Practica No 1
Manejo de Entradas/Salidas (GPIO), Interrupciones y la LCD TFT con el ESP32

Jhon Edison Vargas Vargas - 20152005077, Juanita Acero Cuellar - 20161005012

***Resumen*—Esta practica consiste en diseñar e implementar con el ESP32 un juego de memoria que se basa en encontrar parejas antes de que termine el tiempo o se acaben los intentos, se cuenta con dos tamaños de tablero (4x4 o 6x6) y tres niveles distintos (fácil, medio y difícil) que pueden ser elegidos por el jugador. Para esto se hace uso de un teclado matricial 4x4 y una LCD TFT ILI9163 (o similar) controlada por SPI.**

***Palabras clave*— ESP32, GPIO, Interrupciones**

I. Introducción

El ESP32 es un dispositivo de bajo costo y bajo consumo de energía.
Esta diseñado para dispositivos móviles, dispositivos electrónicos portátiles y aplicaciones IoT.
La integración de Bluetooth, Bluetooth LE y Wi-Fi permite una amplia gama de aplicaciones, el uso de Wi-Fi permite una comunicación de mediano alcance y conectarse a una red LAN y a través de un Router conexión a Internet, mientras que el Bluetooth nos permite conectarse directamente a otro dispositivo como un celular [1].
Para el desarrollo de una aplicación y la implementación en este microcontrolador se puede utilizar el IDE de arduino.

II. Formulación del problema

Diseñar e implementar con el ESP32 un juego de encontrar parejas usando un teclado matricial 4x4 y una LCD TFT ILI9163 controlada por SPI.

El juego contara con las siguientes funcionalidades:

- El jugador podrá seleccionar dos tamaños de tablero al inicio del juego, de 16 o de 36 casillas.
- Se incluye de forma aleatoria 8 o 18 parejas (dependiendo el tamaño del tablero) personalizadas ubicadas en las casillas.
- Por medio del teclado se selecciona cada una de las casillas (al oprimir la tecla de la casilla correspondiente se vera el símbolo personalizado ubicado en esa casilla).
- Si se encuentra una pareja de símbolos estos permanecerán visibles hasta finalizar el juego y la pareja de teclas quedara inhabilitada. Además se incrementara el contador de puntaje.
  En caso de no acertar a la pareja, se volverán a ocultar los símbolos.
- El juego cuenta con un temporizador que se decrementara al iniciar el juego, en caso de que llegue a cero y no se hayan encontrado todas las parejas, el juego finalizara.
- Si se encuentran todas las parejas antes del tiempo y en los intentos estipulados, se podrá continuar con un nuevo tablero.

III. Diseño y modelos de solución

Se coloca la imagen de la Figura 1 para que el jugador pueda elegir por medio del teclado, el tamaño del tablero con el que desea jugar.

Figura 1: Elección tamaño del tablero

Luego se genera la imagen de la Figura 2 para que el jugador elija, igual con el teclado, el nivel de dificultad con el que desea jugar.

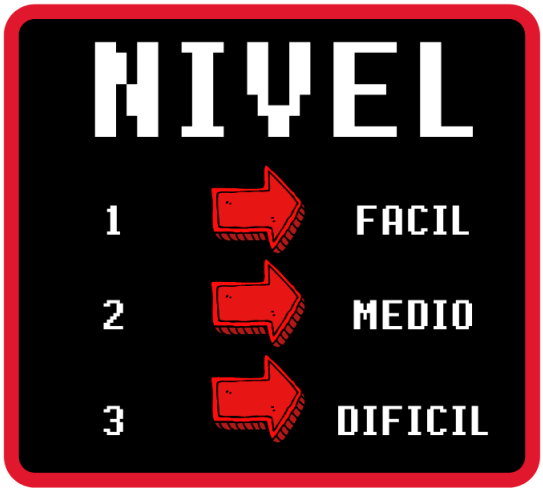

Figura 2: Elección nivel de dificultad

Para crear las parejas en los tableros y asegurarse de que estas no se repitan, es decir que solo se genere una pareja de la misma imagen/color, se selecciona de un array números del 0 al 8 o del 0 al 16 dependiendo del tamaño del tablero.
A continuación se crea un segundo array, al que se carga el arreglo anterior ordenado de forma aleatoria, para anular las probabilidades de que se genere el mismo tablero dos veces.
En la Figura 3 se muestra como se ejecutaron los arreglos para el tablero 4x4.

```
int f, i, r, p, x;
int a[8]= { 0,  1,  2,  3,  4,  5,  6,  7};
int b[8]= { 0,  0,  0,  0,  0,  0,  0,  0};

for (r = 8; r > 0; r--) // here we pick random elements
{
  x = random(r);
  p = a[x];

  for (i = x; x < r; x++) // here we shift all elements
    {   a[x] = a[x+1];}
  b[x] = p;
}
```

Figura 3: Declaración de arreglos

Se genera una matriz de parejas de forma aleatoria. Es decir, se colocan en la matriz dos ubicaciones con el mismo valor, con los números adquiridos en el array (ver Figura 12).

```
for (uint8_t z = 0; z<=7; z++){
  uint8_t pocionx = random(0,4); // crear pareja al azar
  uint8_t pociony = random(0,4); // crear pareja al azar
  uint8_t pocionx2 = random(0,4); // crear pareja al azar
  uint8_t pociony2 = random(0,4); // crear pareja al azar
  uint16_t randon = b[z+1];// Numero random selecionado.

// Crear Pareja x,y.
    if(Array[pocionx][pociony] == 0){
        Array[pocionx][pociony] = randon; //cargar en el arreglo
    }else{
      while(Array[pocionx][pociony]!=0){
        pocionx = random(0,4); // crear pareja al azar
        pociony = random(0,4); // crear pareja al azar
        if(Array[pocionx][pociony] == 0){
          Array[pocionx][pociony] = randon; //cargar en el arreglo
          break;
          }
        }
    }
    // Crear Pareja x2,y2.
    if(Array[pocionx2][pociony2] == 0){
        Array[pocionx2][pociony2] = randon; //cargar en el arreglo
    }else{
      while(Array[pocionx][pociony]!=0){
        pocionx2 = random(0,4); // crear pareja al azar
        pociony2 = random(0,4); // crear pareja al azar
        if(Array[pocionx2][pociony2] == 0){
          Array[pocionx2][pociony2] = randon; //cargar en el arreglo
          break;
          }
        }
     }
```

Figura 4: Creación de matriz de parejas

Se obtienen entonces los siguientes arreglos y la matriz de parejas:

```
 Array A
0 1 2 3 4 5 6 7
 Array B
2 7 3 4 0 1 6 5

 7       3       6       3
0       1       1       5
7       4       4       2
0       2       6       5
```

Figura 5: Resultados obtenidos

Cada numero indica un color/imagen, lo que permite mostrar los símbolos en el tablero. Para seleccionarlos se hace uso de la lectura del teclado mediante interrupciones, en donde la tecla oprimida es convertida en posiciones x, y de la matriz de juego.

```
const uint32_t * misimagenes[8] = {naruto,tokyoghoul,kuroko,KNY,katekyo,jujutsu,haikyuu,deku};
uint16_t count1 = 0;
uint16_t count2 = 0;
uint16_t count3 = 0;
for (uint8_t y = 0; y < 128; y += 32) {
  for (uint8_t x = 0; x < 128; x += 32) {
    tft.fillRect(x, y, 32, 32, BLACK); //fila, columna, altura, ancho
    dibujarImagen(x,y,32,32,misimagenes[Array[count1][count2]],0); //nivel
    tft.drawRect(x, y, 32, 32, BLACK);
    tft.setTextSize(1.5);
    tft.setTextColor(WHITE, Array[count1][count2]);
    tft.setCursor(x + 2, y + 2);
    tft.print(Str1Vals[count3]);

    count2++;
    count3++;
    if (count3 == 16) {
      count1 = 0;
    }
    if (count1 == 4) {
      count1 = 0;
    }
    if(count2 == 4){
      count1++;
      count2 = 0;
    }
  }
}
```

Figura 6: Llenado de la matriz con los símbolos personalizados

Una vez oprimidas las dos posiciones, se procede a realizar las comparaciones para verificar si se encontró o no una pareja.

La misma lógica se repite para todos los niveles del juego y los dos tamaños del tablero.

Para ganar, se necesita encontrar todas las parejas en el tablero antes de que finalice el tiempo o se terminen los intentos.
Para el nivel fácil se programo un tiempo de 120 segundos, para el nivel medio 100 segundos y para el nivel difícil 80 segundos.

En cuanto a los intentos, depende del tamaño del tablero. Para el tablero 4x4 en el nivel fácil se tienen 16 intentos, en el nivel medio 12 intentos y para el nivel difícil 8 intentos.
Para el tablero 6x6, en el nivel fácil se tienen 34 intentos, para el nivel medio 26 intentos y para el nivel difícil 18

intentos.

Una vez completado un tablero se pasa al siguiente nivel, el decir, si se elige un tablero 4x4 nivel fácil, pasara a un tablero 4x4 nivel medio. Una vez jugados y ganados los tres niveles se saltara a un tablero 6x6 en el nivel fácil.

IV. Resultados

A continuación se muestran algunas imágenes del funcionamiento del juego en los diferentes niveles y tamaños del tablero.

Figura 7: Tablero 4x4 Nivel Fácil

Figura 8: Tablero 4x4 Nivel Fácil con un acierto

Figura 9: Tablero 4x4 Nivel Medio

Figura 10: Tablero 4x4 Nivel Medio con un acierto

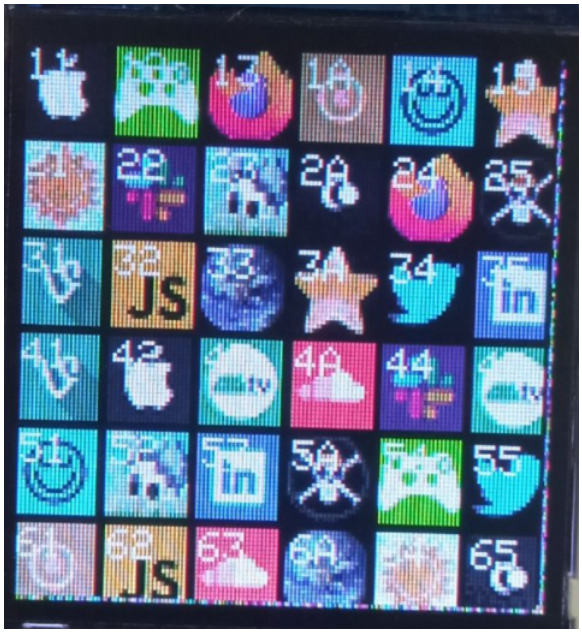

Figura 11: Tablero 6x6 Nivel Difícil

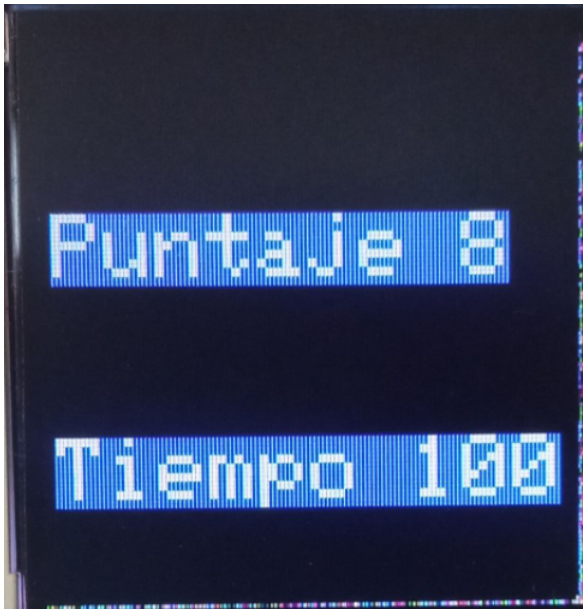

Figura 12: Pantalla puntaje obtenido y tiempo

V. Conclusiones

- Es notorio que una programación recursiva que utilice múltiples veces las mismas lineas de código ayuda a una buena ejecución.
- Es mejor no generar una mayor carga en la RAM que la necesaria, esto se puede ver al crear múltiples timmers contando hasta un millón con un prescaler de 80, se obtuvieron mejores resultados en tiempo de respuesta al utilizar un prescaler de 8.000 y contar hasta 1.000
- Al utilizar imágenes de 22x22 píxeles para el caso del tablero 6x6 notamos que la pantalla cuenta con una resolución suficiente, pero es necesario generar un contraste entre cada imagen, de esta manera poder diferenciar mejor el detalle. Sin embargo, es importante tener en cuenta que las imágenes no pueden tener mucho detalle debido al tamaño de cada una.
- El uso de periféricos aunque es posible utilizarlos con un antirebote por software, notamos que era necesario en todos los casos tener una resistencia pull-up que no permita falsas medidas.

Referencias

[1] Espressif.com. n.d. ESP32 Wi-Fi Bluetooth MCU I Espressif Systems. [online] Available at: <https://www.espressif.com/en/products/socs/esp32> [Accessed 17 August 2021].