Arkanoid con Arduino

Desarrollo de dispositivos IOT - 21791

Universitat de les Illes Balears

23/01/2025

—

Miquel Robles Mclean
41615695R

Índice

Introducción

El proyecto que se presenta a continuación es la creación de un juego Arkanoid multijugador utilizando Arduino y Processing.

Arkanoid es un clásico juego de arcade creado en 1986 que ha entretenido a generaciones de jugadores con su sencilla pero adictiva mecánica de juego. En esta versión del juego, se ha añadido una emocionante dimensión multijugador, donde dos jugadores compiten para romper la mayor cantidad de bloques en un tiempo limitado.

El juego se basa en la colaboración de dos placas Arduino, cada una con funciones específicas. Una de las placas Arduino se encarga de controlar los joysticks y los diferentes dispositivos de salida como LCDs y zumbadores, mientras que la otra gestiona la mecánica del juego, incluyendo el movimiento de las barras y las bolas, así como la detección de colisiones y la actualización de los bloques. Las dos placas se comunican mediante protocolo MQTT con un programa en Processing para que éste pueda mostrar el juego al usuario mediante una interfaz gráfica.

En cuanto a la mecánica de juego, existen dos barras y dos bolas, cada jugador solo puede hacer rebotar su bola en su barra designada mediante un joystick . El objetivo del juego es romper la mayor cantidad de bloques posibles en dos minutos. Ambos jugadores comparten el mismo conjunto de bloques, lo que añade un elemento de competencia directa.

La elección de este juego para el proyecto final está basada en otro Arkanoid que se desarrolló con lenguaje ensamblador para ‘Estructura de computadores II’.

Diseño

Para diseñar el juego se han tenido en cuenta todos los requerimientos del proyecto indicados en su enunciado, por lo que el desarrollo del juego ha necesitado varios elementos hardware y software, que conjuntamente hacen posible que se pueda controlar el juego mediante dos joysticks externos.

La comunicación entre placas y el ordenador se realiza por protocolo MQTT y se explicará más profundamente en el apartado de comunicaciones, pero a modo de resumen, la siguiente imagen refleja la comunicación entre placas, ordenador y elementos hardware.

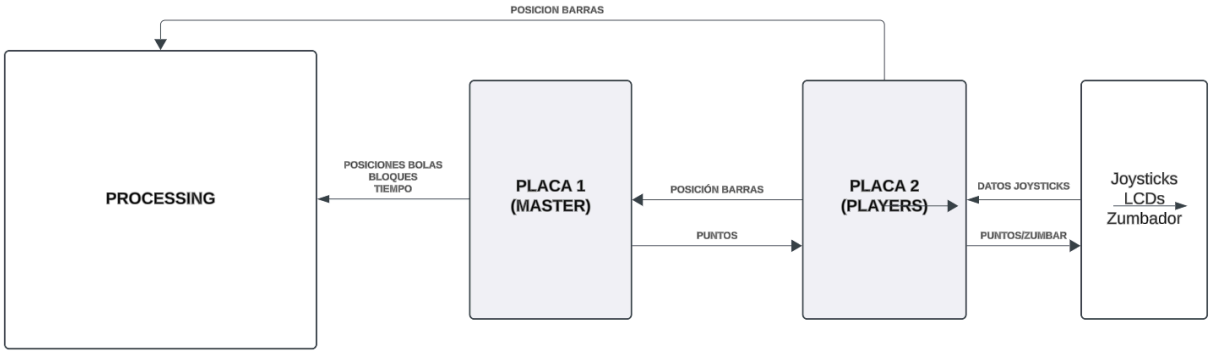


Fig 1: Diagrama de datos entre elementos

Diseño Hardware

Para el hardware se ha tenido en cuenta el uso de dos placas Arduino Mega, así como los diferentes tipos de entradas. Las placas en este caso diferencian dos tipos de control, la primera gestiona la mecánica de juego y la segunda gestiona la comunicación de datos con los dispositivos de entrada / salida.

Se ha decidido usar este diseño porque así se diferencian claramente los dos tipos de comportamientos hardware (control de I/O y gestión del juego). Contrariamente se habia planteado usar un diseño alternativo con el control de cada jugador en una placa diferente, aunque se descartó porque hay elementos comunes en la gestión del juego y hacerlo así sería un poco más confuso.

Placa de Jugadores (placa 2)

Se trata de una placa ArduinoMega 2560.

Las tareas asociadas a esta placa son las de control de los **joysticks**, **pantallas LCD** y **zumbador**, todo gestionado en el programa ArkanoidPlaca2.ino.

Esta placa está conectada al ordenador mediante un cable Serial que enlaza la entrada USB-B del Arduino y el puerto COM3 del ordenador.

Para la comunicación con los dispositivos de entrada/salida cuenta con conexiones con los siguientes dispositivos:

- **Shield ethernet**

  Para poder proveer de conexión ethernet a la placa.

- **Joysticks**

  La placa se conecta con dos joysticks, para controlar el eje X de las barras de ambos jugadores. En este caso el joystick solo está conectado por su salida VrX (eje X), ya que el uso del botón y del eje Y nos son irrelevantes.

- **Zumbador**

  La placa se conecta a un zumbador para que, cada vez que se reciba un punto, zumbe.

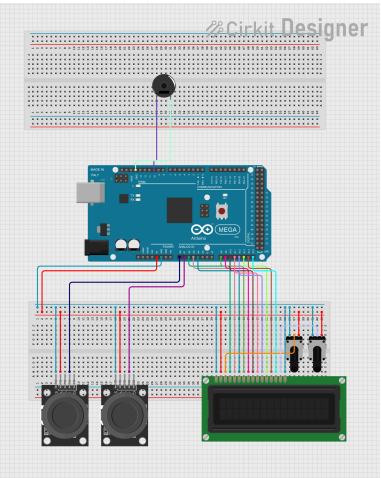
- **LCD**

  La placa se conecta con un LCD en el cual se escribe la puntuación asociada a cada jugador.

- **Protoboard**

  Se conecta una protoboard con los joysticks, zumbador y LCDs para poder proveer de un circuito eléctrico más eficiente y mejor organizado.

A continuación se enlaza una imagen de como quedan las conexiones eléctricas y de datos con todos los elementos.



Placa Master (placa 1)

Se trata de una placa ArduinoMega 2560.

Las tareas asociadas a esta placa son las de gestión y control de la mecánica del juego, todo gestionado en el programa ArkanoidPlaca1.ino.

Al igual que la otra placa, esta placa está conectada al ordenador mediante un cable Serial que enlaza la entrada USB-B del Arduino y el puerto COM4 del ordenador.

Ya que esta placa no realiza ningún control con dispositivos de entrada/salida, solámente requiere de un dispositivo:

- **Shield ethernet**

  Para poder proveer de conexión ethernet a la placa.

Diseño Software

Para el diseño del software se ha seguido con el mismo patrón del diseño hardware ya que van muy ligados de la mano, un software de gestión de mecánica del juego y otro de gestión de entradas y salidas hardware. Además, hay un tercer elemento software muy importante que es el de pintado del juego, en este caso se realiza a través de la interfaz de programación Processing. También se hace uso de un timer para gestionar la frecuencia de envío de datos a través de MQTT, ya que es el protocolo usado para la comunicación entre los diferentes elementos.

Este enfoque, al igual que en el diseño del hardware permite una diferenciación clara entre los diferentes softwares del proyecto, ya que cada uno se encarga de realizar unas ciertas tareas específicas.

Dicho esto, para concretar (aunque se va a hacer más en profundo en el apartado de implementación), el software se reparte en estos tres programas:

- **Control de dispositivos de entrada / salida (ArkanoidPlaca2.ino)**

  Este programa se encarga de leer los datos que se reciben de los dos joysticks y publicar sus valores como posiciones X de las barras en un topic MQTT.

  También se encarga de leer las puntuaciones de otro topic y mostrarlas en un LCD a la vez que se hace sonar un zumbador.

- **Control del juego (ArkanoidPlaca1.ino)**

  Este programa se encarga de leer los datos de la posición de las barras de un topic y con estos datos calcular las posiciones de las bolas, al igual que controlar los rebotes en paredes, barras y bloques.

El programa es el 'Master' y gestiona todos los datos del juego, como puntuaciones, posiciones de bola y tiempo.

Todos estos datos son publicados después en sus respectivos topics para que Processing pueda leerlos.

- **Pintado del juego (ArkanoidPintado.pde)**

  Este programa se encarga de leer los datos que ha publicado el master en los diferentes topics y pintar las barras, las bolas y los bloques en las posiciones correspondientes.

  También se encarga de mantener un estado del juego visible, dividido en tres pantallas (inicio, juego y resultado).

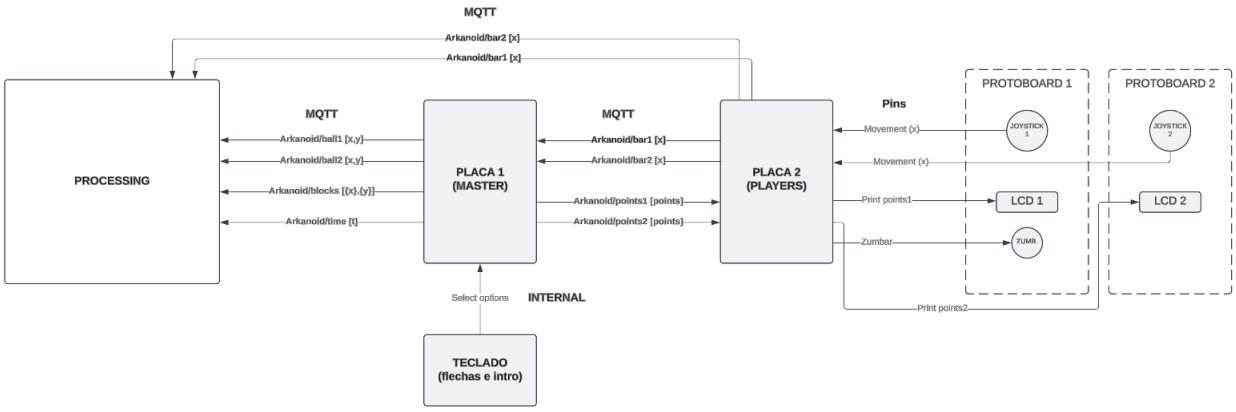
  A través de este programa se puede iniciar una interfaz gráfica para poder jugar al juego.

La implementación software así como los detalles específicos de cada programa se detallan en el apartado de Implementación.

Comunicación entre dispositivos

Las comunicaciones entre los diferentes dispositivos y programas se realizan mayoritariamente mediante el protocolo MQTT, solamente hay una comunicación que se hace a través de hardware (inicio del juego (lectura de teclado -> processing)).

En este siguiente diagrama se especifica la comunicación y los datos y los diferentes topics que se detallarán seguidamente.



Como se puede observar en el diagrama, tanto los dos programas Arduino como Processing usan comunicaciones con protocolo MQTT para enviar o recibir datos. En

cuanto a las comunicaciones con los elementos input/output, se usan lecturas de los diferentes pines de la placa 2.

En los casos de MQTT, los tres programas se conectan al mismo broker para enviar o recibir información y los mensajes se envían en diferentes topics. En esta tabla se puede ver un resumen de esta información:

Broker	Puerto
mqtt://tom.uib.es	1883

Topic	Datos del topic	Tipo de datos	Sincronización	Publicado por	Leído por
Arkanoid/bar1	Coordenada del eje X de la barra 1	char [ ]	Periódica (cada 100ms)	Placa 2	- Placa 1 (calcular rebote de bola) - Processing (pintar la barra)
Arkanoid/bar2	Coordenada del eje X de la barra 2	char [ ]	Periódica (cada 100ms)	Placa 2	- Placa 1 (calcular rebote de bola) - Processing (pintar la barra)
Arkanoid/ball1	Coordenadas X, Y de la bola 1	String	Periódica (cada 6.4 ms)	Placa 1	- Processing (pintar la bola)
Arkanoid/ball2	Coordenadas X, Y de la bola 2	String	Periódica (cada 6.4 ms)	Placa 1	- Processing (pintar la bola)
Arkanoid/blocks	Indica para cada bloque si está activo o no	String	Periódica (cada 6.4 ms)	Placa 1	- Processing (pintar los bloques activos e inactivos)
Arkanoid/points1	Puntuación del jugador 1	char [ ]	Aperiódica (cada vez que se actualiza la puntuación)	Placa 1	- Placa 2 (enviar puntuación a LCD)
Arkanoid/points2	Puntuación del jugador 2	char [ ]	Aperiódica (cada vez que se actualiza la puntuación)	Placa 1	- Placa 2 (enviar puntuación a LCD)
Arkanoid/time	Tiempo restante de juego	char [ ]	Periódica (cada segundo)	Placa 1	- Processing (mostrar tiempo y mostrar pantalla de resultados))

Implementación

Programa para la placa 1

Este programa gestiona las posiciones de la bola y de las barras, así como puntuaciones y tiempo.

Más concretamente, lee desde los topics correspondientes la posición de las barras y en cada bucle modifica la posición de la bola, revisando todas las colisiones posibles (paredes, barras y bloques).

La estructura genérica del programa es la siguiente:

Configuración de red y MQTT:

- Se configuran los parámetros de red (dirección IP, puerta de enlace, DNS).
- Se establece la conexión con un broker MQTT para la comunicación de los datos del juego.

Configuración del juego:

- Se definen constantes para el tamaño del lienzo, paletas, bloques y bolas.
- Se inicializan variables para el seguimiento de la puntuación y el tiempo del juego.

Bucle principal (`loop`)

- Gestiona la conexión MQTT.
- Actualiza la posición de las paletas mediante interpolación.
- Controla la lógica del juego y publica datos cuando el temporizador lo indica.

Para especificar el código, a continuación se describen todas las funciones con pseudocódigo para su mayor entendibilidad:

`setup()` - Configuración principal

Esta función configura los parámetros de red, inicializa el temporizador y resetea la posición de las bolas.

```
INICIO setup()
    Iniciar comunicación serie a 115200 baudios
    Configurar Ethernet con dirección MAC e IP fija
    Configurar servidor MQTT con dirección y puerto
    Inicializar temporizador con preescalado 256 y cuenta de 400
    Iniciar temporizador
    Calcular el ancho de los bloques de acuerdo con el lienzo
    PARA cada fila de bloques
        PARA cada columna de bloques
            Inicializar bloque como activo (true)
        FIN PARA
    FIN PARA
    Resetear la posición y velocidad de las bolas 1 y 2
    Guardar tiempo de inicio del juego
FIN
```

`loop()` - Bucle principal

Administra la conexión con el broker MQTT, suscribiéndose a los temas y actualizando el estado del juego.

```
INICIO loop()
    SI no está conectado a MQTT
        Conectar a MQTT
    FIN SI

    SI no se ha suscrito
        Suscribirse a los temas bar1 y bar2
    FIN SI
    Actualizar posición de paletas mediante interpolación

    SI el juego está en curso Y la bandera de publicación es verdadera
        Llamar a updateGame() para actualizar la lógica del juego
        Publicar datos del juego
        Publicar puntuación
        Reiniciar bandera de publicación
    FIN SI
```

```
    Actualizar temporizador del juego
    Ejecutar bucle MQTT para recibir datos
FIN
```

`updateGame()` - Lógica del juego

Mueve las bolas y las resetea si están inactivas.

```
INICIO updateGame()
    SI bola1 está activa
        Mover bola1 y verificar colisiones
    SINO
        Resetear bola1
    FIN SI

    SI bola2 está activa
        Mover bola2 y verificar colisiones
    SINO
        Resetear bola2
    FIN SI
FIN
```

`moveBall()` - Movimiento de la bola

Controla el movimiento de la bola, la colisión con paredes, paletas y bloques.

Para el cálculo de colisiones de la bola con la paleta, primero, verifica si la bola está dentro del rango vertical de la paleta (`ballY`) y luego comprueba si su posición horizontal (`ballX`) está dentro de los límites de la paleta. Si se detecta la colisión, calcula la posición relativa del centro de la bola respecto al centro de la paleta, normalizando este valor entre `-1` y `1`. Dependiendo del punto de impacto, ajusta la velocidad horizontal de la bola, de modo que si golpea el centro, se moverá recta, y si golpea los bordes, tomará una trayectoria inclinada. Luego, invierte la dirección de la velocidad vertical para simular el rebote y ajusta la posición de la bola justo encima de la paleta para evitar superposiciones. El código se repite para una segunda bola con la misma lógica.

```
INICIO moveBall(ballX, ballY, speedX, speedY, active, paddleX, paddleY, ballNum)
    Actualizar posición de la bola sumando velocidad
    SI colisión con paredes laterales
        Invertir velocidad en X
    FIN SI
```

```
    SI colisión con la pared superior
        Invertir velocidad en Y
    FIN SI

    SI colisión con la paleta correspondiente
        Calcular el punto de impacto relativo
        Ajustar dirección de la bola
    FIN SI

    PARA cada fila de bloques
        PARA cada columna de bloques
            SI colisión con bloque activo
                Desactivar bloque
                Invertir velocidad en Y
                Actualizar puntuación
            FIN SI
        FIN PARA
    FIN PARA

    SI la bola se sale del área de juego
        Desactivar bola
    FIN SI
FIN
```

`callback()` - Manejo de mensajes MQTT

Recibe las posiciones de las paletas y las actualiza.

```
INICIO callback(topic, payload, length)
    Leer mensaje recibido
    Convertir mensaje a número flotante

    SI el mensaje es para la paleta 1
        Actualizar posición objetivo de la paleta 1
    FIN SI

    SI el mensaje es para la paleta 2
        Actualizar posición objetivo de la paleta 2
    FIN SI
FIN
```

`connect()` - Conexión a MQTT

Intenta conectar al broker hasta que se logra una conexión.

```
INICIO connect()
    MIENTRAS no esté conectado
        Intentar conectar a broker MQTT
        SI la conexión es exitosa
            Mostrar mensaje de conexión exitosa
        SINO
            Mostrar error y esperar 2 segundos
        FIN SI
    FIN MIENTRAS
FIN
```

`publishData()` - Publicación de datos

Envía la posición de las bolas y el estado de los bloques.

```
INICIO publishData()
    Convertir posición de bolas en cadenas de texto y publicarlas
    Convertir estado de bloques en una cadena de texto y publicarlo
    Publicar puntuaciones actuales
FIN
```

`resetBall()` - Reiniciar bola

Coloca la bola en una nueva posición y le asigna una dirección aleatoria.

```
INICIO resetBall(ballX, ballY, speedX, speedY, active)
    Posicionar la bola en una coordenada X aleatoria
    Posicionar la bola en una coordenada Y fija
    Asignar velocidad aleatoria en X
    Activar la bola
FIN
```

`interpolatePaddle()` - Movimiento suave de las paletas

Ajusta la posición de la paleta suavemente hacia la posición objetivo.

```
INICIO interpolatePaddle(paddleX, targetPaddleX)
    Actualizar paddleX usando interpolación lineal
    Limitar paddleX dentro de los márgenes de la pantalla
```

```
FIN
```

`updateCountdown()` - Control del tiempo de juego

Actualiza el temporizador y detiene el juego cuando el tiempo ha terminado.

```
INICIO updateCountdown()
    Calcular tiempo transcurrido
    SI el tiempo ha finalizado
        Detener el juego
        Publicar mensaje de tiempo agotado
    SINO
        Calcular minutos y segundos restantes
        Publicar tiempo restante
    FIN SI
FIN
```

Así pues, en modo de resumen el flujo genérico del programa es el siguiente:

`setup()`: Configura red, MQTT y estado inicial del juego.

`loop()`:

- Conecta al broker MQTT si es necesario.
- Actualiza la posición de las paletas.
- Gestiona la lógica del juego y la publicación de datos periódicamente.
- Actualiza el temporizador.

Programa para la placa 2

El programa para la placa 2 lee la posición sobre el eje X del joystick y la envía por MQTT, al mismo tiempo escucha para recibir los puntos y los escribe en el LCD. Tambié hace sonar el zumbador cuando recibe un punto.

La estructura genérica del programa es el siguiente:

Configuración de red y MQTT:

- Se configuran los parámetros de red (dirección IP, puerta de enlace, DNS).
- Se establece la conexión con un broker MQTT para la comunicación de los datos de los dispositivos I/O.

Configuración del juego:

- Se inicializa el LCD y los timers.

Bucle principal (`loop`)

- Gestiona la conexión MQTT.
- Actualiza la posición de las paletas mediante su lectura de los joysticks.

Para especificar el código, a continuación se describen todas las funciones con pseudocódigo para su mayor entendibilidad:

`setup()` - Configuración principal

Inicializa las comunicaciones y el LCD

```
INICIO setup()
    Inicializar comunicación serial
    Inicializar LCD y limpiar pantalla
    Configurar pines de buzzer y joystick
    Configurar red Ethernet con dirección IP, gateway y DNS
    Configurar conexión MQTT con broker
    Iniciar temporizador con intervalo de 100ms
FIN setup
```

`loop()` - Bucle principal

Se lee de los pines y se publica el valor

```
INICIO loop()
    Llamar a la función connect() para asegurar conexión MQTT
    Si no se ha suscrito a los temas
        Suscribirse a los temas de puntos (points1Topic, points2Topic)
    SI publishFlag es verdadero
        Llamar a publishJoystickValues() para publicar valores del joystick
        Resetear publishFlag
    Llamar a mqttClient.loop() para manejar la comunicación MQTT
FIN loop
```

`publishJoystickValues()` - Publicar valores de joystick

Se mapean y publican valores del joystick

```
INICIO publishJoystickValues()
    Leer valores de los joysticks (JOYSTICK_1_X_PIN, JOYSTICK_2_X_PIN)
    Mapear los valores leídos al rango deseado (minMappedValue, maxMappedValue)
    Convertir los valores mapeados a cadenas de texto
    Publicar valores de joystick en los tópicos correspondientes (bar1Topic,
bar2Topic)
FIN publishJoystickValues
```

`callback()` - Se reciben valores por MQTT

Lee valores en los topics y actualiza los puntos

```
INICIO callback(topic, payload, length)
    Copiar el payload en un buffer
    Convertir el mensaje recibido en un número entero
    SI el tema es "points1"
        Si el valor recibido es diferente al anterior
            Actualizar el valor de puntos y mostrar en el LCD
    SINO SI el tema es "points2"
        Si el valor recibido es diferente al anterior
            Actualizar el valor de puntos y mostrar en el LCD
FIN callback
```

`connect()` - Se conecta a MQTT

Conexión a MQTT

```
INICIO connect()
    Mientras no esté conectado a MQTT
        Intentar conectar con el broker MQTT usando el ID de cliente
        SI la conexión es exitosa
            Imprimir mensaje de conexión exitosa
        SINO
            Imprimir código de error de conexión
```

```
            Esperar 5 segundos antes de intentar nuevamente
FIN connect
```

`soundBuzzer()` - Zumbador

Haze sonar el zumbador

```
INICIO soundBuzzer()
    Cambiar el estado del buzzer (encender o apagar)
    Alternar el estado del buzzer
FIN soundBuzzer
```

Programa para processing

El programa de Processing, lee las posiciones de la bola y barra y las pinta en pantalla, además de proveer de todo el pintado del juego.

El juego arranca con una pantalla de inicio, en la que espera a un KeyPressed para pasar a la pantalla de juego. El programa va comparando el tiempo con el string 00:00 y cuando llega a su fin, muestra la pantalla de finalización.

Cabe destacar que también se hace la misma interpolación para el cálculo de posición de la barra, para que el movimiento sea más fluido

La estructura genérica del programa es la siguiente:

Configuración inicial (`settings` y `setup`):

- Define el tamaño del lienzo.
- Inicializa las posiciones de las paletas y bolas.
- Carga imágenes y fuentes.
- Conecta al broker MQTT y se suscribe a los tópicos.

Bucle principal (`draw`):

- Evalúa el estado del juego:
    - Muestra pantalla de inicio.
    - Muestra el juego en ejecución, dibujando elementos.

- Muestra pantalla de "Game Over" cuando el tiempo llega a 0.

Funciones auxiliares:

- **Pantalla de inicio:** Muestra el logo y espera que el jugador inicie el juego.
- **Pantalla del juego:** Dibuja paredes, bloques, paletas y bolas, actualiza posiciones interpolando con MQTT.
- **Pantalla de "Game Over":** Muestra el resultado final y ganador.
- **Conexión a MQTT:** Maneja la conexión y suscripción a los tópicos del juego.
- **Reconexión automática:** Intenta reconectar en caso de pérdida de conexión.
- **Procesamiento de datos recibidos:** Actualiza posiciones de paletas, bolas, puntuación y estado de bloques.

Eventos:

- Detecta la presión de teclas para iniciar o salir del juego.

Para especificar el código, a continuación se describen todas las funciones con pseudocódigo para su mayor entendibilidad:

`setup()` - Configuración principal

Configuración inicial del programa

```
INICIO setup()
    Calcular el ancho de los bloques
    Inicializar posiciones de las paletas
    Cargar imágenes de paredes, paletas, bolas y bloques
    Cargar fuente personalizada
    Inicializar bloques en estado activo
    INTENTAR
        Crear cliente MQTT
        Definir callbacks para manejar conexión, mensajes y entrega
        Conectar al broker MQTT
    ATRAPAR ERROR
        Imprimir mensaje de error
    FIN INTENTAR
FIN setup
```

`draw()` - Control del dibujado

Se controla el dibujado del juego según su estado

```
INICIO draw()
    SI gameState es 0
        Llamar a drawStartScreen()
    SINO SI gameState es 1
        Llamar a drawGameScreen()
        SI tiempo es "00:00"
            Cambiar gameState a 2
        FIN SI
    SINO SI gameState es 2
        Llamar a drawGameOverScreen()
    FIN SI
FIN draw
```

`drawStartScreen()` - Dibujado de la pantalla de inicio

Se dibuja la pantalla inicial

```
INICIO drawStartScreen()
    Establecer fondo negro
    Mostrar logo centrado
    Mostrar texto de autor e información
    Mostrar "PRESS ANY KEY TO START" con efecto parpadeante
FIN drawStartScreen
```

`drawGameScreen()` - Dibujado de la pantalla de juego

Se dibuja la pantalla del juego

```
INICIO drawGameScreen()
    Establecer fondo azul oscuro
    Dibujar paredes y área de puntuación
    Mostrar logo y puntuaciones de jugadores
    Dibujar bloques activos
    Dibujar paletas interpolando posición
    Verificar límites de movimiento de las paletas
    Dibujar bolas
FIN drawGameScreen
```

`drawGameOver()` - Dibujado de la pantalla de finalización

Se dibuja la pantalla de game over

```
INICIO drawGameOverScreen()
    Establecer fondo negro
    Mostrar logo pequeño en la parte superior
    Mostrar mensaje "GAME OVER"
    Determinar y mostrar ganador
    Mostrar mensaje para salir
FIN drawGameOverScreen
```

`connectToBroker()` - Conexión con el broker

Se intenta conectar con el broker

```
INICIO connectToBroker()
    INTENTAR
        Conectar al broker MQTT
        Suscribirse a los temas relevantes (bar1, bar2, ball1, ball2, etc.)
        Imprimir mensaje de éxito
    ATRAPAR ERROR
        Imprimir mensaje de error
    FIN INTENTAR
FIN connectToBroker
```

`reconnect()` - Reconexión con el broker

Se intenta reconectar con el broker

```
INICIO reconnect()
    MIENTRAS no esté reconectado
        INTENTAR
            Conectar al broker MQTT
            Suscribirse a los temas
            Imprimir mensaje de éxito
            Marcar como reconectado
        ATRAPAR ERROR
            Imprimir error
            Esperar 5 segundos antes de reintentar
        FIN INTENTAR
```

```
    FIN MIENTRAS
FIN reconnect
```

`updateBlocks()` - Control de bloques activos

Actualiza el estado de los bloques para su pintado

```
INICIO updateBlocks(payload)
    Dividir payload en elementos individuales
    SI la longitud es correcta
        Actualizar estado de cada bloque
    SINO
        Imprimir mensaje de error
    FIN SI
FIN updateBlocks
```

`keyPressed()` - Detección de pulsación de tecla

Detecta si se ha pulsado una tecla y se cambia el estado del juego

```
INICIO keyPressed()
    SI gameState es 0
        Cambiar gameState a 1 (inicio del juego)
    SINO SI gameState es 2
        Cerrar el programa
    FIN SI
FIN keyPressed
```

Conclusión

Esta práctica ha servido para poner en práctica todos los conocimientos de esta asignatura y ha sido muy entretenida de realizar.

Una de las partes que más tiempo ha tomado ha sido el diseño, ya que no es fácil poder llegar a uno que sea eficiente y factible, aunque la implementación tampoco ha sido nada fácil.

Se han ido creando iteraciones del juego, primero solamente en Processing para después ir creando y añadiendo funciones en arduino

Los diferentes retos a los que me he enfrentado creo que me van a ser muy útiles en la vida universitaria y profesional

Referencias

Para las imágenes: https://github.com/wkeeling/arkanoid

Para la interpolación: https://www.gamedev.net/tutorials/programming/general-and-gameplay-programming/linear-interpolation-explained-r5892/

Para la creación de circuitos: https://app.cirkitdesigner.com/