Meenakshi Shankar
Sr No. 22400
M.Tech EPD

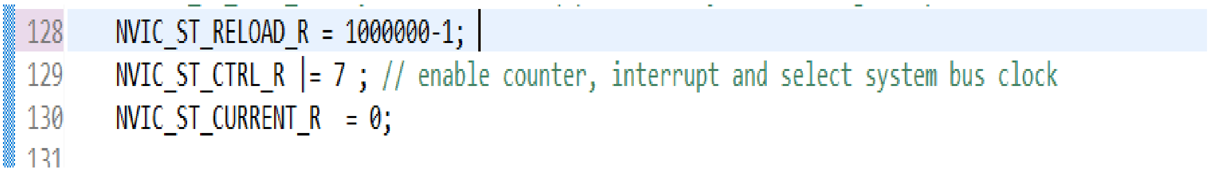
EMBEDDED SYSTEM DESIGN (E3-257)

LAB ASSIGNMENT – 6 and 7

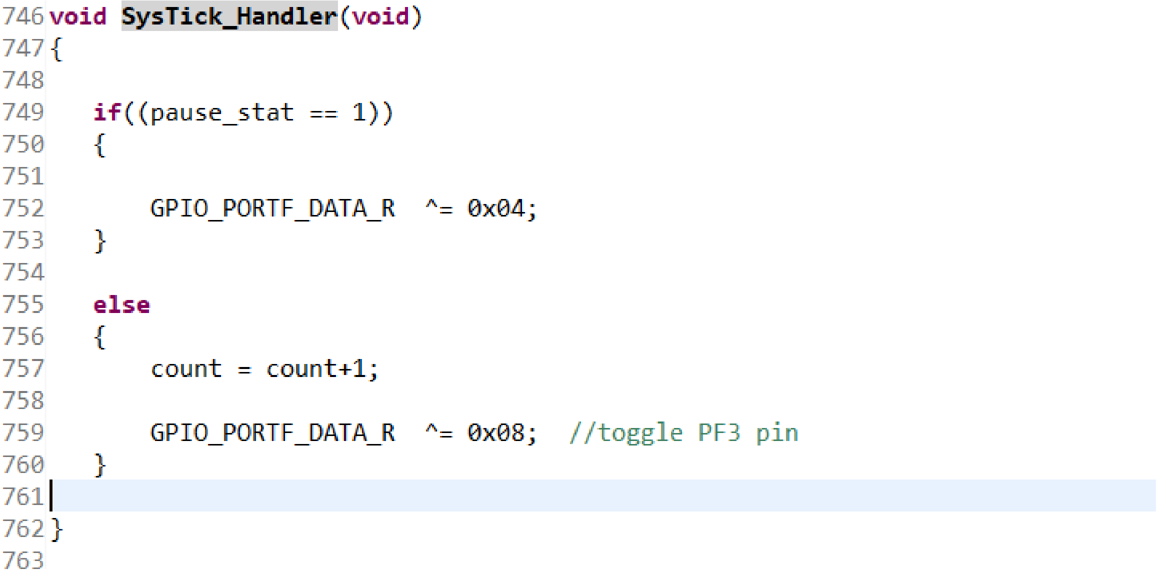
→Lab 6: *Implementing Stop watch Timer*

1. Setting up the SSD for Stop-Watch.

We use the NVIC SysTick Handler to manage the timer setting in the Seven segment display



```
NVIC_ST_RELOAD_R = 1000000-1;
NVIC_ST_CTRL_R |= 7 ; // enable counter, interrupt and select system bus clock
NVIC_ST_CURRENT_R  = 0;
```

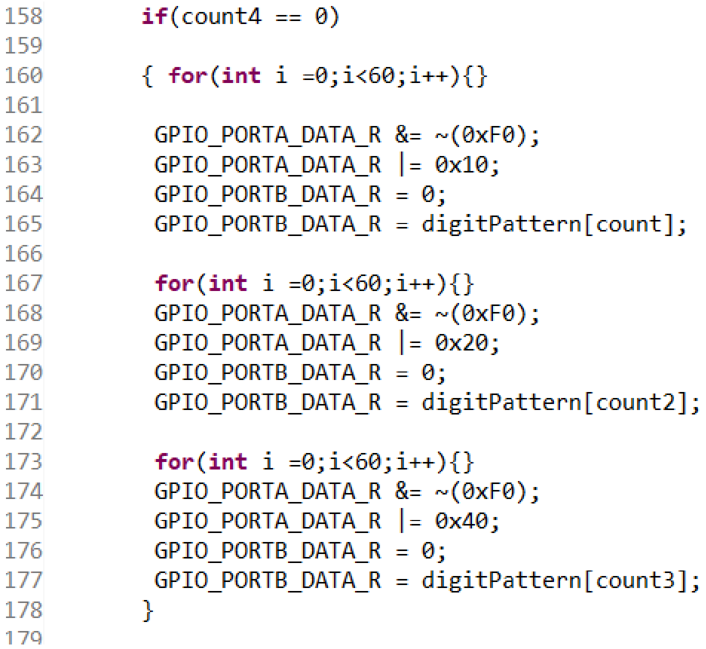


```
void SysTick_Handler(void)
{

    if((pause_stat == 1))
    {

        GPIO_PORTF_DATA_R  ^= 0x04;
    }

    else
    {
        count = count+1;

        GPIO_PORTF_DATA_R  ^= 0x08;  //toggle PF3 pin
    }

}
```

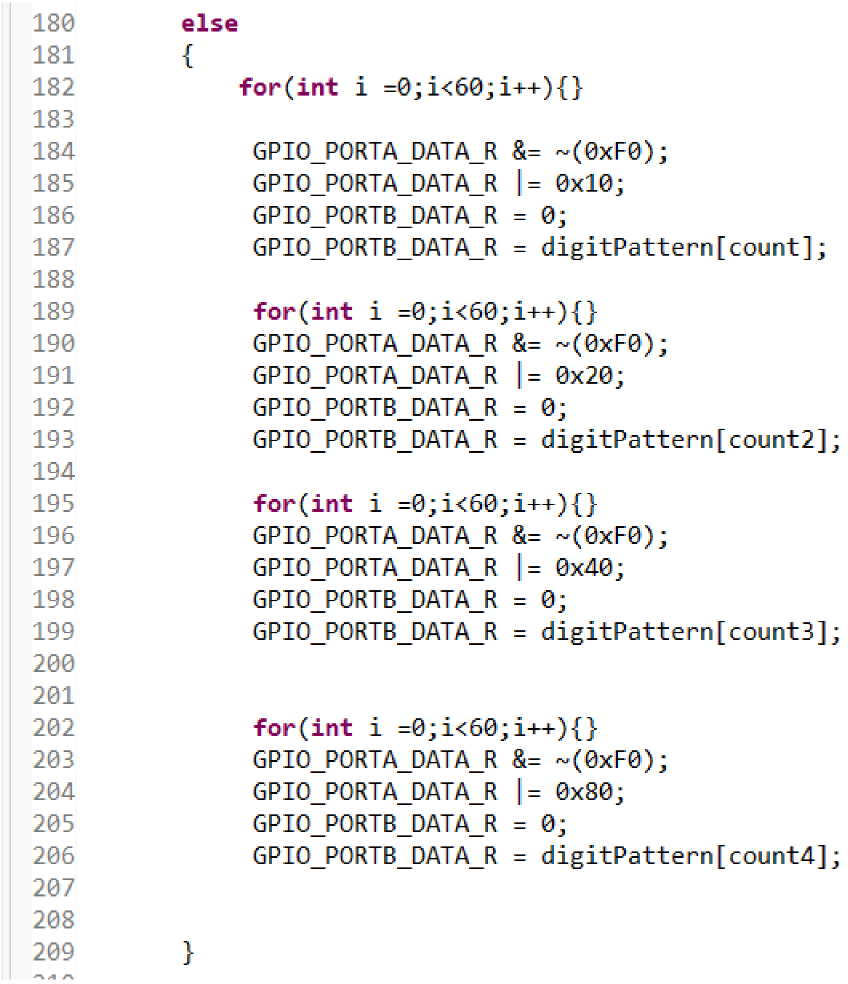
Here the Handler toggles in default and the pause state conditions are checked for when the flag is raised.

Since the 0 pattern in the 4th SSD is interfering with the LCD cmnd mode operation its not printing for seconds with 3rd digit in 0 value and it starts showing when count becomes 100 ms onwards

Meenakshi Shankar
Sr No. 22400
M.Tech EPD



```
if(count4 == 0)

{ for(int i =0;i<60;i++){}

 GPIO_PORTA_DATA_R &= ~(0xF0);
 GPIO_PORTA_DATA_R |= 0x10;
 GPIO_PORTB_DATA_R = 0;
 GPIO_PORTB_DATA_R = digitPattern[count];

 for(int i =0;i<60;i++){}
 GPIO_PORTA_DATA_R &= ~(0xF0);
 GPIO_PORTA_DATA_R |= 0x20;
 GPIO_PORTB_DATA_R = 0;
 GPIO_PORTB_DATA_R = digitPattern[count2];

 for(int i =0;i<60;i++){}
 GPIO_PORTA_DATA_R &= ~(0xF0);
 GPIO_PORTA_DATA_R |= 0x40;
 GPIO_PORTB_DATA_R = 0;
 GPIO_PORTB_DATA_R = digitPattern[count3];
}

```



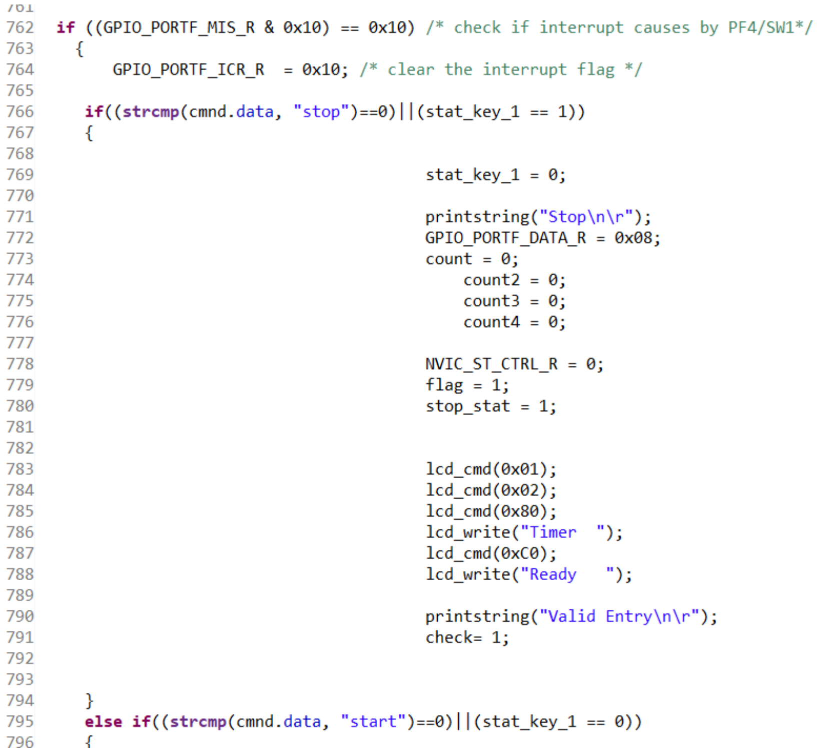
```
else
{
    for(int i =0;i<60;i++){}

     GPIO_PORTA_DATA_R &= ~(0xF0);
     GPIO_PORTA_DATA_R |= 0x10;
     GPIO_PORTB_DATA_R = 0;
     GPIO_PORTB_DATA_R = digitPattern[count];

     for(int i =0;i<60;i++){}
     GPIO_PORTA_DATA_R &= ~(0xF0);
     GPIO_PORTA_DATA_R |= 0x20;
     GPIO_PORTB_DATA_R = 0;
     GPIO_PORTB_DATA_R = digitPattern[count2];

     for(int i =0;i<60;i++){}
     GPIO_PORTA_DATA_R &= ~(0xF0);
     GPIO_PORTA_DATA_R |= 0x40;
     GPIO_PORTB_DATA_R = 0;
     GPIO_PORTB_DATA_R = digitPattern[count3];


     for(int i =0;i<60;i++){}
     GPIO_PORTA_DATA_R &= ~(0xF0);
     GPIO_PORTA_DATA_R |= 0x80;
     GPIO_PORTB_DATA_R = 0;
     GPIO_PORTB_DATA_R = digitPattern[count4];


}
```

Meenakshi Shankar
Sr No. 22400
M.Tech EPD

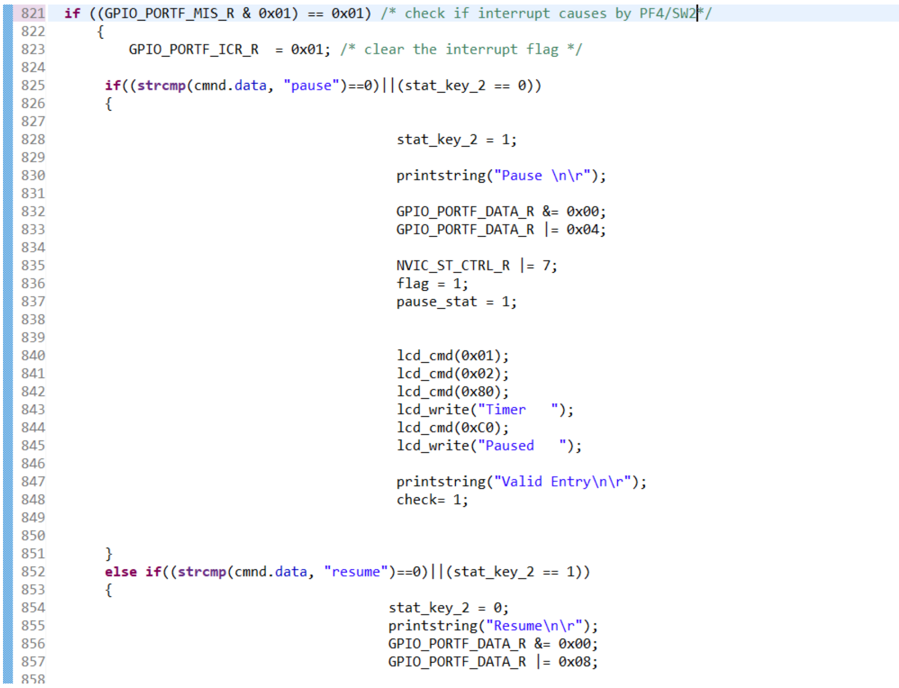
2.Push Buttons are set as follows and works based on:

SW1: Will be used to start and stop.

SW2: Will be used to pause and resume.



```
if ((GPIO_PORTF_MIS_R & 0x10) == 0x10) /* check if interrupt causes by PF4/SW1*/
  {
      GPIO_PORTF_ICR_R  = 0x10; /* clear the interrupt flag */

   if((strcmp(cmnd.data, "stop")==0)||(stat_key_1 == 1))
   {

                                    stat_key_1 = 0;

                                    printstring("Stop\n\r");
                                    GPIO_PORTF_DATA_R = 0x08;
                                    count = 0;
                                        count2 = 0;
                                        count3 = 0;
                                        count4 = 0;

                                    NVIC_ST_CTRL_R = 0;
                                    flag = 1;
                                    stop_stat = 1;


                                    lcd_cmd(0x01);
                                    lcd_cmd(0x02);
                                    lcd_cmd(0x80);
                                    lcd_write("Timer  ");
                                    lcd_cmd(0xC0);
                                    lcd_write("Ready   ");

                                    printstring("Valid Entry\n\r");
                                    check= 1;


   }
   else if((strcmp(cmnd.data, "start")==0)||(stat_key_1 == 0))
   {
```



```
if ((GPIO_PORTF_MIS_R & 0x01) == 0x01) /* check if interrupt causes by PF4/SW2*/
    {
        GPIO_PORTF_ICR_R  = 0x01; /* clear the interrupt flag */

     if((strcmp(cmnd.data, "pause")==0)||(stat_key_2 == 0))
     {

                                    stat_key_2 = 1;

                                    printstring("Pause \n\r");

                                    GPIO_PORTF_DATA_R &= 0x00;
                                    GPIO_PORTF_DATA_R |= 0x04;

                                    NVIC_ST_CTRL_R |= 7;
                                    flag = 1;
                                    pause_stat = 1;


                                    lcd_cmd(0x01);
                                    lcd_cmd(0x02);
                                    lcd_cmd(0x80);
                                    lcd_write("Timer   ");
                                    lcd_cmd(0xC0);
                                    lcd_write("Paused   ");

                                    printstring("Valid Entry\n\r");
                                    check= 1;


     }
     else if((strcmp(cmnd.data, "resume")==0)||(stat_key_2 == 1))
     {
                                   stat_key_2 = 0;
                                   printstring("Resume\n\r");
                                   GPIO_PORTF_DATA_R &= 0x00;
                                   GPIO_PORTF_DATA_R |= 0x08;
```

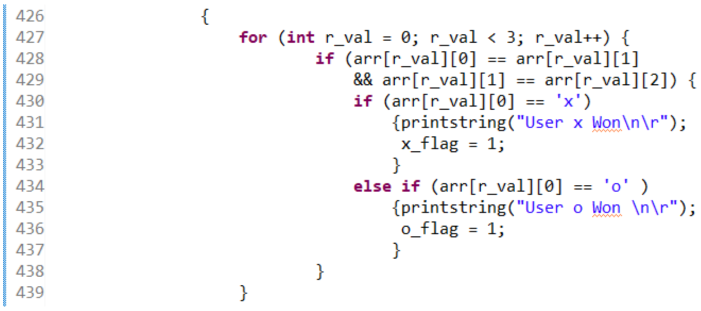
Meenakshi Shankar
Sr No. 22400
M.Tech EPD

Both SW1 and SW2 are configured to falling edge interrupts and the same is checked in the port F handler as above.

→Lab 7: *Tic-Tac-Toe*

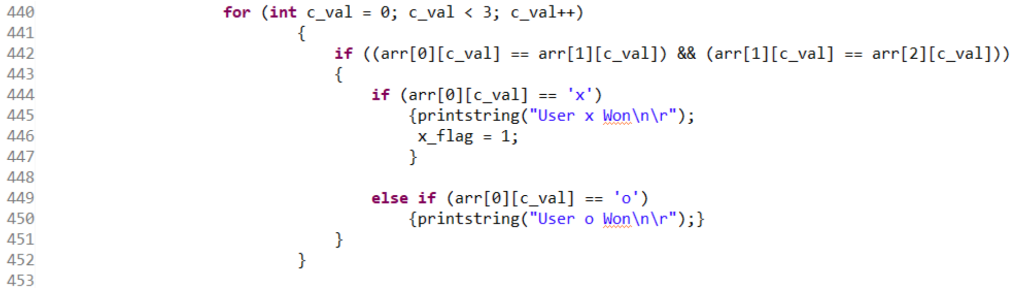
1. Logic:

Detecting a Row Strike.



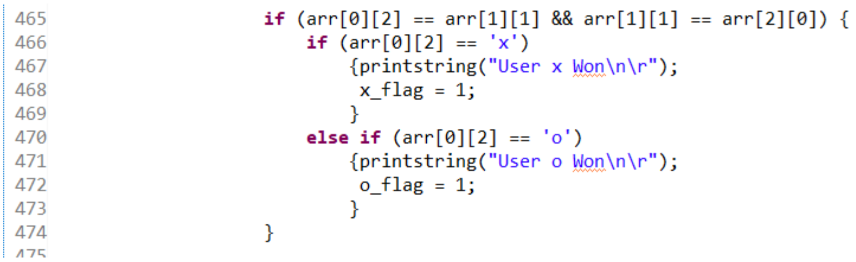
```
{
    for (int r_val = 0; r_val < 3; r_val++) {
            if (arr[r_val][0] == arr[r_val][1]
                && arr[r_val][1] == arr[r_val][2]) {
                if (arr[r_val][0] == 'x')
                    {printstring("User x Won\n\r");
                     x_flag = 1;
                    }
                else if (arr[r_val][0] == 'o' )
                    {printstring("User o Won \n\r");
                     o_flag = 1;
                    }
            }
    }
```

Detecting a Column Strike.



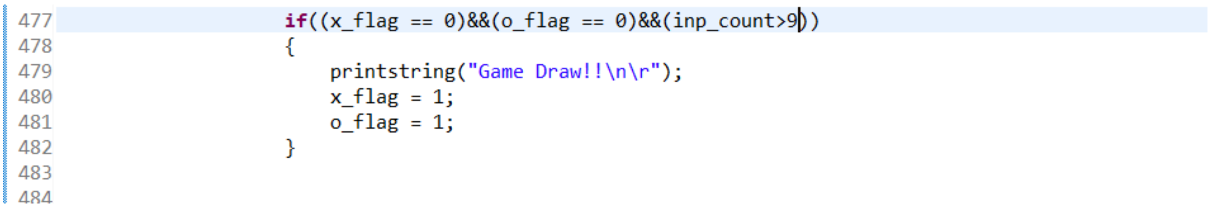
```
for (int c_val = 0; c_val < 3; c_val++)
        {
            if ((arr[0][c_val] == arr[1][c_val]) && (arr[1][c_val] == arr[2][c_val]))
            {
                if (arr[0][c_val] == 'x')
                    {printstring("User x Won\n\r");
                     x_flag = 1;
                    }

                else if (arr[0][c_val] == 'o')
                    {printstring("User o Won\n\r");}
            }
        }
```

Detecting Diagonal Strike.



```
if (arr[0][2] == arr[1][1] && arr[1][1] == arr[2][0]) {
   if (arr[0][2] == 'x')
      {printstring("User x Won\n\r");
       x_flag = 1;
      }
   else if (arr[0][2] == 'o')
      {printstring("User o Won\n\r");
       o_flag = 1;
      }
}
```

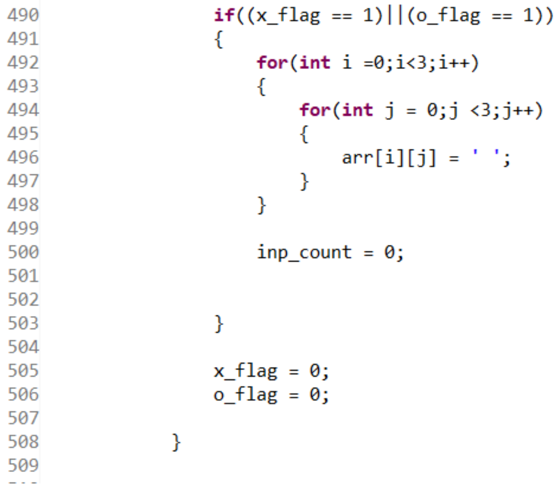
Detecting a Draw Game.



```
if((x_flag == 0)&&(o_flag == 0)&&(inp_count>9))
{
    printstring("Game Draw!!\n\r");
    x_flag = 1;
    o_flag = 1;
}
```

Meenakshi Shankar
Sr No. 22400
M.Tech EPD

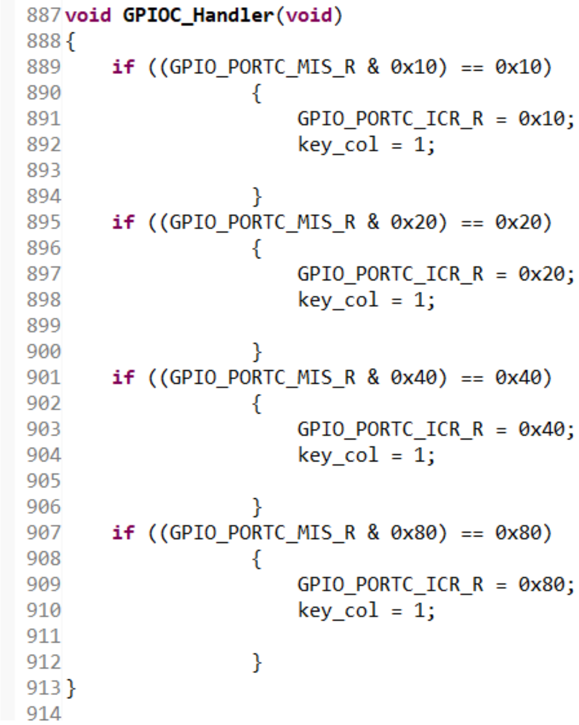
Clearing a Game after result:



```
        if((x_flag == 1)||(o_flag == 1))
        {
            for(int i =0;i<3;i++)
            {
                for(int j = 0;j <3;j++)
                {
                    arr[i][j] = ' ';
                }
            }

            inp_count = 0;


        }

        x_flag = 0;
        o_flag = 0;

    }

```

2. Mapping and Recognizing Key Presses.

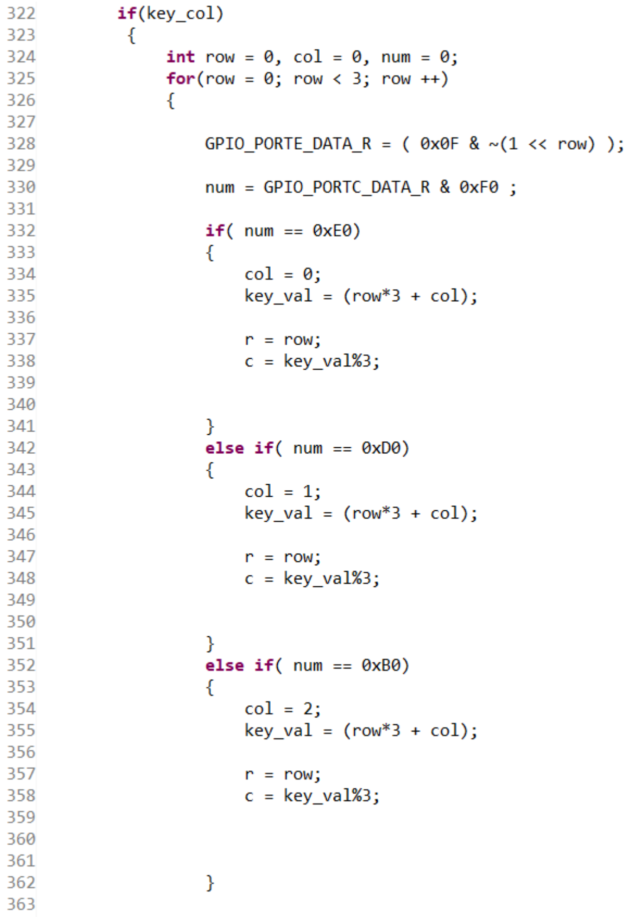
➔ Implementation of col recognition using interrupts:



```
void GPIOC_Handler(void)
{
    if ((GPIO_PORTC_MIS_R & 0x10) == 0x10)
                {
                    GPIO_PORTC_ICR_R = 0x10;
                    key_col = 1;

                }
    if ((GPIO_PORTC_MIS_R & 0x20) == 0x20)
                {
                    GPIO_PORTC_ICR_R = 0x20;
                    key_col = 1;

                }
    if ((GPIO_PORTC_MIS_R & 0x40) == 0x40)
                {
                    GPIO_PORTC_ICR_R = 0x40;
                    key_col = 1;

                }
    if ((GPIO_PORTC_MIS_R & 0x80) == 0x80)
                {
                    GPIO_PORTC_ICR_R = 0x80;
                    key_col = 1;

                }
}

```

Meenakshi Shankar
Sr No. 22400
M.Tech EPD

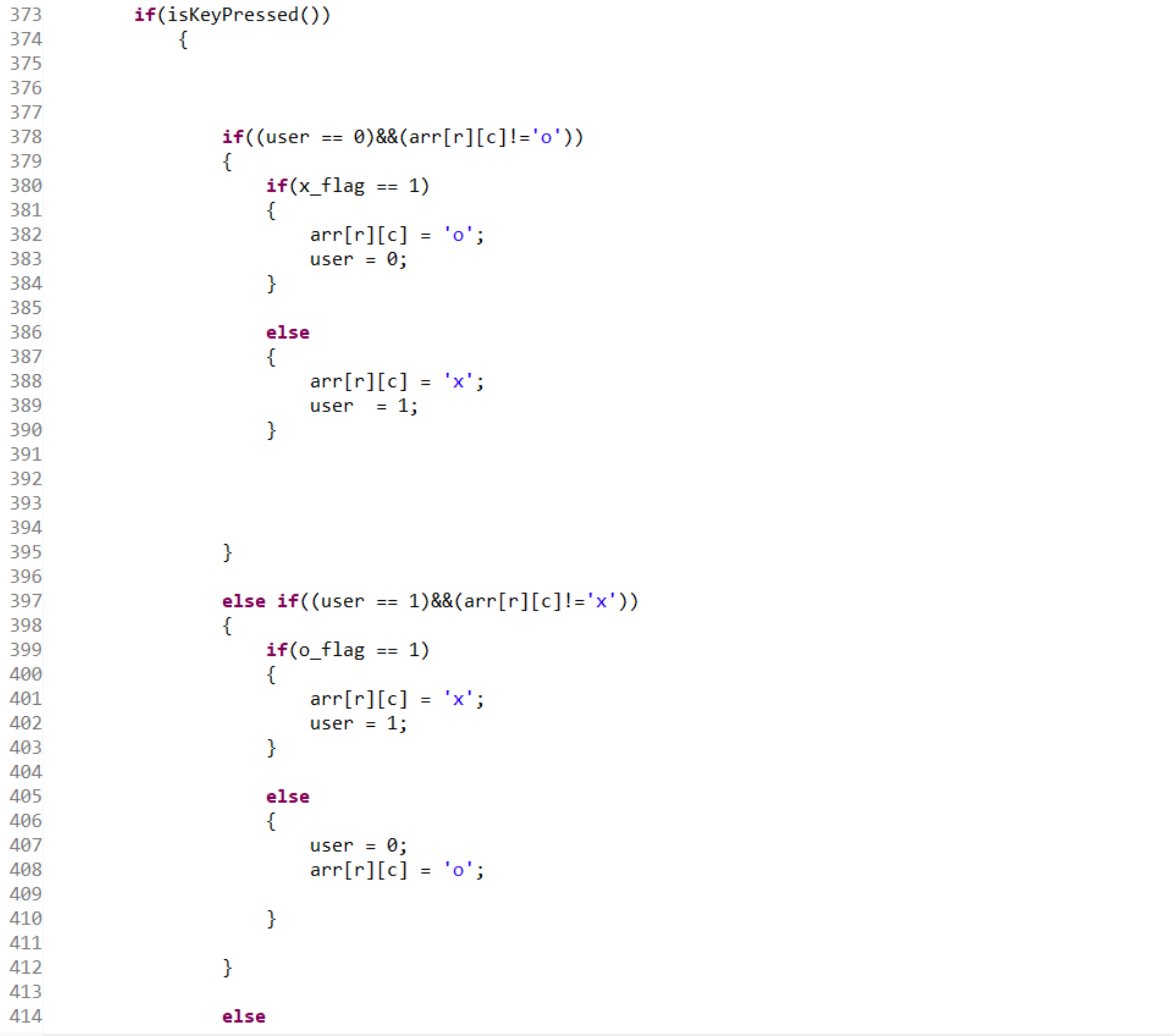
➔ Recognition of the row pressed:



```
if(key_col)
 {
     int row = 0, col = 0, num = 0;
     for(row = 0; row < 3; row ++)
     {

         GPIO_PORTE_DATA_R = ( 0x0F & ~(1 << row) );

         num = GPIO_PORTC_DATA_R & 0xF0 ;

         if( num == 0xE0)
         {
             col = 0;
             key_val = (row*3 + col);

             r = row;
             c = key_val%3;


         }
         else if( num == 0xD0)
         {
             col = 1;
             key_val = (row*3 + col);

             r = row;
             c = key_val%3;


         }
         else if( num == 0xB0)
         {
             col = 2;
             key_val = (row*3 + col);

             r = row;
             c = key_val%3;



         }

```

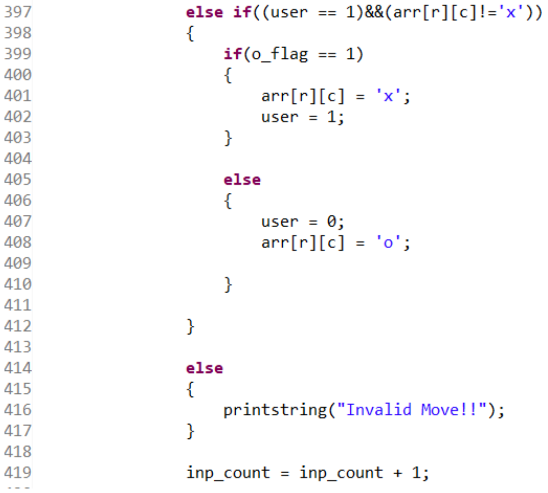
Recognizing and storing the x or o intake to the particular position in arr[][] as mapped from the keypad button position(set based on the winner of previous game. On setup it is X)

Meenakshi Shankar
Sr No. 22400
M.Tech EPD



```
if(isKeyPressed())
    {


        if((user == 0)&&(arr[r][c]!='o'))
        {
            if(x_flag == 1)
            {
                arr[r][c] = 'o';
                user = 0;
            }

            else
            {
                arr[r][c] = 'x';
                user  = 1;
            }




        }

        else if((user == 1)&&(arr[r][c]!='x'))
        {
            if(o_flag == 1)
            {
                arr[r][c] = 'x';
                user = 1;
            }

            else
            {
                user = 0;
                arr[r][c] = 'o';

            }

        }

        else
```

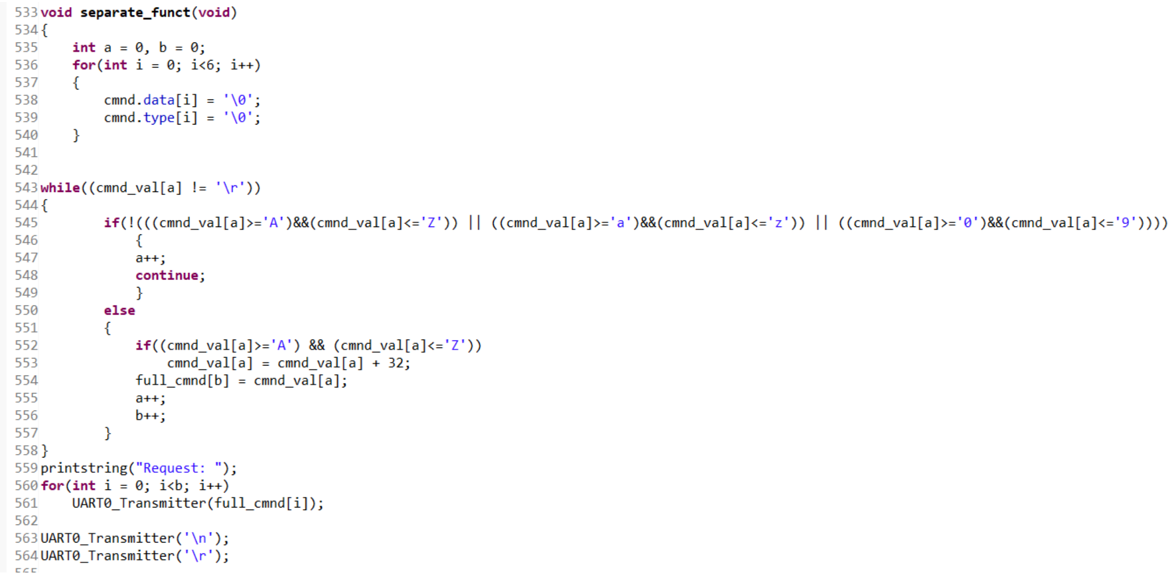
Pressing of already pressed position in the 3x3 matrix is not allowed and is realised as “Invalid Move”



```
else if((user == 1)&&(arr[r][c]!='x'))
{
    if(o_flag == 1)
    {
        arr[r][c] = 'x';
        user = 1;
    }

    else
    {
        user = 0;
        arr[r][c] = 'o';

    }

}

else
{
    printstring("Invalid Move!!");
}

inp_count = inp_count + 1;
```

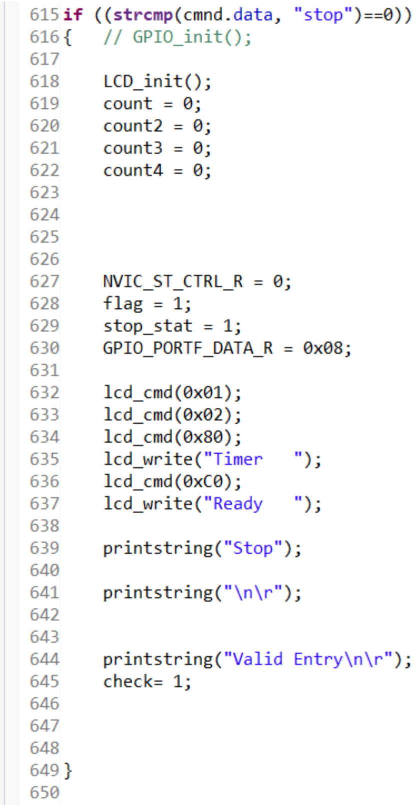
Meenakshi Shankar
Sr No. 22400
M.Tech EPD

3. UART console commands are also accepted. The commands are parsed as below in the separate_funct()



```c
void separate_funct(void)
{
    int a = 0, b = 0;
    for(int i = 0; i<6; i++)
    {
        cmnd.data[i] = '\0';
        cmnd.type[i] = '\0';
    }


while((cmnd_val[a] != '\r'))
{
        if(!(((cmnd_val[a]>='A')&&(cmnd_val[a]<='Z')) || ((cmnd_val[a]>='a')&&(cmnd_val[a]<='z')) || ((cmnd_val[a]>='0')&&(cmnd_val[a]<='9'))))
            {
            a++;
            continue;
            }
        else
        {
            if((cmnd_val[a]>='A') && (cmnd_val[a]<='Z'))
                cmnd_val[a] = cmnd_val[a] + 32;
            full_cmnd[b] = cmnd_val[a];
            a++;
            b++;
        }
}
printstring("Request: ");
for(int i = 0; i<b; i++)
    UART0_Transmitter(full_cmnd[i]);

UART0_Transmitter('\n');
UART0_Transmitter('\r');
```

And different operation conditions are mapped to the received state as below:



```c
if ((strcmp(cmnd.data, "stop")==0))
{   // GPIO_init();

    LCD_init();
    count = 0;
    count2 = 0;
    count3 = 0;
    count4 = 0;




    NVIC_ST_CTRL_R = 0;
    flag = 1;
    stop_stat = 1;
    GPIO_PORTF_DATA_R = 0x08;

    lcd_cmd(0x01);
    lcd_cmd(0x02);
    lcd_cmd(0x80);
    lcd_write("Timer    ");
    lcd_cmd(0xC0);
    lcd_write("Ready    ");

    printstring("Stop");

    printstring("\n\r");


    printstring("Valid Entry\n\r");
    check= 1;




}

```