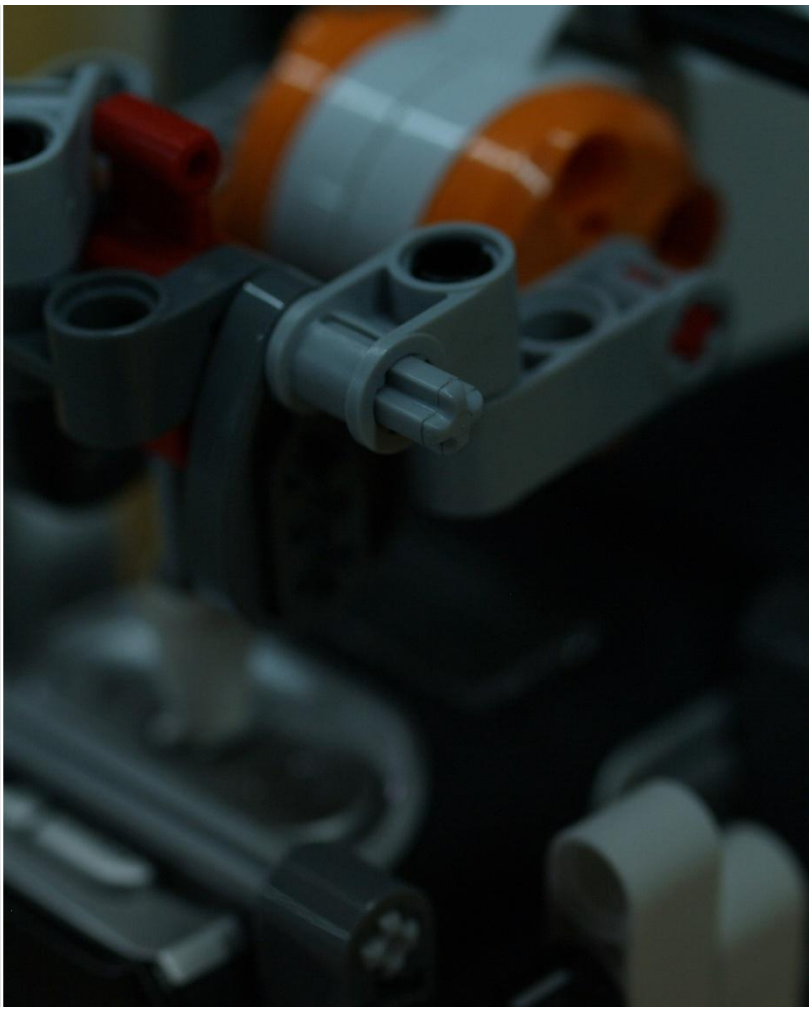
Memoria Trabajo Práctico

ULPGC

2010/2011



Biocibernética Computacional

Alberto Manuel Mireles Suárez
David Guillermo Morales Sáez

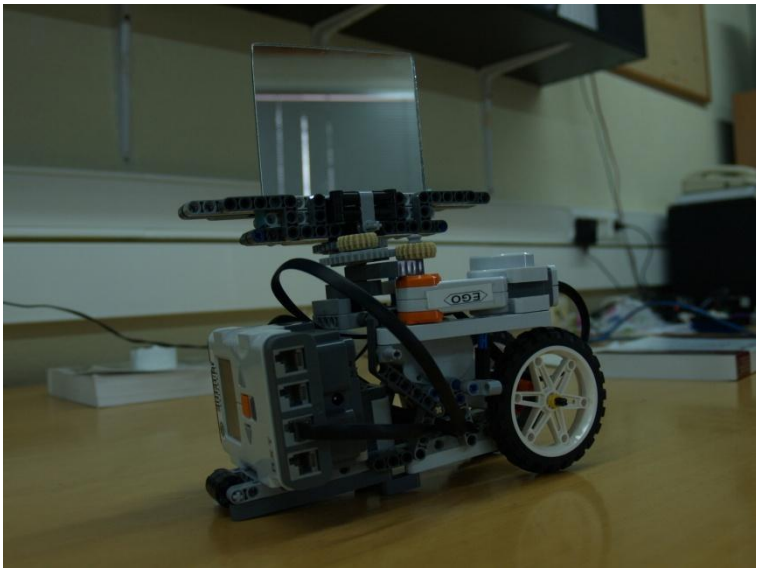
Introducción

El trabajo propuesto para este año se centra en el juego de los años 80 “Deflektor”. Para llevarlo a cabo, hemos de utilizar el sistema robótico LEGO® Mindstorms™ NXT™ e imitar el juego en la realidad, utilizando dos tipos de unidades, unas que se encarguen de generar el láser y otras que se encarguen de redireccionarlo.

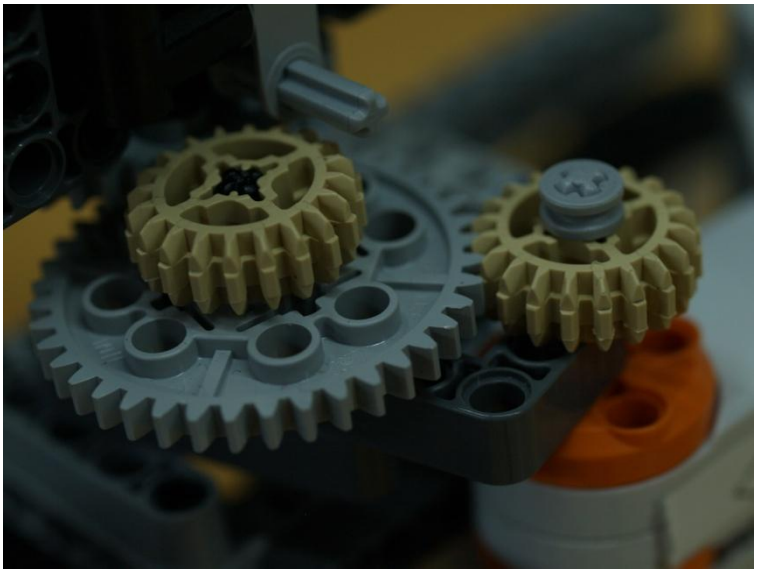
Diseño de las unidades

En el trabajo propuesto, hemos tenido que diseñar dos unidades independientes, cada cual con una finalidad distinta: una plataforma espejo y otra láser. Para el movimiento de las plataformas a lo largo de los raíles hemos utilizado unos sensores de luz para leer la marca de parada debida. Para que las distintas plataformas no se desvien del camino trazado por el raíl, hemos creado una superficie que encaja con el raíl sin llegar a tocar el suelo, impidiendo que las plataformas se salgan de su curso.

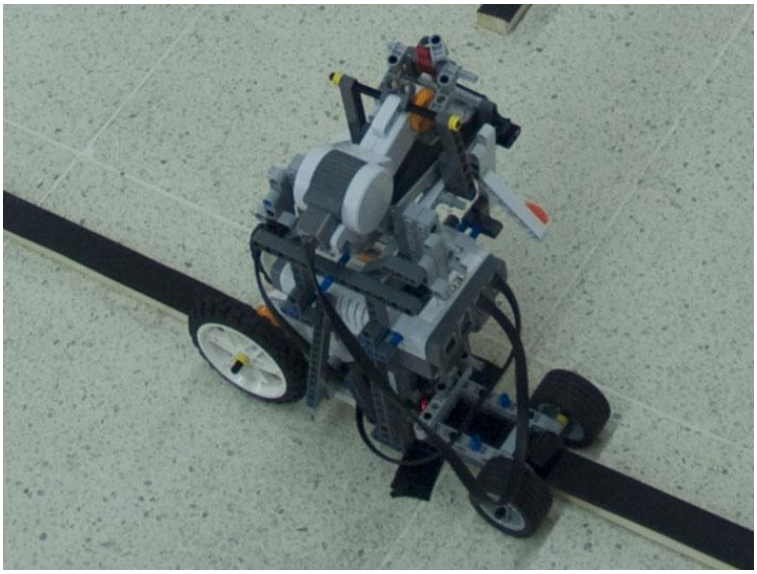
- Plataforma espejo:



La función final de esta plataforma es la de redireccionar los rayos entrantes hacia una posición predeterminada. Para ello, utilizamos la reflexión creada por un espejo. Se definieron 16 posiciones distintas para el espejo, por lo que tuvimos que buscar la manera en la cual el espejo fuese capaz de girar con una precisión de medio grado. Para ello utilizamos un sistema de engranajes 4:1.



- Plataforma láser:



La función final de esta plataforma es la de enviar el láser a la primera plataforma espejo de la ruta calculada para golpear al enemigo en cuestión. Para ello, hace uso de un sistema que hace presión sobre el botón de disparo del puntero láser entregado.

Hay que comentar que hemos diseñado las plataformas para que el “brick” sea de fácil acceso para la manipulación de los programas y para la carga de su batería.

Algoritmo

El algoritmo diseñado para este trabajo no busca una ruta ya prefijada, de forma directa, sino que calcula la matriz de trayectorias posibles de ataque y coloca los espejos maximizando el número de trayectorias posibles. Tras calcular las posiciones de los espejos, calculamos la ruta de ataque para cada plataforma láser contrario. Para poder ver bien el diseño, vamos a explicar detalladamente cada bloque del algoritmo:

- Inicialización y obtención de las posiciones enemigas

  Inicializamos las variables de control, la matriz y las posiciones de los obstáculos. Además, nos conectamos con el servidor via Bluetooth para recibir las posiciones de los enemigos. Éstas vienen con el formato (x,y), siendo 'x' un 1 si el enemigo está en activo (no ha sido descalificado) y un 0 en caso contrario; e 'y' la posición del enemigo en su raíl (entre 0 y 3).

- Calculamos todas las trayectorias de ataque directo

  Recorremos la matriz en base a las posibles trayectorias de ataque (en diagonal, horizontal o vertical en función del enemigo), incrementando su valor en el caso que haya un posible ataque. En el caso de encontrarnos con un obstáculo, dejamos de recorrer la matriz en esa trayectoria y continuamos con la siguiente.

- Seleccionamos las posiciones de los espejos

  Leyendo la matriz por filas (sólo las impares), comprobamos cuál es el valor mayor (siempre ha de ser impar, en el caso par lo ignoramos) para marcarlo como la posición de un espejo. En el caso en el que hayan varias posiciones igualadas al máximo, comprobamos cuales tienen más conexiones con los espejos ya establecidos y la escogemos. Esto lo hacemos siempre tras intentar calcular todas las posiciones.

- Calculamos la posición del primer espejo de las rutas

  Compramos cuál de los espejos tiene más número de conexiones y en el caso de un empate, escogemos el primero.

- Calculamos las tres rutas

    - Escogemos atacante

      Recalculamos las posibles rutas de ataque directo para la plataforma láser y guardamos sus posiciones. Tras esto, comprobamos cuantos atacantes hay. En el caso de haber un atacante, lo marcamos como atacante final; si no hay ninguno, marcamos el ataque como nulo; en otro caso, comprobamos si los espejos están alineados entre sí, en este caso se escoge el más lejano y marcamos el más cercano para que se ponga en paralelo a la trayectoria del haz. En el caso en el que no estén alineados, escogemos aquel con menos conexiones.

- Creamos la ruta

  Comprobamos la cantidad de espejos que nos faltan por colocar. Si nos queda uno, comprobamos si está conectado con el espejo inicial y el espejo que va en la tercera posición. Si es así, el espejo será el segundo en recibir el rayo, en caso contrario, el rayo partirá del espejo inicial al ya calculado. Si nos quedan dos espejos por conectar, comprobamos las conexiones que tienen entre sí y entre el espejo inicial y el final. En función de estas conexiones, los colocamos debidamente.

- Calculamos los ángulos de los espejos

  Hemos desarrollado una función que calcula la posición del espejo (0-15) en base a la posición del origen del rayo, la posición del espejo que hará la reflexión y la posición del espejo que recibirá el rayo o el enemigo.

- Enviamos las posiciones al servidor

  Nos conectamos con el servidor y le enviamos una posición del espejo por vez, es decir, hasta que el servidor no nos informe que el espejo anterior ha llegado a su posición no enviamos la siguiente posición. Hay que comentar que el programa nativo del espejo está en constante funcionamiento, esperando a recibir órdenes del servidor y, cuando las recibe, va a la posición indicada en el primer parámetro y coloca el espejo en la posición indicada en el segundo parámetro del vector pasado y actualiza sus valores internos. Tras esto notifica al servidor la finalización del movimiento y vuelve a esperar un nuevo mensaje.

- Completamos el ataque

  Nos dirigimos a la posición del raíl que concuerde con el espejo primero de la ruta de ataque y disparamos. Tras esto, volvemos a enviar las posiciones al servidor, pero para el ataque al siguiente enemigo, modificando sólo las posiciones de los espejos de las plataformas espejo. Esto lo hacemos tres veces.

Problemas Encontrados

Nos hemos encontrado principalmente dos tipos de problemas: los problemas del sistema de juego y los problemas inherentes de los NXT. Los primeros son, básicamente, la falta de homogeneidad en los raíles, estando algunos torcidos y algunos, inclusive, con un *twisting*. Esto ha ocasionado gran parte de los problemas a la hora de la alineación de los distintos raíles y al movimiento de las plataformas.

Por otro lado, tenemos los problemas inherentes de los NXT. En primer lugar, los escasos recursos computacionales que disponen los bricks nos obligaron a reducir considerablemente el algoritmo, reduciendo su efectividad drásticamente. Además, la poca fiabilidad de los motores nos ha impedido que las posiciones de los espejos sean exactas, debido a ciertas holguras de los motores, tanto en la rotación como en el propio movimiento de las plataformas.

Conclusiones

Gracias a este trabajo, hemos descubierto las dificultades que entraña el diseñar un sistema con elementos poco precisos como son los NXT, además de sus faltas tecnológicas a la hora de procesamiento y memoria. Por otro lado, hemos podido aprender mejor el funcionamiento del sistema de comunicación Bluetooth.

Anexo

Código láser:

```
// PARAMETROS
#define potencia 41 // potencia del motor
#define iteraciones 10 // n√∫mero de iteraciones en las lecturas
#define SERVERBOX 1
#define NXTBOX 0
#define CONN_TIMEOUT 15000
#define obs_ay 0
#define obs_ax 1
#define obs_by 3
#define obs_bx 0
#define obs_cy 8
#define obs_cx 1
#define obs_dy 3
#define obs_dx 8
// DEFINICIONES
#define EYE_VALUE     SENSOR_2
// VARIABLES GLOBALES
int pa,pb,pc,pos,grado_act, ang_a[3], ang_b[3], ang_c[3], ang_d[3], e_a, e_b, e_c, e_d;
bool conec_a[3], conec_b[3], conec_c[3], conec_d[3];
inline int diagon(int x1, int y1, int x2, int y2)
{
        return ((x1-x2 == y1-y2)||(x2-x1==y2-y1)||(x1-x2==y2-x1)||(x2-x1==y1-y2));
}
inline int n_conex(int a)
{
    int conex = 0;
    switch(a)
    {
    case 0:
        if(conec_d[0])
            conex++;
        if(conec_d[1])
            conex++;
        if(conec_d[2])
            conex++;
        return conex;
    case 1:
        if(conec_b[0])
            conex++;
        if(conec_b[1])
            conex++;
        if(conec_b[2])
            conex++;
        return conex;
    case 2:
        if(conec_a[0])
            conex++;
        if(conec_a[1])
            conex++;
        if(conec_a[2])
            conex++;
        return conex;
    case 3:
        if(conec_c[0])
            conex++;
        if(conec_c[1])
            conex++;
        if(conec_c[2])
            conex++;
        return conex;
    }
}
inline bool conectados(int a, int b)
{
    if((a==-1)||(b==-1))
        return false;
    switch(a)
    {
    case 0:
        switch(b)
        {
```

```
        case 1:
            return conec_d[1];
        case 2:
            return conec_d[0];
        case 3:
            return conec_d[2];
        }
    case 1:
        switch(b)
        {
        case 0:
            return conec_b[2];
        case 2:
            return conec_b[0];
        case 3:
            return conec_b[1];
        }
    case 2:
        switch(b)
        {
        case 0:
            return conec_a[2];
        case 1:
            return conec_a[0];
        case 3:
            return conec_a[1];
        }
    case 3:
        switch(b)
        {
        case 0:
            return conec_c[2];
        case 1:
            return conec_c[1];
        case 2:
            return conec_c[0];
        }
    }
}
inline int dame_pos(int ind)
{
    switch(ind)
    {
    case 0:
        return e_d;
    case 1:
        return e_b;
    case 2:
        return e_a;
    case 3:
        return e_c;
    }
}
inline void pon_ang(int ind, int ang, int n)
{
    switch(ind)
    {
    case 0:
        ang_d[n] = ang;
        break;
    case 1:
        ang_b[n] = ang;
        break;
    case 2:
        ang_a[n] = ang;
        break;
    case 3:
        ang_c[n] = ang;
        break;
    }
}
inline int halla_ang(int xa, int ya, int xb, int yb, int xc, int yc)
{
    if(ya==yb) // A y B estan alineados horizontalmente
    {
        if(xb==xc) // B y C est·n alineados verticalmente
        {
```

```
        if(yb<yc)
        {
            if(xa<xb)
                return 10;
            else
                return 6;
        }
        else
        {
            if(xa<xb)
                return 14;
            else
                return 2;
        }
    }
    else
    {
        if(xb<xc)
        {
            if(yb<yc)
            {
                if(xa<xb)
                    return 9;
                else
                    return 5;
            }
            else
            {
                if(xa<xb)
                {
                    return 15;
                }
                else
                    return 3;
            }
        }
        else
        {
            if(yb<yc)
            {
                if(xa<xb)
                    return 11;
                else
                    return 7;
            }
            else
            {
                if(xa<xb)
                    return 13;
                else
                {
                    return 1;
                }
            }
        }
    }
}
else
{
    if(xa==xb)
    {
        if(xb==xc)
            return 4;
        else
        {
            if(xb<xc)
            {
                if(ya<yb)
                {
                    if(yb<yc)
                        return 3;
                    else
                    {
                        if(yb>yc)
                            return 1;
                    }
                }
```

```
            else
            {
                if(yb<yc)
                    return 7;
                else
                {
                    if(yb>yc)
                        return 5;
                }
            }
        }
        else
        {
            if(ya<yb)
            {
                if(yb<yc)
                    return 13;
                else
                {
                    if(yb>yc)
                        return 15;
                    else
                        return 14;
                }
            }
            else
            {
                if(yb<yc)
                    return 9;
                else
                {
                    if(yb>yc)
                        return 11;
                    else
                        return 10;
                }
            }
        }
    }
}
else
{
    if(xa<xb)
    {
        if(ya<yb)
        {
            if(xb<xc)
            {
                if(yb<yc)
                    return 2;
                else
                {
                    if(yb>yc)
                        return 0;
                }
            }
            else
            {
                if(xb>xc)
                {
                    if(yc>yb)
                        return 12;
                    else
                        return 13;
                }
                else
                {
                    if(yb<yc)
                        return 11;
                    else
                        return 15;
                }
            }
        }
        else
        {
            if(xb<xc)
```

```
        {
            if(yb<yc)
                return 8;
        }
        else
        {
            if(xb>xc)
            {
                if(yc<yb)
                    return 12;
                else
                    return 11;
            }
            else
            {
                if(yb<yc)
                    return 9;
                else
                    return 13;
            }
        }
    }
}
else
{
    if(ya<yb)
    {
        if(xb<xc)
        {
            if(yb<yc)
                return 4;
        }
        else
        {
            if(xb>xc)
            {
                if(yc>yb)
                    return 6;
                else
                {
                    if(yc<yb)
                        return 0;
                    else
                        return 15;
                }
            }
            else
            {
                if(yb<yc)
                    return 5;
                else
                    return 1;
            }
        }
    }
    else
    {
        if(xb<xc)
        {
            if(yb<yc)
                return 8;
            else
            {
                if(yb>yc)
                {
                    if(xa!=xc)
                        return 10;
                    else
                        return 4;
                }
                else
                {
                                                    return 9;
                }
            }
        }
        else
```

```
                    {
                        if(xb>xc)
                        {
                            if(yc>yb)
                            {
                                if(ya==yc)
                                    return 8;
                                else
                                    return 4;
                            }
                            else
                            {
                                if(yb==yc)
                                    return 9;
                                else
                                    return 2;
                            }
                        }
                        else
                        {
                            if(yb<yc)
                                return 7;
                            else
                                return 3;
                        }
                    }
                }
            }
        }
    }
}
inline int lectura()
{
        int i, salida=0;
        for(i=0; i<iteraciones; i++)
        {
                salida += EYE_VALUE;
        }
        salida /= iteraciones;
        return salida;
}

sub retardo(int tiempo){
        for(int i =0;i<tiempo*20;i++){
                int j=2;
        }
}

sub ve_a_pos(int destino){
        if(destino==pos)
                return;
        if(pos==-1)
        {
                OnFwd(OUT_A, potencia);
                int luz_old = lectura();
                int luz;
                while(true)
                {
                        luz = lectura();
                        if(luz_old+95 <= luz)
                        {
                                Off(OUT_A);
                                retardo(1000);
                                break;
                        }
                        luz_old = luz;
                        retardo(50);
                }
                pos=0;
                return;
        }
        for(int i=0;i<destino;i++)
        {
                OnFwd(OUT_A, potencia);
                int luz_old = lectura();
                int luz;
                while(true)
```

```
		{
			luz = lectura();
			if(luz_old+95 <= luz)
			{
				Off(OUT_A);
				retardo(1000);
				break;
			}
			luz_old = luz;
			retardo(50);
		}
	}
	pos=destino;
}

task main() {
	long first_tick = CurrentTick();
	string msg;
	char enemies[7];
	pos=-1;
	SetSensorType(S2, SENSOR_TYPE_LIGHT_ACTIVE);
	SetSensorMode(S2, SENSOR_MODE_RAW);
	ResetSensor(S2);
	ve_a_pos(0);
	while(ReceiveMessage(NXTBOX,true,enemies));
	if(enemies[1]==1)
		pa = enemies[2];
	else
		pa = -1;
	if(enemies[3]==1)
		pb = enemies[4];
	else
		pb = -1;
	if(enemies[5]==1)
		pc = enemies[6];
	else
		pc = -1;
	KeepAliveType kaArgs;
	grado_act = 0;
	//COMIENZA ALGORITMO
	//DECLARACION DE VARIABLES
	int ruta_a[4], ruta_b[4], ruta_c[4], pd, aux, fila, dir, column, ini, sent, inix,
incr, unos, valor,vector[81];
	//Insertamos los obstaculos
	vector[((9*obs_ax)+obs_ay)]=-1;
	vector[((9*obs_bx)+obs_by)]=-1;
	vector[((9*obs_cx)+obs_cy)]=-1;
	vector[((9*obs_dx)+obs_dy)]=-1;
    //Calcular trayectorias
	for(int i=0;i<81;i++) vector[i]=0;
    for(int j=0; j<3; j++)
    {
        switch(j)
        {
        case 0:
            sent = 1;
            ini = pa;
            inix = 0;
                    valor = pa;
            break;
        case 1:
            sent = -1;
            ini = pb;
            inix = 8;
                    valor = pb;
            break;
        case 2:
            sent = 1;
            ini = 0;
            inix = pc;
                    valor = pb;
            break;
        }
           if(valor == -1)
                    continue;
        for(int i=0;i<3;i++)
           {
```

```
        switch(i)
        {
        case 0:
            dir = -1;
            break;
        case 1:
            dir = 0;
            break;
        case 2:
            dir = 1;
            break;
        }
        column = (j!=2)?ini*2+1+dir:0;
        fila = (j!=2)?inix:inix*2+1+dir;
        aux = vector[(fila*9)+column];
        while((aux!=-1)&&(fila<9)&&(column<9)&&(column>-1)&&(fila>-1))
        {
            vector[(fila*9)+column]=vector[(fila*9)+column]+1;
                        fila+=(j!=2)?sent:dir;
            column+=(j!=2)?dir:sent;
                        if((fila<9)&&(column<9)&&(column>-1)&&(fila>-1))
                                aux = vector[(fila*9)+column];
        }
    }
}
for(int j=1; j<9; j+=2)
{
    int max = 1;
    int vmax;
    for(int i=1; i<9; i+=2)
        if(vector[(j*9)+max]<vector[(j*9)+i])
            max = i;
    vmax = vector[(j*9)+max];
    for(int i=1; i<9; i+=2)
    {
        if(vmax==vector[(j*9)+i])
                        vector[(j*9)+i]=1;
        else
                        vector[(j*9)+i]=0;
    }
}
for(int j=1; j<9; j+=2)
{
    for(int i=0; i<9; i+=2)
                vector[(j*9)+i]=0;
}
for(int j=0; j<9; j+=2)
{
    for(int i=0; i<9; i++)
                vector[(j*9)+i]=0;
}
   vector[((9*obs_ax)+obs_ay)]=-1;
   vector[((9*obs_bx)+obs_by)]=-1;
   vector[((9*obs_cx)+obs_cy)]=-1;
   vector[((9*obs_dx)+obs_dy)]=-1;
for(int j=1; j<9; j+=2)
{
    unos = 0;
    for(int i=1; i<9; i+=2)
        if(vector[(j*9)+i] == 1)
            unos++;
    if(unos > 1)
    {
        int max = 1;
        int nconext=0, nconex, winner;
        for(int i=1; i<9; i+=2)
        {
            nconex = 0;
            if(vector[(j*9)+i] == 1)
            {
                fila = j;
                column = i;
                aux = vector[(fila*9)+column];
                for(int k=0; k<6; k++)
                {
                    fila = j;
                    column = i;
```

```
                        switch(k)
                        {
                        case 0:
                            sent = -1;
                            dir = -1;
                            break;
                        case 1:
                            sent = -1;
                            dir = 0;
                            break;
                        case 2:
                            sent = -1;
                            dir = 1;
                            break;
                        case 3:
                            sent = 1;
                            dir = 1;
                            break;
                        case 4:
                            sent = 1;
                            dir = 0;
                            break;
                        case 5:
                            sent = 1;
                            dir = -1;
                            break;
                        }
                        while((aux!=-1)&&(fila<9)&&(column<9)&&(column>-1)&&(fila>-1))
                        {
                            fila += sent;
                            column += dir;
                                                    if((fila<9)&&(column<9)&&(column>-
1)&&(fila>-1))
                                                            aux                             =
vector[(fila*9)+column];
                            if(aux == 1)
                            {
                                nconex++;
                                break;
                            }
                        }
                    }
                    if(nconex>nconext)
                    {
                                                winner = i;
                        nconext = nconex;
                    }
                }
            }
            for(int i=1; i<9; i+=2)
                if(i!=winner)
                                        vector[(j*9)+i]=0;
        }
    }
    for(int j=1; j<9; j+=2)
    {
        for(int i=1; i<9; i+=2)
        {
            int nconex = 0;
            if(vector[(j*9)+i] == 1)
            {
                for(int k=0; k<6; k++)
                {
                    fila = j;
                    column = i;
                    aux = vector[(fila*9)+column];
                    switch(k)
                    {
                    case 0:
                        sent = -1;
                        dir = -1;
                        break;
                    case 1:
                        sent = -1;
                        dir = 0;
                        break;
                    case 2:
```

```
                    sent = -1;
                    dir = 1;
                    break;
                case 3:
                    sent = 1;
                    dir = 1;
                    break;
                case 4:
                    sent = 1;
                    dir = 0;
                    break;
                case 5:
                    sent = 1;
                    dir = -1;
                    break;
                }
                fila += sent;
                column += dir;
                while((aux!=-1)&&(fila<9)&&(column<9)&&(column>-1)&&(fila>-1))
                {
                                        aux = vector[(fila*9)+column];
                    fila += sent;
                    column += dir;
                    if(aux >= 1)
                    {
                        nconex++;
                        break;
                    }
                }
            }
                         vector[(j*9)+i]=nconex;
        }
    }
}
inix = 0;
int iniy = 0;
for(int j=1; j<9; j+=2)
{
    for(int i=1; i<9; i+=2)
    {
        if(vector[(j*9)+i]>vector[(iniy*9)+inix])
        {
            if(!(((obs_ay==j)&&(obs_ax==8))||((obs_by==j)&&(obs_bx==8))||
                 (((obs_cy==j))&&(obs_cx==8))||((obs_dy==j)&&(obs_dx==8))))
            {
                inix = i;
                iniy = j;
            }
            break;
        }
    }
}
pd = (iniy-1)/2;
for(int i=1; i<9; i+=2)
{
    if(vector[(9)+i]>0)
    {
        e_d = (i-1)/2;
        break;
    }
}
for(int i=1; i<9; i+=2)
{
    if(vector[(27)+i]>0)
    {
        e_b = (i-1)/2;
        break;
    }
}
for(int i=1; i<9; i+=2)
{
    if(vector[(45)+i]>0)
    {
        e_a = (i-1)/2;
        break;
    }
}
```

```
for(int i=1; i<9; i+=2)
{
    if(vector[(63)+i]>0)
    {
        e_c = (i-1)/2;
        break;
    }
}
// Calculamos el ataque a la plataforma Espejo A
   for(int w=0; w<3; w++)
   {
          if(((w==0)&&(pa==-1))||((w==1)&&(pb==-1))||((w==2)&&(pc==-1)))
                 continue;
          int atacantes[4]={-1,-1,-1,-1};
          int t_atacantes = 0, objx;
          switch(w)
          {
          case 0:
                 objx = pa;
                 for(int j=0; j<3; j++)
                 {
                        fila = 0;
                        switch(j)
                        {
                        case 0:
                               dir = -1;
                               column = pa*2;
                               break;
                        case 1:
                               dir = 0;
                               column = pa*2+1;
                               break;
                        case 2:
                               dir = 1;
                               column = pa*2+2;
                               break;
                        }
                        aux = vector[column];
                        fila+=1;
                        column+=dir;
                        while((aux!=-1)&&(fila<9)&&(column<9)&&(column>-1))
                        {
                               if(aux>=1)
                               {
                                      atacantes[t_atacantes]=fila-1;
                                      t_atacantes++;
                               }
                               aux = vector[(fila*9)+column];
                               fila+=1;
                               column+=dir;
                        }
                 }
                 break;
          case 1:
                 objx = pb;
                 for(int j=0; j<3; j++)
                 {
                        fila = 8;
                        switch(j)
                        {
                        case 0:
                               dir = -1;
                               column = pb*2;
                               break;
                        case 1:
                               dir = 0;
                               column = pb*2+1;
                               break;
                        case 2:
                               dir = 1;
                               column = pb*2+2;
                               break;
                        }
                        aux = vector[(72)+column];
                        fila--;
                        column+=dir;
                        while((aux!=-1)&&(column<9)&&(column>-1)&&(fila>-1))
```

```
                {
                        if(aux>=1)
                        {
                                atacantes[t_atacantes]=fila+1;
                                t_atacantes++;
                        }
                        aux = vector[(fila*9)+column];
                        fila--;
                        column+=dir;
                }
        }
        break;
case 2:
        objx = pc;
        for(int j=0; j<3; j++)
        {
                column = 0;
                switch(j)
                {
                case 0:
                        sent = -1;
                        fila = pc*2;
                        incr = -1;
                        break;
                case 1:
                        sent = 0;
                        fila = pc*2+1;
                        incr = 0;
                        break;
                case 2:
                        sent = 1;
                        fila = pc*2+2;
                        incr = 1;
                        break;
                }
                aux = vector[(fila*9)];
                fila+=sent;
                column++;
                while((aux!=-1)&&(fila<9)&&(column<9)&&(fila>-1))
                {
                        if(aux>=1)
                        {
                                atacantes[t_atacantes++]=fila-incr;
                                //t_atacantes++;
                        }
                        aux = vector[(fila*9)+column];
                        fila+=sent;
                        column++;
                }
        }
        break;
}
for(int i=0; i<t_atacantes; i++)
{
        if(pd*2+1==atacantes[i])
        {
                t_atacantes--;
                if(t_atacantes!=0)
                        atacantes[i]=-1;
                break;
        }
}
for(int j=1; j<9; j+=2)
{
        int colu = 0;
        for(int i=1; i<9; i+=2)
        {
                if(vector[(j*9)+i]>=1)
                {
                        colu = i;
                        break;
                }
        }
        for(int k=0; k<3; k++)
        {
                fila = j+1;
                sent = 1;
```

```
                                switch(k)
                                {
                                case 0:
                                        dir = -1;
                                        column = colu-1;
                                        break;
                                case 1:
                                        dir = 0;
                                        column = colu;
                                        break;
                                case 2:
                                        dir = 1;
                                        column = colu+1;
                                        break;
                                }
                                aux = vector[(fila*9)+column];
                                while((aux!=-1)&&(fila<9)&&(column<9)&&(column>-1)&&(fila>-
1))
                                {
                                        if(aux>=1)
                                        {
                                                switch((j-1)/2)
                                                {
                                                case 0:
                                                        switch((fila-1)/2)
                                                        {
                                                        case 1:
                                                                conec_d[1]=1;
                                                                conec_b[2]=1;
                                                                break;
                                                        case 2:
                                                                conec_d[0]=1;
                                                                conec_a[2]=1;
                                                                break;
                                                        case 3:
                                                                conec_d[2]=1;
                                                                conec_c[2]=1;
                                                                break;
                                                        }
                                                        break;
                                                case 1:
                                                        switch((fila-1)/2)
                                                        {
                                                        case 2:
                                                                conec_b[0]=1;
                                                                conec_a[0]=1;
                                                                break;
                                                        case 3:
                                                                conec_b[1]=1;
                                                                conec_c[1]=1;
                                                                break;
                                                        }
                                                        break;
                                                case 2:
                                                        switch((fila-1)/2)
                                                        {
                                                        case 3:
                                                                conec_a[1]=1;
                                                                conec_c[0]=1;
                                                                break;
                                                        }
                                                        break;
                                                }
                                        }
                                        fila+=sent;
                                        column+=dir;
                                        if((fila==9)||(fila==-1)||(column==9)||(column==-
1))
                                                break;
                                        aux = vector[(fila*9)+column];
                                }
                        }
                }
                for(int i=0; i< 4; i++)
                        atacantes[i] = (atacantes[i]-1)/2;
                int atac_pred=0;
                bool nochoca[4];
```

```
		if(t_atacantes==0)
		{
			// Atacamos con el espejo que recibe el rayo del Maestro
			atac_pred = -1;
		}
		else
		{
			if(t_atacantes==1)
			{
				// Atacaremos con el ¨nico atacante
				atac_pred = atacantes[0];
				if(atac_pred==-1)
				{
					atac_pred = atacantes[1];
				}

	if(((dame_pos(atac_pred)==dame_pos(pd))&&(dame_pos(atac_pred)==objx)))
					atac_pred = -1;
			}
			else
			{
				// Atacaremos con uno de los espejos a decidir
				for(int i=0; i<t_atacantes; i++)
				{
					if(atacantes[i]==-1)
						continue;
					for(int j=i+1; j<t_atacantes; j++)
					{
						if(atacantes[j]==-1)
							continue;
						else
						{
							if(n_conex(i)>n_conex(j))
							{
								if(conectados(atacantes[i],
atacantes[j]))
								{

	if(((dame_pos(atacantes[j])==objx)&&(dame_pos(atacantes[i])==objx)))

	nochoca[atacantes[j]] = true;
									atac_pred            =
atacantes[i];
									continue;
								}
								else
									atac_pred            =
atacantes[j];
							}
							else
							{
								if(conectados(atacantes[i],
atacantes[j]))
								{

	if(((dame_pos(atacantes[j])==objx)&&(dame_pos(atacantes[i])==objx)))

	nochoca[atacantes[i]] = true;
									atac_pred            =
atacantes[j];
									continue;
								}
								atac_pred = atacantes[i];
							}
						}
					}
				}
			}
		}
		// Creamos la ruta
		int ruta[4]={-1, -1, -1, -1};
		t_atacantes = 0;
		int auxi[2] = {-1, -1};
		ruta[0] = pd;

		for(int i=0; i<4; i++)
		{
```

```
                        if(i!=pd)
                        {
                                if((i==atac_pred)||(nochoca[i]))
                                {
                                        if(nochoca[i])
                                        {
                                                if(ruta[3]!=-1)
                                                {
                                                        if(ruta[2]!=-1)
                                                                ruta[1] = ruta[2];
                                                        ruta[2] = ruta[3];
                                                        ruta[3] = i;
                                                }
                                                else
                                                        ruta[3] = i;
                                        }
                                        else
                                        {

        if((dame_pos(atac_pred)==dame_pos(pd))&&(dame_pos(pd)==objx))
                                                {
                                                        ruta[3]=-1;
                                                        continue;
                                                }
                                                if(ruta[3]!=-1)
                                                {

        if((dame_pos(ruta[3])==dame_pos(pd))&&(dame_pos(i)==dame_pos(pd)))
                                                        {
                                                                ruta[2] = -1;
                                                        }
                                                        else
                                                        {
                                                                if(ruta[3]!=-1)
                                                                {
                                                                        if(ruta[2]!=-1)
                                                                                ruta[1]       =
ruta[2];
                                                                        ruta[2] = i;
                                                                }
                                                        }
                                                }
                                                else
                                                        ruta[3] = i;
                                        }
                                }
                                else
                                {
                                        auxi[t_atacantes] = i;
                                        t_atacantes++;
                                }
                        }
                }
                if(atac_pred!=-1)
                {
                        if(t_atacantes==1)
                        {
                                if((conectados(pd,          auxi[0]))&&(conectados(auxi[0],
ruta[2])))
                                {

        if((dame_pos(pd)==dame_pos(auxi[0]))&&(dame_pos(pd)==dame_pos(atac_pred)))
                                                ruta[1] = -1;
                                        else
                                                ruta[1] = auxi[0];
                                }
                                else
                                        ruta[1] = -1;
                        }
                        else
                        {
                                if(conectados(pd,auxi[0]))
                                {
                                        if(conectados(auxi[0], auxi[1]))
                                        {
                                                if(conectados(atac_pred, auxi[1]))
                                                {
```

```
        if((dame_pos(atac_pred)==dame_pos(auxi[1]))&&(dame_pos(auxi[1])==dame_pos(pd)))
                                                {
                                                        ruta[1] = -1;
                                                        ruta[2] = -1;
                                                }
                                                else
                                                {
                                                        ruta[1] = auxi[0];
                                                        ruta[2] = auxi[1];
                                                }
                                        }
                                        else
                                        {
                                                if(conectados(auxi[0], atac_pred))
                                                {

        if((dame_pos(atac_pred)==dame_pos(auxi[0]))&&(dame_pos(auxi[0])==dame_pos(pd)))
                                                                ruta[1] = -1;
                                                        else
                                                                ruta[1] = auxi[0];
                                                }
                                                else
                                                {
                                                        ruta[1] = -1;
                                                }
                                                ruta[2] = -1;
                                        }
                                }
                                else
                                {
                                        if(conectados(atac_pred, auxi[0]))
                                        {

        if((dame_pos(atac_pred)==dame_pos(auxi[0]))&&(dame_pos(pd)==dame_pos(auxi[0])))
                                                        ruta[1] = -1;
                                                else
                                                        ruta[1] = auxi[0];
                                        }
                                        else
                                        {
                                                if((conectados(atac_pred,
auxi[1]))&&(conectados(pd, auxi[1])))
                                                        ruta[1] = auxi[1];
                                                else
                                                        ruta[1] = -1;
                                        }
                                        ruta[2] = -1;
                                }
                        }
                        else
                        {
                                if(conectados(pd, auxi[1]))
                                {
                                        if(conectados(atac_pred, auxi[0]))
                                        {
                                                ruta[1] = auxi[1];
                                                ruta[2] = auxi[0];
                                        }
                                        else
                                        {
                                                if(conectados(atac_pred, auxi[1]))
                                                        ruta[1] = auxi[1];
                                                else
                                                        ruta[1] = -1;
                                                ruta[2] = -1;
                                        }
                                }
                                else
                                {
                                        ruta[1] = -1;
                                        ruta[2] = -1;
                                }
                        }
                }
        }
        else
```

```
		{
			ruta[1] = -1;
			ruta[2] = -1;
			ruta[3] = -1;
		}
		if(ruta[3]==-1)
		{
			ruta[3] = ruta[2];
			ruta[2] = -1;
		}
		switch(w)
		{
		case 0:
			ruta_a[0] = ruta[0];
			ruta_a[1] = ruta[1];
			ruta_a[2] = ruta[2];
			ruta_a[3] = ruta[3];
			break;
		case 1:
			ruta_b[0] = ruta[0];
			ruta_b[1] = ruta[1];
			ruta_b[2] = ruta[2];
			ruta_b[3] = ruta[3];
			break;
		case 2:
			ruta_c[0] = ruta[0];
			ruta_c[1] = ruta[1];
			ruta_c[2] = ruta[2];
			ruta_c[3] = ruta[3];
			break;
		}
	}
	if(pa!=-1)
	{
		pon_ang(0, 4, 0);
		pon_ang(1, 4, 0);
		pon_ang(2, 4, 0);
		pon_ang(3, 4, 0);
		if(ruta_a[1] == -1)
		{
			if(ruta_a[2] == -1)
			{
				if(ruta_a[3]==-1)
				{
					pon_ang(ruta_a[0],	halla_ang(4,	pd,
dame_pos(ruta_a[0]), pd, pa, -1), 0);
				}
				else
				{
					pon_ang(ruta_a[0],	halla_ang(4,	pd,
dame_pos(ruta_a[0]), pd, dame_pos(ruta_a[3]), ruta_a[3]), 0);
					pon_ang(ruta_a[3],	halla_ang(dame_pos(ruta_a[0]),
ruta_a[0], dame_pos(ruta_a[3]), ruta_a[3], pa, -1), 0);
				}
			}
			else
			{
				pon_ang(ruta_a[0],  halla_ang(4,  pd,  dame_pos(ruta_a[0]),
pd, dame_pos(ruta_a[2]), ruta_a[2]), 0);
				pon_ang(ruta_a[2],	halla_ang(dame_pos(ruta_a[0]),
ruta_a[0], dame_pos(ruta_a[2]), ruta_a[2], dame_pos(ruta_a[3]), ruta_a[3]), 0);
				pon_ang(ruta_a[3],	halla_ang(dame_pos(ruta_a[2]),
ruta_a[2], dame_pos(ruta_a[3]), ruta_a[3], pa, -1), 0);
			}
		}
		else
		{
			pon_ang(ruta_a[0],  halla_ang(4,  pd,  dame_pos(ruta_a[0]),  pd,
dame_pos(ruta_a[1]), ruta_a[1]), 0);
			if(ruta_a[2]==-1)
			{
				pon_ang(ruta_a[1],	halla_ang(dame_pos(ruta_a[0]),	pd,
dame_pos(ruta_a[1]), ruta_a[1], dame_pos(ruta_a[3]), ruta_a[3]), 0);
				pon_ang(ruta_a[3],	halla_ang(dame_pos(ruta_a[1]),
ruta_a[1], dame_pos(ruta_a[3]), ruta_a[3], pa, -1), 0);
			}
			else
```

```
                        {
                                pon_ang(ruta_a[1],    halla_ang(dame_pos(ruta_a[0]),    pd,
dame_pos(ruta_a[1]), ruta_a[1], dame_pos(ruta_a[2]), ruta_a[2]), 0);
                                pon_ang(ruta_a[2],          halla_ang(dame_pos(ruta_a[1]),
ruta_a[1], dame_pos(ruta_a[2]), ruta_a[2], dame_pos(ruta_a[3]), ruta_a[3]), 0);
                                pon_ang(ruta_a[3],          halla_ang(dame_pos(ruta_a[2]),
ruta_a[2], dame_pos(ruta_a[3]), ruta_a[3], pa, -1), 0);
                        }
                }
        }
        if(pb!=-1)
        {
                pon_ang(0, 4, 1);
                pon_ang(1, 4, 1);
                pon_ang(2, 4, 1);
                pon_ang(3, 4, 1);
                if(ruta_b[1] == -1)
                {
                        if(ruta_b[2] == -1)
                        {
                                if(ruta_b[3] == -1)
                                {
                                        pon_ang(ruta_b[0],         halla_ang(4,         pd,
dame_pos(ruta_b[0]), pd, pb, 4), 1);
                                }
                                else
                                {
                                        pon_ang(ruta_b[0],         halla_ang(4,         pd,
dame_pos(ruta_b[0]), pd, dame_pos(ruta_b[3]), ruta_b[3]), 1);
                                        pon_ang(ruta_b[3],    halla_ang(dame_pos(ruta_b[0]),
ruta_b[0], dame_pos(ruta_b[3]), ruta_b[3], pb, 4), 1);
                                }
                        }
                        else
                        {
                                pon_ang(ruta_b[0],  halla_ang(4,  pd,  dame_pos(ruta_b[0]),
pd, dame_pos(ruta_b[2]), ruta_b[2]), 1);
                                pon_ang(ruta_b[2],          halla_ang(dame_pos(ruta_b[0]),
ruta_b[0], dame_pos(ruta_b[2]), ruta_b[2], dame_pos(ruta_b[3]), ruta_b[3]), 1);
                                pon_ang(ruta_b[3],          halla_ang(dame_pos(ruta_b[2]),
ruta_b[2], dame_pos(ruta_b[3]), ruta_b[3], pb, 4), 1);
                        }
                }
                else
                {
                        pon_ang(ruta_b[0],  halla_ang(4,  pd,  dame_pos(ruta_b[0]),  pd,
dame_pos(ruta_b[1]), ruta_b[1]), 1);
                        if(ruta_b[2]==-1)
                        {
                                pon_ang(ruta_b[1],    halla_ang(dame_pos(ruta_b[0]),    pd,
dame_pos(ruta_b[1]), ruta_b[1], dame_pos(ruta_b[3]), ruta_b[3]), 1);
                                pon_ang(ruta_b[3],          halla_ang(dame_pos(ruta_b[1]),
ruta_b[1], dame_pos(ruta_b[3]), ruta_b[3], pb, 4), 1);
                        }
                        else
                        {
                                pon_ang(ruta_b[1],    halla_ang(dame_pos(ruta_b[0]),    pd,
dame_pos(ruta_b[1]), ruta_b[1], dame_pos(ruta_b[2]), ruta_b[2]), 1);
                                pon_ang(ruta_b[2],          halla_ang(dame_pos(ruta_b[1]),
ruta_b[1], dame_pos(ruta_b[2]), ruta_b[2], dame_pos(ruta_b[3]), ruta_b[3]), 1);
                                pon_ang(ruta_a[3],          halla_ang(dame_pos(ruta_b[2]),
ruta_b[2], dame_pos(ruta_b[3]), ruta_b[3], pb, 4), 1);
                        }
                }
        }
        if(pc!=-1)
        {
                pon_ang(0, 4, 2);
                pon_ang(1, 4, 2);
                pon_ang(2, 4, 2);
                pon_ang(3, 4, 2);
                if(ruta_c[1] == -1)
                {
                        if(ruta_c[2] == -1)
                        {
                                if(ruta_c[3]==-1)
                                {
```

```
                                        pon_ang(ruta_c[0],          halla_ang(4,          pd,
dame_pos(ruta_c[0]), pd, -1, pc), 2);
                                }
                                else
                                {
                                        pon_ang(ruta_c[0],          halla_ang(4,          pd,
dame_pos(ruta_c[0]), pd, dame_pos(ruta_c[3]), ruta_c[3]), 2);
                                        pon_ang(ruta_c[3],    halla_ang(dame_pos(ruta_c[0]),
ruta_c[0], dame_pos(ruta_c[3]), ruta_c[3], -1, pc), 2);
                                }
                        }
                        else
                        {
                                pon_ang(ruta_c[0],  halla_ang(4,  pd,  dame_pos(ruta_c[0]),
pd, dame_pos(ruta_c[2]), ruta_c[2]), 2);
                                pon_ang(ruta_c[2],           halla_ang(dame_pos(ruta_c[0]),
ruta_c[0], dame_pos(ruta_c[2]), ruta_c[2], dame_pos(ruta_c[3]), ruta_c[3]), 2);
                                pon_ang(ruta_c[3],           halla_ang(dame_pos(ruta_c[2]),
ruta_c[2], dame_pos(ruta_c[3]), ruta_c[3], -1, pc), 2);
                        }
                }
                else
                {
                        pon_ang(ruta_c[0],   halla_ang(4,   pd,   dame_pos(ruta_c[0]),   pd,
dame_pos(ruta_c[1]), ruta_c[1]), 2);
                        if(ruta_c[2]==-1)
                        {
                                if(ruta_c[3]==-1)
                                {
                                        pon_ang(ruta_c[1],    halla_ang(dame_pos(ruta_c[0]),
pd, dame_pos(ruta_c[0]), pd, -1, pc), 0);
                                }
                                else
                                {
                                        pon_ang(ruta_c[1],    halla_ang(dame_pos(ruta_c[0]),
pd, dame_pos(ruta_c[1]), ruta_c[1], dame_pos(ruta_c[3]), ruta_c[3]), 2);
                                        pon_ang(ruta_c[3],    halla_ang(dame_pos(ruta_c[1]),
ruta_c[1], dame_pos(ruta_c[3]), ruta_c[3], -1, pc), 2);
                                }
                        }
                        else
                        {
                                pon_ang(ruta_c[1],    halla_ang(dame_pos(ruta_c[0]),    pd,
dame_pos(ruta_c[1]), ruta_c[1], dame_pos(ruta_c[2]), ruta_c[2]), 2);
                                pon_ang(ruta_c[2],           halla_ang(dame_pos(ruta_c[1]),
ruta_c[1], dame_pos(ruta_c[2]), ruta_c[2], dame_pos(ruta_c[3]), ruta_c[3]), 2);
                                pon_ang(ruta_a[3],           halla_ang(dame_pos(ruta_c[2]),
ruta_c[2], dame_pos(ruta_c[3]), ruta_c[3], -1, pc), 2);
                        }
                }
        }
        //SE PREPARA TODO PARA ENVIAR AL SERVIDOR
        for(fila=0;fila<3;fila++)
        {
                for(int i=0;i<4;i++)
                {
                        char rist[3];
                        switch(i){
                                case 3:  //Espejo D
                                        rist[0] = i;
                                        rist[1] = e_d;
                                        rist[2] = ang_d[fila];
                                        break;
                                case 1:  //Espejo B
                                        rist[0] = i;
                                        rist[1] = e_b;
                                        rist[2] = ang_b[fila];
                                        break;
                                case 0:  //Espejo A
                                        rist[0] = i;
                                        rist[1] = e_a;
                                        rist[2] = ang_a[fila];
                                        break;
                                case 2:  //Espejo C
                                        rist[0] = i;
                                        rist[1] = e_c;
                                        rist[2] = ang_c[fila];
```

```
                                        break;
                        }
                        SendMessage(SERVERBOX, rist);
                        while((ReceiveMessage(NXTBOX,true,msg) == 64))
                        {
                                SysKeepAlive(kaArgs);
                        }
                }
                first_tick = CurrentTick();
                while(((CurrentTick()     -     first_tick)     <     CONN_TIMEOUT)     &&
(ReceiveMessage(NXTBOX,true,msg) == 64))
                {
                        SysKeepAlive(kaArgs);
                }
                ve_a_pos(pd);
                OnFwd(OUT_B,20);
                while(!(ButtonPressed(BTNCENTER, false)))
                {
                        SysKeepAlive(kaArgs);
                }
                //APAGA
                Off(OUT_B);
        }
}
```

Código espejo:

```
// PARAMETROS
#define potencia 34 // potencia del motor
#define iteraciones 10 // n√∫mero de iteraciones en las lecturas
#define SERVERBOX 1
#define NXTBOX 0

// DEFINICIONES
#define EYE_VALUE     SENSOR_2

// VARIABLES GLOBALES
mutex motor;
int pos;
int grado_act;

inline int lectura()
{
	int i, salida=0;
	for(i=0; i<iteraciones; i++)
	{
		salida += EYE_VALUE;
	}
	salida /= iteraciones;
	return salida;
}

inline void dispara(){
  PlayTone(200,200);
	OnFwd(OUT_B,20);
	Wait(9000);
}

inline void apaga(){
	OnRev(OUT_B,50);
	Wait(100);
	Off(OUT_B);
}

sub ve_a_pos(int destino){

	if(destino==pos) return;
	if(destino<pos){
		for(int i=0;i<pos-destino;i++)
		{
			OnRev(OUT_A, potencia-8);
			int luz_old = lectura();
			int luz;
			while(true)
			{
				string salida = NumToStr(luz_old);
				salida = StrCat("Luz ant = ", salida);
				TextOut(0, LCD_LINE1, salida);
				luz = lectura();
				salida = NumToStr(luz);
				salida = StrCat("Luz act = ", salida);
				TextOut(0, LCD_LINE2, salida);
				if(luz_old+100 <= luz)
				{
					PlayTone(100,100);
					Off(OUT_A);
					Wait(1000);
					break;
				}
				luz_old = luz;
				Wait(10);
```

```
                        }
                }
        }
        else
        {
                for(int i=0;i<destino-pos;i++)
                {
                        OnFwd(OUT_A, potencia);
                        int luz_old = lectura();
                        int luz;
                        while(true)
                        {
                                string salida = NumToStr(luz_old);
                                salida = StrCat("Luz ant = ", salida);
                                TextOut(0, LCD_LINE1, salida);
                                luz = lectura();
                                salida = NumToStr(luz);
                                salida = StrCat("Luz act = ", salida);
                                TextOut(0, LCD_LINE2, salida);
                                if(luz_old+100 <= luz)
                                {
                                        PlayTone(100,100);
                                        Off(OUT_A);
                                        Wait(1000);
                                        break;
                                }
                                luz_old = luz;
                                Wait(10);
                        }
                }
        }
        pos=destino;
}

sub rota_Espejo(int grado_pos)
{
        ResetRotationCount(OUT_B);
        if(grado_act==grado_pos)
                return;
        int angulo=(grado_pos-grado_act);
        RotateMotor(OUT_B,20,angulo*(-90));
        grado_act=grado_pos;

}

task main() {
        SetSensorType(S2, SENSOR_TYPE_LIGHT_ACTIVE);
        SetSensorMode(S2, SENSOR_MODE_RAW);
        grado_act = 0;
        pos=0;
        ResetSensor(S2);
        while(true){
                string msg;
                while( ReceiveMessage(NXTBOX,true,msg));
                ResetScreen();
                SendMessage(SERVERBOX, "0");
                ve_a_pos(msg[0]);
                rota_Espejo(msg[1]);

                TextOut(0, LCD_LINE1, "Posición: ");
                NumOut(30, LCD_LINE1, pos);
                TextOut(0, LCD_LINE2, "Espejo: ");
                NumOut(30, LCD_LINE2, grado_act);

                SendMessage(SERVERBOX, "FINIQUITADO");
        }
}
```