Juego: mide tu capacidad de reacción

Por Koldo Olaskoaga

Fotos por Koldo Olaskoaga

En este artículo se presenta el proceso de creación de un programa sencillo que mide la capacidad de reacción del usuario. Se utiliza como entorno de programación Open Lab Roberta.

Reto

Crear un dispositivo que permita medir la velocidad de reacción ante un estímulo luminoso.

Para resolver este reto voy a utilizar el mismo dispositivo de juegos que utilicé en el artículo sobre LEGO MINDSTORMS del número 15 de Hispabrick Magazine que posteriormente completé y publiqué con más detalle en Issuu[1]: tres botones de diferentes colores con tres luces de los mismos colores conectados todos ellos al NXT. Sin embargo, lo he actualizado al EV3 tal y como puede verse en la imagen y en este caso utiliza la luz propia del EV3.

Como estímulo utilizaré la luz del propio EV3 y tras pulsar la tecla correspondiente se mostrará en pantalla el tiempo conseguido.

Programa

El objetivo final va a ser crear un programa que tenga las siguientes características:

- Se encenderá una luz verde, naranja o roja y habrá de pulsarse la tecla correspondiente.
- La luz se encenderá en un margen de tiempo entre 2 y 5 segundos.
- Se podrán hacer tres intentos y tras cada uno se mostrará el tiempo conseguido en el intento y el mejor de los conseguidos hasta el momento.
- Se muestran las instrucciones necesarias en la pantalla.

Empezar con todo ello a la vez no es recomendable, siempre conviene dividir los retos en pequeños retos más sencillos de resolver para poco a poco combinarlos y crear algo más complejo. Por eso vamos a empezar con el que corresponde a un solo botón cubriendo objetivos paso a paso.

En todo caso, antes de crear el programa en el lenguaje elegido, conviene escribir el algoritmo correspondiente en lenguaje natural, es decir, en nuestro propio lenguaje.

Paso nº 1

Objetivo: cuando se encienda la luz verde habrá que pulsar el botón verde y a continuación se mostrará el tiempo transcurrido en la pantalla.

Veamos cuales son los pasos a seguir, es decir, el algoritmo:

1. Crear una variable para poder almacenar el tiempo pasa hasta pulsar la tecla
2. Encender la luz verde (EV3)
3. Poner en marcha el cronómetro (los temporizadores o cronómetros en el EV3 están en marcha continuamente, así que lo que hay que hacer es ponerlo a cero cuando se quiere empezar a contar el tiempo)
4. Esperar hasta que se pulse el botón verde (puerto 1 del EV3)
5. Almacenar el valor de lectura del cronómetro en una variable (los temporizadores no se pueden detener, pero es posible asignar a una variable su valor cuando se desee, de modo que se pueda registrar el valor en el momento deseado).
6. Mostrar la lectura en la pantalla

Una vez que tenemos el algoritmo, ya lo podemos convertir en un programa en el lenguaje que se desee. En este caso voy a utilizar Open Roberta Lab (NEPO de aquí en adelante), sobre el que se publicó un artículo en el anterior número de Hispabrick Magazine, aunque hacerlo con EV3-G sería muy similar. Hay que seleccionar el modo NEPO-bloques experto ya que hay algún bloque no presente en el modo principiante.

El programa: en NEPO las variables se definen en el mismo bloque de inicio del programa, para lo que solo hay que pulsar sobre el signo más. De este modo podremos darle un nombre, indicarle de qué tipo de variable se trata (hay 9 tipos disponibles) y asignarle un valor inicial.

Tras encender la luz verde del EV3, hay que poner a cero el temporizador, hay cinco disponibles y en este caso vamos a utilizar el número 1.

El proceso de creación de la instrucción que va a pausar la ejecución del programa hasta que se pulse la tecla verde (que está sobre el sensor de contacto conectado al puerto 1) es muy similar al utilizado en Scratch. Aquí utilizamos el bloque **esperar hasta**, que se encuentra en el menú **Control** > **Esperar** al que hay que conectar una operación lógica. En este caso es ver si el sensor de contacto conectado al puerto 1 está pulsado.

Una vez que el programa pueda continuar adelante asignará a la variable Tiempo el valor que en ese momento tenga el temporizador número 1, un valor que se encontrará en milisegundos.

Ya solo queda mostrar en pantalla el tiempo obtenido.

Y ahora toca ejecutarlo y observar lo que pasa que es lo siguiente:

- la luz verde se enciende nada más iniciar el programa
- No muestra la lectura en la pantalla ya que nada más pulsar la tecla verde el programa finaliza.

La razón es que no hemos marcado los tiempos de ejecución que nos interesan, no le hemos pedido que espere un poco para encender la luz verde tras la puesta en ejecución y no le hemos dado tiempo para poder ver la lectura en pantalla, porque de hecho sí la ha mostrado, pero una vez la ha mostrado, el programa ha finalizado de inmediato y no la hemos podido apreciar. Son dos aspectos que vamos a mejorar introduciendo varios cambios.

Qué vamos a cambiar: una vez que el programa se inicie esperará un periodo de tiempo aleatorio entre 2 y 5 segundos antes de encender la luz y al final del programa incluiremos una espera de 5 segundos para que nos de tiempo de ver el tiempo conseguido (los cambios aparecen en negrita en el algoritmo).

1. Crear una variable para poder almacenar el tiempo
2. **Esperar un tiempo aleatorio entre 2 y 5 segundos**
3. Encender la luz verde
4. Poner en marcha el cronómetro
5. Esperar hasta que se pulse el botón verde (puerto 1)
6. Almacenar el valor de lectura del cronómetro en una variable
7. Mostrar la lectura en la pantalla
8. **Esperar 5 segundos**

Un tiempo aleatorio entre 2 y 5 segundos es como lanzar un dado para ver cómo cae, solo que en este caso el número de opciones es mucho más alto. Dado que el bloque de control **esperar** pide que el tiempo se dé en milisegundos habrá que calcular un número entre 2000 y 5000, aunque también se puede calcular u número entre 2 y 5 y a continuación multiplicar el resultado por 1000.

Y en la parte final del programa se añade un bloque **esperar** para poder leer el resultado.

Puede observarse que los bloques que ya estaban en el primer programa han sido contraídos de tal modo que ocupan menor espacio. Ello se puede hacer haciendo clic con el botón derecho sobre el bloque que se desee contraer.

Al ejecutar el nuevo programa podemos ver que se han solucionado los dos problemas anteriormente mencionados y que hemos conseguido de modo satisfactorio responder a lo que se pedía en el paso nº 1. Sin embargo, en el reto se pide que se puedan jugar hasta tres rondas seguidas, así que pasemos al siguiente paso.

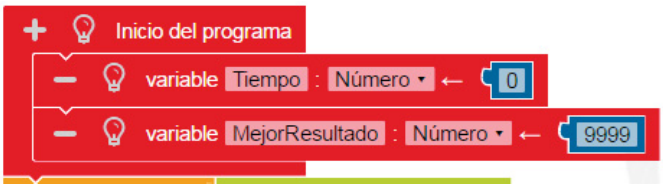
Paso nº 2

Objetivo: cuando se inicie el programa aparecerá en pantalla el nombre del juego y un mensaje que pida pulsar la tecla verde para seguir. Tras una espera entre 2 y 5 segundos se encenderá la luz verde y al pulsar la tecla verde se mostrará en pantalla el tiempo de reacción y el mejor tiempo conseguido. A los dos segundos el programa continuará. Todo esto lo hará tres veces antes de acabar.

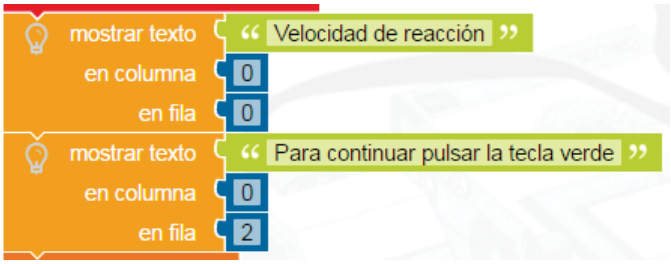
El nuevo algoritmo será el siguiente:

1. Crear las variables Tiempo y MejorResultado
2. Mostrar en pantalla el nombre del juego y un texto que pida pulsar la tecla verde para continuar
3. Repetir los siguientes pasos
    - **a. Esperar a que pulse la tecla verde**
    - b. Esperar un tiempo aleatorio entre 2 y 5 segundos
    - c. Encender la luz verde
    - d. Poner en marcha el cronómetro
    - e. Esperar hasta que se pulse la tecla verde
    - f. Almacenar el valor de lectura del cronómetro en una variable
    - **g. Comparar la nueva lectura con la almacenada en la variable MejorResultado**
        - **i. Si el valor de la variable Tiempo es menor que el valor de MejorResultado almacenar en MejorResultado el nuevo valor**
    - **h. Mostrar la lectura en la pantalla, así como el mejor resultado hasta el momento**
    - i. Esperar 2 segundos

Aunque en el paso anterior ya hemos visto cómo crear una variable e inicializarla, vamos a ver aquí una particularidad sobre la nueva variable MejorResultado. Esta variable ha de almacenar el mejor resultado que se consiga entre todos los intentos, para lo que en cada uno de ellos comparará el tiempo obtenido con su valor (paso 3g del algoritmo). Para ver si ha habido mejora comparará el nuevo resultado con su valor, y si es menor lo sustituirá. Así que de valor inicial hay que asignarle un valor lo suficientemente grande para que tras el primer intento, haga la comparación y al ser menor que su valor inicial lo sustituya por el nuevo. Un valor que en un principio podemos asegurar que será superior al tiempo de reacción es 9999 milisegundos, pero en lugar de este podíamos haber elegido cualquier otro valor que fuese lo suficientemente alto.

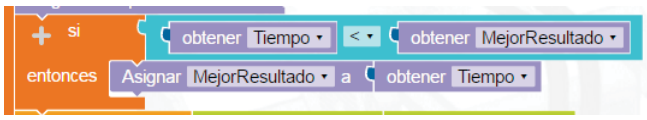


Una vez que se ha puesto el programa en marcha mostrará en pantalla el nombre del juego e instrucciones asociadas en dos filas diferentes.

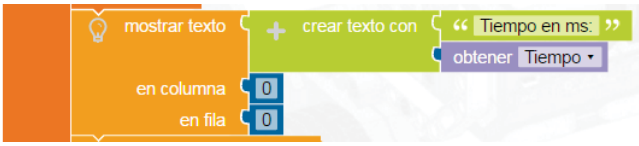


A continuación comienza un **Bucle** que se repetirá 3 veces, este bloque lo encontramos en el menú **Control**. La primera instrucción que encontramos en su interior es la espera a que se pulse la tecla verde (sensor de contacto en puerto 1) para que se dé inicio al nuevo intento.

Lo que sigue a continuación es lo mismo que teníamos en el paso nº 1 hasta que llegamos a la comparación del nuevo resultado con el mejor. En este caso tendremos que utilizar una estructura de control condicional, es decir, una de las que aparecen en el menú **Control** > **Decisiones**. En este caso, si el resultado tiene un valor menor que el almacenado en la variable **MejorResultado**, sustituirá su valor por el nuevo, en caso contrario cerrará este bloque sin generar ningún cambio.



Ahora ya solo queda presentar los resultados. Si bien se puede hacer del mismo modo que en el paso anterior, vamos a ver cómo combinar el resultado con el texto que queramos. En el menú **Texto** tenemos varios bloques que permiten manipular cadenas de texto. Uno de ellos es el de la siguiente imagen que permite generar, se puede decir así, una frase combinando un texto y el resultado obtenido. Con el mejor resultado haríamos algo similar, aunque deberíamos mostrarlo en una fila diferente.



Si bien el flujo del programa se desarrolla del modo esperado, todavía quedan algunas cositas a corregir:

- la luz verde se mantiene encendida todo el tiempo: hay que apagarla cuando ya no la necesitemos.
- la anchura de la pantalla es limitada, así que si queremos mostrar un texto que supere su anchura, deberemos cortar el texto y pedirle que muestre cada fragmento en una línea diferente.
- las cadenas de texto aparecen superpuestas: cuando le pedimos que muestre una nueva cadena de texto, solo elimina las zonas que sobrescribe, así que hay que borrar la pantalla previamente.

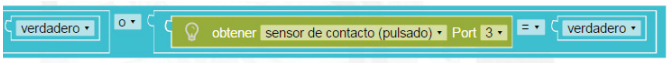
Estos mejoras quedan en manos de cada uno, ya que no suponen ninguna dificultad añadida.

Paso nº 3

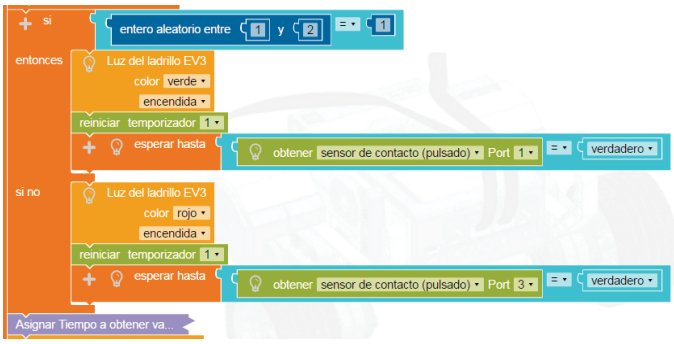
Objetivo: incorporar una segunda tecla. Ahora no se sabrá cuál de las dos se va a encender y habrá que pulsar la tecla adecuada para que el programa siga adelante.

1. Crear las variables
2. Mostrar en pantalla el nombre del juego y un texto que pida pulsar cualquier tecla para continuar
3. Repetir los siguientes pasos
    a. Esperar a que pulse alguna tecla
    b. Borrar la pantalla
    c. Esperar un tiempo aleatorio entre 2 y 5 segundos
    d. Establecer de modo aleatorio qué luz se va a encender:
        i. Si sale la primera: encender la luz verde, poner en marcha el cronómetro y esperar a que se pulse el botón verde
        ii. Si no: encender la luz roja, poner en marcha el cronómetro y esperar a que se pulse el botón rojo
    e. Almacenar el valor de lectura del cronómetro en una variable
    f. Apagar la luz
    g. Comparar la nueva lectura con la almacenada en la variable MejorResultado
        i. Si es menor que el valor de MejorResultado almacenar en MejorResultado el nuevo valor
    h. Mostrar la lectura en la pantalla, así como el valor mínimo y un mensaje que pida pulsar una tecla para continuar
    i. Esperar a que las teclas no estén pulsadas
4. Esperar 2 segundos

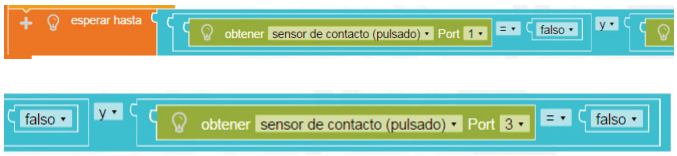
El programa se inicia igual, sin embargo, ahora se puede pulsar cualquiera de las teclas que entran en el juego para continuar, en este caso la verde y la roja. Esto requiere modificar el bloque que esperaba que se pulsase una tecla para poner en marchar el intento. Ahora habrá que esperar a que se pulse una o la otra para lo que se introducirá un operador lógico O. Dada la longitud del bloque se presenta partido en dos imágenes.



Una vez borrada la pantalla y habiendo esperado el tiempo necesario para que la luz se encienda hay que decidir qué luz encender y esperar a que se pulse la tecla correspondiente. Para tomar la decisión vamos a calcular un número aleatorio entre uno y dos, que es como echar a cara o cruz. Si sale el 1 pues verde y si sale 2 rojo. Tras encender la luz se reinicia el temporizador y se espera a que se pulse la tecla que se corresponde al color mostrado. Si se pulsa la tecla errónea no pasará nada, sino que seguirá la espera hasta que se pulse la tecla adecuada. Cuando se pulse el programa seguirá adelante y asignará el resultado a la variable **Tiempo**.



Tras apagar la luz se comprueba si se ha mejorado o no el tiempo de la misma manera que se había hecho antes y, tras presentar los resultados en la pantalla, se indica que para continuar hay que pulsar una de las teclas, es decir, ahora podremos pasar al siguiente intento sin tener que esperar dos segundos como en el paso anterior . Y ahí viene la necesidad del paso del algoritmo **Esperar a que las teclas no estén pulsadas**. Hay que ser consciente que las instrucciones de un programa informático se ejecutan muy rápido, así que puede suceder que al eliminar esa espera de 2 segundos el programa siga con el siguiente bucle sin detenerse para mostrar el resultado, ello es debido a que la ejecución del programa puede ser más rápida que el tiempo que necesitemos para soltar la tecla pulsada. EV3-G dispone de una opción para el sensor de contacto que es esperar a que se haya pulsado y soltado, algo que resolvería este problema, sin embargo, esa opción todavía no está disponible en NEPO. En este caso la diferencia con la operación lógica anterior es que ahora las dos deben estar no pulsadas, así que utilizamos la operación lógica Y.



Por dónde seguir

Los anteriores no son los únicos algoritmos válidos para este reto, seguro que se pueden mejorar. A continuación se presentan algunas ideas de mejora:

- Modificarlo para que se puedan utilizar las tres teclas
- En lugar de que el estímulo ante el que hay que reaccionar sea una luz utilizar un sonido o varias notas que se podrían asociar a las diferentes teclas.
- Además de el mejor resultado en otra línea muestra la media de los intentos.
- Si se pulsa una tecla antes de tiempo o una tecla errónea suena un aviso y se pierde el juego.
- Utilizar las herramientas para convertir los milisegundos en segundos antes de presentar los resultados en la pantalla
- Crear los programas con un entorno de programación diferente
- El programa pide el nombre del jugador y crea o carga un archivo con el histórico de resultados. El programa registra los resultados en el archivo. Esto por ahora no podría hacerse con NEPO, pero sí con otros entornos de programación que permiten el uso de archivos.

[1] https://issuu.com/koldo_lrobotikas/docs/memorygame

#

Sobre el autor

About.me/koldo_olaskoaga