UNIVERSIDAD DE GRANADA

GRADO DE INGENIERÍA INFORMÁTICA

PERIFÉRICOS Y DISPOSITIVOS DE INTERFAZ HUMANA

TRABAJO DE TEORÍA CURSO 2021/2022

DESARROLLO DE VIDEOJUEGO "FLAPPY BIRD" CON ARDUINO

Alumnos: Daniel Carrasco Moreno y Antonio Fernández Ortega

Índice

1. Introducción

¿Qué es Arduino?

Arduino es una compañía de **desarrollo de programa y hardware libres**, así como una sociedad mundial que diseña y manufactura placas de desarrollo de hardware para crear dispositivos digitales y dispositivos interactivos que logren identificar y mantener el control de objetos de todo el mundo real.

Los productos que vende son distribuidos como Hardware y Programa Independiente, bajo la Licencia Pública Gral. de **GNU (GPL)** y la Licencia Pública Gral. Limitada de **GNU (LGPL),** permitiendo la manufactura de las placas Arduino y repartición del programa por cualquier sujeto.

Principalmente el hardware radica de un **microcontrolador Atmel AVR**, conectado bajo la configuración de "sistema mínimo" sobre una placa de circuito impreso a la que se le tienen la posibilidad de conectar placas de extensión (shields) por medio de la disposición de los puertos de ingreso y salida presentes en la placa escogida.

El programa de Arduino se basa principalmente en 2 recursos:

- Un ámbito de desarrollo (**IDE**) (basado en el **ámbito de processing** y en la composición del lenguaje de programación **Wiring**).
- El cargador de arranque (**bootloader**) que es ejecutado de manera automática dentro del microcontrolador en cuanto este se enciende.

Comentado [1]: Processing es un lenguaje y un IDE basado en Java que se utiliza para la producción de proyectos multimedia de diseño digital.

Su principal objetivo era el de servir de herramienta a artistas ajenos a la programación, para que aprendieran las bases del lenguaje de manera gráfica instantánea y visual de la información.

Básicamente, es un lenguaje que permite la creación de programas de manera visual, incluso usando Realidad Aumentada(VR)

Comentado [2]: Wiring es una plataforma iniciada en 2004 por Hernando Barragán, cuya idea era crear una plataforma para programar y generar prototipos de circuitos electrónicos. Con Wiring podemos controlar multitud de dispositivos conectados a un microcontrolador.

El IDE utiliza como lenguaje principal, C/C++.

Un poco de historia… Flappy Bird

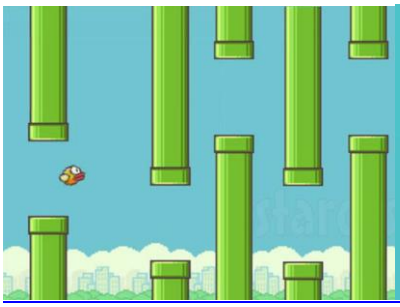
Flappy Bird fue un juego para dispositivos móviles desarrollado por Nguyen Hà Đông (Dong Nguyen) por .GEARS Studios, un pequeño desarrollador independiente. El juego fue publicado el 24 de mayo de 2013.

El juego en sí es bastante sencillo, el jugador controla un pájaro intentando volar entre filas de tuberías verdes y el objetivo es pasar a través de ellas sin tocarlas. La escena se va desplazando lateralmente conforme va avanzando el pájaro.

El juego rápidamente alcanzó el éxito ya que llegó a ser el más descargado en todas las tiendas de apps, para sorpresa del creador. Pero esto tuvo un giro inesperado de los acontecimientos…

El 9 de febrero de 2014, el creador anuncio en su cuenta de Twitter que eliminaría el juego de todas las tiendas ya que debido a su gran éxito, Nguyen llegó a recibir hasta **90.000 dólares al día** con un pequeño **banner** publicitario que aparecía en la parte inferior de la pantalla y sintió que su persona estaba en una posición complicada.

Aunque más tarde afirmó en una entrevista con Rolling Stone que la razón principal fué que había recibido e-mails de fans alegando que su juego era tan adictivo como el crack y causaba bastante frustración, algo que se aleja bastante de la intención inicial de su creador: disfrutar y relajar a sus jugadores.



2. Desarrollo del proyecto

Nosotros vamos a realizar una implementación bastante sencilla y básica de Flappy Bird pero conservando la esencia principal del videojuego: esquivar una serie de obstáculos que avanzan hacia nosotros.

En primer lugar, como soporte hardware, hemos utilizado un kit de Arduino Uno del servicio de préstamo de la biblioteca de la E.T.S.I.I.T, ya que para este sencillo proyecto, no necesitamos demasiados componentes.

Este trabajo es una versión “low-cost” del Flappy Bird original con el mínimo número de componentes y con una programación bastante sencilla.

Concepto principal del juego en Arduino

La idea principal para desarrollar el videojuego ha sido sencilla.

El pájaro ha sido representado con el carácter “0” y las tuberías/obstáculos con el carácter “#”.

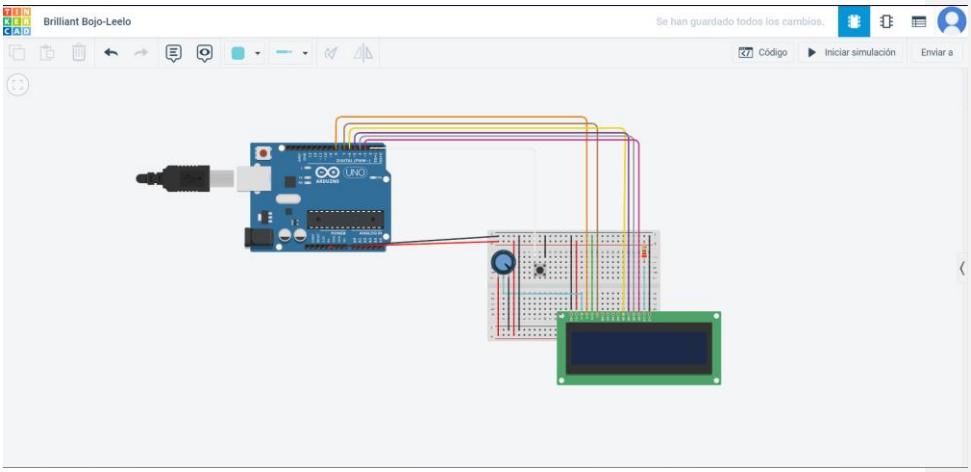
Hemos declarado dos variables string con diferentes combinaciones del carácter “#”, uno para la fila superior y otra para la fila inferior de la pantalla.

El objetivo principal del juego es pasar el “pájaro” (el 0) a través de los obstáculos simplemente pulsando el botón.

Materiales utilizados

- Arduino UNO motherboard
- 16x2 LCD Display
- Push Button
- Connecting Wires
- Breadboard
- 5V Power Supply
- Potenciómetro de 10 kohmios
- Resistencia de 220 ohmios

Diseño del circuito



Img 1: Esquema del circuito completo a través de *Tinkercad*

Como hemos mencionado anteriormente en la lista de componentes, vamos a necesitar únicamente una pantalla LCD de 16x2 , la placa de Arduino Uno y un botón interruptor.

Además, para regular el contraste de la pantalla, vamos a hacer uso de un potenciómetro de una resistencia de 10 kohmios.

Al realizar un primer montaje , nos dió problemas el potenciómetro para la pantalla.

Entonces lo que hicimos fue añadir una resistencia de 220 ohmios conectada al pin 15 de la pantalla, que se corresponde con LED de ánodo, ya que a dicho LED le llegaba una corriente superior a su límite máximo. La corriente sin la resistencia era de 23.7 kA y el máximo permitido para ese LED era de 20 mA. Colocando dicha resistencia de 220 ohmios, el problema quedaba solucionado y todo funcionaba correctamente.

Respecto a las conexiones de la pantalla LCD:

- Conectamos las patillas RS y E a los pines digitales 8 y 7 de la placa Arduino UNO. Y los pines de datos D4-D7 se conectan a los pines digitales de E/S : 6,5,4 y 3 de la placa de Arduino Uno.

En cuanto al botón va conectado entre el pin 2 de la placa base y tierra.
El resto de conexiones se puede apreciar fácilmente en **img 1**.

3. Código utilizado: Explicación

Inicialización de variables:

```
#include<LiquidCrystal.h>
LiquidCrystal lcd (8,7,6,5,4,3); //RS, E, D4, D5, D6, D7
int ledPin = 13;
int buttonPin = 2;

int ledToggle;
int previousState = HIGH;
unsigned int previousPress;
volatile int buttonFlag;
int buttonDebounce = 20;
int i=0;
int pos=0;
int Speed=100;
String typea    = "                    ##        ######     ##       ####     ####       ########       ##                ";
String typeaa   = "                ##       ####          ##     ####       ##         ##           ##                    ";


int  typeA[]    = {22,23,32,33,34,35,36,37,43,44,52,53,54,55,61,62,63,64,72,73,74,75,76,77,78,79,87,88};//28
int typeAA[]    = {17,18,26,27,28,29,40,41,47,48,49,50,58,59,69,70,82,83};//18
```

Respecto a las variables, cabe destacar que guardamos dos Strings con la disposición de los obstáculos (representados con #) y luego 2 vectores que almacenan la posición de cada String. El superior ocupa 28 posiciones y el inferior 18.

```
void setup()
{
  lcd.begin(16,2);
  lcd.begin(16,2);
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(2), button_ISR, CHANGE);
  while(ledToggle==0)
  {
    lcd.clear();
    delay(500);
    lcd.print("   START GAME   ");
    delay(500);
  }
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print('O');
  //lcd.print(bala.substring(1,3));

}
```

img 2: Función setup()

Función setup():

La función setup, como sabemos, es la primera función que se ejecuta en el programa y se utiliza para configurar, inicializar variables, comenzar a usar librerías, etc… Esta función es el núcleo de todos los programas de Arduino y se usa para el control activo de la placa.

En primer lugar, se inicializa el número de celdas y filas de la pantalla LCD, en este caso 2 filas y 16 columnas.

Luego, únicamente muestra START GAME en la pantalla hasta que no pulsemos el botón, que entonces, dará comienzo el juego.

En la función **attachInterrupt(digitalPinToInterrupt(2), button_ISR, CHANGE)**, lo que hacemos es establecer que vamos a leer la información del botón del pin número 2 de los pines digitales e/s de la placa base de Arduino, y traducir el pin digital real al número (int) de interrupción correspondiente. Para ello, vamos a hacer uso de la función button_ISR que realizará los cambios de estado del botón .

El último parámetro indica que dispare la interrupción (es decir, el cambio de posición del pájaro) cuando detecte que ha cambiado el valor del pin número 2 correspondiente al botón.

Función loop()

```
void loop()
{

while(1)
 {
   for(i=0;i<89;i++)//
   {
    if(pos==0)
    {
      lcd.setCursor(1,0);
      lcd.print(typea.substring(i+1,16+i));
    }
    else
    {
      lcd.setCursor(0,0);
      lcd.print(typea.substring(i,16+i));
    }


    if(pos==1)
    {
      lcd.setCursor(1,1);
      lcd.print(typeaa.substring(i+1,16+i));
    }
    else
    {
      lcd.setCursor(0,1);
      lcd.print(typeaa.substring(i,16+i));
    }
```

```
    for(int j=0;j<28;j++)
    {
      if(typeA[j]==i)
      {
        if(pos==0)
        {
          gameover();
        }
      }
    }
    for(int j=0;j<18;j++)
    {
      if(typeAA[j]==i)
      {
        if(pos==1)
        {
          gameover();
        }
      }
    }
    delay(Speed);
  }
   winner();
 }
}
```

En la función loop básicamente, detecta cuando el pájaro está arriba o abajo y cambia su posición en pantalla. Luego, en 2 bucles detecta cuándo el pájaro toca un obstáculo según se encuentre en la fila de arriba o en la de abajo.

En caso de tocar un obstáculo, hemos perdido la partida y se llama a la función gameover().

Función gameover():

```
void gameover (void)
{
  detachInterrupt(digitalPinToInterrupt(2));
  while(1)
  {
    lcd.clear();
    delay(500);
    lcd.print("    GAME OVER   ");
    delay(500);
  }
}
```

Si pasamos todos los obstáculos, es decir, no entramos en ningún momento en la condición if(pierdo) de los 2 bucles (el que controla la parte de arriba y la de abajo) , hemos ganado y se llama a la función winner().

Función winner():

```
void winner (void)
{
  detachInterrupt(digitalPinToInterrupt(2));
  while(1)
  {
    lcd.clear();
    delay(500);
    lcd.print(" WINNER! WINNER!");
    lcd.setCursor(0,1);
    lcd.print(" CHICKEN DINNER ");
    delay(500);
  }
}
```

Función button_ISR

```
void button_ISR()
{
   buttonFlag = 1;
   if((millis() - previousPress) > buttonDebounce && buttonFlag)
  {
    previousPress = millis();
    if(digitalRead(buttonPin) == LOW && previousState == HIGH)
    {
      ledToggle =! ledToggle;
      digitalWrite(ledPin, ledToggle);
      previousState = LOW;
      if(ledToggle)
      {
       lcd.setCursor(0,0);
       lcd.print('O');
       lcd.setCursor(0,1);
       lcd.print(' ');
       pos=0;
      }
      else
      {
       lcd.setCursor(0,1);
       lcd.print('O');
       lcd.setCursor(0,0);
       lcd.print(' ');
       pos=1;
      }

    }

    else if(digitalRead(buttonPin) == HIGH && previousState == LOW)
    {
      previousState = HIGH;
    }
    buttonFlag = 0;
  }
}
```

Esta función se llama en el setup para actualizar el valor del botón. En resúmen, lo que hace esta función es detectar las pulsaciones en el botón, guardar el estado previo y actualizarlo a HIGH o a LOW, para saber cuándo se ha pulsado y por tanto, cuándo se debe “subir” o “bajar” nuestro pájaro en la pantalla.

4. Conclusión y posibles mejoras

Una vez comprobado el funcionamiento del juego, pudimos observar que una vez finalizado la partida, ya hayamos ganado o perdido, no se reiniciaba el juego para poder jugar de nuevo, sin necesidad de reconectar el arduino desde el USB.

Para ello, hemos establecido las siguientes modificaciones:

```
void (*myReset) (void) = 0x0;

void apagar(){

  lcd.clear(); lcd.print("Restart"); delay(1000); lcd.clear(); myReset();

}
```

Con esto lo que hacemos es reiniciar nuestro programa para que vuelva a ejecutar la función setup() y por tanto vuelva a mostrar el “START GAME” para iniciar una nueva partida.

Esta nueva función se llamaría desde las funciones gameover y winner:

```
void gameover (void)
{
  detachInterrupt(digitalPinToInterrupt(2));
  lcd.clear();
  while(1)
  {
    lcd.clear();
    delay(500);
    lcd.print("   GAME OVER   ");
    delay(500);
    apagar();
  }
}
```

```
void winner (void)
{
  detachInterrupt(digitalPinToInterrupt(2));
  while( digitalRead(buttonPin) == previousState  )
  {
    lcd.clear();
    delay(500);
    lcd.print(" WINNER! WINNER!");
    lcd.setCursor(0,1);
    lcd.print(" CHICKEN DINNER ");
    delay(500);
    apagar();
  }
  previousState = HIGH;
}
```

Entonces, una vez acabada la partida, se reinicia la pantalla, mostrando “Restart” y volvería a lanzar el mensaje de bienvenida esperando a que pulsemos el botón de nuevo y volvería a empezar.

Con esta mejora logramos una ejecución en bucle y más natural de cómo funcionaría un juego de plataformas de este tipo en la realidad, sin necesidad de enchufar y desenchufar el cable de Arduino.

Segunda modificación

Tras ejecutar el juego unas cuantas veces, hemos detectado que los obstáculos se desplazaban demasiado rápido para poder esquivarlos y, en resúmen, hacía el juego prácticamente “imposible” de ganar.

Para ello, lo que hemos hecho ha sido modificar el parámetro Speed,(que es utilizado como “delay” al finalizar cada iteración del bucle principal del programa) aumentando su valor de 100ms a 200ms. Esto se traducirá en un avance más pausado de los obstáculos, lo cuál permite una jugabilidad más equilibrada.

```
#include<LiquidCrystal.h>
LiquidCrystal lcd (8,7,6,5,4,3); //RS, E, D4, D5, D6, D7
int ledPin = 13;
int buttonPin = 2;

int ledToggle;
int previousState = HIGH;
unsigned int previousPress;
volatile int buttonFlag;
int buttonDebounce = 20;
int i=0;
int pos=0;
int Speed=200;
String typea    = "                    ##       ######    ##       ####     ####      ########      ##                ";
String typeaa   = "               ##      ####         ##    ####      ##        ##          ##                      ";
```

Modificación de la inicialización de la variable Speed.

5. Bibliografía

- https://www.electronicshub.org/flappy-bird-game-using-arduino/
- https://www.tinkercad.com/things/cKVzIVYfCdo-flappy-bird-pdih
- https://www.tinkercad.com/things/d5Q0JM9nYrG-flappy-bird