Universidad Autónoma de Coahuila

FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA

Filtros Digitales
Proyecto Tres en Raya

Estudiantes:

Brandon Salas Huerta

Erick Isaí Garza Zamora

Iván Uziel Dávila Domínguez

Profesor:

Jorge Adrián Osuna Gonzales

25/Abril/2022

Tres en Raya

La función principal de este proyecto es principalmente simular la jugabilidad del minijuego comúnmente conocido como “el gato”, y esto se llevará a cabo con una representación mediante el uso de LEDs, displays y un botón haciendo uso de todos los métodos anteriormente vistos en las prácticas ya trabajadas.

MATERIALES:

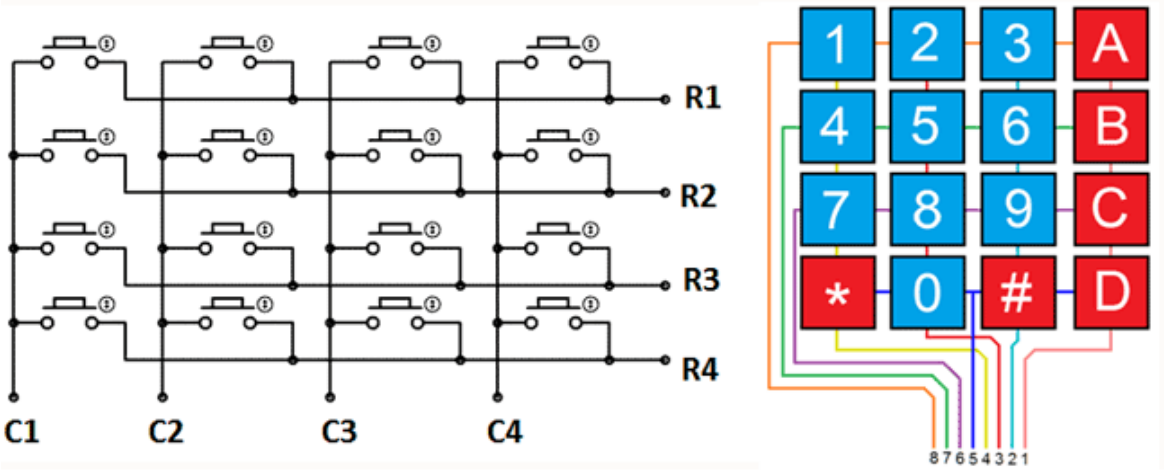
- Interruptor 3 pasos ON/OFF/ON (1)
- Resistencias 330Ω (22)
- Resistencias 10KΩ (5)
- Diodo rectificador 1N4007 (3)
- LED amarillo (9)
- LED azul (9)
- Display 7 segmentos cátodo común (2)
- DsPIC30F4013 (1)
- Decodificador 74LS47 (2)
- Teclado de membrana (1)
- Protoboard (2)
- Placa con pistas para soldar (1)
- Cable 3mts
- Cautín
- Estaño

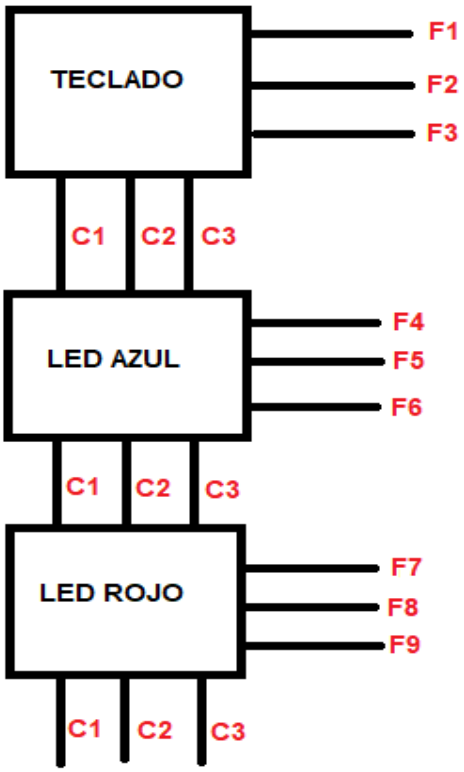
JUGABILIDAD:

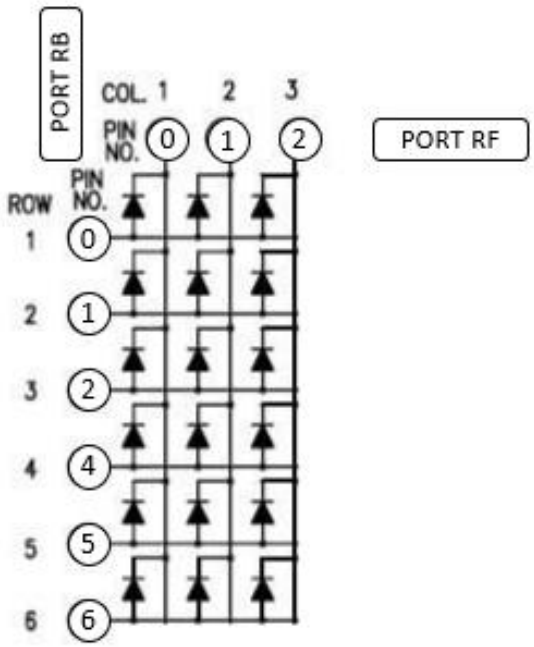
Básicamente el modo de juego del proyecto consiste en utilizar el teclado de membrana con una matriz de 3x3 que será el campo de juego, así iremos marcando los turnos respectivos del jugador 1 y del jugador 2 y esto mediante una representación que se visualizará en un panel de LEDs que se distingue por dos colores siendo así uno para cada jugador (azul y amarillo) los cuales estarán coordinados de acuerdo al botón del teclado de membrana ubicado en la respectiva fila-columna que se haya presionado por los jugadores, de modo

que al completar las 3 marcas en raya se obtenga la victoria del respectivo jugador que logró formar el conjunto de marcas que pueden cumplir las condiciones de victoria (diagonal, horizontal, vertical), al ganar un jugador, tendremos 2 displays (contadores) para diferenciar las victorias de cada jugador respectivamente, los cuales se irán acumulando a un máximo de 10 victorias y al lograrse dicha cantidad se volvería a reiniciar desde 0 ambos contadores, de ser que no se quiera llegar a las 10 victorias para esto haremos uso del interruptor de 3 pasos, pues, con este mismo podremos reiniciar tanto el panel de LEDs como los números en los contadores de victoria de cada jugador accionando hacia arriba para reiniciar los contadores y hacia abajo para reiniciar el panel de LEDs respectivamente, en caso contrario, al no lograr cumplir ninguna condición de victoria por las distintas marcas en los LEDs debido a la colocación aleatoria de estas mismas, el juego indicará que nadie ganó y procederá a reiniciarse para proseguir nuevamente con un juego desde 0.

DIAGRAMA DE CONEXIONES:







CÓDIGO CON EXPLICACIÓN:

Primera Parte – Declaraciones y configuraciones de pines y puertos del micro

```
/*
 * File:   Gato_.c
 * Author: jsala
 *
 * Created on 12 de abril de 2022, 04:12 PM
 */


// DSPIC30F4013 Configuration Bit Settings

// 'C' source line config statements

// FOSC
#pragma config FOSFPR = FRC             // Oscillator (Internal Fast RC (No change to Primary Osc Mode bits))
#pragma config FCKSMEN = CSW_FSCM_OFF   // Clock Switching and Monitor (Sw Disabled, Mon Disabled)

// FWDT
#pragma config FWPSB = WDTPSB_1        // WDT Prescaler B (1:16)
#pragma config FWPSA = WDTPSA_1       // WDT Prescaler A (1:512)
#pragma config WDT = WDT_OFF            // Watchdog Timer (Disabled)

// FBORPOR
#pragma config FPWRT = PWRT_OFF          // POR Timer Value (64ms)
#pragma config BODENV = BORV20          // Brown Out Voltage (Reserved)
#pragma config BOREN = PBOR_ON          // PBOR Enable (Enabled)
#pragma config MCLRE = MCLR_EN          // Master Clear Enable (Enabled)

// FGS
#pragma config GWRP = GWRP_OFF          // General Code Segment Write Protect (Disabled)
#pragma config GCP = CODE_PROT_OFF      // General Segment Code Protection (Disabled)

// FICD
```

```

// FICD
#pragma config ICS = ICS_PGD            // Comm Channel Select (Use PGC/EMUC and PGD/EMUD)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.


#include <xc.h>

int main(void) {


    OSCCONbits.POST = 0;
    T1CONbits.TCS = 0; //Ponmos este en 0 para decir que usaremos el clock interno
    T1CONbits.TGATE = 0; //Ponmos este en ,este es para sincronizar los pulsos externos con los intrnos,no lo usaremos por el momento
    T1CONbits.TCKPS = 3; //Es el prescalador le decimos que queremos la opcion 4 (256)  o podemos poner en binario 0b11
    T1CONbits.TSYNC = 1; //Le ponemos 0 , no nos interesa sincronizarlo,ya que estamos usando el osilador interno del nucleo
    PR1 = 10000; // Un pequeño delay para captar mejor las pulsaciones
    T1CONbits.TON = 1; //Activamos el timer


    ADPCFGbits.PCFG0 = 1;
    ADPCFGbits.PCFG1 = 1;
    ADPCFGbits.PCFG2 = 1;
    ADPCFGbits.PCFG3 = 1; //Al ponerlos en 1 le decimos que lo usaremos como DIGITAL
    ADPCFGbits.PCFG4 = 1;
    ADPCFGbits.PCFG5 = 1;
    ADPCFGbits.PCFG6 = 1;
    ADPCFGbits.PCFG7 = 1;
    ADPCFGbits.PCFG8 = 1;
```

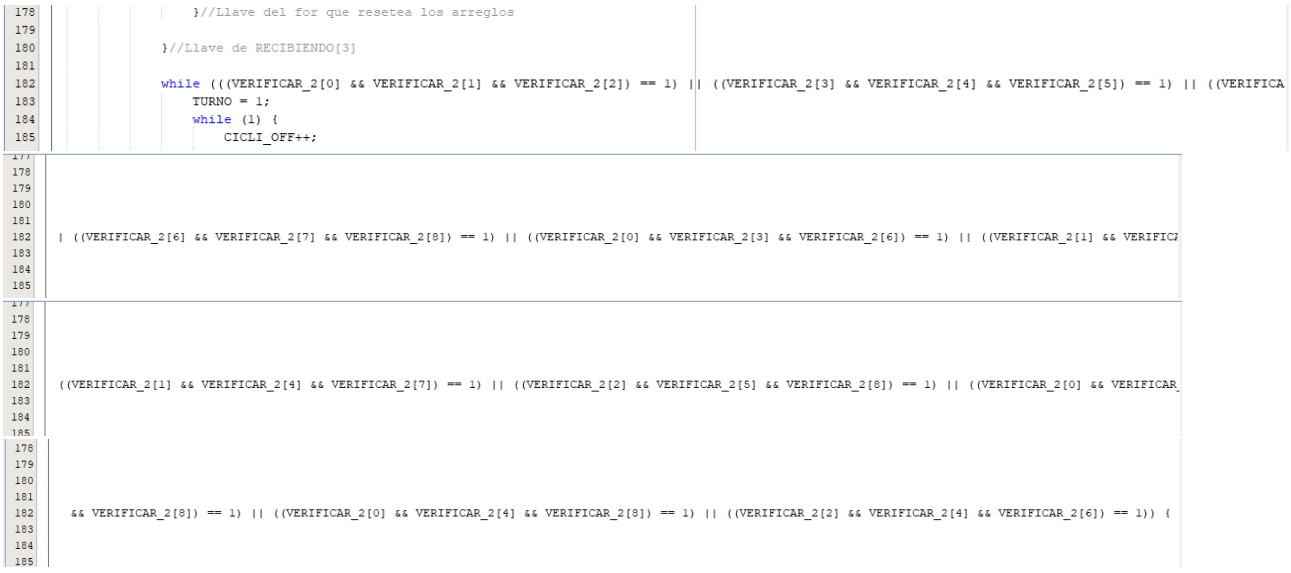
```
    ADPCFGbits.PCFG8 = 1;
    ADPCFGbits.PCFG9 = 1;
    ADPCFGbits.PCFG10 = 1;
    ADPCFGbits.PCFG11 = 1;
    ADPCFGbits.PCFG12 = 1;

    C1CTRLbits.REQOP = 1; //Desabilitamos el comparador de RF's 1 y 2
    U2MODEbits.UARTEN = 1; //Desabilitamos el modulo UART de RF's 3 4 5
    U1MODEbits.UARTEN = 1;



    TRISBbits.TRISB0 = 0;
    TRISBbits.TRISB1 = 0;
    TRISBbits.TRISB2 = 0; //Configuramos desde RB0 hasta RB9 como salidas
    TRISBbits.TRISB3 = 0; //los usaremos para prender el display
    TRISBbits.TRISB4 = 0;
    TRISBbits.TRISB5 = 0;
    TRISBbits.TRISB6 = 0;
    TRISBbits.TRISB7 = 0;
    TRISBbits.TRISB8 = 0;
    TRISBbits.TRISB9 = 0;
    TRISBbits.TRISB10 = 0;
    TRISBbits.TRISB11 = 0;
    TRISBbits.TRISB12 = 0;



    TRISFbits.TRISF0 = 0;
    TRISFbits.TRISF1 = 0; // Lo configuramos como salida para darles 1 logico
    TRISFbits.TRISF3 = 0; // y asi controlar las columnas
```

Parte 2 – Decalracion de Arreglos que guardaran los datos de nuestro juego ,mostraran el display y detectaran las señales de entrada

```
    TRISFbits.TRISF3 = 0; // y asi controlar las columnas
    TRISFbits.TRISF5 = 0;



    TRISDbits.TRISD0 = 1;
    TRISDbits.TRISD1 = 1; //configuramos desde el RD0 al RD4
    TRISDbits.TRISD2 = 1; //como entrada de la botonera
    TRISDbits.TRISD3 = 1; //

    TRISCbits.TRISC13 = 0;
    TRISCbits.TRISC14 = 0;
    TRISAbits.TRISA11 = 0;



    unsigned short RECIBIENDO[] = {
        0xE, // Recibimos un cero 1110 en el puerto RD0
        0xD, // Recibimos un cero 1101 en el puerto RD1
        0xB, // Recibimos un cero 1011 en el puerto RD2
        0x7 //  Recibimos un cero 0111 en el puerto RD3
    };

    unsigned COLUMNAS[] = {
        0x00FE,
        0x00FD,
        0x00F7
    };
    unsigned DISPLAY_1[] = {//usamos 0000 0**0 0000 0000 del puerto B para el display
        0x0000, //Cero
        0x0200, //Uno
```

```
        0x0200, //Uno
        0x0400, //Dos
        0x0600, //Tres
        0x0008, //Cuatro
        0x0208, //Cinco
        0x0408, //Seis
        0x0608, //Siete
    };
    unsigned DISPLAY_2[] = {//usamos 000* *000 0000 0000 del puerto B para el display
        0x0000, //Cero
        0x0800, //Uno
        0x1000, //Dos
        0x1800, //Tres
        0x0080, //Cuatro
        0x0880, //Cinco
        0x1080, //Seis
        0x1880 //Siete
    };
    unsigned IMPRIMIR[] = {//aqui guardamos los numeros para prender los leds
        0, 0, 0, 0, 0, 0, 0, 0

    };
    int TURNO = 1;
    int x = 0;
    int y = 0;
    int DELAY = 200;
    int DELAY_2 = 19000;
    int GANAR_1 = 0;
    int GANAR_2 = 0;
    int CICLI_OFF = 0;
    unsigned short VERIFICAR_1[] = {0, 0, 0, 0, 0, 0, 0, 0, 0 //Se encarga de verificar cuales leds estan encendidos AMARILLOS
```

```
    unsigned short VERIFICAR_1[] = {0, 0, 0, 0, 0, 0, 0, 0, 0 //Se encarga de verificar cuales leds estan encendidos AMARILLOS
    };
    unsigned short VERIFICAR_2[] = {0, 0, 0, 0, 0, 0, 0, 0, 0 //Se encarga de verificar cuales leds estan encendidos AZULES
    };

    /*
    DELAY - Lo ponemos en cada for que checa que boton fue presionado,esto evita que algun led de la misma
    fila parezca que esta un poco encendido

    ¿Por que sumamos lo IMPRIMIR? - Esto por que el led 1 es 0001,el led 2 0010 y el led 3 0100
    si queremos encender 1 y 2 = 0001 + 0010 = 0011 o el 3
    *
    o encender todos 1 2 y 3 leds = 0001 + 0010 + 0100 = 0111 o el 7
    */

    while (1) {


        if (TURNO == 1) { //siempre estaremos alternando entre los turnos
            CICLI_OFF = 0;
            if (PORTD == RECIBIENDO[3]) { // El 4to boton usado para resetear los leds
                for (x = 0; x <= 9; x++) {
                    TURNO = 1;
                    IMPRIMIR[x] = 0x0000; //recorre todo el arreglo poniendolo en 0 y apagando los leds
                    VERIFICAR_1[x] = 0; //Ponemos en 0 el arreglo indicando que todos los LEDs AMARILLOS estan apagados
                    VERIFICAR_2[x] = 0; //Ponemos en 0 el arreglo indicando que todos los LEDs AZULES estan apagados

                }//Llave del for que resetea los arreglos

            }//Llave de RECIBIENDO[3]
```



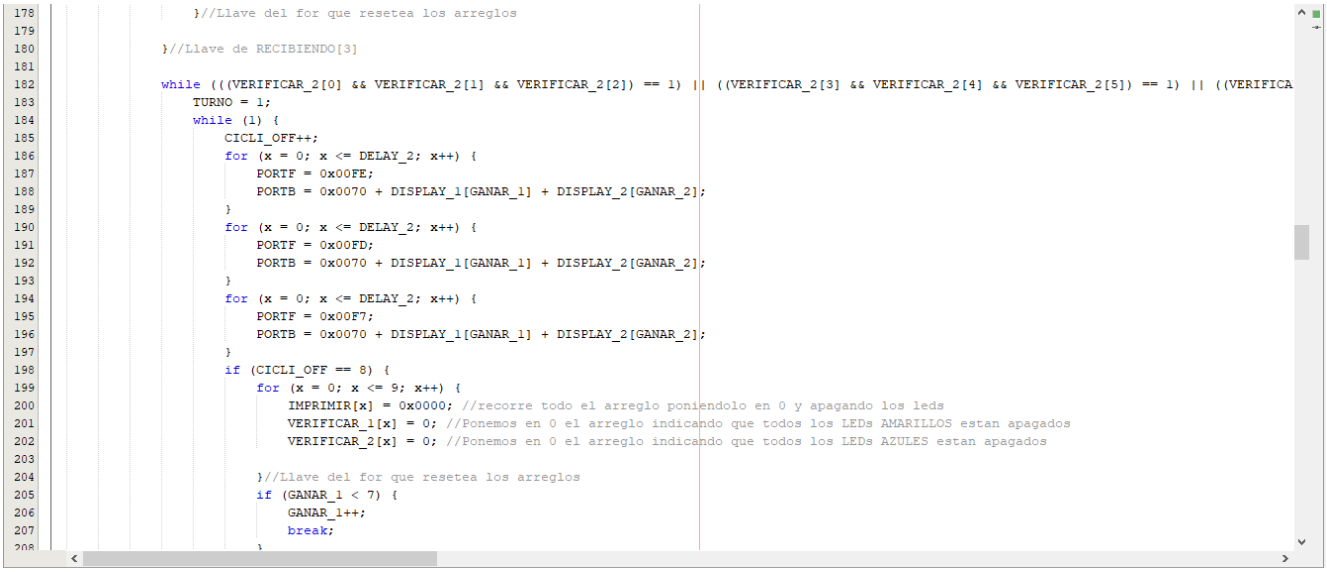
```
178                 }//Llave del for que resetea los arreglos
179
180             }//Llave de RECIBIENDO[3]
181
182             while (((VERIFICAR_2[0] && VERIFICAR_2[1] && VERIFICAR_2[2]) == 1) || ((VERIFICAR_2[3] && VERIFICAR_2[4] && VERIFICAR_2[5]) == 1) || ((VERIFICA
183                 TURNO = 1;
184                 while (1) {
185                     CICLI_OFF++;
```

```
178
179
180
181
182 | ((VERIFICAR_2[6] && VERIFICAR_2[7] && VERIFICAR_2[8]) == 1) || ((VERIFICAR_2[0] && VERIFICAR_2[3] && VERIFICAR_2[6]) == 1) || ((VERIFICAR_2[1] && VERIFICA
183
184
185
```

```
178
179
180
181
182 ((VERIFICAR_2[1] && VERIFICAR_2[4] && VERIFICAR_2[7]) == 1) || ((VERIFICAR_2[2] && VERIFICAR_2[5] && VERIFICAR_2[8]) == 1) || ((VERIFICAR_2[0] && VERIFICAR_
183
184
185
```

```
178
179
180
181
182  && VERIFICAR_2[8]) == 1) || ((VERIFICAR_2[0] && VERIFICAR_2[4] && VERIFICAR_2[8]) == 1) || ((VERIFICAR_2[2] && VERIFICAR_2[4] && VERIFICAR_2[6]) == 1)) {
183
184
185
```

(esto fue la extensión de la línea de código 182 y su condición puesto que no cabe en una sola captura de pantalla)



```
178                 }//Llave del for que resetea los arreglos
179
180             }//Llave de RECIBIENDO[3]
181
182             while (((VERIFICAR_2[0] && VERIFICAR_2[1] && VERIFICAR_2[2]) == 1) || ((VERIFICAR_2[3] && VERIFICAR_2[4] && VERIFICAR_2[5]) == 1) || ((VERIFICA
183                 TURNO = 1;
184                 while (1) {
185                     CICLI_OFF++;
186                     for (x = 0; x <= DELAY_2; x++) {
187                         PORTF = 0x00FE;
188                         PORTB = 0x0070 + DISPLAY_1[GANAR_1] + DISPLAY_2[GANAR_2];
189                     }
190                     for (x = 0; x <= DELAY_2; x++) {
191                         PORTF = 0x00FD;
192                         PORTB = 0x0070 + DISPLAY_1[GANAR_1] + DISPLAY_2[GANAR_2];
193                     }
194                     for (x = 0; x <= DELAY_2; x++) {
195                         PORTF = 0x00F7;
196                         PORTB = 0x0070 + DISPLAY_1[GANAR_1] + DISPLAY_2[GANAR_2];
197                     }
198                     if (CICLI_OFF == 8) {
199                         for (x = 0; x <= 9; x++) {
200                             IMPRIMIR[x] = 0x0000; //recorre todo el arreglo poniendolo en 0 y apagando los leds
201                             VERIFICAR_1[x] = 0; //Ponemos en 0 el arreglo indicando que todos los LEDs AMARILLOS estan apagados
202                             VERIFICAR_2[x] = 0; //Ponemos en 0 el arreglo indicando que todos los LEDs AZULES estan apagados
203
204                         }//Llave del for que resetea los arreglos
205                         if (GANAR_1 < 7) {
206                             GANAR_1++;
207                             break;
208                         }
```

```
                    GANAR_1++;
                    break;
                }

            }

        }

    }//WHILE que verifica si GANAMOS en los AZULES


    for (x = 0; x <= DELAY; x++) {
        PORTF = 0x00FE; //RF0 0000 0000 1111 1110  Desabilitamos las demas columnas para leer si presionamos algun boton de la columna 1
        PORTB = IMPRIMIR[0] + IMPRIMIR[1] + IMPRIMIR[2] + DISPLAY_1[GANAR_1] + DISPLAY_2[GANAR_2]; //imprimimos la suma de los botones presionados
        if (VERIFICAR_2[0] == 0) { //con este if IMPEDIMOS prender ambos LEDs de cada seccion
            if (PORTD == RECIBIENDO[0]) {
                IMPRIMIR[0] = 0x0001; // 0001
                VERIFICAR_1[0] = 1; //indicamos que el led Fila 1 Columna 1 esta ENCENDIDO
                TURNO = 2; //Con esto nos vamos a encender los LEDs AZULES
            }
        }

        if (VERIFICAR_2[1] == 0) {
            if (PORTD == RECIBIENDO[1]) {
                IMPRIMIR[1] = 0x0002; // 0010
                VERIFICAR_1[1] = 1; //indicamos que el led Fila 2 Columna 1 esta ENCENDIDO
                TURNO = 2;
                ;
            }
        }

```

```
        if (VERIFICAR_2[2] == 0) {
            if (PORTD == RECIBIENDO[2]) {
                IMPRIMIR[2] = 0x0004; // 0100
                VERIFICAR_1[2] = 1; //indicamos que el led Fila 3 Columna 1 esta ENCENDIDO
                TURNO = 2;
            }
        }
    } // Llave del ciclo 1 que verifica la columna 1 AMARILLOS

    for (x = 0; x <= DELAY; x++) {
        PORTF = 0x00FD; //RF1 000 0000 1111 1101
        PORTB = IMPRIMIR[3] + IMPRIMIR[4] + IMPRIMIR[5] + DISPLAY_1[GANAR_1] + DISPLAY_2[GANAR_2];
        if (VERIFICAR_2[3] == 0) {
            if (PORTD == RECIBIENDO[0]) {
                IMPRIMIR[3] = 0x0001;
                VERIFICAR_1[3] = 1;
                TURNO = 2;
            }
        }
        if (VERIFICAR_2[4] == 0) {
            if (PORTD == RECIBIENDO[1]) {
                IMPRIMIR[4] = 0x0002;
                VERIFICAR_1[4] = 1;
                TURNO = 2;
            }
        }
        if (VERIFICAR_2[5] == 0) {
            if (PORTD == RECIBIENDO[2]) {
                IMPRIMIR[5] = 0x0004;
                VERIFICAR_1[5] = 1;
```

```
                    VERIFICAR_1[5] = 1;
                    TURNO = 2;
                }
            }
        }// Llave del ciclo 1 que verifica la columna 2 AMARILLOS

        for (x = 0; x <= DELAY; x++) {
            PORTF = 0x00F7; //RF3 0000 0000 1111 0111
            PORTB = IMPRIMIR[6] + IMPRIMIR[7] + IMPRIMIR[8] + DISPLAY_1[GANAR_1] + DISPLAY_2[GANAR_2];
            if (VERIFICAR_2[6] == 0) {
                if (PORTD == RECIBIENDO[0]) {
                    IMPRIMIR[6] = 0x0001;
                    VERIFICAR_1[6] = 1;
                    TURNO = 2;
                }
            }
            if (VERIFICAR_2[7] == 0) {
                if (PORTD == RECIBIENDO[1]) {
                    IMPRIMIR[7] = 0x0002;
                    VERIFICAR_1[7] = 1;
                    TURNO = 2;
                }
            }
            if (VERIFICAR_2[8] == 0) {
                if (PORTD == RECIBIENDO[2]) {
                    IMPRIMIR[8] = 0x0004;
                    VERIFICAR_1[8] = 1;
                    TURNO = 2;
                }
            }
```

```
                }
            }// Llave del ciclo 1 que verifica la columna 3 AMARILLOS

        }// termina TURNO 1------------------------------------------------------------
        if (TURNO == 2) {

            CICLI_OFF = 0;
            if (PORTD == RECIBIENDO[3]) {
                for (x = 0; x <= 9; x++) {
                    TURNO = 1;
                    IMPRIMIR[x] = 0x0000; //recorre todo el arreglo poniendolo en 0 y apagando los leds
                    VERIFICAR_1[x] = 0; //Ponemos en 0 el arreglo indicando que todos los LEDs AMARILLOS estan apagados
                    VERIFICAR_2[x] = 0; //Ponemos en 0 el arreglo indicando que todos los LEDs AZULES estan apagados

                }//Llave del for que resetea los arreglos
            }

            while (((VERIFICAR_1[0] && VERIFICAR_1[1] && VERIFICAR_1[2]) == 1) || ((VERIFICAR_1[3] && VERIFICAR_1[4] && VERIFICAR_1[5]) == 1) || ((VERIFICA
                TURNO = 1;
                while (1) {
                    CICLI_OFF++;
                    for (x = 0; x <= DELAY_2; x++) {
                        PORTF = 0x00FE;
                        PORTB = 0x0007 + DISPLAY_1[GANAR_1] + DISPLAY_2[GANAR_2];
                    }
                    for (x = 0; x <= DELAY_2; x++) {
                        PORTF = 0x00FD;
                        PORTB = 0x0007 + DISPLAY_1[GANAR_1] + DISPLAY_2[GANAR_2];
                    }
                    for (x = 0; x <= DELAY_2; x++) {
```

```
&& VERIFICAR_1[5]) == 1) || ((VERIFICAR_1[6] && VERIFICAR_1[7] && VERIFICAR_1[8]) == 1) || ((VERIFICAR_1[0] && VERIFICAR_1[3] && VERIFICAR_1[6]) == 1)
```

```
VERIFICAR_1[6]) == 1) || ((VERIFICAR_1[1] && VERIFICAR_1[4] && VERIFICAR_1[7]) == 1) || ((VERIFICAR_1[2] && VERIFICAR_1[5] && VERIFICAR_1[8]) == 1) ||
```

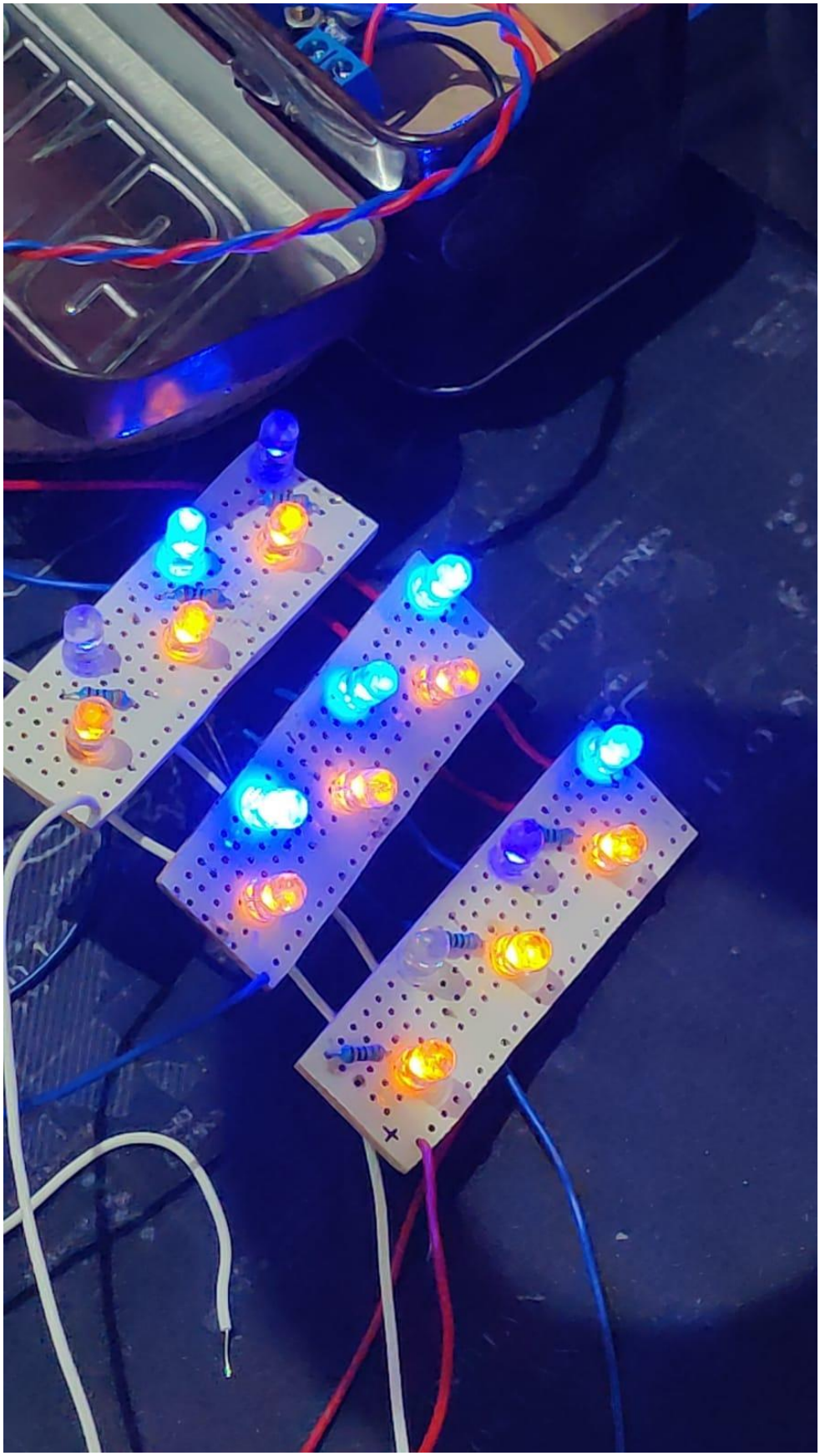
```
&& VERIFICAR_1[8]) == 1) || ((VERIFICAR_1[0] && VERIFICAR_1[4] && VERIFICAR_1[8]) == 1) || ((VERIFICAR_1[2] && VERIFICAR_1[4] && VERIFICAR_1[6]) == 1)) {
```

(esto fue la extensión de la línea de código 312 y su condición puesto que no cabe en una sola captura de pantalla)

```
                }
                for (x = 0; x <= DELAY_2; x++) {
                    PORTF = 0x00F7;
                    PORTB = 0x0007 + DISPLAY_1[GANAR_1] + DISPLAY_2[GANAR_2];
                }
                if (CICLI_OFF == 8) {
                    for (x = 0; x <= 9; x++) {
                        IMPRIMIR[x] = 0x0000; //recorre todo el arreglo poniendolo en 0 y apagando los leds
                        VERIFICAR_1[x] = 0; //Ponemos en 0 el arreglo indicando que todos los LEDs AMARILLOS estan apagados
                        VERIFICAR_2[x] = 0; //Ponemos en 0 el arreglo indicando que todos los LEDs AZULES estan apagados

                    }//Llave del for que resetea los arreglos
                    if (GANAR_2 < 7) {
                        GANAR_2++;
                        break;
                    }
                }

            }

        } // Llave de el WHILE que verifica si ganamos


        for (x = 0; x <= DELAY; x++) {
            PORTF = 0x00FE; //RF0 0000 0000 1111 1110
            PORTB = IMPRIMIR[0] + IMPRIMIR[1] + IMPRIMIR[2] + DISPLAY_1[GANAR_1] + DISPLAY_2[GANAR_2];
            if (VERIFICAR_1[0] == 0) {
                if (PORTD == RECIBIENDO[0]) {
                    IMPRIMIR[0] = 0x0010;
                    VERIFICAR_2[0] = 1;
                    TURNO = 1;
                }
            }
            if (VERIFICAR_1[1] == 0) {
                if (PORTD == RECIBIENDO[1]) {
                    IMPRIMIR[1] = 0x0020;
                    VERIFICAR_2[1] = 1;
                    TURNO = 1;
                }
            }
            if (VERIFICAR_1[2] == 0) {
                if (PORTD == RECIBIENDO[2]) {
                    IMPRIMIR[2] = 0x0040;
                    VERIFICAR_2[2] = 1;
                    TURNO = 1;
                }
            }
        }

        for (x = 0; x <= DELAY; x++) {
            PORTF = 0x00FD; //RF1 000 0000 1111 1101
            PORTB = IMPRIMIR[3] + IMPRIMIR[4] + IMPRIMIR[5] + DISPLAY_1[GANAR_1] + DISPLAY_2[GANAR_2];
            if (VERIFICAR_1[3] == 0) {
                if (PORTD == RECIBIENDO[0]) {
                    IMPRIMIR[3] = 0x0010;
                    VERIFICAR_2[3] = 1;
                    TURNO = 1;
                }
            }
```

```
383
384                 if (VERIFICAR_1[4] == 0) {
385                     if (PORTD == RECIBIENDO[1]) {
386                         IMPRIMIR[4] = 0x0020;
387                         VERIFICAR_2[4] = 1;
388                         TURNO = 1;
389                     }
390                 }
391
392                 if (VERIFICAR_1[5] == 0) {
393                     if (PORTD == RECIBIENDO[2]) {
394                         IMPRIMIR[5] = 0x0040;
395                         VERIFICAR_2[5] = 1;
396                         TURNO = 1;
397                     }
398                 }
399             }
400
401             for (x = 0; x <= DELAY; x++) {
402                 PORTF = 0x00F7; //RF3 0000 0000 1111 0111
403                 PORTB = IMPRIMIR[6] + IMPRIMIR[7] + IMPRIMIR[8] + DISPLAY_1[GANAR_1] + DISPLAY_2[GANAR_2];
404                 if (VERIFICAR_1[6] == 0) {
405                     if (PORTD == RECIBIENDO[0]) {
406                         IMPRIMIR[6] = 0x0010;
407                         VERIFICAR_2[6] = 1;
408                         TURNO = 1;
409                     }
410                 }
411
412                 if (VERIFICAR_1[7] == 0) {
413                     if (PORTD == RECIBIENDO[1]) {
```

```
407                         VERIFICAR_2[6] = 1;
408                         TURNO = 1;
409                     }
410                 }
411
412                 if (VERIFICAR_1[7] == 0) {
413                     if (PORTD == RECIBIENDO[1]) {
414                         IMPRIMIR[7] = 0x0020;
415                         VERIFICAR_2[7] = 1;
416                         TURNO = 1;
417                     }
418                 }
419
420                 if (VERIFICAR_1[8] == 0) {
421                     if (PORTD == RECIBIENDO[2]) {
422                         IMPRIMIR[8] = 0x0040;
423                         VERIFICAR_2[8] = 1;
424                         TURNO = 1;
425                     }
426                 }
427             }
428
429         }// TURNO 2
430     }
431
432     return 0;
433 }
434
```




OLEUM

Producto Final