








Tutorial para implementar el juego PONG minimalista en Arduino

labtec@umce.cl

jonnhatan.garcia@umce.cl

Materiales necesarios

<ul style="list-style-type: none">1 Arduino UNO	
<ul style="list-style-type: none">1 Display 4 Matriz De Puntos Led 8x8 Max7219	
<ul style="list-style-type: none">1 Buzzer	<p>KY-006</p> 
<ul style="list-style-type: none">1 Módulo de joystick analógico de 3 ejes para Arduino	
<ul style="list-style-type: none">Cables varios	



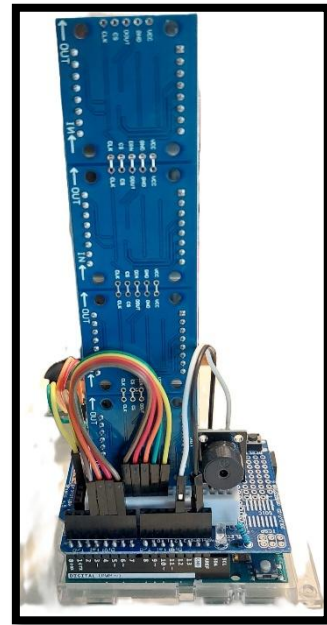
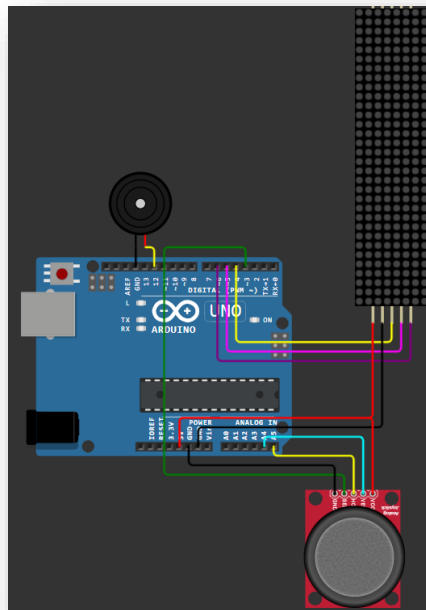
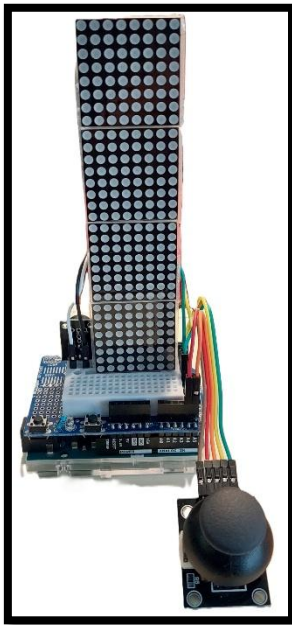
Este documento explica cómo implementar el clásico juego PONG en una versión minimalista para Arduino, usando 4 matrices LED MAX7219 (32 filas x 8 columnas) y controlado por un joystick analógico. El juego está diseñado para funcionar con hardware sencillo y de bajo costo, y se juega en solitario con la matriz dispuesta verticalmente. Utiliza la librería LedControl para gestionar las matrices LED y el buzzer para efectos de sonido. El código fuente está comentado en español para facilitar su comprensión y modificación.

Características generales

Todos los juegos están diseñados para ser minimalistas y funcionar con el mismo hardware sencillo y de bajo costo. El documento incluye el código fuente completo en Arduino para cada juego, con comentarios explicativos en español. Se detallan las funciones principales, la inicialización de hardware, la gestión de entradas y salidas, y la visualización en matrices LED. Se utilizan animaciones y sonidos para mejorar la experiencia de juego.

Cómo implementar el juego PONG minimalista en Arduino

Imagen n°1: de izquierda a derecha se muestra i) vista frontal, ii) diagrama de cableado, y iii) vista posterior del hardware utilizado.



Hardware necesario

Construye el circuito de acuerdo con el diagrama de cableado mostrado en la imagen n°1. Los pines que controlan la matriz led configurada de forma vertical son: DIN_PIN 4, CS_PIN 5, CLK_PIN 6. La matriz fue configurada para que las piezas se muevan de forma vertical, por lo tanto la matriz dispone de 8



columnas y 32 filas para el movimiento de las piezas del juego. Los pines del joystick se conectan a: JOY_X (salida horizontal) al pin analógico A5; y JOY_Y (salida vertical) al pin analógico A4, y el botón de rotación (botón integrado SW del joystick) se conecta al pin digital 3 del Arduino UNO. El Buzzer se conecta al pin digital 12 de la placa y a GND.

Estructura básica del código PONG

Inicialización

- Se utiliza la librería LedControl para controlar las matrices LED.
- Se definen los pines para el joystick y el buzzer.
- Se inicializan las matrices LED y las variables del juego.

Variables principales

- **x, y:** Coordenadas de la pelota.
- **downDir, rightDir:** Direcciones de movimiento de la pelota (vertical y horizontal).
- **paddleTop, paddleBottom:** Posición de la paleta (controlada por el joystick).
- **playerScore, computerScore:** Puntuaciones del jugador y del “ordenador”.
- **ballsMissed:** Contador de pelotas perdidas.
- **gamePace:** Velocidad del juego.

Lógica del juego

Bucle principal (loop)

- El juego se ejecuta mientras ninguna puntuación llegue a 9 y no se hayan perdido demasiadas pelotas.
- Se dibuja la pelota y la paleta, se evalúa la jugada y se mueve la pelota.
- Se ajusta la velocidad y se limpia la pantalla tras cada jugada.
- Si se pierde una pelota, se muestra la pantalla de “Game Over” y se reinician las variables.

Lógica principal

- **Lectura del joystick:** El jugador mueve la paleta verticalmente usando el joystick.
- **Movimiento de la pelota:** La pelota rebota en los bordes y en la paleta. Si no se intercepta, suma punto al “ordenador” y se reinicia la jugada.
- **Colisiones:** Se detectan colisiones con los bordes y la paleta para cambiar la dirección de la pelota.
- **Puntuación:** Se incrementa la puntuación del jugador al interceptar la pelota y la del ordenador si se pierde.
- **Animaciones y sonidos:** Se reproducen sonidos con el buzzer en cada rebote y al perder la pelota.

Funciones clave

- **setPixel(x, y, state):** Enciende o apaga un LED en la posición indicada.
- **paddlePosition():** Lee el joystick y calcula la posición de la paleta.



- **displayBall():** Dibuja la pelota en la pantalla.
- **displayPaddle():** Dibuja la paleta en la pantalla.
- **evaluatePlay():** Evalúa si la pelota ha sido interceptada o perdida, y actualiza puntuaciones y direcciones.
- **moveBall():** Actualiza la posición de la pelota según la dirección.
- **kickoff():** Reinicia la posición de la pelota y la paleta para una nueva jugada.
- **displayCountdown():** Muestra una cuenta regresiva antes de iniciar.
- **displayGameOver():** Muestra la animación de fin de juego y reproduce sonidos.

Sugerencias para adaptar el código

- Puedes ajustar la velocidad del juego modificando la variable gamePace.
- El brillo de las matrices LED se puede ajustar con setIntensity.
- El código está comentado en español para facilitar la comprensión y modificación.

Código fuente PONG minimalista

```
/* PONG minimalista para 4 matrices MAX7219 (32x8)
 * Controlado con joystick analógico.
 * Utiliza la librería LedControl para manejar las matrices LED.
 * Adaptado de varios ejemplos de PONG en Arduino.
 * by LabTec
 * octubre/2025
 */

#include <LedControl.h>

#define DIN_PIN 4
#define CS_PIN 5
#define CLK_PIN 6
#define NUM_OF_MATRIX 4

#define JOY_Y A4
#define JOY_SW 3
#define BUZZER 12

LedControl lc = LedControl(DIN_PIN, CLK_PIN, CS_PIN, NUM_OF_MATRIX);

// Dimensiones lógicas del tablero (4 matrices x 8 columnas = 32x8)
const int MATRIX_WIDTH = 32; // 4 matrices de 8 columnas
const int MATRIX_HEIGHT = 8; // Altura fija

int x, y;
boolean downDir, rightDir;

int paddleTop, paddleBottom;

const int initGamePace = 300;
int gamePace;

int playerScore = 0;
int computerScore = 0;
int ballsMissed = 0;
const int MAX_BALLS_MISSED = 1;
```



```
void setPixel(int x, int y, bool state) {
  int matrixId = y / 8;
  int localY = y % 8;
  int rotatedRow = x;
  int rotatedCol = 7 - localY;
  lc.setLed(matrixId, rotatedRow, rotatedCol, state);
}

void setup() {
  for (int i = 0; i < NUM_OF_MATRIX; i++) {
    lc.shutdown(i, false);
    lc.setIntensity(i, 8);
    lc.clearDisplay(i);
  }
  Serial.begin(9600);
  randomSeed(analogRead(0));
  pinMode(JOY_SW, INPUT_PULLUP);
  pinMode(BUZZER, OUTPUT);

  kickoff(1, playerScore, computerScore);
  displayCountdown();
  tone(BUZZER, 1000);
  delay(100);
  noTone(BUZZER);
}

void loop() {
  while ((playerScore < 9) && (computerScore < 9) && (ballsMissed < MAX_BALLS_MISSED)) {
    displayBall();
    displayPaddle();
    evaluatePlay();
    moveBall();

    if (gamePace > 0) {
      gamePace--;
    }
    delay(gamePace);
    clearAllMatrices();
  }

  if (ballsMissed >= MAX_BALLS_MISSED) {
    displayGameOver();
    delay(5000);
    // Reset game
    playerScore = 0;
    computerScore = 0;
    ballsMissed = 0;
  } else {
    delay(3000);
    playerScore = 0;
    computerScore = 0;
  }
  kickoff(1, playerScore, computerScore);
  displayCountdown();
}

int paddlePosition() {
  int sensorValue = analogRead(JOY_Y);
  Serial.println(sensorValue);
  int pos = map(sensorValue, 0, 1023, 0, 6);
  Serial.println(pos);
}
```



```
return constrain(pos, 0, 6);
}

void displayBall() {
  setPixel(x, y, true);
}

void displayPaddle() {
  paddleTop = paddlePosition();    // posición superior
  paddleBottom = paddleTop + 1;    // ahora cubre tres LEDs
  for (int i = paddleTop; i <= paddleBottom; i++) {
    setPixel(i, 0, true);          // dibuja la paleta vertical en la columna 0
  }
}

void evaluatePlay() {
  if (y == 31) {
    downDir = false;
    tone(BUZZER, 800, 50);
    return;
  }

  if (y == 0) {
    if ((x == paddleTop) || (x == paddleBottom)) {
      downDir = true;
      tone(BUZZER, 1000, 50);
      playerScore++;
    } else {
      computerScore++;
      ballsMissed++; // Incrementar contador de pelotas perdidas
      kickoff(1, playerScore, computerScore);
    }
    return;
  }

  if (x == 0) {
    rightDir = true;
    tone(BUZZER, 500, 30);
  }

  if (x == 7) {
    rightDir = false;
    tone(BUZZER, 500, 30);
  }

  if (y == 15 || y == 16) {
    if (random(0, 20) == 0) {
      rightDir = !rightDir;
      tone(BUZZER, 600, 20);
    }
  }
}

void moveBall() {
  y += downDir ? 1 : -1;
  x += rightDir ? 1 : -1;
  x = constrain(x, 0, 7);
  y = constrain(y, 0, 31);
}

void kickoff(int player, int pScore, int cScore) {
```



```
delay(500);
delay(1500);

clearAllMatrices();

if (player == 1) {
    y = 1;
    x = 3;
    downDir = true;
} else {
    y = 30;
    x = 4;
    downDir = false;
}

rightDir = random(0, 2);
gamePace = initGamePace;
}

void displayCountdown() {
    for (int i = 3; i >= 1; i--) {
        clearAllMatrices();
        displayDigit(i,0); // Mostrar el número en la posición izquierda
        delay(1000); // Esperar 1 segundo
    }
    clearAllMatrices();
}

void clearAllMatrices() {
    for (int i = 0; i < NUM_OF_MATRIX; i++) {
        lc.clearDisplay(i);
    }
}

void displayDigit(int digit, int pos) {
    const int digits[10][3][5] = {
        {{1,1,1,1,1},{1,0,0,0,1},{1,1,1,1,1}}, // 0 ok
        {{0,0,0,0,1},{1,1,1,1,1},{0,1,0,0,1}}, // 1 ok
        {{1,1,1,0,1},{1,0,1,0,1},{1,0,1,1,1}}, // 2 ok
        {{1,1,1,1,1},{1,0,1,0,1},{1,0,1,0,1}}, // 3 ok
        {{1,1,1,1,1},{0,0,1,0,0},{1,1,1,0,0}}, // 4 ok
        {{1,0,1,1,1},{1,0,1,0,1},{1,1,1,0,1}}, // 5 ok
        {{1,0,1,1,1},{1,0,1,0,1},{1,1,1,1,1}}, // 6 ok
        {{1,1,1,1,1},{1,0,0,0,0},{1,0,0,0,0}}, // 7 ok
        {{1,1,1,1,1},{1,0,1,0,1},{1,1,1,1,1}}, // 8 ok
        {{1,1,1,1,1},{1,0,1,0,1},{1,1,1,0,1}} // 9 ok
    };
    // Ensure the digit is valid
    if (digit < 0 || digit > 9) return; // Invalid digit, exit

    int digitWidth = 5;
    int digitHeight = 3;
    int startCol = (MATRIX_WIDTH - digitWidth) / 2; // Centrado horizontal
    int startRow = (MATRIX_HEIGHT - digitHeight) / 2; // Centrado vertical
    for (int row = 0; row < digitHeight; row++) {
        for (int col = 0; col < digitWidth; col++) {
            bool ledState = digits[digit][row][col];
            int globalX = startCol + col;
            int globalY = startRow + row;
            if (globalX >= 0 && globalX < MATRIX_WIDTH && globalY >= 0 && globalY < MATRIX_HEIGHT) {
                int matrixIndex = NUM_OF_MATRIX - 1 - (globalX / 8);
```



```
int localX = globalX % 8;
lc.setLed(matrixIndex, globalY, localX, ledState);
}
}
}
}

void displayGameOver() {
clearAllMatrices();
const byte N[8] = { //N
B00000000,
B01111110,
B00001100,
B00011000,
B00110000,
B01111110,
B00000000,
B00000000
};
const byte I[8] = { //I
B00000000,
B00000000,
B01000010,
B01111110,
B01000010,
B00000000,
B00000000,
B00000000
};
const byte F[8] = { //F
B00000000,
B01000000,
B01001000,
B01001000,
B01111110,
B00000000,
B00000000,
B00000000
};

// Mostrar cada letra en su matriz correspondiente
for (int row = 0; row < 8; row++) {
lc.setRow(1, row, N[row]); // Matriz 1: N
lc.setRow(2, row, I[row]); // Matriz 2: I
lc.setRow(3, row, F[row]); // Matriz 3: F
}

// Efecto de sonido de game over
for (int i = 0; i < 3; i++) {
tone(BUZZER, 300 - (i * 50), 300);
delay(400);
}
noTone(BUZZER);
}
```

Espero que todo funcione.