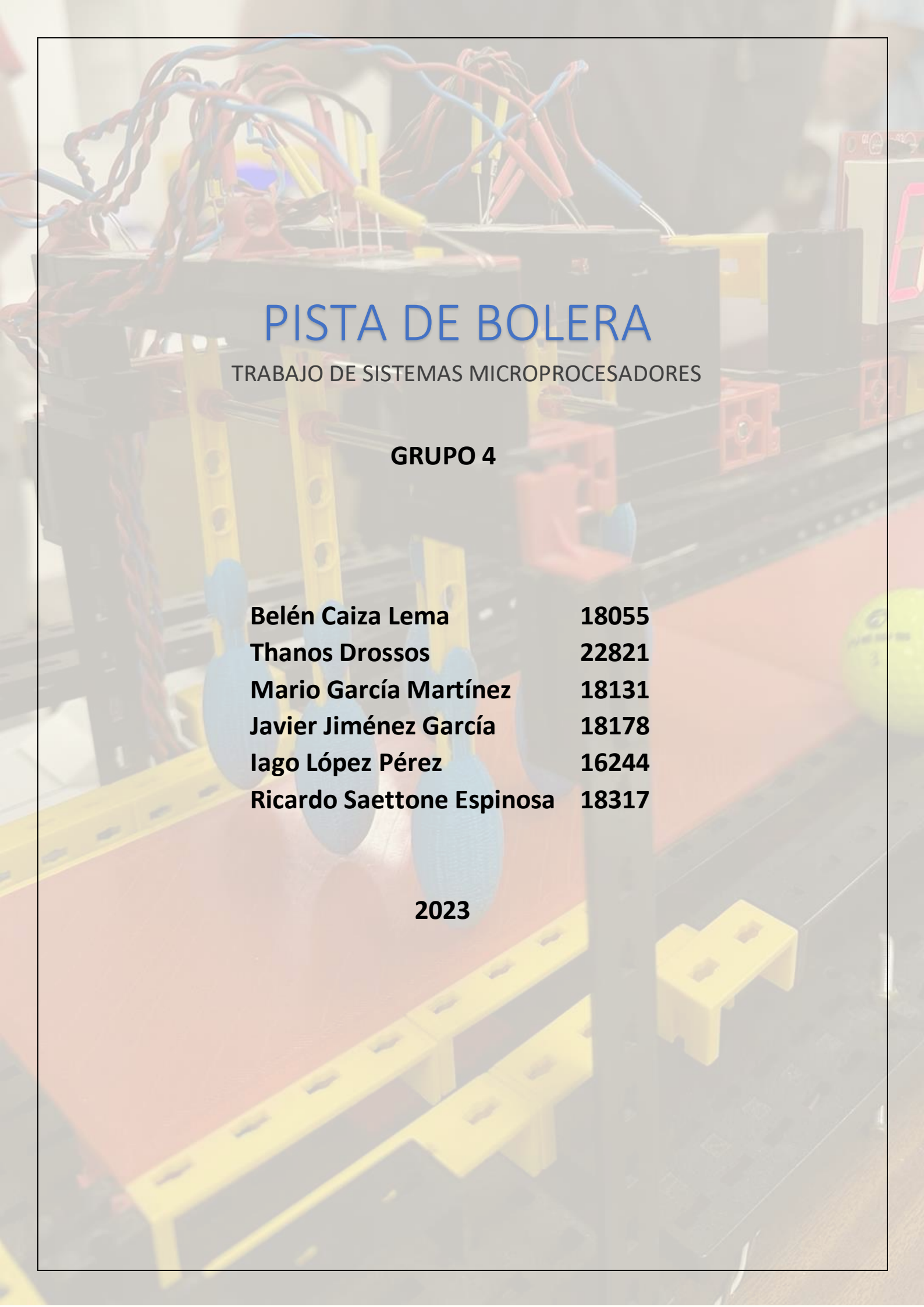


PISTA DE BOLERA

TRABAJO DE SISTEMAS MICROPROCESADORES

GRUPO 4

Belén Caiza Lema	**18055**
Thanos Drossos	**22821**
Mario García Martínez	**18131**
Javier Jiménez García	**18178**
Iago López Pérez	**16244**
Ricardo Saettone Espinosa	**18317**

2023

ÍNDICE

Enlace del código: https://github.com/Thanos002/Bolera_Full_v2

1 PLANTEAMIENTO DE LA SOLUCIÓN

La maqueta de la bolera se puede resolver de forma cómoda a través de una máquina de estados. En nuestro caso hemos desarrollado una máquina de estados que sigue un planteamiento secuencial, considerando que es la opción más adecuada para el funcionamiento que tiene la bolera.

1.1 DESCRIPCIÓN DE LA SOLUCIÓN

En primer lugar, hemos utilizado dos timers. El timer cero lo hemos configurado para que interrumpta temporalmente cada 5 milisegundos por comparación. Principalmente sirve para manejar el funcionamiento de los displays y su tasa de refresco. Por otro lado el timer uno lo hemos preparado para que interrumpa temporalmente cada 3 segundos, también por comparación. Lo utilizamos principalmente para el control de la duración del juego y para tareas que requieran el uso de tiempos largos.

Los displays que muestran la puntuación funcionan con un refresco temporal de 5 milisegundos, que alterna la señal del selector y permiten observar con claridad la puntuacion. Complementario a este refresco se desarrollan otras funciones que permiten la representación de qué número en función de la puntuación como variable global. Para el manejo de los displays y la señal del selector utilizamos el puerto B.

Los bolos permiten el control de la puntuación anteriormente comentada. Estos funcionan con el grupo PCINT2, utilizando concretamente los PCINT 16 a PCINT 21. Estas interrupciones se habilitan al pulsar la señal de disparo y se deshabilitan mediante una máscara que se aplica en su propia interrupción al ser tocado un bolo.

Los motores funcionan principalmente con el puerto L y algunos bits de los puertos D y K. Se les habilita con la señal ‘enable’ y se les indica su sentido con la señal ‘dirección’.

Por otro lado, las señales de los fines de carrera se manejan con el puerto D. El botón de disparo (SW6) y los que gobiernan la posición del lanzador (SW2) se manejan mediante interrupciones, el resto como entradas normales.

Para el último elemento a manejar, el led del lanzador, se emplea un bit del puerto L.

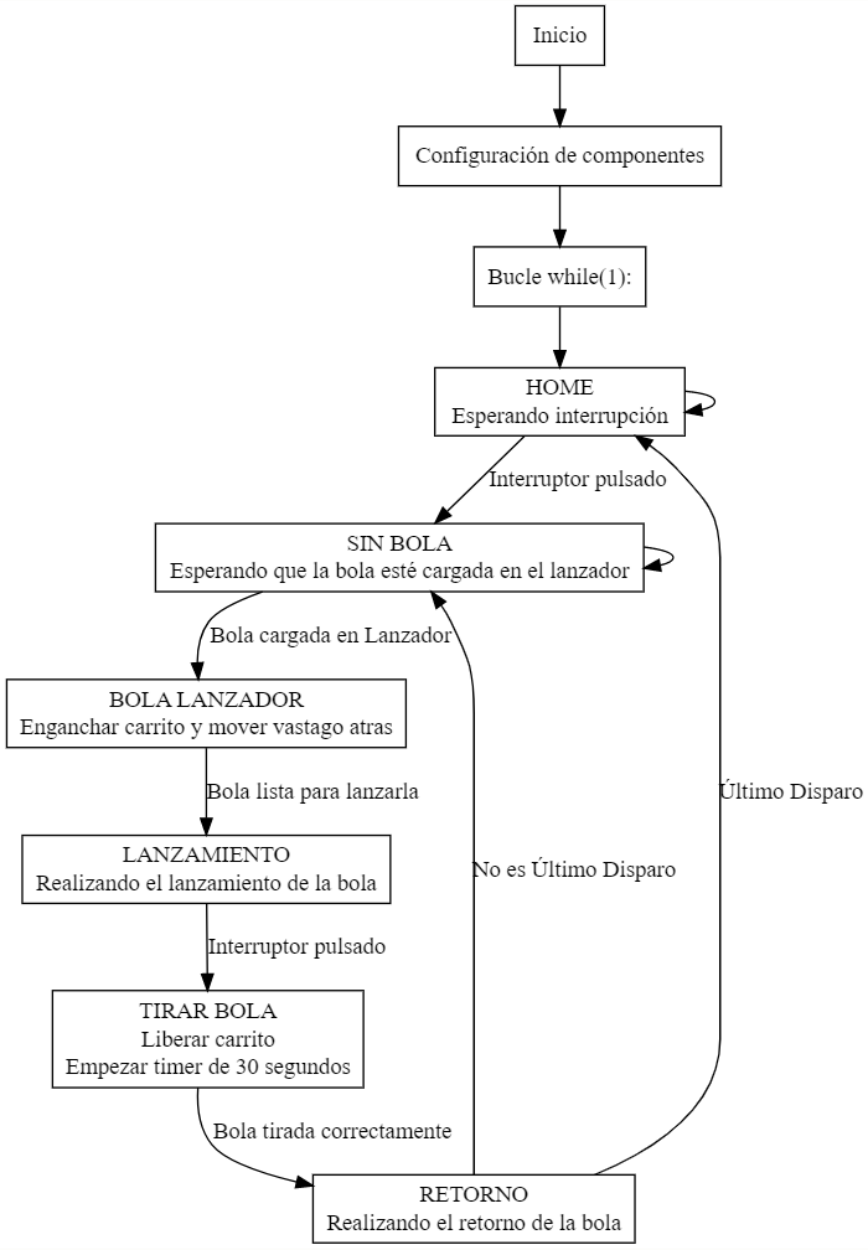
Por resumir, las partes que emplean interrupciones son los bolos mediante el puerto K(PCINT2) y las señales SW2 y SW6 mediante el puerot D (INT0 e INT2).

Cabe destacar que nuestra solución sigue un proceso secuencial en el que hemos utilizado delays. Para ello hemos utilizado la librería externa <util/delay.h>, comprobando previamente que no realizaba interrupciones y que funcionaba como en la primera práctica de la asignatura.

1.2 FLUJOGRAMA

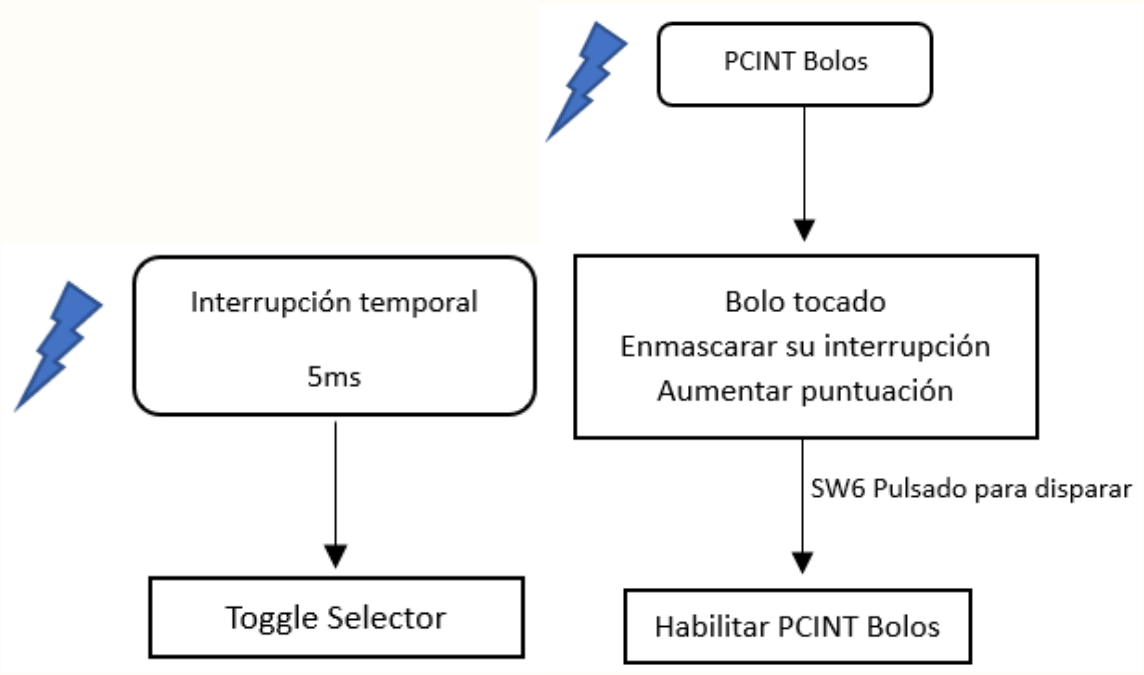
Principalmente tenemos dos bloques : el propio funcionamiento y secuencia de la pista de la bolera y por otro lado la lógica para los displays y la puntuación, que trabaja en paralelo al funcionamiento fundamental de la bolera.

El flujograma del proceso principal consiste en la máquina de estados que hemos implementado como solución :



Flujograma principal

Los flujogramas que reflejan la idea principal de los displays y la cuenta de la puntuación serían los siguientes:

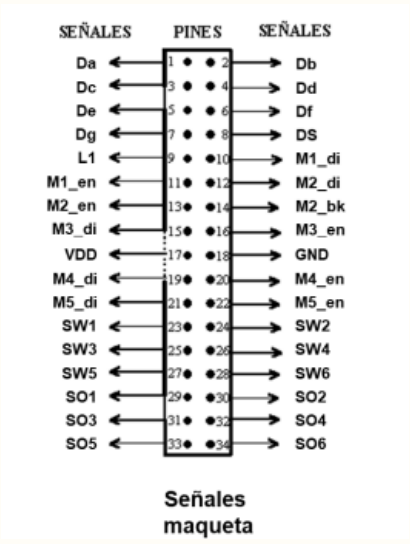


Flujograma bolos y selector

2 TABLA DE CONEXIONES

A continuación, se mostrarán los pines de la maqueta asociados a los pines de la placa Atmega640 utilizada en el proyecto.

Hemos utilizado el conector J1 de la placa para realizar el conexionado.



Señales maqueta

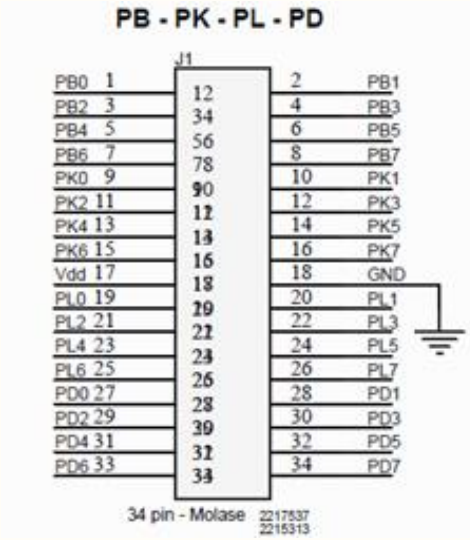


TABLA 1- PUERTOS Y PINES DE ENTRADA

SEÑALES ENTRADAS	PUERTOS	PINES ATMEGA	PINES MAQUETA
SO1	PK0/PCINT16	9	29
SO2	PK1/PCINT17	10	30
SO3	PK2/PCINT18	11	31
SO4	PK3/PCINT19	12	32
SO5	PK4/PCINT20	13	33
SO6	PK5/PCINT21	14	34
SW1	PD6	33	23
SW2	PD0/INT0	27	24
SW3	PD1/INT1	28	25
SW4	PD4	31	26
SW5	PD3/INT3	30	27
SW6	PD2/INT2	29	28

Tabla 2- Puertos y pines de salida

SEÑALES SALIDAS	PUERTOS	PINES ATMEGA	PINES MAQUETA
M1_di	PK6	15	10
M1_en	PL0	19	11
M2_di	PK7	16	12
M2_en	PL1	20	13
M2_bk	PL3	22	14
M3_di	PL2	21	15
M3_en	PL4	23	16
M4_di	PL5	24	19
M4_en	PL7	26	20
M5_di	PD5	32	21
M5_en	PD7	34	22

L1	PL6	25	9
Da	PB0	1	1
Db	PB1	2	2
Dc	PB2	3	3
Dd	PB3	4	4
De	PB4	5	5
Df	PB5	6	6
Dg	PB6	7	7
DS	PB7	8	8

3 CÓDIGOS DE LA SOLUCIÓN

El proyecto en global está particionado en un main.c principal en el que incluimos distintos ficheros de cada una de las partes por separado tales como configuraciones, lanzador, elevadores de carga/retorno, fichero para macros y funciones inline, manejo del display…

Queremos aprovechar este apartado para comentar que hemos realizado simulaciones de la maqueta a través de ficheros de estímulos. Realizamos la implementación fase a fase de cada componente. Seguimos un desarrollo guiado por pruebas, es decir, hacer una cosa, probarla, hacer una cosa, probarla…. Y ver que se iban cumpliendo los requisitos que habíamos predefinido para desarrollar.

En el propio código del proyecto está comentada la solución, sin embargo, las principales funciones o lógica que nos gustaría comentar aquí son las siguientes:

- **Principales estados de nuestro proyecto, que se manejarán con un switch en el main**:

```
typedef enum {HOME, SIN_BOLA, BOLA_LANZADOR, LANZAMIENTO, TIRAR_BOLA,
RETORNO, FINAL} States;
```

- **Configuración del timer1:**

```
void setup_timer1(){            //Preparado para interrumpir cada 3 seg
      cli();
      setBit(TCCR1B,CS10);      //Prescalado 1024
      setBit(TCCR1B,CS12);
      setBit(TCCR1B,WGM12);
      setBit(TIMSK1,OCIE1A);
      OCR1A  = 23437;           // timer para interrumpir cada 3 segundos
      sei();
```

```
}
```

- **Algunas de las funciones inline empleadas:**

```
#define setBit(P, B) (P |= (1 << B))
#define clearBit(P, B) ((P) &= ~(1 << B))
#define toggleBit(P, B) ((P)^=(1 << B))
#define setOutput(P, B) ((P) |= (1 << B))
#define setInput(P, B) ((P) &= ~(1 << B))
```

- **Principales funciones que gobiernan el funcionamiento de la maqueta:**

```
// M1: Elevador de Cargas
    void bajaEC();
    void subeEC();

// M2: Giro vertical del lanzador
    void girarLanzador(int dir);
    void girarLanzador(int dir);
    void girarLanzador(int dir);
    void frenoLanzador();
    void pararLanzador();

// M3: Vastago
    void moverVastagoAdelante();
    void moverVastagoAtras();
    void pararVastago();

// M4: Carrito
    void engancharCarrito();
    void liberarCarrito();
    void pararCarrito();

// M5: Elevador de retorno
    void bajaER();
    void subeER();

// LED
    void encenderLED();
    void apagarLED();
    void parpadearLED();

// SENSORES SW
// Sensor SW1:
    int getSensor1();
// Sensor SW2:
    int getSensor2();
// Sensor SW3:
    int getSensor3();
// Sensor SW4:
    int getSensor4();
// Sensor SW5:
    int getSensor5();
// Sensor SW6:
    int getSensor6();

// DISPLAY
    void setDisplay(int numero);
    void DisplayUpdater();

// BOLOS
    void habilitarInterrupcionesSensores();
    void deshabilitarInterrupcionesSensores()
    void OnPinChangeBolos();
    void setPuntuacion(uint8_t num);
```

4 PARTICIÓN DEL TRABAJO

- Parte 1: Control de lanzador. Movimiento y lanzado de la bola y su luz.
  Thanos y Ricardo
- Parte 2: Control del sistema de retorno de las bolas y del interruptor de disparo.
  Iago y Belén
- Parte 3: Control de los bolos y de los displays de 7 segmentos e integración del sistema (medida del paso del tiempo, conteo de puntos, etc.)
  Javier y Mario + Thanos en integración