

## **Versión 2024.02**

**Carrera:** INGENIERIA EN INFORMATICA

**Asignatura:** 3631-Fundamentos de sistemas embebidos.

**Tema:** Máquinas de estado finito hardware y software (micropython)

**Unidad:** 5.0 y 5.1

**Objetivo:** Comprender el funcionamiento de FSMs

**Competencia/s a desarrollar:**

- Concepción, diseño y desarrollo de proyectos de ingeniería en sistemas de información / informática.
- Gestión, planificación, ejecución y control de proyectos de ingeniería en sistemas de información / informática.
- Utilización de técnicas y herramientas de aplicación en la ingeniería en sistemas de información / informática.
- Generación de desarrollos tecnológicos y/o innovaciones tecnológicas.
- Desarrollo de una actitud profesional emprendedora.
- Aprendizaje continuo.
- Actuación profesional ética y responsable.
- Comunicación efectiva.
- Desempeño en equipos de trabajo.
- Identificación, formulación y resolución de problemas de ingeniería en sistemas de información/informática.

**Descripción de la actividad:**

1-Tiempo estimado de resolución: 2 semanas

2-Metodología: Implementar en logisim-evolution y Wokwi

3-Forma de entrega: No obligatoria

4:Metodología de corrección y feedback al alumno: Presencial y por Miel.

**NOTA:** En MleL tienen los archivos de logisim-evolution para muchas de las FSM de esta unidad.

## F- Máquinas de estado finito

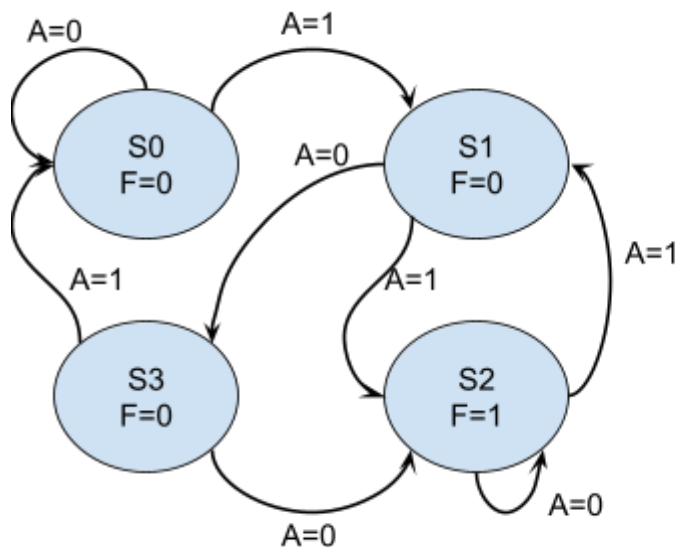
**F.01** Utilizando un contador binario síncrono ascendente de 3 bits (simil hoja 3 de la teoría) implemente un **secuenciador** que genere el siguiente código: 0000 → 0001 → 0011 → 0111 → 1111 → 1110 → 1100 → 1000. Grafique en logisim utilizando el osciloscopio digital.

**F.02** Utilizando un contador de código one hot (anillo) de 10 bits implemente un secuenciador que genere código BCD. Grafique en logisim utilizando el osciloscopio digital. Nota: repase el codificador simple.

**F.03** Utilizando un contador Johnson de 5 bits implemente un secuenciador que genere código de Aiken ABCD (0000 ; 0001 ; 0010 ; 0011 ; 0100 ; 1011 ; 1100 ; 1101 ; 1110 ; 1111). Grafique en logisim utilizando el osciloscopio digital. Nota: Tenga en cuenta que el código Johnson de 5 bits es: 00000 → 00001 → 00011 → 00111 → 01111 → 11111 → 11110 → 11100 → 11000 → 10000. Esto quiere decir que la tabla de verdad es de 5 variables (32 combinaciones) pero solo 10 están definidas. Comience definiendo la tabla de verdad completa, completando con Don't care en las combinaciones que nunca se producen. Luego utilice un simplificador Karnaugh online para cada una de las funciones de salida del código Aiken. Otra opción es directamente implementar cada función con un MUX en logisim.

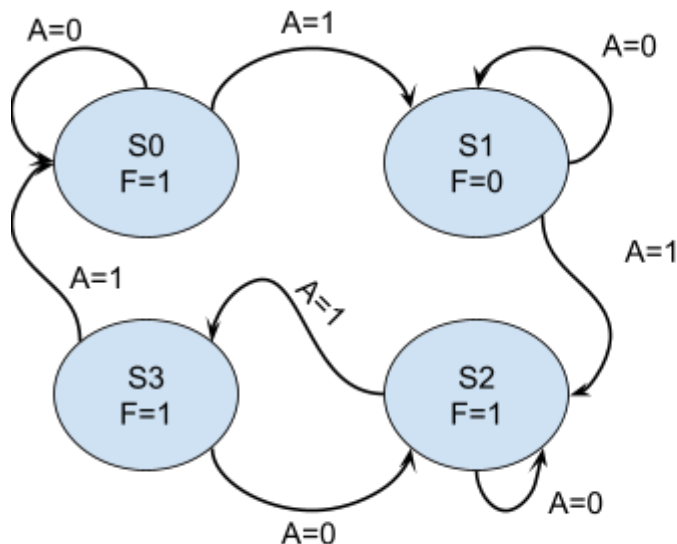
**F.04** Utilizando un contador Johnson de 3 bits y un MUX (similar a la página 6 de la teoría pero con otro contador) genere una señal periódica con un ciclo de actividad de 2/6. Grafique en logisim utilizando un osciloscopio digital. Nota: tenga en cuenta que si bien el MUX tiene 3 entradas de selección, **no todas las combinaciones se van a generar en el contador Johnson**, y el orden en el que se recorren las entradas del MUX es distinto del contador binario.

**F.05** Dado el siguiente diagrama de transición de estados, complete la tabla de verdad de  $S'$  e implemente la máquina de estados en Logisim. La entrada es A. La salida es F.



$S_1 S_0$	A	$S'_1 S'_0$
0 0	0	
0 0	1	
0 1	0	
0 1	1	
1 0	0	
1 0	1	
1 1	0	
1 1	1	

**F.06** Dado el siguiente diagrama de transición de estados, complete la tabla de verdad de  $S'$  e implemente la máquina de estados en Logisim. La entrada es A. La salida es F.



$S_1 S_0$	A	$S'_1 S'_0$
0 0	0	
0 0	1	
0 1	0	
0 1	1	
1 0	0	
1 0	1	
1 1	0	
1 1	1	

**F.07** Dada la FSM de la máquina de gaseosas de la teoría. Se modifica el detector de billetes para que soporte billetes de \$200 generando la salida 11. El precio de la gaseosa sigue siendo \$150. Modifique la FSM para que soporte pagar con \$200. Recuerda que sigue sin dar vuelto.

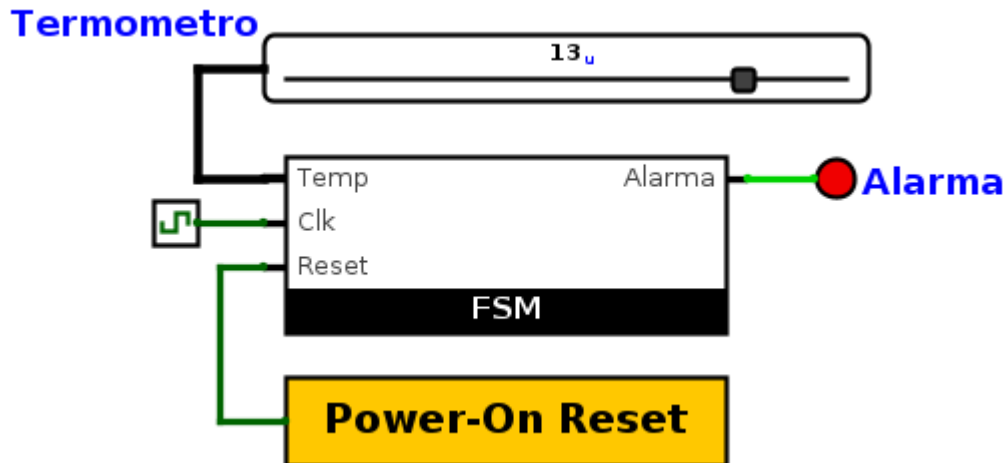
**F.08** Utilizando FF tipo D (y lógica combinacional) implemente un contador síncrono binario descendente de 3 bits. Plantee la tabla de verdad de las transiciones.

**F.09** Modifique la FSM **SemaforoProgramable** para que los tiempos sean: 12 en S0, 5 en S1, 14 en S2 y 2 en S3. (Ver archivo en MleL)

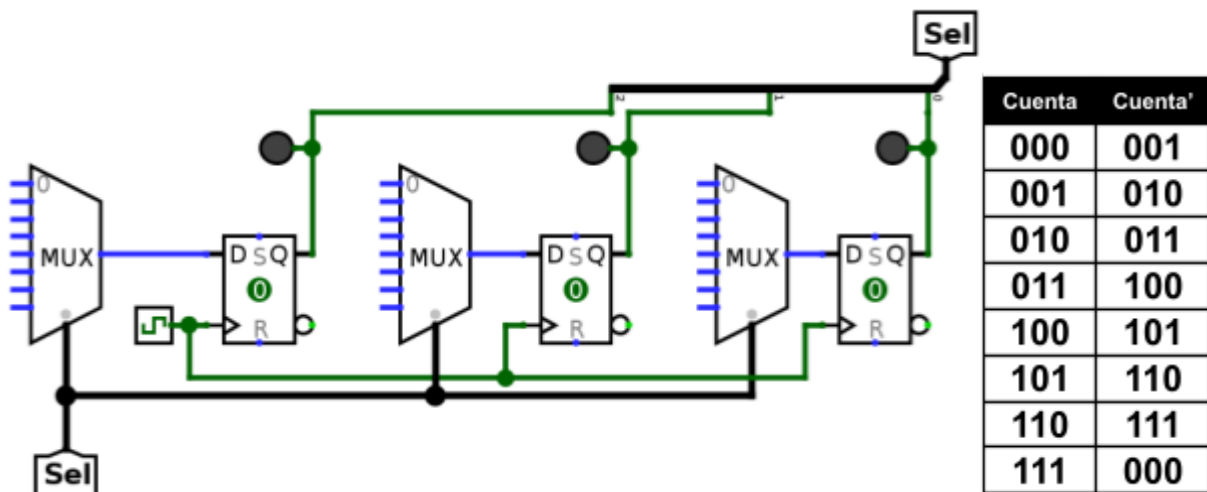
**F.10** Modifique la FSM **TecladoCajaFuerteProgramable** para que la clave sean los últimos 4 dígitos de su DNI. (Ver Archivo en MleL)

**F.11** Diseñe una FSM que funcione como alarma de temperatura para un frigorífico. Utilice el componente slider de 4 bits. Este valor representa la temperatura (de 0 a 15 grados). La FSM debe encender la alarma (un uno en la salida de alarma) si la temperatura supera los 12 grados. En el caso que comience a bajar, debe mantener la alarma encendida hasta que la temperatura baje de 8 grados (ciclo de histéresis). Represente el diagrama de transición de

estados y la tabla de verdad correspondiente. Puede elegir implementarlo como una FSM programable o simplemente como una máquina de moore o mealy.



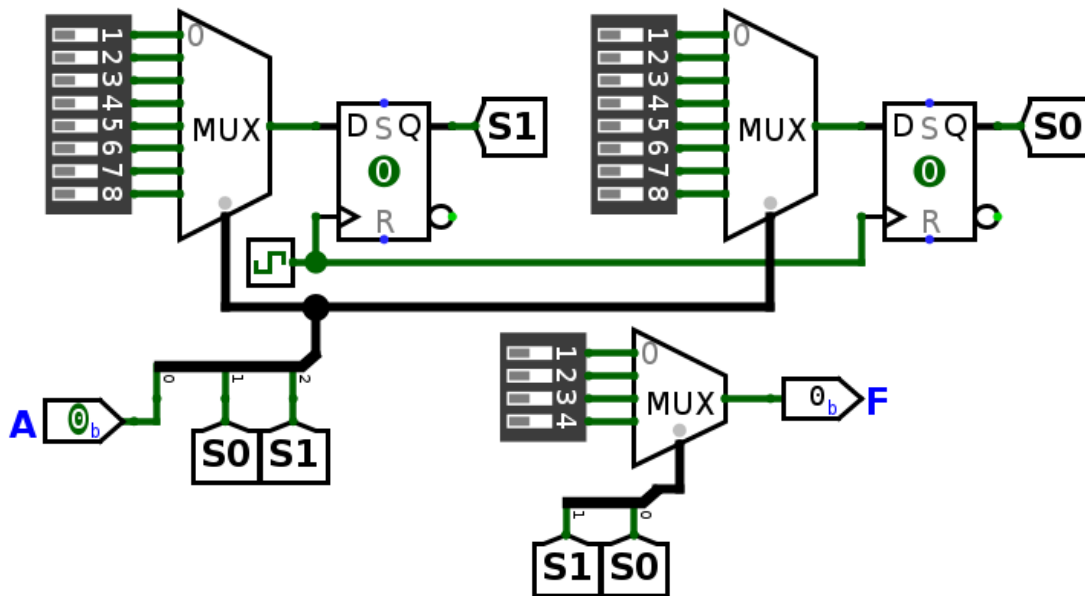
**F. 12** Dado el siguiente circuito lógico programable, los multiplexores funcionan como LUTs (LookUp Tables) seleccionando el estado actual (la salida de los 3 FF) y calculando el nuevo estado. Indique los valores que deben tomar las entradas de datos de los MUX para que se comporte como un contador síncrono binario ascendente.



**F. 13** Modifique el circuito anterior para que se comporte como un contador binario descendente.

**F. 14** Modifique el circuito de F.12 para que se comporte como un contador binario ascendente módulo 5 (es decir, cuenta 000, 001, 010, 011, 100).

**F. 15** Diseñe una máquina de estados en donde existe una entrada  $A$  y una salida  $F$ . En el caso que el valor de  $A$  en 3 pulsos de clock consecutivos sea 011, la salida  $F$  toma el valor 1, en cualquier otro caso toma el valor 0. Implemente la FSE utilizando el siguiente circuito. Las entradas de datos de los MUX se encuentran conectadas a DIP-Switches que permiten ingresar ceros o unos.



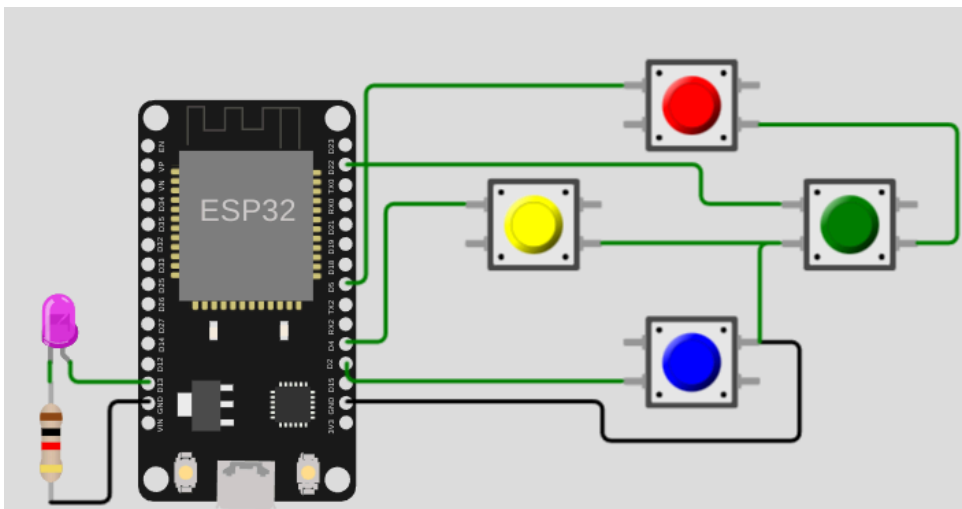
## G- Máquinas de estado en Micropython (simuladas en Wokwi)

**G.01** Implemente un diseño que cuente números positivos desde 0000 hasta 9999. Utilice un display TM1637 para indicar el número. Agregue dos pulsadores, uno para incrementar la cuenta y otro para decrementar la cuenta. Cuando se produce  $9999+1$  el próximo valor es 0000. Cuando se produce  $0000-1$  el próximo valor es 9999. Cada pulsador debe funcionar como una FSM para soportar bouncing.

**G.02** Modifique el diseño SemaforoFSMIntermitente para que tenga 6 segundos de verde titilante antes de pasar a amarillo.

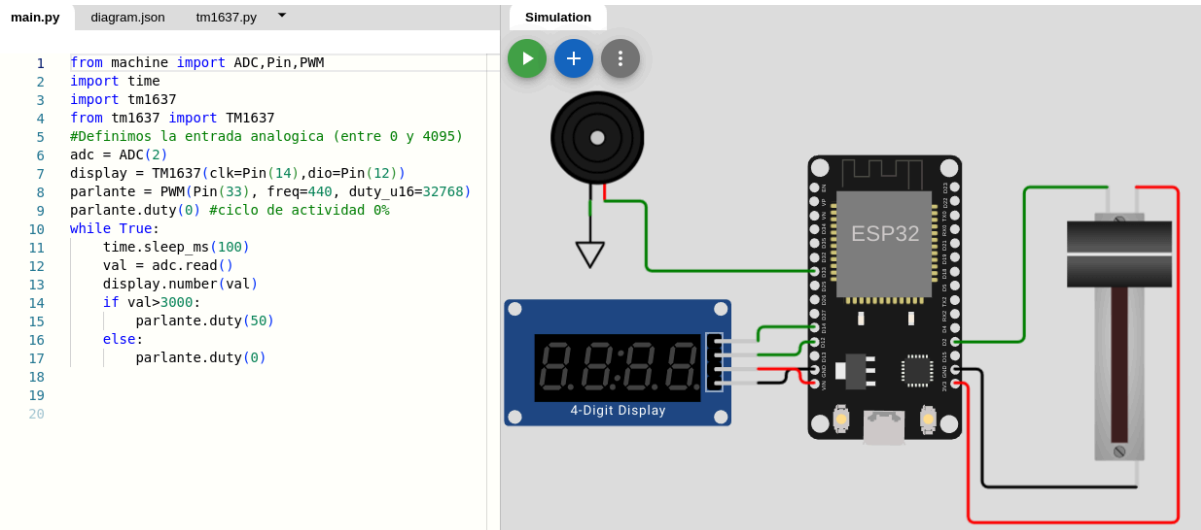
**G.03** Modifique el diseño SemaforoFSMIntermitente para que tenga otra fase completa (rojo, amarillo, verde) de forma tal que cuando una fase este en rojo la otra este en verde (respetando los tiempos de amarillo en cada fase). Elija los tiempos que considere adecuados. Debe soportar intermitente con un único switch para ambas fases.

**G.04** Dado el siguiente diseño con 4 botones (arriba-Rojo , abajo-Azul, izquierda-Amarillo, derecha-Verde) y un led de salida, diseñe una maquina de estados que encienda el led si el usuario ingresa la secuencia : Arriba, Arriba, Abajo, Abajo, Izquierda, Derecha, Izquierda, Derecha. Una vez ingresada la secuencia, cualquier botón apaga el led. Tenga en cuenta que debe soportar Bouncing en los botones, por ende, piense que cada botón en sí es una máquina de estado para eliminar el bouncing (con una variable de estado por botón que detecta si el botón fue apretado o no). Luego existe una máquina de estado general para llevar el estado de la secuencia, en donde cada transición se produce en base al botón apretado.



**G.05** Modifique el diseño ADCSlider para que haga sonar el parlante cuando la temperatura ingresada (simulada con el slider) sobrepasa los 3500. Luego cuando baja de ese valor, debe mantener el parlante hasta que la temperatura desciende de los 1500. Se provee como

*ejemplo un código que sólo acciona el parlante cuando la temperatura sobrepasa los 3000 (sin FSM).*



**G.06** Modifique el diseño CajitaDeMusica para que toque una secuencia de notas distinta. En caso de no saber música complete la canción actual con las siguientes notas:

^Do (523) Sol (329) Mi (659)  
La (440) Si (493) Sib(466) La (440)  
Sol(329) ^Mi(659) ^Sol(659) ^La (698)  
^Fa(698) ^Sol (659) ^Mi (659) ^Do(523)  
^Re(587) Si (493)

