



Proyecto Final.

Realización de un sistema medidor de reflejos usando *Raspberry Pi*.

Autores:

Rubén San-Segundo

Roberto Rodríguez Zurrenero

Francisco Tirado Andrés

Ramiro Utrilla Gutierrez

Dpto. de Ingeniería Electrónica

Fecha última revisión: Enero-2016

Contenido

1	Introducción	3
2	Objetivos	3
2.1	Objetivos específicos.....	3
2.2	Materiales necesarios	3
3	Descripción funcional del sistema.....	3
3.1	Descripción general.....	3
3.2	Descripción detallada.....	4
4	Subsistema hardware.....	4
5	Subsistema software	6
5.1	Modo de operación.....	6
5.1.1	Estado S1 - wait_start	7
5.1.2	Estado S2 - wait_push.....	7
5.1.3	Estado S3 - wait_end.....	7
5.1.4	Estado S4 - exctn_wait_start.....	7

1 Introducción

La práctica a desarrollar en la segunda parte de la asignatura SDG2 consiste en la implementación de un dispositivo de entrenamiento y evaluación de reflejos basado en la interacción del usuario mediante diodos LED y botones.

En esta práctica se pretende desarrollar un proyecto de cierta complejidad a partir de los conocimientos adquiridos en las sesiones previas de la asignatura. Por ello, se recomienda seguir los criterios de diseño que se han proporcionado en las distintas prácticas.

En el proyecto se utilizará una *Raspberry Pi* para realizar un sistema real de evaluación de reflejos, usando: programación en C, interrupciones, temporización, máquinas de estados...

2 Objetivos

El objetivo fundamental de la práctica es el desarrollo de una aplicación en un sistema empotrado basado en un microcontrolador de altas prestaciones. Se utiliza una *Raspberry Pi* que es un sistema de referencia y permite la utilización de un Sistema Operativo Linux.

Habrá que realizar también un pequeño subsistema hardware que permita la interacción del usuario con la plataforma *Raspberry Pi*.

2.1 Objetivos específicos

- Mejora de conocimientos de programación en C
- Diseño lógico basado en máquinas de estados aplicado a un sistema real
- Manejo avanzado de entradas y salidas digitales
- Manejo de interrupciones
- Manejo de concepto de tiempo
- Integración de todos los conceptos en un único sistema final

2.2 Materiales necesarios

- *Raspberry Pi B+*
- 5 diodos LED y resistencias de polarización
- 5 Pulsadores y resistencias de *pull-up*
- Cables para el conexionado
- Entorno de desarrollo basado en eclipse

3 Descripción funcional del sistema

3.1 Descripción general

El interfaz del sistema de medida de reflejos consiste en 5 pulsadores, cada uno con su correspondiente led asociado. Uno de estos pulsadores se encarga de la función START/END; los otros cuatro se encargan de la función estímulo-respuesta para la medida de reflejos.

Una vez empezada la ejecución, el sistema encenderá un led aleatorio y medirá el tiempo (en milisegundos) que el usuario tarda en pulsar su botón asociado para apagarlo. Esto se repetirá un número de veces definido por el desarrollador, siendo 10 el mínimo.

Finalmente, se extraerán una serie de medidas en base a los tiempos de respuesta capturados. Para reiniciar una ronda se deberá volver a pulsar START.

3.2 Descripción detallada

A continuación, se explica la funcionalidad básica detallada del sistema que el alumno deberá cumplir para superar la práctica.

En primer lugar, se debe inicializar y configurar en la *Raspberry Pi* los puertos de entrada/salida necesarios.

Seguidamente, debe iniciarse la rutina de la máquina de estados finitos encargada de controlar el flujo del sistema. Este deberá ser como sigue:

1. Se espera hasta que se da la orden de START con el botón correspondiente.
2. Se calcula aleatoriamente el primer led a encender y se espera hasta que se pulse el botón correspondiente. Al pulsar se apaga el led y se calcula el tiempo transcurrido entre estímulo y respuesta.
3. Este proceso se repetirá un número de veces determinado por el desarrollador, siendo el mínimo de 10, momento en el cual terminará el juego.
4. Si se pulsa un botón incorrecto en algún momento (botón no asociado al led que está encendido) se registra el error. Este error añade un tiempo de respuesta de penalización de 3 segundos. Si se falla más de 3 veces se termina el juego.
5. Al terminar el juego se calcularán las estadísticas en base a los tiempos medidos. Estas estadísticas deben contener **al menos**: tiempo medio, tiempo máximo, tiempo mínimo y número de fallos.
6. Al pulsar el botón START se mostrarán por pantalla las estadísticas calculadas en la partida.
7. Si se vuelve a pulsar el botón START empezará de nuevo el juego.

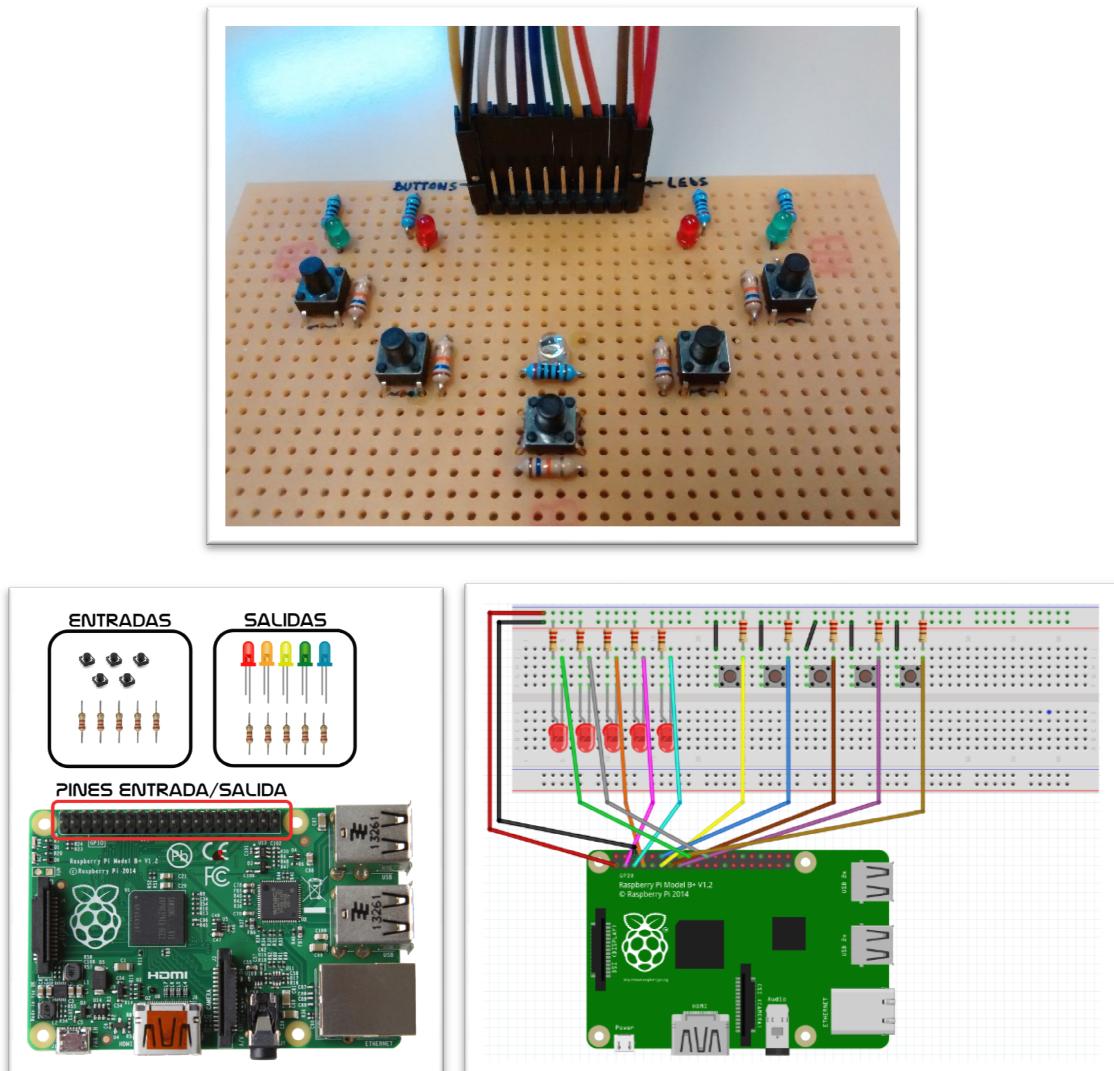
4 Subsistema hardware

El subsistema hardware necesario para la realización está basado en 5 diodos led y 5 pulsadores. El alumno deberá calcular los resistores de polarización necesarios, así como los resistores de *pull-up*.

Los pulsadores estarán conectados a 5 entradas de la *Raspberry Pi*, y los diodos led estarán conectados a 5 salidas, todos ellos a elegir por el alumno.

Es importante tener en cuenta que el voltaje en los pines de entrada y salida utilizados debe ser siempre 3.3V. Si no es así se corre el riesgo de dañar la *Raspberry Pi* de forma irreversible.

Para la realización del montaje hardware en laboratorio se remite al alumno al enunciado de la práctica 3. Alternativamente, se muestra un esquema que podría ser montado en casa sin necesidad del entrenador del laboratorio.



Ejemplo de montaje alternativo.

Se recomienda al alumno que antes de iniciar el desarrollo de esta práctica final realice un pequeño programa software para la verificación de que todas las conexiones hardware se han realizado correctamente y funcionan de la forma esperada. Por ejemplo, un bucle en el que se explora cada uno de los pulsadores y, en el caso de que uno de ellos este pulsado, se encienda el led correspondiente a ese pulsador durante el tiempo en el que este pulsado. De esta manera, comprobamos si las conexiones de los pulsadores y de los leds están realizadas de forma correcta y en los pines de entrada/salida deseados.

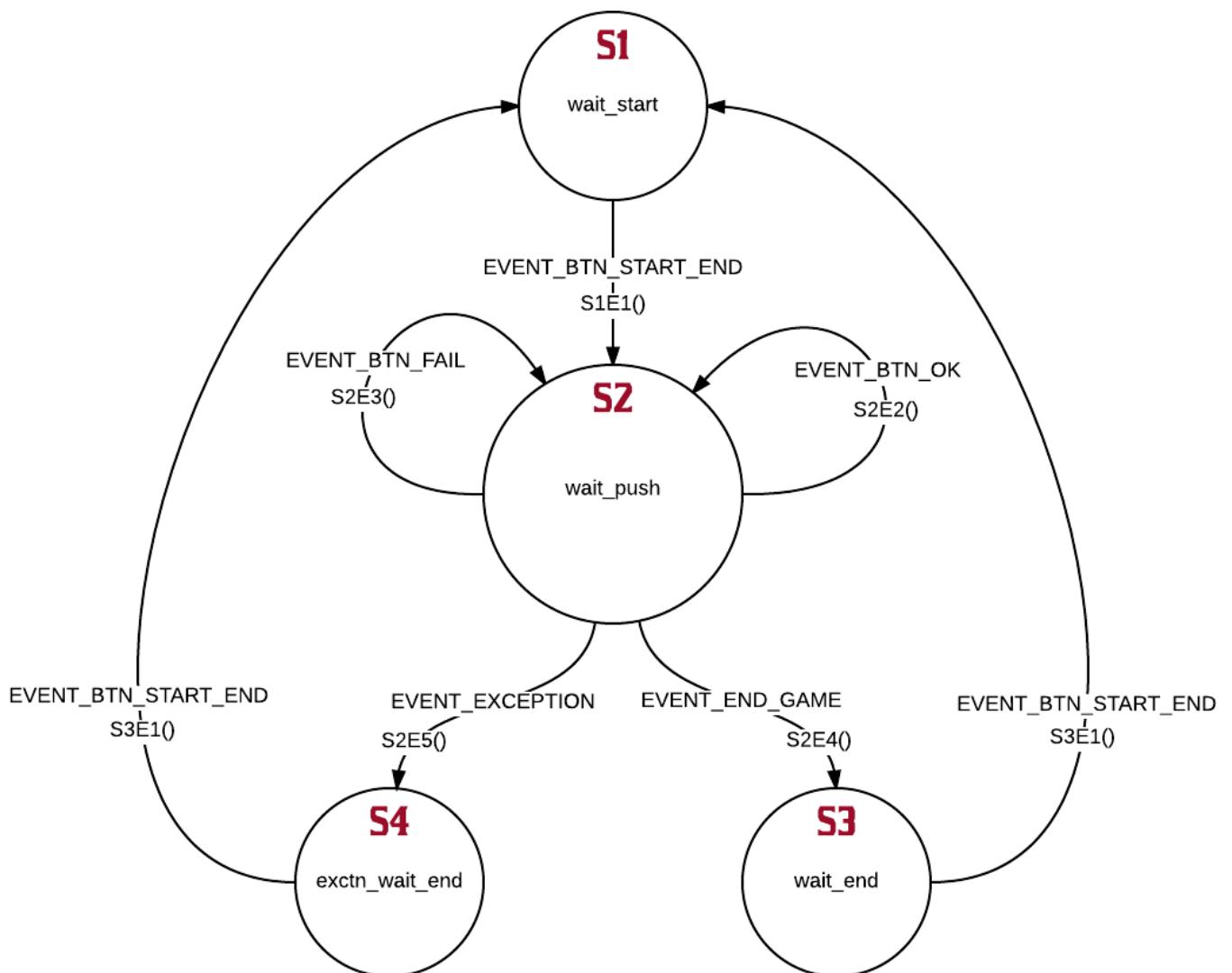
5 Subsistema software

El subsistema software deberá integrar todos los conceptos aprendidos durante las sesiones previas de la asignatura. El software estará basado en una **máquina de estados** con manejo de entrada/salida, interrupciones y temporización.

5.1 Modo de operación

El sistema completo propuesto se basa en una máquina de cuatro estados (S1-S4), como se muestra en la Figura. Como se ha comentado previamente, el interfaz de comunicación con el usuario consiste en cinco leds, cada uno con su correspondiente botón asociado. Cuatro de ellos son de aviso del juego, y el quinto es el de START/END.

El sistema cuenta con varias constantes predefinidas: *repeticiones*, *tiempo de penalización*



y *número máximo de fallos*.

5.1.1 Estado S1 - `wait_start`

El sistema espera que el botón de START/END sea pulsado para iniciar el juego. En este estado todos los leds excepto el asociado a START/END están apagados. Si se pulsa el botón START/END (EVENT_BTN_START_END) se apaga dicho led y se pasa al estado S2.

5.1.2 Estado S2 - `wait_push`

Este es el estado en el que se desarrolla el juego a no ser que se produzcan eventos anómalos tales como el final del juego, o alguna excepción.

En este estado se calcula de forma aleatoria que led de los 4 disponibles hay que encender y lo enciende. En ese momento se obtiene la marca temporal para calcular el tiempo que tarda el usuario en pulsar un botón. En este punto pueden suceder varias cosas:

- Si el botón pulsado **Sí es el correcto** (EVENT_BTN_OK): se guarda el valor del tiempo que ha tardado. Se inicia otra vez el proceso de calcular el siguiente led a encender y extraer la marca de tiempo actual.
- Si el botón pulsado **NO es el correcto** (EVENT_BTN_FAIL), se aumenta la variable de fallos en uno y se guarda un tiempo con un valor de penalización (*penalty_time*). Se inicia otra vez el proceso de calcular el siguiente led a encender y extraer la marca de tiempo actual.
- Si se sobrepasa el valor de *time_out* definido previamente, se llama al evento de botón pulsado NO correcto (EVENT_BTN_FAIL), se marca como fallo y se le añade un tiempo de penalización.
- Si el número de fallos supera el valor de *btn_fail_max* se finaliza el juego y pasa al estado S3.
- Si el número rondas es mayor al valor preestablecido se finaliza el juego y pasa al estado S3.

Este proceso se repite hasta que se haya completado un número predeterminado de *rounds* (pasa a estado S3), se hayan producido más fallos de los permitidos (pasa a estado S3) o se produzca una excepción (pasa a estado S4).

5.1.3 Estado S3 - `wait_end`

A este estado se llega cuando se ha finalizado el juego. En este estado se enciende el led asociado a START/END .

Si se pulsa el botón START/END (EVENT_BTN_START_END) se calculan y se muestran por pantalla las estadísticas de partida: tiempo medio, tiempo máximo, tiempo mínimo y número de fallos. A continuación, se pasa al estado S1.

5.1.4 Estado S4 - `exctn_wait_start`

La captura de excepciones, como la pulsación de varios botones a la vez, se considerará una mejora al sistema básico propuesto.

Si se captura una excepción se salta a este estado S4, se muestra por pantalla el motivo de la excepción, se hace parpadear el led asociado al botón START/END y se espera a que el botón START/END (EVENT_BTN_START_END) sea pulsado para volver al estado S1 reiniciando de este modo el juego.