check the occupied positions of the matrix. If any of the above combinations are valid, then we can declare a winner. This is an example for *diagonal*, but the calculations for *row* or *column* are similar.

```
484  void resetGame() {
485    for (int i = 5; i <= 13; i++) {
486      digitalWrite(i, LOW);
487    }
488
489    p11 = p12 = p13 = 0;
490    p21 = p22 = p23 = 0;
491    p31 = p32 = p33 = 0;
492
493    m1 = m2 = 0;
494
495    for (int i = 0; i < 10; i++) {
496      player_1_matrix[i] = 0;
497      player_2_matrix[i] = 0;
498    }
499
500    for (int i = 0; i < 20; i++) {
501      blink_position[i] = 0;
502    }
503
504    blinking = false;
505    gameOver = false;
506    current_player = 1;
507    Serial.println("Incepe o noua runda. Player 1 incepe!");
508  }
```

In order to start a new round, we have to clear all arrays and variables that were used at the previous steps.

```
510  void allLedsUp(int l1, int l2, int l3) {
511    for (int i = 5; i <= 13; i++) {
512      if (i != l1 && i != l2 && i != l3) {
513        digitalWrite(i, LOW);
514      }else {
515        digitalWrite(i, HIGH);
516      }
517    }
```

The final step when we declare a winner, we emphasize the winner's moves by lighting up the corresponding LEDs.

7. Testing&Debugging

Test that each LED lights up when its corresponding pin is activated. If a LED does not light up, make sure that the wire corresponding to that pin is properly plugged in. Also ensure pull-down resistors are correctly connected.

8. Conclusion

This project made me understand better how to use an Arduino and how you can have fun while building something to have fun with later. It was a great experience that improved both my coding and hardware skills. I wish I knew all of these earlier!