

Universidad Politécnica de Madrid

Grado en Ingeniería Electrónica y Automática

Asignatura

**INFORMÁTICA**

Curso 2018-2019



Universidad Politécnica de Madrid

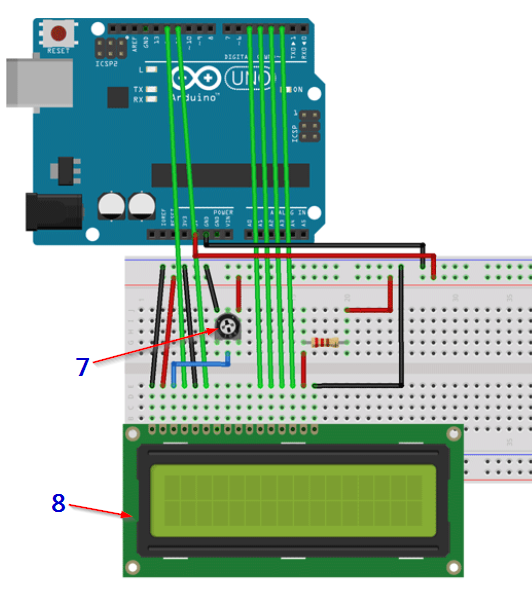
Grado en Ingeniería Electrónica y Automática

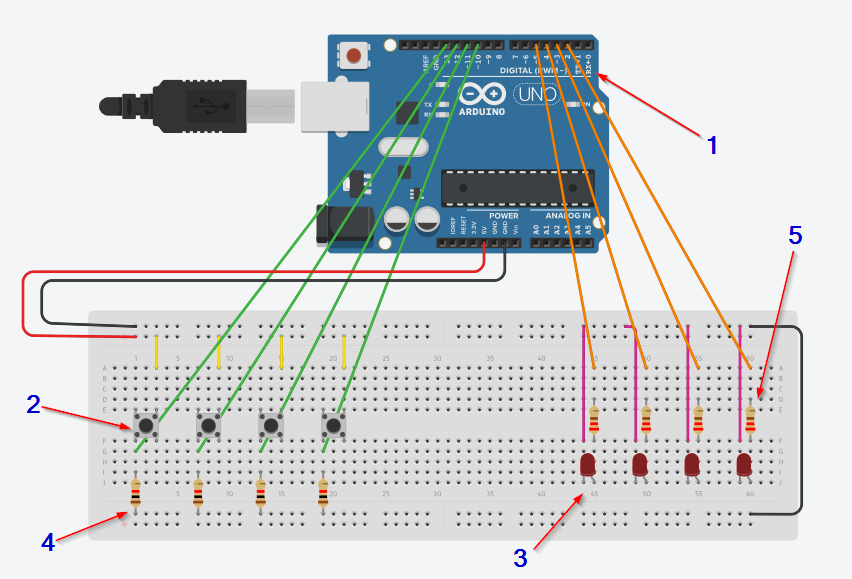
Datos del Grupo

*Javier Pío González Alday (54639)*

*Jesús García Sánchez (54622)*

# Introducción

El proyecto realizado consiste en la programación de 2 juegos con cierta similitud**: SIMÓN DICE** y **DESCIFRA EL CÓDIGO**. Ambos utilizan una placa de Arduino UNO junto una distribución de componentes electrónicos concreta (ver en la imagen), haciendo posible el transcurso de las partidas mediante básicamente 4 leds y 4 botones. Además, se ha incluido una pantalla LCD con el fin de comunicarse con el jugador durante la partida.



Componentes utilizados:

1 . – Placa Arduino UNO.

2 . – 4 pulsadores (normalmente cerrados).

3 . – 4 diodos LED.

4 . – Resistencias de 220 ohmios.

5 . – Resistencias de 1000 ohmios.

6 . – Protoboard.

7 . – Potenciómetro (10K).

8 . – Pantalla LCD. (La disposición de la imagen sirve para hacer una idea de la disposición del LCD, aunque los pines utilizados no coinciden.)

# Desarrollo

**SIMÓN DICE:**

Inspirado en el juego infantil de imitación de gestos, este programa se basa en mostrarle al jugador una sucesión de leds, que se encienden y apagan sucesivamente, para que acto seguido el jugador reproduzca dicha secuencia a través de los botones de la placa.

En caso de que el jugador acierte la secuencia, pasará a la siguiente ronda; en el caso de que la secuencia introducida por el jugador sea errónea, este habrá perdido el juego.

A medida que pasan las rondas y con el fin de aumentar la dificultad, en cada ronda se aumentará en 1 los leds a recordar de la secuencia, llegando a un máximo de rondas por partida, que el usuario determinará antes de comenzar la partida.

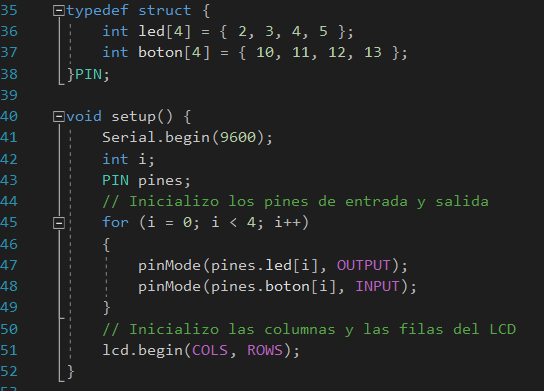
Observaciones:

* Debido al funcionamiento interno de la placa, cada vez que perdamos, nuestro programa se reiniciará y comenzará una nueva partida.
* La reducida memoria SRAM de la placa (Arduino UNO) a condicionado que a la hora de la utilización de vectores de tamaño variable no podamos utilizar la función “realloc” con este fin. Como solución al problema, se ha programado un menú con el que se decide que cantidad de rondas se jugarán.

Respecto a la comprensión del código:

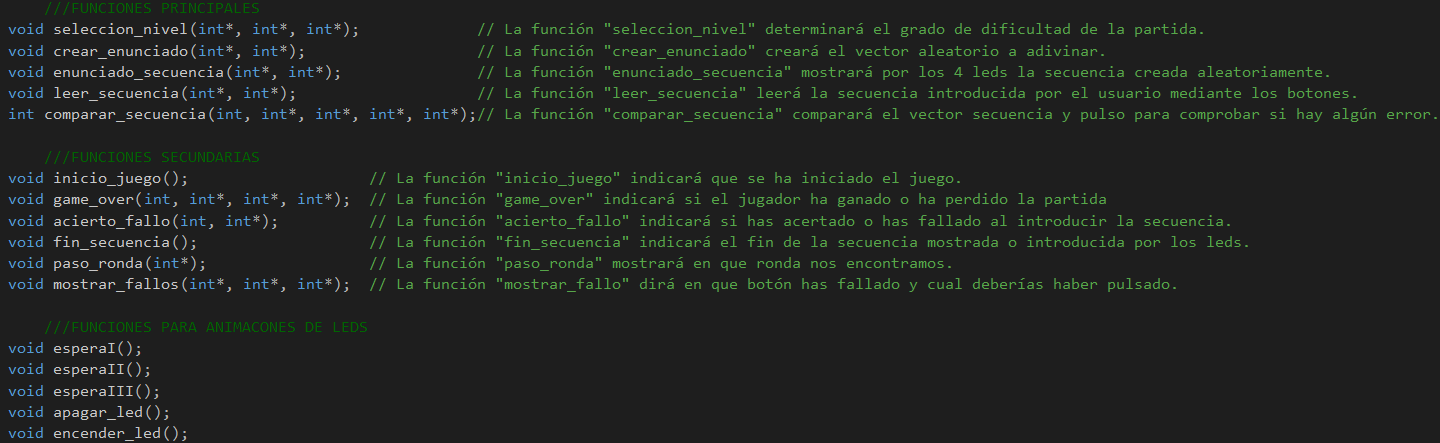
Hay varias consideraciones que se deberían tener en cuenta a la hora de comprender el código:

A la hora de inicializar los pines de entrada y salida se ha optado por crear un tipo de variable “PIN” con 2 vectores con pines a inicializar para que más tarde en el “setup()”. De esta manera no es necesario declarar variables globales y la inicialización es más limpia.



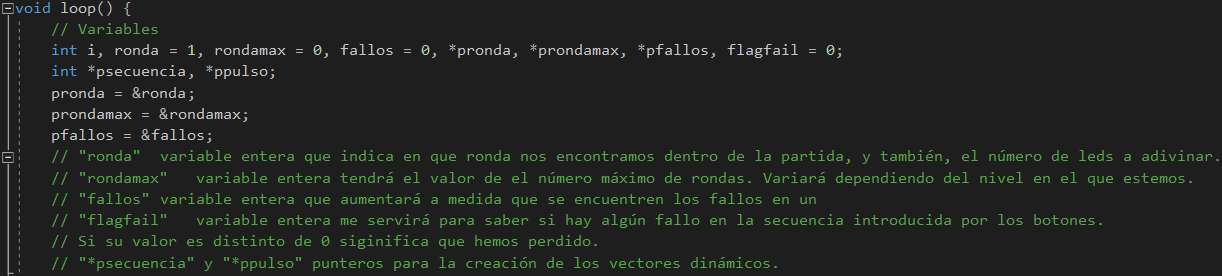
La estructura de este código consiste en la utilización de **distintas funciones** que podemos clasificar en **4 bloques**:

* **Funciones propias de Arduino**, tales como “void setup()” o “void loop()”.
* **Funciones principales**, con las que se realiza el flujo principal del programa.
* **Funciones secundarias**, que se utilizan con el fin de controlar las distintas fases en las que nos encontramos durante la partida.
* **Funciones de animaciones de leds**, con menor importancia que las anteriores.

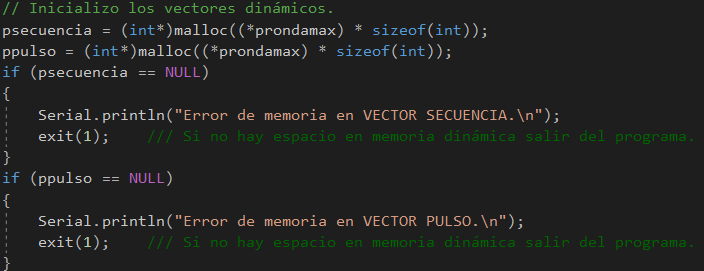


También se remarca el uso de la librería “LiquidCrystal.h”, con el fin de agilizar la escritura de mensajes a través de **la pantalla LCD 16x2** antes mencionada. Las funciones relacionadas con esta librería son “lcd.clear()” con la que vacío de caracteres el monitor, “lcd.setCursor()” con la que decido donde escribir el mensaje , y “lcd.print()” con la que escribo los mensajes.

Otra cosa para tener en cuenta son los **mensajes que recibe el usuario desde el Arduino** durante la partida dependiendo de sus jugadas. Este programa está íntegramente diseñado para que el usuario interactúe con la disposición de componentes sin necesidad de mirar nunca la pantalla del ordenador. Pero cabe destacar que podemos encontrar ciertas funciones tales como “Serial.print()” o “Serial.println()” con las que escribimos en el monitor serial del Arduino mensajes con el fin de comprobar que el flujo de juego se está realizado de manera óptima. **Estos mensajes no son necesarios para el jugador**, ya que en algunos casos contiene las soluciones de la ronda en la que se encuentra.

Las **variables** inicializadas en la función “loop()” son solamente del tipo entero, incluyendo sus punteros del mismo tipo. Mediante los punteros se irán variando los valores de esta mediante las funciones de paso por referencia.

Flujo del programa:

La idea principal del programa es **comparar 2 vectores** de la misma longitud. El primero (“psecuencia”) de ellos estará **creado aleatoriamente**, con la información de los leds que se encenderán y que tendremos que adivinar. El otro (“ppulso”) tendrá la misma longitud, pero estará inicialmente vacío. Este lo iremos rellenando con **la información de los botones pulsados** por el jugador. Ambos vectores se crean mediante asignación dinámica de memoria, con el tamaño determinado por el usuario en un principio.

Estos vectores están rellenados con números del 0 al 3, siendo estas las posiciones de los vectores que contienen los pines concretos de los leds y los botones.

Ya declaradas las variables y creados los vectores, **comenzaría la partida**. Primero se rellenaría el vector “psecuencia” aleatoriamente mediante la función “crear\_enunciado”.

Acto seguido entraríamos en un bucle “while” del que para salir de él se pueden dar **dos casos**: llegamos al **número máximo de rondas** (rondamax), lo que significaría que hemos ganado; o la variable “flagfail” toma un **valor distinto de 0** (1) a causa de haber fallado al introducir nosotros la secuencia por los botones.

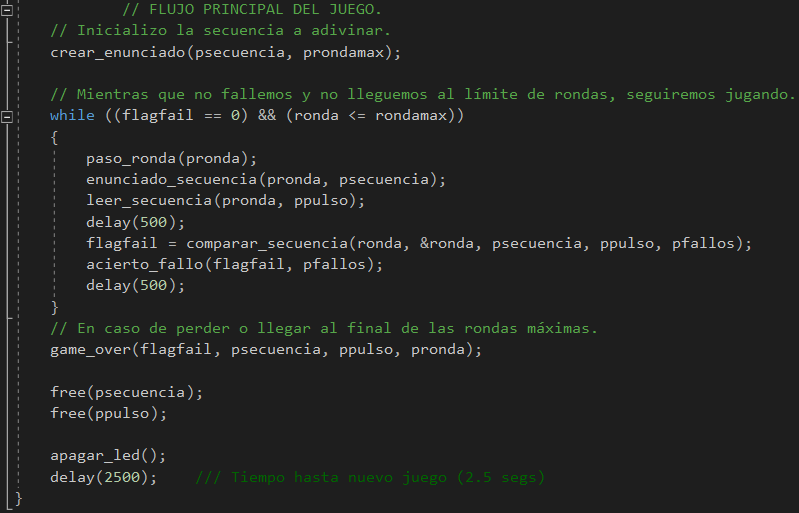
Dentro de este bucle están las funciones que irán desarrollando el transcurso del juego. Después de indicar en que ronda estamos (paso\_ronda), la función “enunciado\_secuencia” mostrará los leds a recordar encendiéndolos y apagándolos en la protoboard.

A continuación, se encuentra la función “leer\_secuencia” que se encargará de crear el vector “ppulso” a partir de los botones pulsdos por el usuario.

Para decidir si hemos acertado o no, utilizamos la función “comparar\_secuencia”, a la que le introducimos la ronda en la que está el jugador, los vectores principales del programa y una variable entera que contabilizará los fallos en caso de existir. El valor que retorna esta función puede ser 0 (no hay fallos) o 1 (hay uno o varios errores).

Finalmente, la función “acierto\_fallo” se encarga de informarle al usuario ya sea por el LCD o por el monitor serial si ha cometido algún error o no.

Al salir del bucle, de cualquiera de las 2 formas posibles, se ejecuta la función “game\_over”, que dependiendo del valor de “flagfail” detecta si hemos ganado o perdido.

Al llegar al final del “loop()”, los vectores se liberarán y todas las variables se reiniciarán, volviendo al inicio del código, y comenzando así una nueva partida.

PD: Los detalles concretos de cada función están comentados dentro del código entregado.

**DESCIFRA EL CÓDIGO:**

El segundo juego del proyecto consiste en una versión simplificada del juego anterior, pero con unas ciertas variaciones en las reglas y en las funciones.

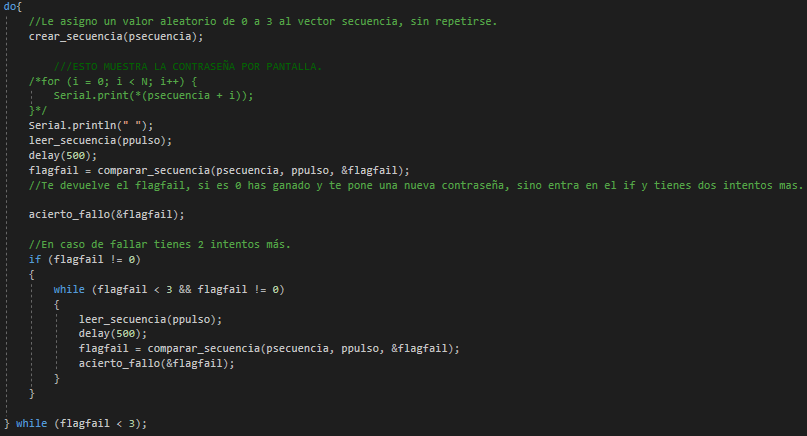
El jugador tendrá que adivinar contraseñas de 4 dígitos pulsando los botones dispuestos en la protoboard e utilizando la información mostrada por el LCD y el monitor serial conjuntamente.

Estas contraseñas de 4 dígitos se crean de manera aleatoria y con la particularidad de que ninguno de sus dígitos pueda repetirse. Es decir, solo pulsaremos cada botón una vez por ronda.

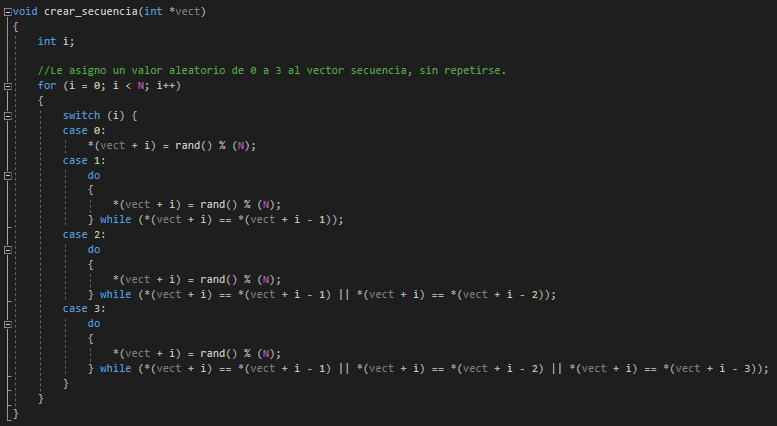
En cada ronda se crea una nueva contraseña, y el usuario tiene 3 intentos para adivinarla. En caso de fallar al introducir una contraseña, mediante el LCD el usuario podrá llevar la cuenta de los intentos restantes, mediante el monitor serial se le dará información de que botones son los que ha pulsado erróneamente. En caso de no acertar el código en ninguno de los intentos, el jugador habrá perdido la partida. En cambio, si se pasa de ronda los intentos se reiniciarán.

Respecto a la comprensión del código:

Debido a la gran similitud con el primer código, comentaremos algunos de los elementos que los diferencian entre sí.

Las principales diferencias consisten en que ahora solamente tenemos 2 vectores de 4 elementos cada uno a comparar, y que el jugo, en caso de acertar siempre, sería infinito, ya que el único criterio de salida del bucle principal del programa en consumir todos los intentos de una ronda.

Como se puede observar, la manera de implementar los intentos en el juego se lleva a cabo con el valor de la variable “flagfail” que irá cariando de valor entre 0 y 3 dependiendo de los fallos que hayamos cometido en la ronda. Cundo fallamos la primera vez, entramos en un “while” dentro de un “if” , del que no saldremos hasta que “flagfail” no se reinicie a 0 al introducir la contraseña correcta.

Otro cambio a remarcar es la manera en la que se rellena el vector secuencia. Este tendrá 4 elementos del 0 al 3 que no se podrán repetir en la misma contraseña.

Con este fin creamos una función (crear\_secuencia), que dependiendo del valor que se esté rellenado compararemos los valores anteriores del vector que se está creando y no rellenaremos el siguiente elemento hasta que no tenga un valor distinto a los valores que le preceden.

El flujo principal del juego es similar al del Simón Dice: Se crea el vector aleatorio, se introduce el segundo vector por los botones y finalmente se comparan ambos.

# Conclusiones

La realización de este trabajo ha supuesto de una gran dedicación por parte de los integrantes del equipo, ya que ninguno de los dos estaba familiarizado con este tipo de componentes (Arduino). Debido a esto, en un principio la gran mayoría del tiempo se dedicó a la búsqueda de información acerca del funcionamiento de la placa y componentes externos a esta. Pasada esta fase, a la hora de programar fue difícil encontrar la manera de juntar los conocimientos de lenguaje C adquiridos en clase y el Arduino, pero finalmente nos dimos cuenta de su similitud.

Este trabajo nos ha motivado realmente en el ámbito de la electrónica, y por ello estamos deseando poder profundizar en dicho tema, ya sea con el Arduino o con cualquier componente parecido, para aumentar nuestro conocimiento en la materia.

# Bibliografía

* Simulador de Arduino: <https://www.tinkercad.com>
* Pantalla LCD: <https://programarfacil.com/tutoriales/fragmentos/arduino/texto-en-movimiento-en-un-lcd-con-arduino/>
* Y en la gran mayoría foros como: <https://forum.arduino.cc/>