PROYECTO PSOC5

Autores: Robin Costas del Moral

José Manuel Herrera Vera

José Antonio Padial Molina

Miguel Ángel Rispal Martínez

Contenido

INTRODUCCIÓN

Hemos creado un juego basado en el juego clásico de “SPACE INVADERS” el cual consiste en manejar una nave espacial y disparar a unos alienígenas.

Los alienígenas son de diferentes colores y tamaños, descienden desde la parte de arriba de la matriz de led hacia abajo y nosotros debemos de dispararle para eliminarlos.

Cada monstruo según su velocidad y tipo nos da una puntuación u otra al matarlo.

El juego termina cuando un alienígena nos toca.

El jugador interactúa con el juego de la siguiente forma:

- Pedal izquierdo: Para disparar.
- Pedal derecho: Para subir y bajar.
- Volante: Derecha e izquierda para movernos e intentar esquivar a los alienígenas.

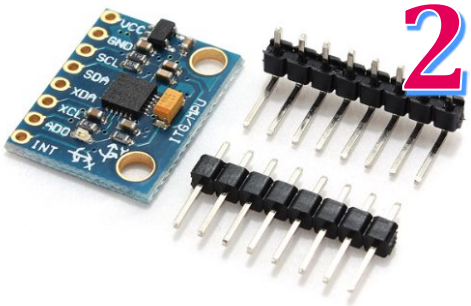
MATERIALES UTILIZADOS

Pantalla LCD Modulo Verde Indicadores 2 x 16 caracteres para Arduino

1

- Uso: La usamos para mostrar datos:
  - Al comienzo mostramos el proceso de calibración.
  - Durante el juego mostramos la puntuación.
  - Al final muestra “GAME OVER”.

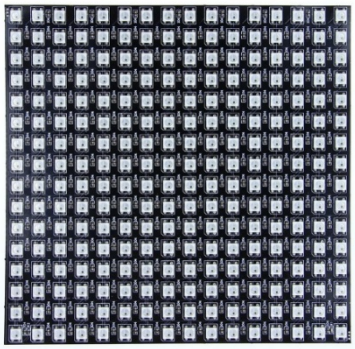
SODIAL(R) Modulo MPU-6050 de 3 ejes giroscopio + 3 Modulo Acelerómetro GY-521 Eje para Arduino



2

- Uso:
  - Calculamos los grados de inclinación del volante.
  - En función de los grados calculados con el giroscopio, nos movemos en el eje X de la matriz de leds.

ALITOVE For Arduino WS2812B LED Rainbow Matrix 16x16 256 Pixels



3

- Uso:
  - Mostrar jugador.
  - Mostrar enemigos.
  - Mostrar disparos.

Pedales (2 Potenciómetros)

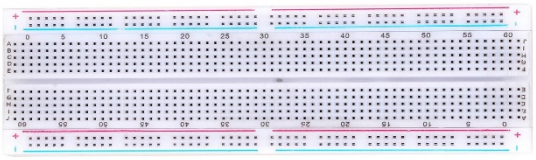
4

- Uso:
  - El pedal izquierdo lo usamos para disparar.
  - El pedal derecho lo usamos para movernos en el eje Y.

830 Protoboard Placa Breadboard PCB Circuito Sin Soldadura MB -102

5

- Uso:
  - Para hacer las conexiones necesarias.



Cables Puente para Placas Prototipo (Protoboard) para Arduino, Kit de Cables para Arduino

- Uso:
  - Para hacer las conexiones necesarias.

WINGONEER MB102 Tablero 3.3V/5V la fuente de alimentación del módulo 3.3V/5V para Arduino

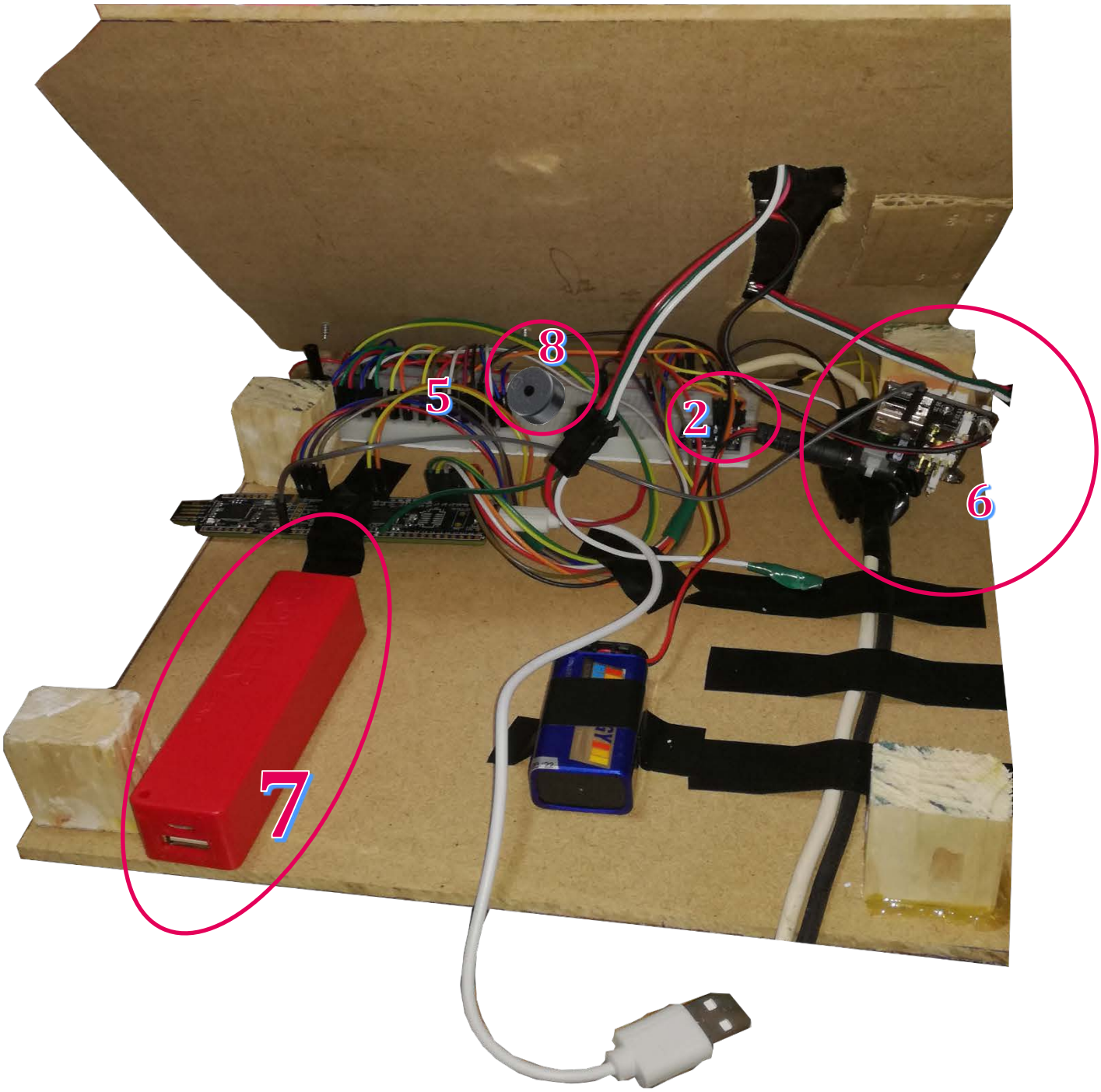
- Uso:
  - Para alimentar la matriz de leds.

POWERBANK

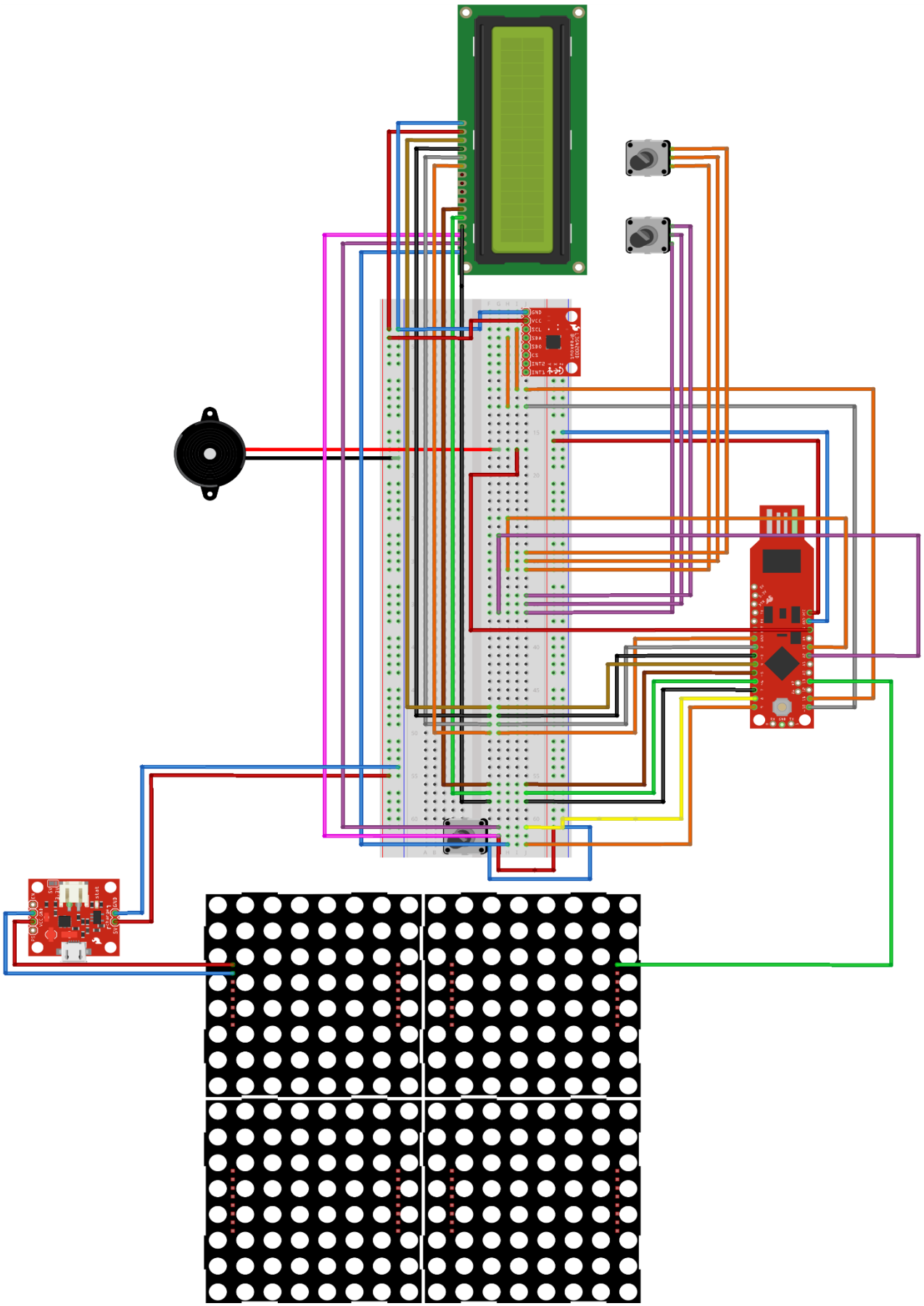
- Uso:
  - Para alimentar la PSOC 5.

BUZZER

- Uso:
  - Pitido corto cuando matamos a un monstruo.
  - Pitido largo cuando matan al personaje.


8
5
2
6
7

Esquema de conexiones



EXPLICACIÓN DEL CÓDIGO

• Declaración de objetos:

Los monstruos los vamos a representar con un struct que contienen:

- int posx; //Representa la coordenada x donde se encuentra
- int posy; //Representa la coordenada y donde se encuentra
- int tipo; //Representa el tipo de monstruo (1-4)
- int velocidad; //Velocidad del monstruo
- int muerto; //Si está muerto o no
- int contador;
- uint32 color; //Color del monstruo

Las balas las vamos a representar con un struct que contienen:

- int posx; //Representa la coordenada x donde se encuentra
- int posy; //Representa la coordenada y donde se encuentra
- int disparada; //Entero que representa si dicha bala está disparada o en la recámara; 0 no está disparada y 1 si está disparada

- Funciones:

void movimiento(monstruo *m)

Esta función hace descender al monstruo y dependiendo de una variable aleatoria solo desciende, desciende y gira a la derecha o desciende y gira a la izquierda.

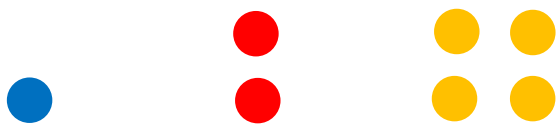
void imprimirMonstruo(monstruo m)

Le pasamos el monstruo por parámetro y lo primero que hacemos es comprobar si el monstruo está muerto o no.
Si está vivo pintamos el monstruo, si está muerto no (0 muerto; 1 vivo).

Tenemos 3 tipos de monstruos:

Según que monstruo sea lo pintamos de una forma o de otra.

Si es de tipo 1 dibujamos un pixel azul;
Si es de tipo 2 dibujamos dos pixeles rojos;
Si es de tipo 3 dibujamos 4 pixeles naranjas.



TIPO 1 TIPO 2 TIPO 3

void imprimirBalas()

Recorremos el array de balas comprobando en que cada posición el valor de la variable disparada.

En caso de valer 1 pintamos la bala en el lugar que le corresponde en la matriz de leds según “posx” y “posy”, en caso contrario, se pinta en la parte de abajo de forma ordenada (recámara).

int detectarColision(int x, int y, monstruo m)

Comprobamos si la “x” y la “y” pasadas por parámetro coinciden con la “posx” y “posy” del monstruo.

En caso afirmativo, devuelve un 1, en caso contrario, devuelve 0.

void moverBalas()

Recorremos el array de balas comprobando en que cada posición el valor de la variable disparada.

En caso de valer 1 incrementamos la variable “posy”, si “posy” supera el valor 63 marcamos la variable disparada y “posy” a 0.

int main()

Antes de llegar al bucle infinito a modo de resumen, lo que hacemos es inicializar todos los componentes, los monstruos y las balas.

Además, realizamos la calibración del giroscopio para asegurarnos así unos valores correctos.

Dentro del bucle infinito:

1. Tomamos valores del giroscopio y de los potenciómetros.
2. Imprimimos en la LCD los puntos que va obteniendo el jugador.
3. Tenemos un switch donde vemos en que rango se encuentra el valor del giroscopio (ángulo Y). En función del valor del ángulo Y incrementamos o decrementamos la posición X del jugador.
4. Llamamos a detectar colisión pasándole cada uno de los monstruos y las coordenadas del jugador. En caso de valer 1 ponemos la variable muerto a 1.
5. Comprobamos el valor de la variable muerto y en caso de ser 1, imprimimos una cruz roja en la matriz de leds. Además, hacemos sonar un zumbador.
6. Comprobamos si hay balas en la recámara, en caso afirmativo, las disparamos.
7. Recorremos el array de balas haciendo en cada posición una llamada a la función de detectarColisión() pasando como parámetros las coordenadas de la bala y un monstruo. En caso de haber colisión ponemos la variable muerto del monstruo a 1 y aumentamos la puntuación en función del tipo de monstruo.
8. Por último, llamamos a las funciones de imprimir y de movimiento. Y realizamos un delay de 50ms.

- Inicialización de los componentes:

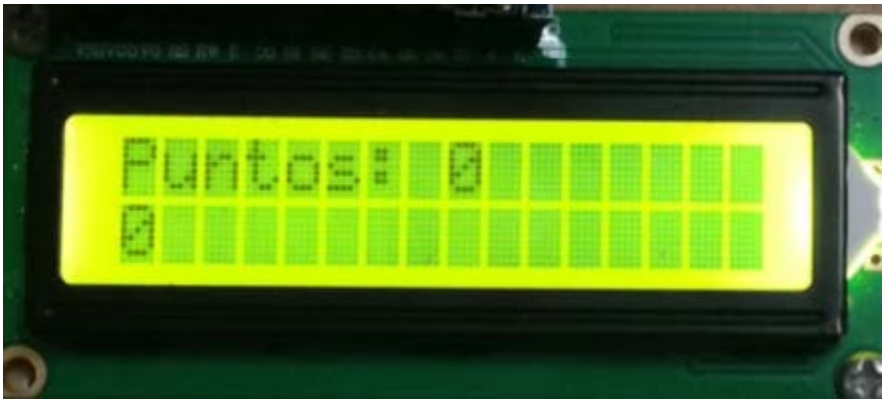
  - StripLights_Pixel(jugador,col,StripLights_LTBLUE);
    - //Para pintar cada pixel de la matriz de leds
  - I2C_MPU6050_Start();
    - //Para iniciar el I2C del giroscopio
  - StripLights_Start();
    - //Iniciar matriz de leds
  - MUX_Start();
    - //Iniciar multiplexor
  - MPU6050_init();
    - //Iniciar giroscopio
  - LCD_Start();
    - //Iniciar la LCD
  - LCD_ClearDisplay();
    - //Limpiar la LCD
  - ADC_Start();
    - //Iniciar el conversor analógico-digital
  - ADC_StartConvert();
    - //Iniciar la conversión
  - PWM_Init();
    - //Iniciar el PWM

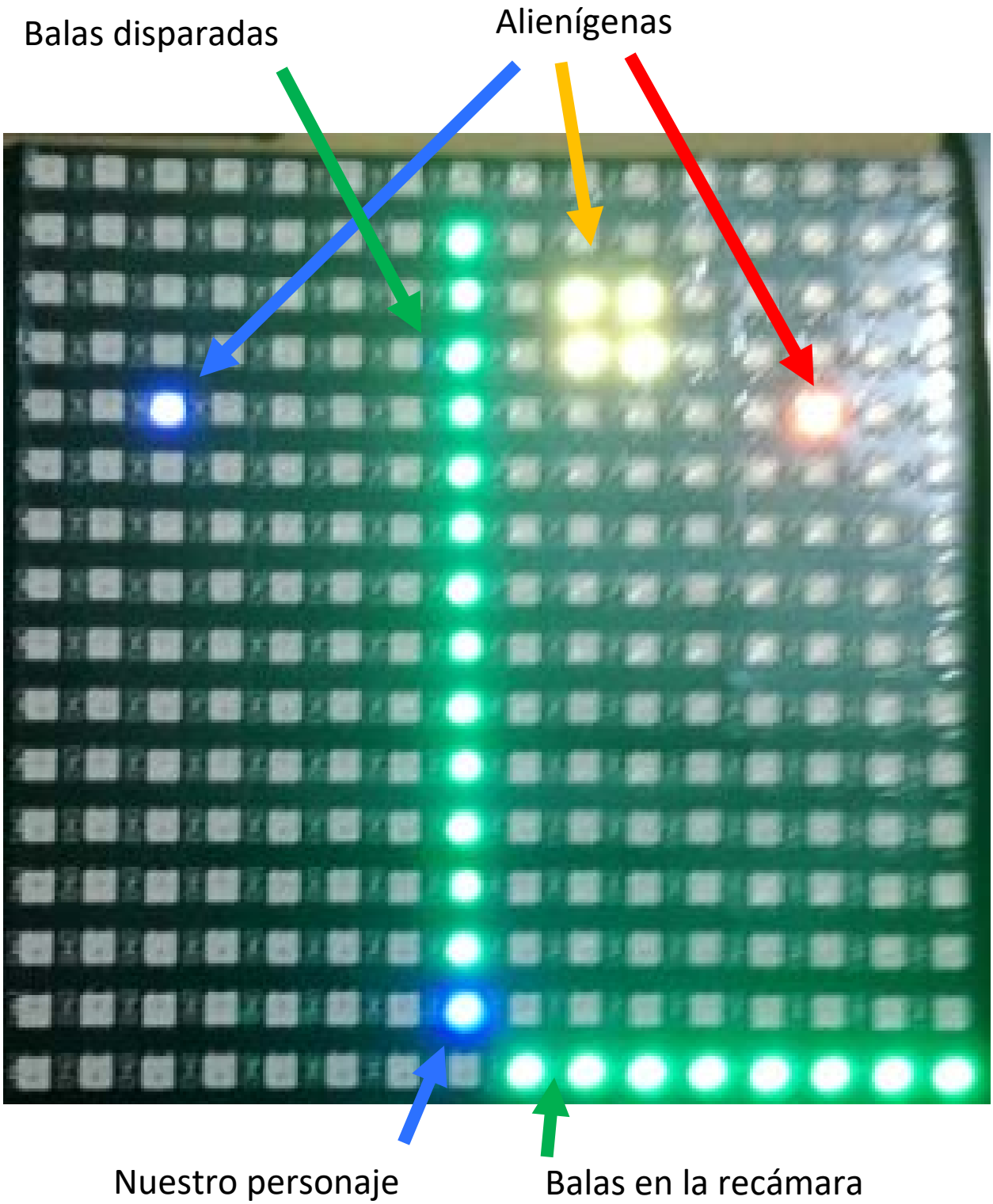
Funcionamiento del juego

Lo primero que hará nuestro juego es calibrar el giroscopio y nos muestra el porcentaje por pantalla:

Una vez calibrado ya podemos empezar a jugar.

Durante el juego la pantalla LCD nos va mostrando la puntuación actual.

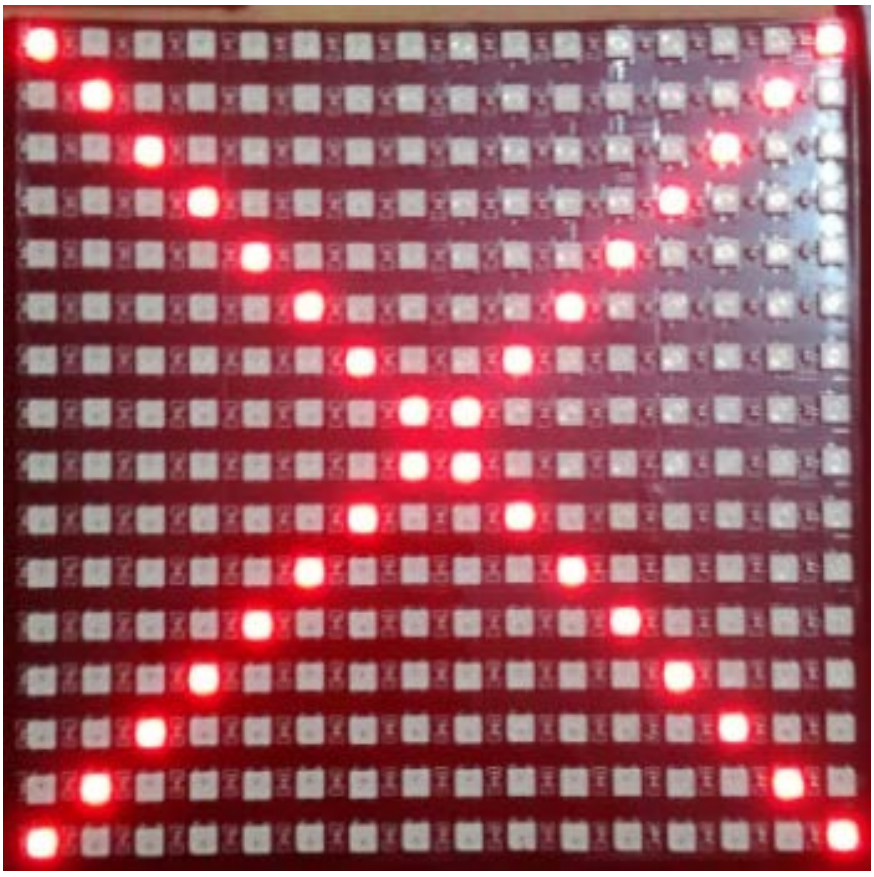



Balas disparadas
Alienígenas
Nuestro personaje
Balas en la recámara

Cada vez que matamos un alienígena suena un pitido.

Cuando nos matan y acaba el juego, suena un pitido constante y en la pantalla LCD nos muestra la puntuación total y un mensaje de “Game Over”.

Además, en la matriz de leds aparecerá una cruz roja.



```c
#include <project.h>
#include <mpu6050.h>
#include <stdio.h>
#include <math.h>
#include<stdlib.h>

uint16 ADCResult;
int16_t CAX, CAY, CAZ; //current acceleration values
int16_t CGX, CGY, CGZ; //current gyroscope values
int16_t CT;            //current temperature
int h=0,i=0,dato=0;

int vel_bala=0;
int balas_activas = 0;

int jugador=0;
int col=0;

int puntuacion=0;
int muerto = 0;
int reset = 0;


void imprimirJugador(){
    StripLights_Pixel(jugador,col,StripLights_LTBLUE);
}

typedef struct monstruo monstruo;
struct monstruo{
  int posx;
  int posy;
  int tipo;
  int velocidad;
  int muerto;
  int contador;
  uint32 color;

};

typedef struct structbalas structbalas;
struct structbalas{
  int posx;
  int posy;
  int disparada;
};

structbalas balas[16];

void movimiento(monstruo *m){

    if(m->contador==m->velocidad){
        m->posy--;
        if(m->posy==1){
            m->posy=15;
            m->tipo=(rand()%3)+1;
            m->muerto=0;
        }
        int dir = rand()%3;


        switch (dir){
            case 0:
                if((m->posx%4)==0){
                    m->posx++;
                }
                else if(m->tipo==1 && m->posx%4==1)
                    m->posx++;
                else
                    m->posx--;
                break;

```

```
            case 2:
                if(m->posx%4==3){
                    m->posx--;
                }
                else
                    m->posx++;
                break;
        }

        m->contador=0;
    }
    else{
        m->contador++;
    }
}

void imprimirMonstruo(monstruo m){
    if(m.muerto == 0){
        if(m.tipo == 1 || m.tipo==2)
           StripLights_Pixel(m.posx,m.posy-1,m.color);
        if(m.tipo == 1){
            StripLights_Pixel(m.posx-1,m.posy-1,m.color);
            StripLights_Pixel(m.posx-1,m.posy,m.color);
        }

        StripLights_Pixel(m.posx,m.posy,m.color);
    }

}

void imprimirBalas(){
    int j=0;
    for(int i=0; i<16; i++){
        if(balas[i].disparada==1 && balas[i].posy<15)
            StripLights_Pixel(balas[i].posx,balas[i].posy,StripLights_LTGREEN);
        else if(balas[i].posy==0){
            StripLights_Pixel(j,0,StripLights_LTGREEN);
            j++;
        }
    }
}

int detectarColision(int x, int y, monstruo m){
    if(m.muerto == 0){
        if(m.tipo == 1){
            if(x == m.posx || x == m.posx -1){
                if(y == m.posy || y == m.posy -1){
                    return 1;
                }
            }
            return 0;
        }
        if(m.tipo == 2){
            if(x == m.posx){
                if(y == m.posy || y == m.posy -1){
                    return 1;
                }
            }
            return 0;
        }
        else{
            if(x == m.posx){
                if(y == m.posy)
                    return 1;
            }
            return 0;
        }
    }
    else
        return 0;
}
```

```
143
144   void moverBalas(){
145       for(int i=0; i<16; i++){
146           if(balas[i].disparada==1){
147               if(balas[i].posy > 63){
148                   balas[i].disparada=0;
149                   balas[i].posy=0;
150                   balas_activas--;
151               }
152               else{
153                   balas[i].posy++;
154               }
155
156           }
157       }
158   }
159
160   int main()
161   {
162       uint8_t canal = 0;
163       int32 adcread;
164       float voltaje;
165       int j = 0; //for loop increment variable
166       char buf[50]; //just to hold text values in for writing to LCD
167
168
169       monstruo monstruo1;
170       monstruo1.tipo=0;
171       monstruo1.posx=0;
172       monstruo1.posy=15;
173       monstruo1.velocidad=3;
174       monstruo1.muerto=0;
175       monstruo1.contador=0;
176       monstruo1.color=0x009933;
177
178       monstruo monstruo2;
179       monstruo2.tipo=1;
180       monstruo2.posx=4;
181       monstruo2.posy=15;
182       monstruo2.velocidad=5;
183       monstruo2.muerto=0;
184       monstruo2.contador=0;
185       monstruo2.color=StripLights_LTYELLOW;
186
187       monstruo monstruo3;
188       monstruo3.tipo=2;
189       monstruo3.posx=8;
190       monstruo3.velocidad=4;
191       monstruo3.contador=0;
192       monstruo3.muerto=0;
193       monstruo3.posy=15;
194
195       monstruo3.color=StripLights_LTRED;
196
197       monstruo monstruo4;
198       monstruo4.tipo=0;
199       monstruo4.posx=12;
200       monstruo4.posy=15;
201       monstruo4.velocidad=3;
202       monstruo4.contador=0;
203       monstruo4.muerto=0;
204       monstruo4.color=0x221111;
205
206
207       I2C_MPU6050_Start();
208
209       CyGlobalIntEnable;
210
211       StripLights_Start();
212       StripLights_DisplayClear(StripLights_BLACK);
213
```

```
214         MUX_Start();
215 
216         MPU6050_init();
217         MPU6050_initialize();
218 
219         LCD_Start();
220         LCD_ClearDisplay();
221         ADC_Start();
222         ADC_StartConvert();
223         ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
224 
225         PWM_Init();
226 
227         LCD_Position(0,0);
228         float accel_ang_x;
229         float accel_ang_y;
230         LCD_Position(0,0);
231         sprintf(buf, "Calibrando:");
232         LCD_PrintString(buf);
233         LCD_Position(1,2);
234         LCD_PrintString("Espere...");
235 
236         for(int i=0; i<16; i++){
237             balas[i].posx=0;
238             balas[i].posy=0;
239             balas[i].disparada=0;
240         }
241 
242 
243         for(j=0; j<100; j++)
244         {
245             MPU6050_getMotion6t(&CAX, &CAY, &CAZ, &CGX, &CGY, &CGZ, &CT);
246 
247             accel_ang_x=atan(CAX/sqrt(pow(CAY,2) + pow(CAZ,2)))*(180.0/3.14);
248             accel_ang_y=atan(CAY/sqrt(pow(CAX,2) + pow(CAZ,2)))*(180.0/3.14);
249 
250             LCD_Position(0,12);
251             sprintf(buf, "%d%c",j,'%');
252             LCD_PrintString(buf);
253 
254             CyDelay(50);
255         }
256 
257         LCD_ClearDisplay();
258 
259         for (;;){
260 
261             MUX_FastSelect(canal);
262             adcread=ADC_GetResult32();
263             voltaje=floor(((5.000/1048576)*adcread));
264             MPU6050_getMotion6t(&CAX, &CAY, &CAZ, &CGX, &CGY, &CGZ, &CT);
265 
266             accel_ang_x=ceil(atan(CAX/sqrt(pow(CAY,2) + pow(CAZ,2)))*(180.0/3.14));
267             accel_ang_y=ceil(atan(CAY/sqrt(pow(CAX,2) + pow(CAZ,2)))*(180.0/3.14));
268 
269             /*LCD_Position(0,0);
270             sprintf(buf, "X:%2.0f",accel_ang_x);
271             LCD_PrintString(buf);
272 
273             LCD_Position(0,8);
274             sprintf(buf, "Y:%2.0f",accel_ang_y);;
275             LCD_PrintString(buf);
276 
277             LCD_Position(1,canal*8);
278 
279             LCD_PrintString("V");
280             LCD_PrintNumber(canal);
281             LCD_PrintString(": ");
282             LCD_PrintNumber(voltaje);*/
283 
284             LCD_Position(0,0);
```

```
285             LCD_PrintString("Puntos: ");
286             LCD_PrintNumber(puntuacion);
287             canal=1-canal;
288             LCD_Position(1,0);
289             LCD_PrintNumber(balas_activas);
290             LCD_PrintString("  ");
291
292             voltaje++;
293
294             switch((int) accel_ang_y){
295                 case 39 ... 90:
296                     jugador=15;
297                     if(canal==0){
298                         if(col<voltaje){
299                             col++;
300                         }
301                         else if(col>voltaje)
302                             col--;
303                     }
304                     break;
305
306                 case 33 ... 38:
307                     jugador=14;
308                     if(canal==0){
309                         if(col<voltaje)
310                             col++;
311                         else if(col>voltaje)
312                             col--;
313                     }
314                     break;
315
316                 case 27 ... 32:
317                     jugador=13;
318                     if(canal==0){
319                         if(col<voltaje)
320                             col++;
321                         else if(col>voltaje)
322                             col--;
323                     }
324                     break;
325
326                 case 22 ... 26:
327                     jugador=12;
328                     if(canal==0){
329                         if(col<voltaje)
330                             col++;
331                         else if(col>voltaje)
332                             col--;
333                     }
334                     break;
335
336                 case 16 ... 21:
337                     jugador=11;
338                     if(canal==0){
339                         if(col<voltaje)
340                             col++;
341                         else if(col>voltaje)
342                             col--;
343                     }
344                     break;
345
346                 case 11 ... 15:
347                     jugador=10;
348                     if(canal==0){
349                         if(col<voltaje)
350                             col++;
351                         else if(col>voltaje)
352                             col--;
353                     }
354                     break;
355
```

```
356             case 6 ... 10:
357                 jugador=9;
358                 if(canal==0){
359                     if(col<voltaje)
360                         col++;
361                     else if(col>voltaje)
362                         col--;
363                 }
364                 break;
365 
366             case 0 ... 5:
367                 jugador=8;
368                 if(canal==0){
369                     if(col<voltaje)
370                         col++;
371                     else if(col>voltaje)
372                         col--;
373                 }
374                 break;
375 
376             case -6 ... -1:
377                 jugador=7;
378                 if(canal==0){
379                     if(col<voltaje)
380                         col++;
381                     else if(col>voltaje)
382                         col--;
383                 }
384                 break;
385 
386             case -11 ... -7:
387                 jugador=6;
388                 if(canal==0){
389                     if(col<voltaje)
390                         col++;
391                     else if(col>voltaje)
392                         col--;
393                 }
394                 break;
395 
396             case -16 ... -12:
397                 jugador=5;
398                 if(canal==0){
399                     if(col<voltaje)
400                         col++;
401                     else if(col>voltaje)
402                         col--;
403                 }
404                 break;
405 
406             case -22 ... -17:
407                 jugador=4;
408                 if(canal==0){
409                     if(col<voltaje)
410                         col++;
411                     else if(col>voltaje)
412                         col--;
413                 }
414                 break;
415 
416             case -27 ... -23:
417                 jugador=3;
418                 if(canal==0){
419                     if(col<voltaje)
420                         col++;
421                     else if(col>voltaje)
422                         col--;
423                 }
424                 break;
425 
426             case -33 ... -28:
```

```
                jugador=2;
                if(canal==0){
                    if(col<voltaje)
                        col++;
                    else if(col>voltaje)
                        col--;
                }
                break;

            case -37 ... -34:
                if(canal==0){
                    if(col<voltaje)
                        col++;
                    else if(col>voltaje)
                        col--;
                }
                break;

            default:
                jugador=0;
                if(canal==0){
                    if(col<voltaje)
                        col++;
                    else if(col>voltaje)
                        col--;
                }
                break;
        }

        if(detectarColision(jugador,col,monstruo1) == 1) muerto=1;
        if(detectarColision(jugador,col,monstruo2) == 1) muerto=1;
        if(detectarColision(jugador,col,monstruo3) == 1) muerto=1;
        if(detectarColision(jugador,col,monstruo4) == 1) muerto=1;

        if(muerto == 1){
            PWM_Start();
            StripLights_MemClear(StripLights_BLACK);
            LCD_Position(0,0);

            LCD_PutChar(LCD_CUSTOM_0);LCD_PutChar(LCD_CUSTOM_0);LCD_PutChar(LCD_CUSTOM
            _0);
            LCD_PrintString("Game Over!");

            LCD_PutChar(LCD_CUSTOM_0);LCD_PutChar(LCD_CUSTOM_0);LCD_PutChar(LCD_CUSTOM
            _0);
            LCD_Position(1,0);
            LCD_PrintString("Puntos: ");
            LCD_PrintNumber(puntuacion);

            while(true){

                for(int i=0; i<8; i++){
                        StripLights_Pixel(i,i,StripLights_LTRED);
                        StripLights_Pixel(i,15-i,StripLights_LTRED);
                        StripLights_Pixel(15-i,i,StripLights_LTRED);
                        StripLights_Pixel(15-i,15-i,StripLights_LTRED);
                        StripLights_Trigger(1);
                        CyDelay(50);
                }
                CyDelay(500);
                StripLights_MemClear(StripLights_BLACK);
            }
        }

        if(canal==1 && balas_activas<16){
            int libre=0;
            while(balas[libre].disparada==1){
                libre++;
            }
            if(voltaje == 2){
                if(vel_bala < 4){
```

```
                    vel_bala++;
                }
                else{
                    balas[libre].disparada =1;
                    balas[libre].posx = jugador;
                    balas[libre].posy = col + 1;
                    balas_activas++;
                    vel_bala=0;
                }
            }
            if(voltaje == 3){
                if(vel_bala < 3){
                    vel_bala++;
                }
                else{
                    balas[libre].disparada =1;
                    balas[libre].posx = jugador;
                    balas[libre].posy = col + 1;
                    balas_activas++;
                    vel_bala=0;
                }
            }
            if(voltaje == 4){
                if(vel_bala < 2){
                    vel_bala++;
                }
                else{
                    balas[libre].disparada =1;
                    balas[libre].posx = jugador;
                    balas[libre].posy = col + 1;
                    balas_activas++;
                    vel_bala=0;
                }
            }
            if(voltaje >= 5){
                    balas[libre].disparada =1;
                    balas[libre].posx = jugador;
                    balas[libre].posy = col + 1;
                    balas_activas++;
                    vel_bala=0;

            }

        }

        for(int i=0; i<16; i++){
            if(balas[i].disparada==1){
                if(detectarColision(balas[i].posx,balas[i].posy,monstruo1) == 1){
                    monstruo1.muerto=1;
                    PWM_Start();
                    CyDelay(10);
                    PWM_Stop();
                    if(monstruo1.tipo==1)
                        puntuacion+=50;
                    else if(monstruo1.tipo==2)
                        puntuacion+=100;
                    else
                        puntuacion+=200;
                }
                if(detectarColision(balas[i].posx,balas[i].posy,monstruo2) == 1){
                    monstruo2.muerto=1;
                    PWM_Start();
                    CyDelay(10);
                    PWM_Stop();
                    if(monstruo2.tipo==1)
                        puntuacion+=50;
                    else if(monstruo2.tipo==2)
                        puntuacion+=100;
                    else
                        puntuacion+=200;
                }
```

```
                if(detectarColision(balas[i].posx,balas[i].posy,monstruo3) == 1){
                    monstruo3.muerto=1;
                    PWM_Start();
                    CyDelay(10);
                    PWM_Stop();
                    if(monstruo3.tipo==1)
                        puntuacion+=50;
                    else if(monstruo3.tipo==2)
                        puntuacion+=100;
                    else
                        puntuacion+=200;
                }
                if(detectarColision(balas[i].posx,balas[i].posy,monstruo4) == 1){
                    monstruo4.muerto=1;
                    PWM_Start();
                    CyDelay(10);
                    PWM_Stop();
                    if(monstruo4.tipo==1)
                        puntuacion+=50;
                    else if(monstruo4.tipo==2)
                        puntuacion+=100;
                    else
                        puntuacion+=200;
                }
            }
        }

        StripLights_MemClear(StripLights_BLACK);
        imprimirJugador();
        imprimirBalas();
        imprimirMonstruo(monstruo1);
        imprimirMonstruo(monstruo2);
        imprimirMonstruo(monstruo3);
        imprimirMonstruo(monstruo4);
        StripLights_Trigger(1);

        movimiento(&monstruo1);
        movimiento(&monstruo2);
        movimiento(&monstruo3);
        movimiento(&monstruo4);
        moverBalas();

        CyDelay(50);
    }
}

```