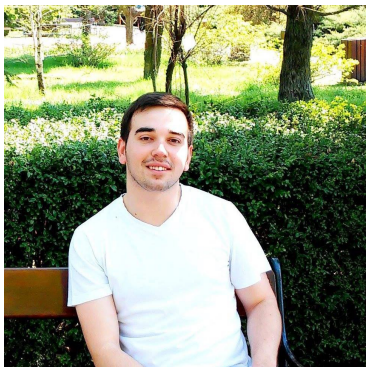
Project name: TIC-TAC-TOE Game

1. **Students involved:**

   1. **Luca Mihai-Adrian**

adrian.luca001@gmail.com

   B. **Magas Emanuel-Vladut**



vladut.magas@yahoo.com

3. Summary

The project's scope is to create a "tic-tac-toe" game on a development board (XMC 4700 Relax Kit Series V1) which can be played by any user, regardless of their knowledge in embedded systems.

4. Introduction

4.1. Generalities

Tic-tac-toe (also known as noughts and crosses or Xs and Os) is a game for two players, *X* and *O*, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.

4.2. Utility

Tic-tac-toe is a game which can be played by people of all ages, any time, anywhere. The game is usually used in enhancing the players thinking by teaching the concepts of good sportsmanship and the branch of artificial intelligence that deals with the searching of game trees.

4.3. Resources

Hardware:

1. Infineon XMC 4700/4800 Relax Kit Series-V1
2. HMI_OLED-V1-
3. UNI_EXT01-V2

Software:

1. DAVE 4.3.2 (64 bit edition)

5. Implementation

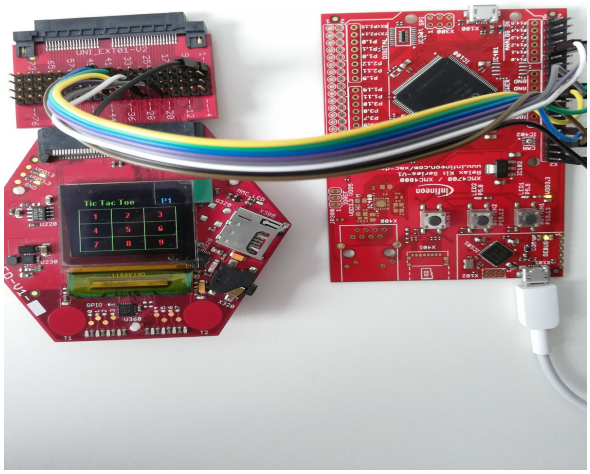
The implementation was done using DAVE, which is a software development tool that uses the C/C++ programming language. The project relies on the game's strategy: the player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game. In order to accomplish this, we used the three buttons provided by the development board:

1. The reset button (RESET) in order to reset the game or start a new one.
2. BUTTON1 (P15.13) is used to select the square in which the player wants to mark X or O.
3. BUTTON2 (P15.12) is used in order to apply the marking and allow the next player to make its move.

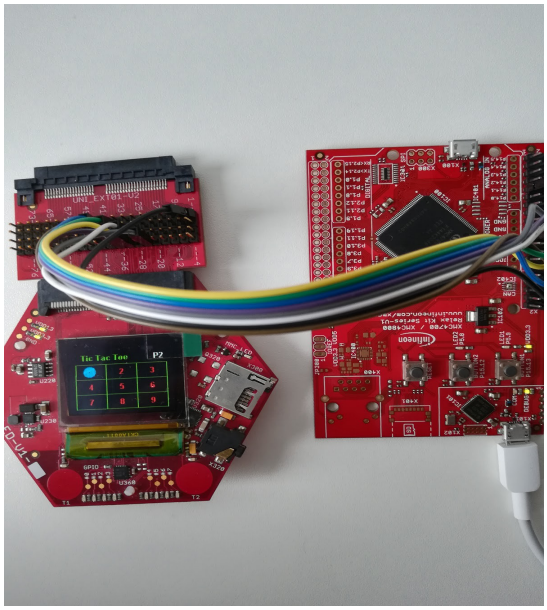
The graphic interface for the game is displayed using the HMI_OLED-V1- which uses the Segger emWin library.

6. Results/demonstrative sequence

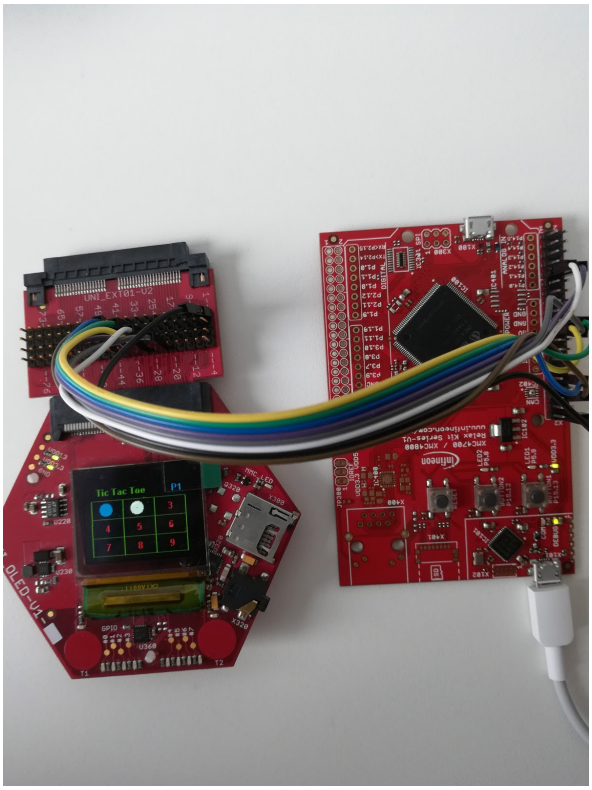
1. The start of the game



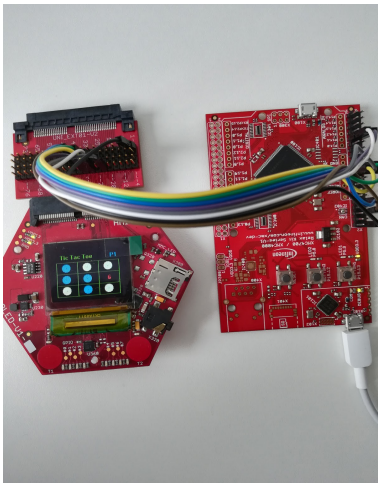
2. Player one marked it's square



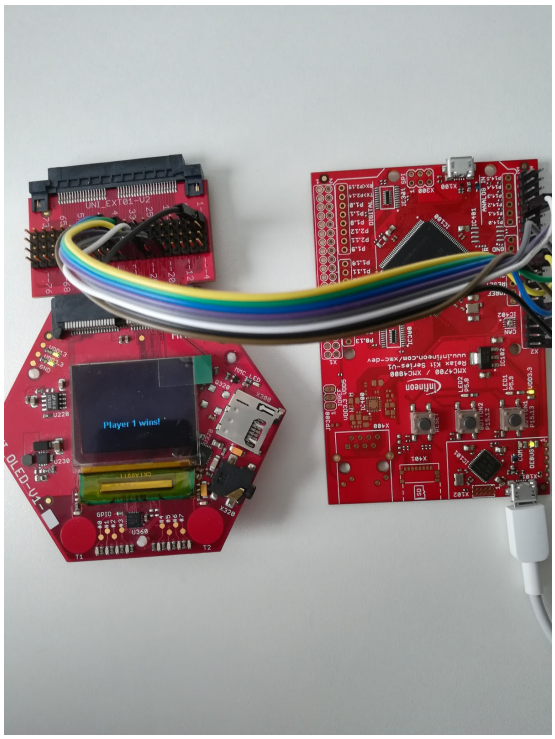
3. Player 2 choose its square



4. After a few moves



5. The result of the game



7. Source code

```
#include <DAVE.h>

int main(void)
{
 DAVE_Init();
 GUI_Init();

 int i;
 int ret_inc=1, ret_set=1, incr=1, pln=1 ;
 int f1=0,f2=0,f3=0,f4=0,f5=0,f6=0,f7=0,f8=0,f9=0;
 int c1=0,c2=0,c3=0,c4=0,c5=0,c6=0,c7=0,c8=0,c9=0;
```

```
while(1)
{

    if(ret_set==0)
    {
    switch(incr)
    {
    case 1:
    if(f1!=1)
    {
    f1=1;
        if(pln==1)
        {
        if(c1==0)
        c1=1;
        }
        else
        {
        if(c1==0)
        c1=2;
        }
    if(pln==1)
    pln=2;
    else
    pln=1;
    }
        break;
    case 2:
    if(f2!=1)
    {
    f2=1;
        if(pln==1)
          {
          if(c2==0)
          c2=1;
          }
          else
          {
```

```
        if(c2==0)
        c2=2;
        }
if(pln==1)
pln=2;
else
pln=1;
}
    break;
case 3:
if(f3!=1)
{
f3=1;
if(pln==1)
        {
        if(c3==0)
        c3=1;
        }
        else
        {
        if(c3==0)
        c3=2;
        }
if(pln==1)
pln=2;
else
pln=1;
}
    break;
case 4:
if(f4!=1)
{
f4=1;
if(pln==1)
        {
        if(c4==0)
        c4=1;
        }
```

```
        else
        {
        if(c4==0)
        c4=2;
        }
if(pln==1)
pln=2;
else
pln=1;
}
break;
case 5:
if(f5!=1)
{
f5=1;
if(pln==1)
        {
        if(c5==0)
        c5=1;
        }
        else
        {
        if(c5==0)
        c5=2;
        }
if(pln==1)
pln=2;
else
pln=1;
}
    break;
case 6:
if(f6!=1)
{
f6=1;
if(pln==1)
        {
        if(c6==0)
```

```
        c6=1;
        }
        else
        {
        if(c6==0)
        c6=2;
        }
if(pln==1)
pln=2;
else
pln=1;
}
    break;
case 7:
if(f7!=1)
{
f7=1;
if(pln==1)
        {
        if(c7==0)
        c7=1;
        }
        else
        {
        if(c7==0)
        c7=2;
        }
if(pln==1)
pln=2;
else
pln=1;
}
    break;
case 8:
if(f8!=1)
{
f8=1;
if(pln==1)
```

```
        {
        if(c8==0)
        c8=1;
        }
        else
        {
        if(c8==0)
        c8=2;
        }
if(pln==1)
pln=2;
else
pln=1;
}
    break;
case 9:
if(f9!=1)
{
f9=1;
if(pln==1)
        {
        if(c9==0)
        c9=1;
        }
        else
        {
        if(c9==0)
        c9=2;
        }
if(pln==1)
pln=2;
else
pln=1;
}
    break;
}

incr=1;
```

```
    GUI_Clear();
    }
GUI_SetFont(&GUI_Font16B_1);

GUI_SetColor(GUI_GREEN);
GUI_DispStringAt("Tic Tac Toe", 18, 20);
GUI_DrawLine(10,40,150,40);
GUI_DrawLine(10,120,150,120);
GUI_DrawLine(10,40,10,120);
GUI_DrawLine(150,40,150,120);
GUI_DrawLine(57,40,57,120);
GUI_DrawLine(104,40,104,120);
GUI_DrawLine(10,67,150,67);
GUI_DrawLine(10,94,150,94);

GUI_SetColor(GUI_RED);
if(f1==0)
  GUI_DispStringAt("1", 31, 46);
else
{
    if(c1==1)
    GUI_SetColor(GUI_BLUE);
    if(c1==2)
    GUI_SetColor(GUI_WHITE);
    for(i=1;i<10;i++)
    GUI_DrawCircle(31, 53, i);
}
GUI_SetColor(GUI_RED);
if(f2==0)
    GUI_DispStringAt("2", 78, 46);
else
{
    if(c2==1)
    GUI_SetColor(GUI_BLUE);
    if(c2==2)
    GUI_SetColor(GUI_WHITE);
    for(i=1;i<10;i++)
    GUI_DrawCircle(78, 53, i);
```

```
}
GUI_SetColor(GUI_RED);
if(f3==0)
    GUI_DispStringAt("3", 125, 46);
else
{
    if(c3==1)
    GUI_SetColor(GUI_BLUE);
    if(c3==2)
    GUI_SetColor(GUI_WHITE);
    for(i=1;i<10;i++)
    GUI_DrawCircle(125, 53, i);
}
GUI_SetColor(GUI_RED);
if(f4==0)
    GUI_DispStringAt("4", 31, 73);
else
{
    if(c4==1)
    GUI_SetColor(GUI_BLUE);
    if(c4==2)
    GUI_SetColor(GUI_WHITE);
    for(i=1;i<10;i++)
    GUI_DrawCircle(31, 80, i);
}
GUI_SetColor(GUI_RED);
if(f5==0)
    GUI_DispStringAt("5", 78, 73);
else
{
    if(c5==1)
    GUI_SetColor(GUI_BLUE);
    if(c5==2)
    GUI_SetColor(GUI_WHITE);
    for(i=1;i<10;i++)
    GUI_DrawCircle(78, 80, i);
}
GUI_SetColor(GUI_RED);
```

```
if(f6==0)
    GUI_DispStringAt("6", 125, 73);
else
{
    if(c6==1)
    GUI_SetColor(GUI_BLUE);
    if(c6==2)
    GUI_SetColor(GUI_WHITE);
    for(i=1;i<10;i++)
    GUI_DrawCircle(125, 80, i);
}
GUI_SetColor(GUI_RED);
if(f7==0)
    GUI_DispStringAt("7", 31, 100);
else
{
    if(c7==1)
    GUI_SetColor(GUI_BLUE);
    if(c7==2)
    GUI_SetColor(GUI_WHITE);
    for(i=1;i<10;i++)
    GUI_DrawCircle(31, 106, i);
}
GUI_SetColor(GUI_RED);
if(f8==0)
    GUI_DispStringAt("8", 78, 100);
else
{
    if(c8==1)
    GUI_SetColor(GUI_BLUE);
    if(c8==2)
    GUI_SetColor(GUI_WHITE);
    for(i=1;i<10;i++)
    GUI_DrawCircle(78, 106, i);
}
GUI_SetColor(GUI_RED);
if(f9==0)
    GUI_DispStringAt("9", 125, 100);
```

```
 else
 {
     if(c9==1)
     GUI_SetColor(GUI_BLUE);
     if(c9==2)
     GUI_SetColor(GUI_WHITE);
     for(i=1;i<10;i++)
     GUI_DrawCircle(125, 106, i);
 }

 if(pln==1)
 {
     GUI_SetColor(GUI_BLUE);
     GUI_DispStringAt("P1", 130, 20);
 }
 else
 {
     GUI_SetColor(GUI_WHITE);
     GUI_DispStringAt("P2", 130, 20);
 }
 GUI_SetColor(GUI_RED);

 if((c1==1&&c2==1&&c3==1)||(c4==1&&c5==1&&c6==1)||(c7==1&&c8==1&&c9==1)||(c1==1&&c4==1&&c7==1)||
(c2==1&&c5==1&&c8==1)||(c3==1&&c6==1&&c9==1)||(c1==1&&c5==1&&c9==1)||(c3==1&&c5==1&&c7==1))
 {
     GUI_Clear();
     GUI_SetColor(GUI_BLUE);
     GUI_DispStringAt("Player 1 wins!", 31, 80);
     break;
 }
 else
 {
 if((c1==2&&c2==2&&c3==2)||(c4==2&&c5==2&&c6==2)||(c7==2&&c8==2&&c9==2)||(c1==2&&c4==2&&c7==2)||
(c2==2&&c5==2&&c8==2)||(c3==2&&c6==2&&c9==2)||(c1==2&&c5==2&&c9==2)||(c3==2&&c5==2&&c7==2))
{
        GUI_Clear();
        GUI_SetColor(GUI_WHITE);
        GUI_DispStringAt("Player 2 wins!", 31, 80);
        break;
```

```
}
 }

 label2:
 ret_inc = DIGITAL_IO_GetInput(&inc);

 if(ret_inc==0)
 {
     for( int i=0;i<0x1fffff;i++);
     incr++;
     if(incr>9)
     incr=1;
 }
     ret_set=DIGITAL_IO_GetInput(&set);
     if(ret_set==0)
     goto label;
     else
     goto label2;
 label: ;

 }
}
```

8. Bibliography

1. http://embedac.ro/CI/index.html
2. https://www.segger.com/products/user-interface/emwin/
3. https://www.infineon.com/dgdl/Infineon-Board_User_Manual_XMC4700_XMC4800_Relax_Kit_Series-UM-v01_02-EN.pdf?fileId=5546d46250cc1fdf01513f8e052d07fc
4. https://www.infineon.com/dgdl/Board_Users_Manual_Standard_HMI_Card_R1.0.pdf?fileId=db3a304335f1f4b60135f24d0fe50038
5. https://en.wikipedia.org/wiki/Tic-tac-toe