TRABAJO MICROCONTROLADORES:

JUAN DE DIOS HERRERA HURTADO

DNI:47562083-S

Simón dice

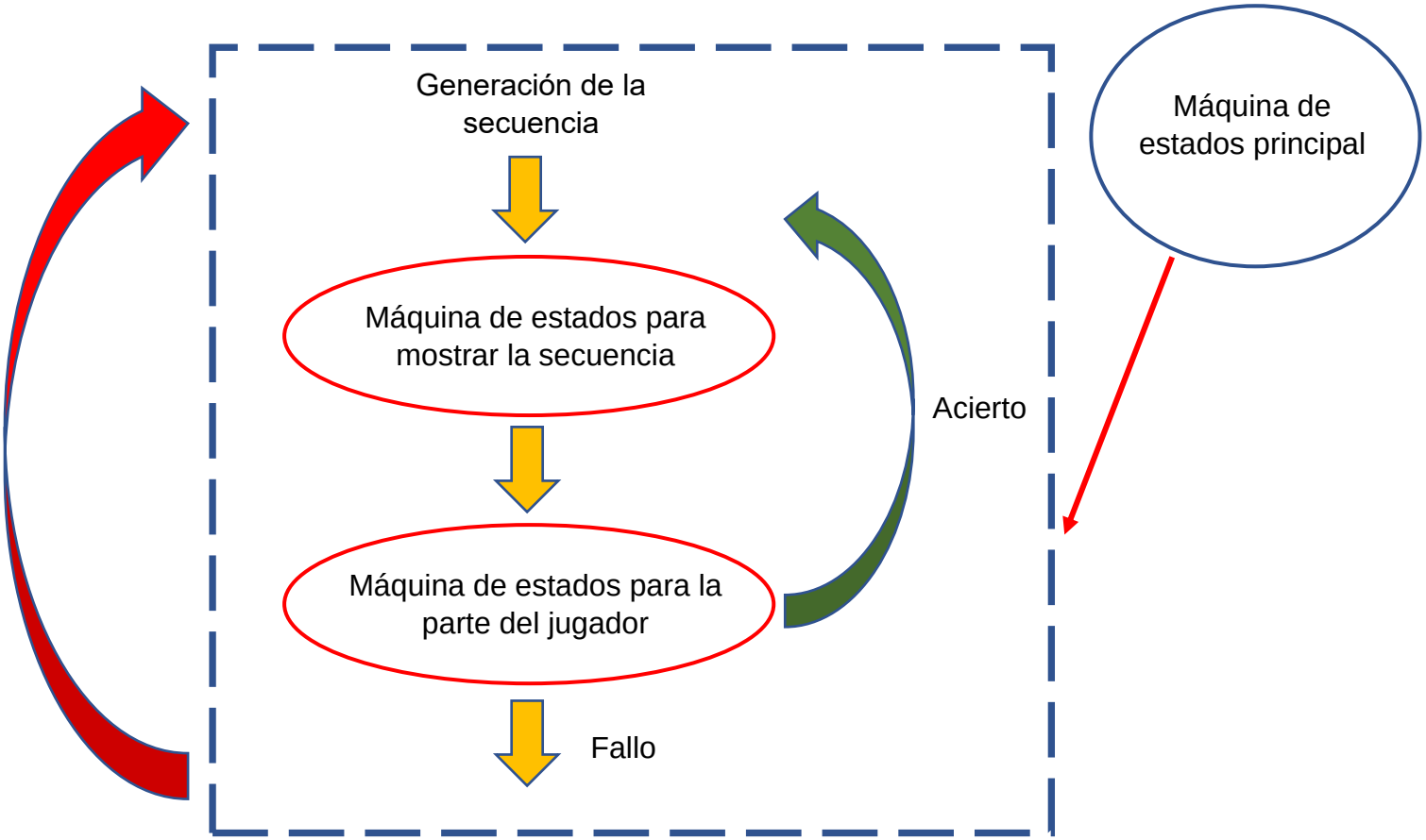
Índice

Introducción

El programa realizado corresponde al enunciado del famoso juguete de “Simón dice”. El objeto de este trabajo es la implementación de dicho juego en los dispositivos de la marca “Texas Instruments” proporcionados por el profesor, en lenguaje C con la ayuda del software “Code Composer Studio”.

Resolución del trabajo explicada

Mi programa se compone a groso modo de una máquina de estados principal que rige todo el “while (1)”, y dos submáquinas de estados (una para la parte de mostrar la secuencia, y la otra para la parte del jugador) que se encuentran dentro de la máquina de estados la principal. El esquema es el siguiente y muestra también las posibles transiciones:



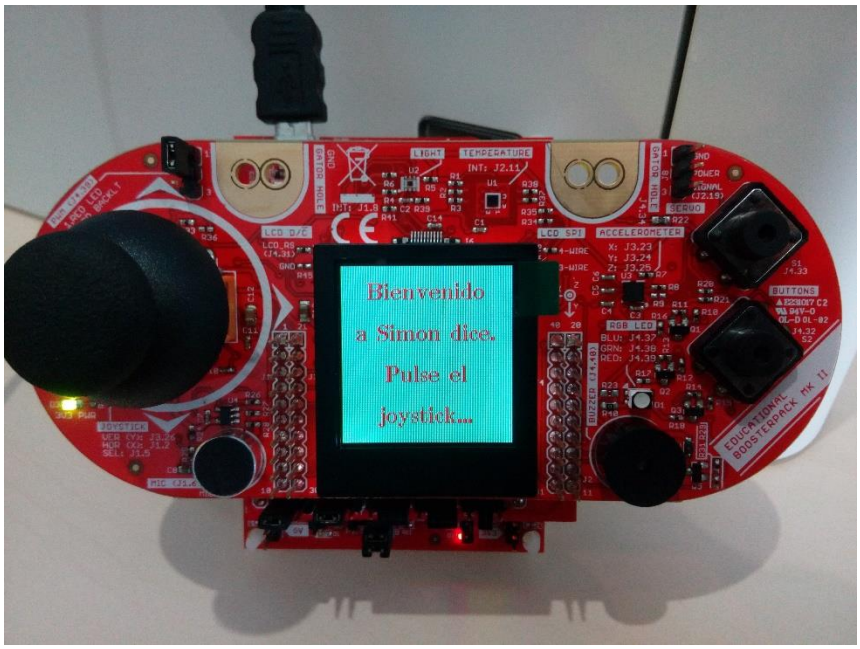
Ambas submáquinas de estados funcionan correctamente entre sí, gracias a la máquina de estados principal.

Todo el programa lo controla la variable “tiempo”. Esta variable es la encargada de temporizar los estados (aquellos que lo necesiten), contar el tiempo que se muestra cada elemento de la secuencia y la separación entre ellos, el tiempo que le queda al jugador para realizar correctamente el siguiente paso de la ronda e incluso es la variable que se usa para crear la secuencia aleatoria, esto último es interesante hacerlo con esta variable debido a que siempre está cambiando.

A continuación, pasaré a explicar qué hace cada estado de las máquinas de estados (los estados de la máquina principal 8 y 9 aparecen detrás del 1 porque los creé al final, lo mismo con el 9, solo eso):

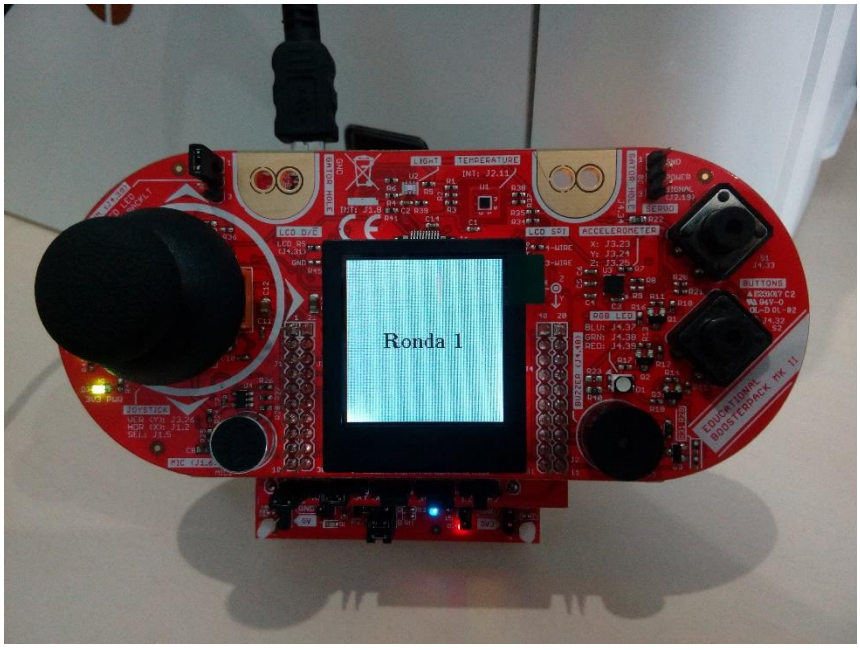
-Estado 1: se muestra una pantalla de bienvenida en la que suena una secuencia de notas que se repiten periódicamente. Además, se pide al jugador que pulse el joystick, es en ese momento cuando se capta el valor de “tiempo” y se crea la secuencia aleatoria. Para ello se modifica la semilla con “srand(tiempo)” y luego con un bucle for que se repite 32 veces, se crea la secuencia gracias a la función “rand()”.

Siempre para todos los estados, en el momento que se vaya a pasar a uno diferente, borro la pantalla y reseteo “tiempo” en caso de haberse utilizado. Tras esto se pasa al “estado” 8.



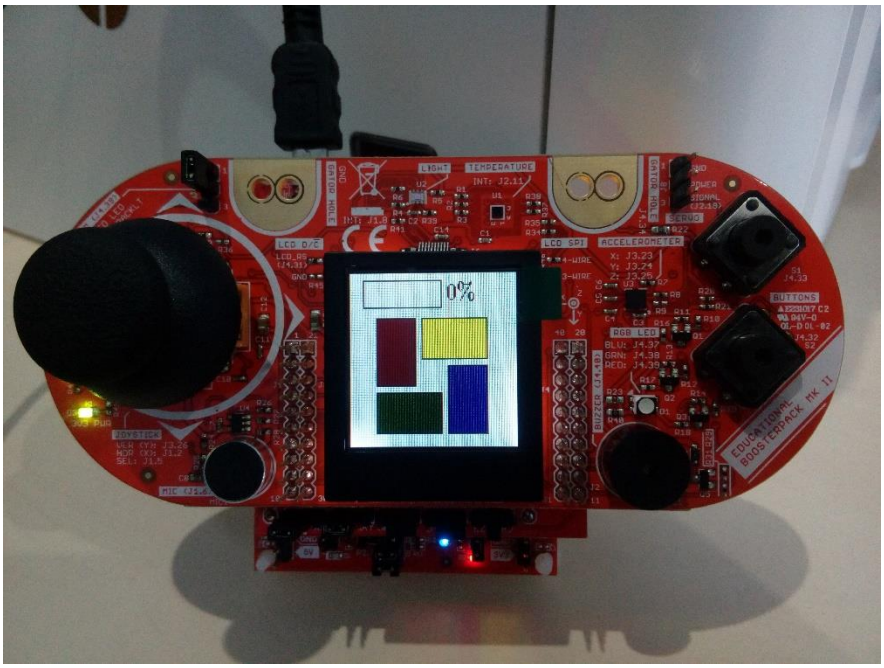
-Estado 8: se pregunta si la persona tiene problemas para distinguir los colores. Para ello se usarán los botones y la variable que se verá afectada por la elección del jugador es “problemavision”. Una vez ha respondido el jugador, se pasa al “estado” 2.

-Estado 2: se muestra la ronda por la que se va durante 1 segundo y se pasa al “estado” 3. Como de costumbre, se resetea “tiempo”.



-Estado 3: se pinta la pantalla de juego y se entra en la submáquina de estados para mostrar la secuencia. Esta submáquina de estados se encuentra dentro de un bucle for cuya variable para seguir en el bucle es “t”, de manera que cuando se llegue a la longitud deseada de la secuencia, se pueda salir de esta máquina de estados:

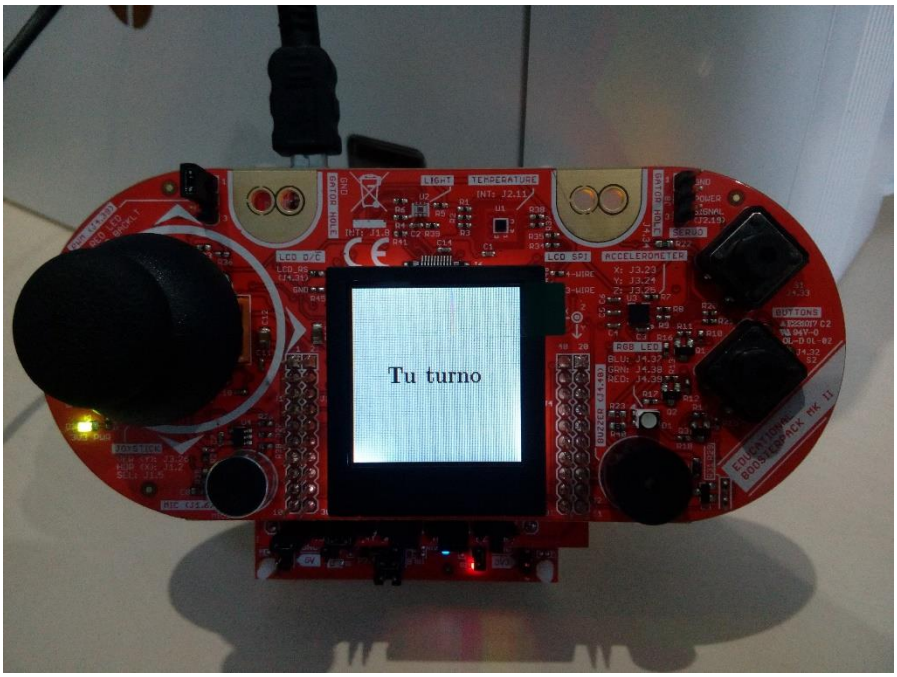
- Estado_aux 1: se lee el elemento de la secuencia, según su valor que estará comprendido entre 1 y 4, suena una nota. En caso de tener el jugador problemas de visión, aparece un círculo sobre los rectángulos, en caso contrario, lo que he planteado para distinguir qué rectángulo toca, es ponerlos de primeras a un color oscuro, y cuando le toque a ese rectángulo, rellenarlo de un color más llamativo. Cada elemento se muestra durante 2 segundos. Se pasa al “estado_aux” 2.



- Estado_aux 2: una vez mostrado el elemento, se vuelve a la pantalla original, es decir, ningún rectángulo destacando del resto o con un círculo en su interior. Pasados 0,5 segundos, incrementamos “t”. En caso de que “t” sea igual “ronda”, ya habremos llegado hasta el elemento de la secuencia deseado y saldremos del bucle for, si no es así, volveremos a “estado_aux” 1.

Una vez salido del bucle, pasamos al “estado” 4.

-Estado 4: se indica durante 1 segundo al jugador que va a comenzar a jugar. Y pasamos al “estado” 5.



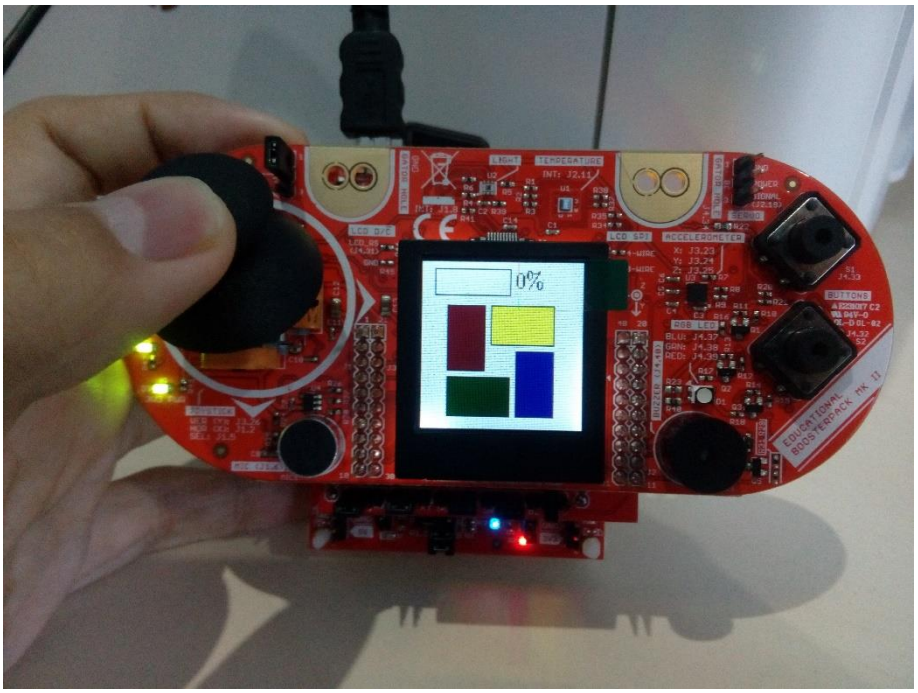
-Estado 5: es el estado destinado al jugador. Se vuelve a pintar la pantalla de juego. Nuevamente, se entra en este caso en la segunda submáquina de estados que también se encuentra dentro de un bucle for:

- Estado_prima 1: se lee la componente x del joystick y se guarda en “x”, y lo mismo con el eje y, pero en este caso guardando en la variable “y”. Según los valores, se pintarán los rectángulos más claros o en caso de ser un jugador con problemas de visión, un círculo dentro del rectángulo correspondiente. A continuación, se muestra una tabla de cómo funciona el estado.

Lectura del joystick	Valor de “eje”	Color	Nota
x < 200	1	Rojo	DO
x > 800	2	Azul	MI
y < 200	4	Verde	SI
y > 800	3	Amarillo	SOL
Ningún caso anterior	0	Todo a su color original	No suena nada

La última fila se usa en “estado_prima” 2.

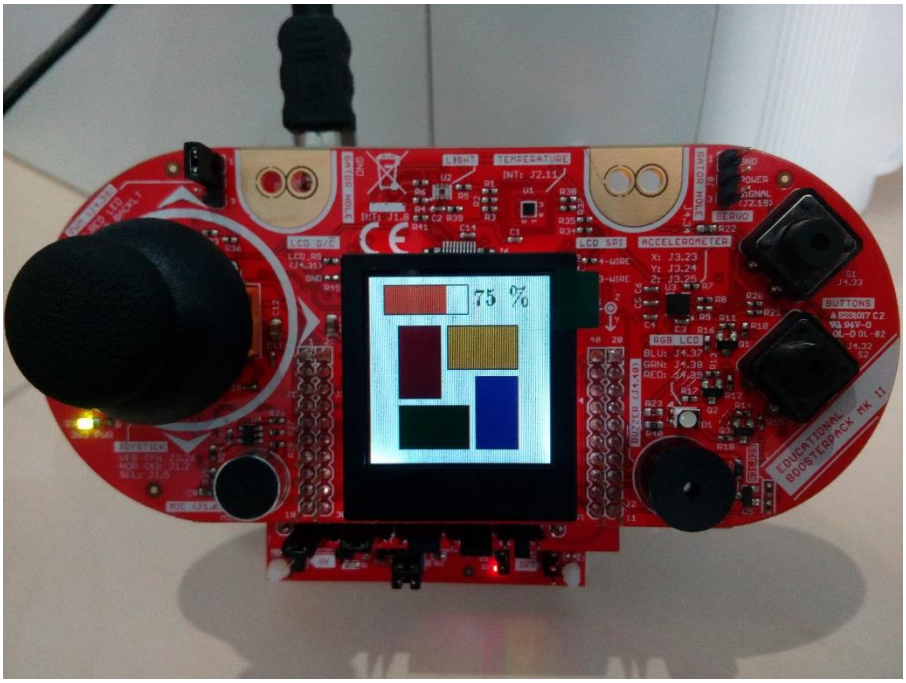
A partir de esta tabla, si coincide “eje” con el valor que toque de la secuencia y estamos dentro del tiempo para hacer cada paso (4 segundos), pasamos a “estado_prima” 2. En caso de fallar o pasarnos de tiempo, asignamos a “t” el valor de ronda para salir automáticamente del bucle for y de la máquina de estados. Además, se pone a 1 “fallo” y se suma a “puntuacion” el “progreso” que se llevase de esa ronda. Y ya fuera del bucle, se resetea “fallo” y “progreso” y se pasa al “estado” 6 (en caso de fallo).



- Estado_prima 2: solo accedemos a este estado en caso de acertar. Se sigue leyendo los ejes del joystick. Para validar el paso de la ronda, el joystick debe volver al centro, de no ser así, si se está acertando el rectángulo, pero no devuelve el jugador el joystick al centro, pasados los 4 segundos, se considera fallo y se realiza lo comentado en el “estado_prima” 1 para cuando se falla.

  En caso de devolver el joystick al centro, es en este momento cuando se para de contar los 4 segundos, se resetea “tiempo” y se pasa al “estado_prima” 3.

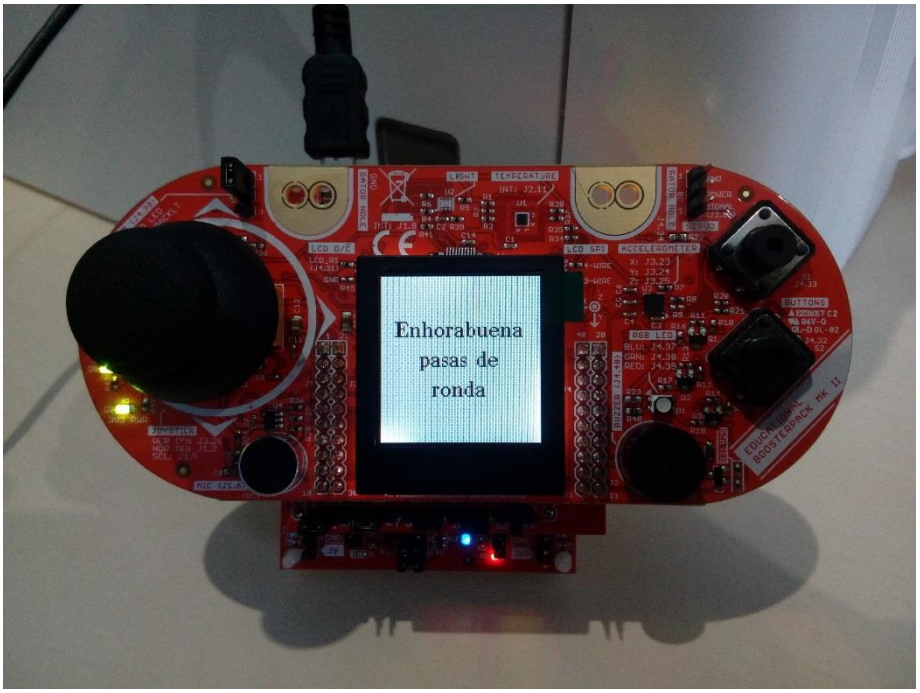
- Estado_prima 3: aumentamos el “progreso” de la ronda. Incrementamos la variable “t” que rige el bucle for. Actualizamos el progreso de la ronda, tanto la cadena como la barra de progresión y, volvemos al “estado_prima” 1 para continuar con el siguiente elemento de la secuencia.



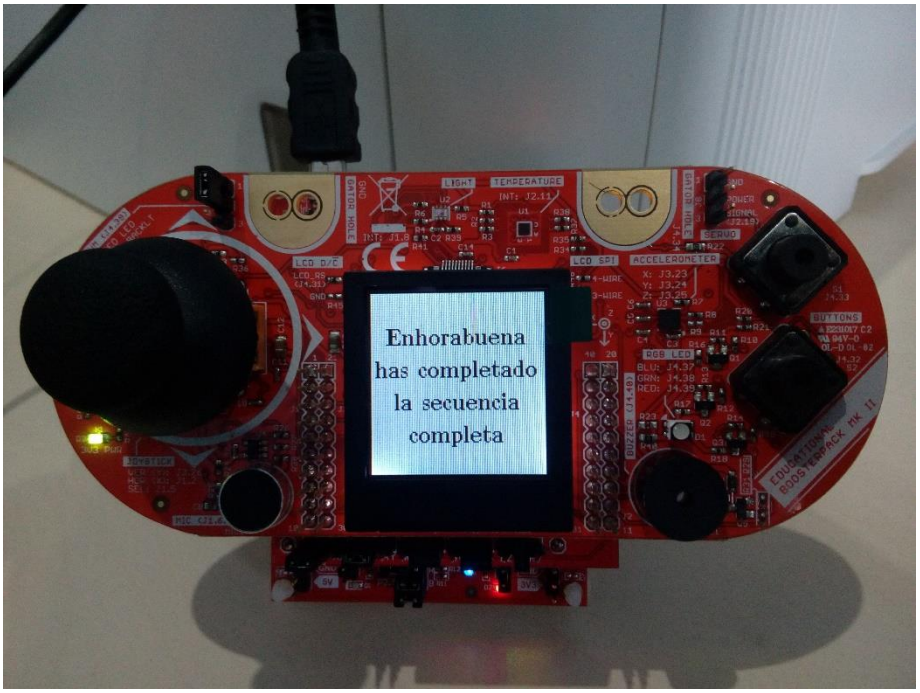
Si se sale del bucle sin fallar, aumentamos la ronda. Sumamos todo el progreso de esa ronda a la puntuación final y reseteamos “progreso” para la siguiente ronda. Se pasa al “estado” 7.

Si se completa la secuencia al completo, una vez salimos del bucle for, aumentamos “fase” para reducir los tiempos de la próxima secuencia, reiniciamos “ronda” a 1, reiniciamos “progreso” y “puntuacion”. Pasamos al “estado” 9.

-Estado 7: se da la enhorabuena durante 1,5 segundos por superar la ronda mientras suena una secuencia de notas. Pasamos al “estado” 2.



-Estado 9: se felicita al jugador durante 1,5 segundos por completar la secuencia mientras suena una secuencia de notas diferente. Se crea una nueva secuencia y se vuelve al “estado” 2.



-Estado 6: se muestra al jugador la puntuación alcanzada en la partida durante 2 segundos. Reseteamos “problemavision”, “fallo”, el buzzer deja de emitir sonido y le volvemos a dar a “ronda” el valor 1. Y finalmente se vuelve al “estado” 1 para comenzar una nueva partida.

Por último, en las interrupciones de los botones (incluido el botón del joystick), se sale del modo de bajo consumo al pulsarlos y en la interrupción del convertidor lo mismo al terminar la conversión en cuestión.

En el caso de la interrupción del timer, se incrementa “tms” y salimos del modo de bajo consumo cada 300 milisegundos. Además, se incrementa “tiempo” siempre que estemos en un estado que lo necesite.

En cuanto a la parte inicial del programa, he declarado todo aquello que he necesitado para poder completar el programa. Y he configurado los pines P2.6 y P2.7 para poder hacer uso del buzzer y los pines P1.1, P1.2 y P2.5 declarados como entrada para poder usarlos como botones.