Versión 261C.01

Carrera: INGENIERÍA EN INFORMÁTICA

Asignatura: 3631 - Fundamentos de sistemas embebidos

Tema: FSM Hardware y Software (Micropython)

Unidad: 5.0 $\sim$ 5.1

Objetivo: Diseñar máquinas de estado integrando los conocimientos de lógica combinacional y secuencial

Competencias a desarrollar:

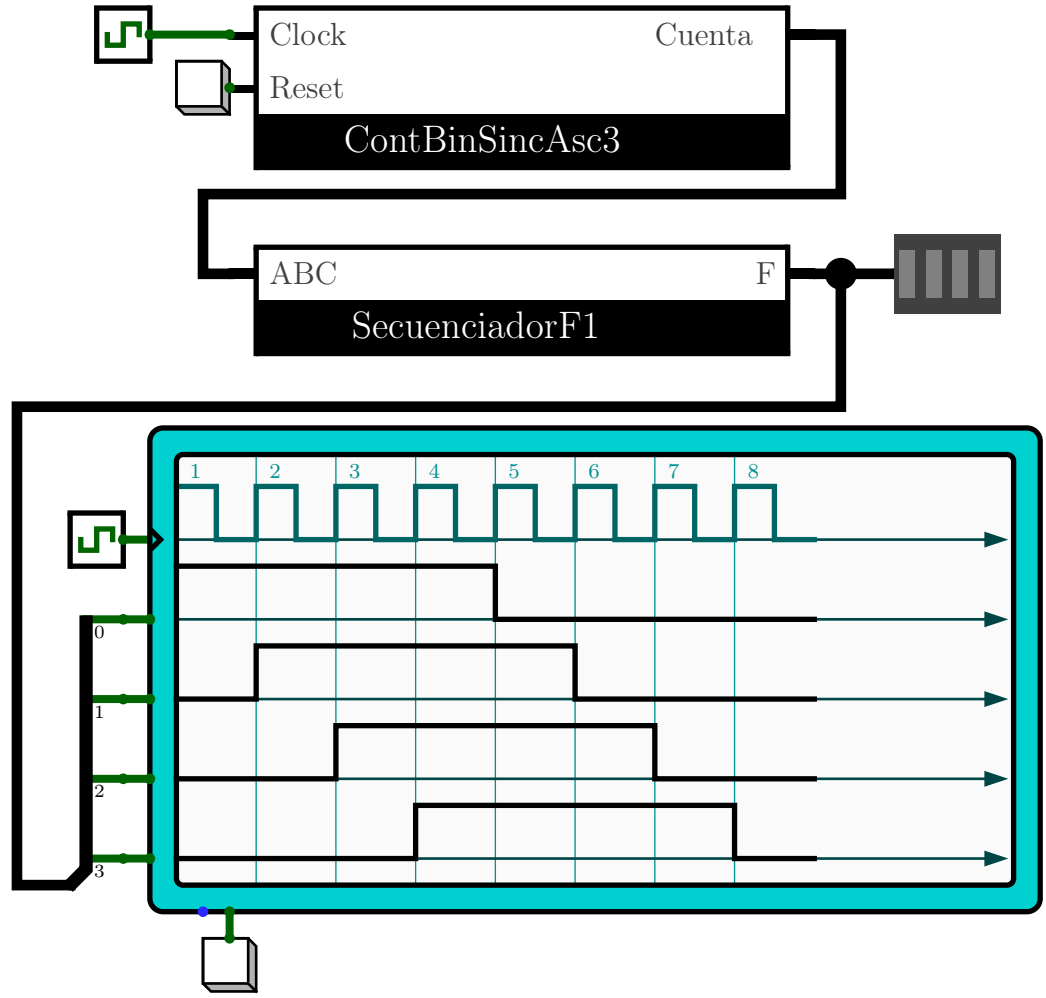
- Concepción, diseño y desarrollo de proyectos de ingeniería en informática.
- Gestión, planificación, ejecución y control de proyectos de ingeniería en informática.
- Utilización de técnicas y herramientas de aplicación en la ingeniería en informática.
- Generación de desarrollos tecnológicos y/o innovaciones 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 informática

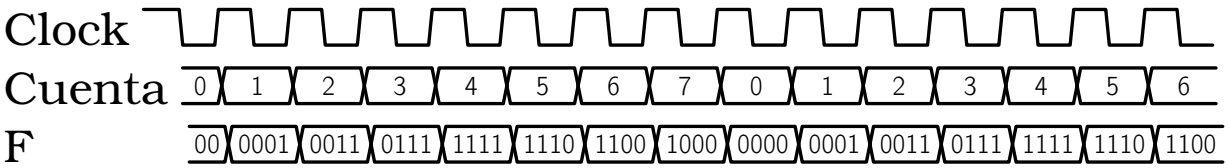
Descripción de la actividad:

1. Tiempo estimado de resolución: 2 semanas
2. Metodología: Ejercicios verificados en simuladores
3. Forma de entrega: No obligatoria
4. Metodología de corrección y feedback al alumno: Presencial y por Miel.

F- Máquinas de Estado Finito (FSM) Hardware

F.1 Utilizando un contador binario sincrónico 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. Se agrega una barra de leds para visualizar la secuencia.



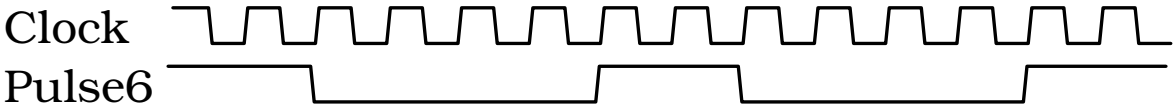


F.2 Utilizando un contador de código one hot (anillo) de 10 bits implemente un secuenciador que genere código BCD. Grafique en logisim-evolution utilizando el osciloscopio digital. *Nota: repase el codificador simple.*

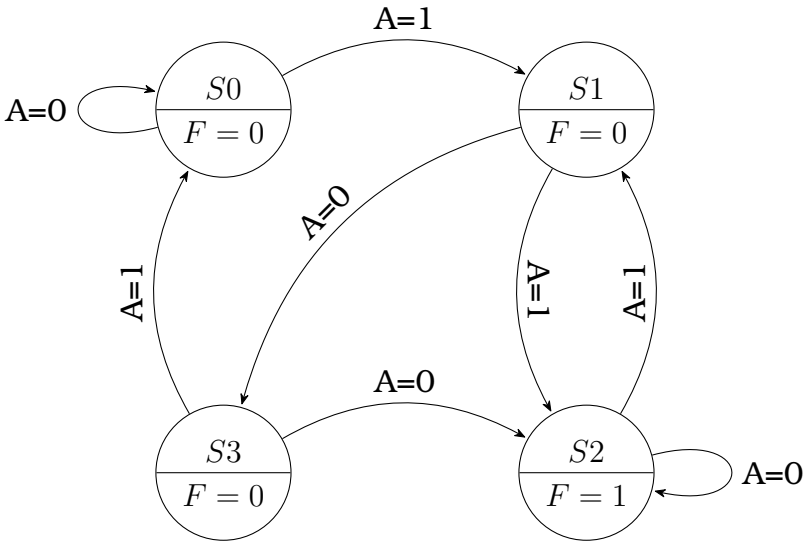
F.3 Utilizando un contador Johnson de 5 bits implemente un secuenciador que genere código de Aiken ABCD (0000 ; 0001 ; 0010 ; 0011 ; 0100 ; 1011 ; 1100 ; 1101 ; 1110; 1111). Grafique en logisim-evolution utilizando el osciloscopio digital. *Nota: Tenga en cuenta que el código Johnson de 5 bits es: 00000 → 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 o ROM en logisim-evolution.

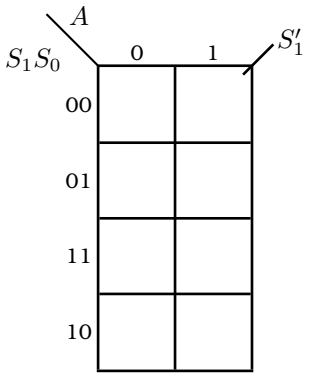
F.4 Utilizando un contador Johnson de 3 bits y un MUX (similar a la página 6 de la teoría pero con otro contador) genere una señal periódica con un ciclo de actividad de 2/6. Grafique en logisim-evolution utilizando un osciloscopio digital. *Nota: tenga en cuenta que si bien el MUX tiene 3 entradas de 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.*

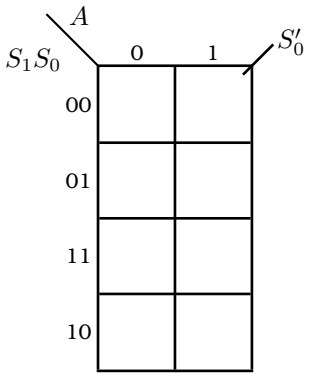


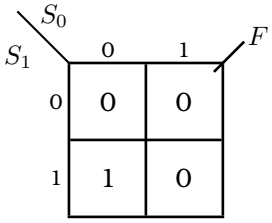
F.5 Dado el siguiente diagrama de transición en que representa una máquina de estados, complete la tabla que calcula el estado siguiente ($S_1'S_0'$) en base al estado actual (S_1S_0) y la entrada A. Se brinda el valor de la salida F en función del estado actual. Simplifique por Karnaugh las funciones $S_1'(S_1, S_0, A)$ y $S_0'(S_1, S_0, A)$. Realice la implementación de las funciones e implemente el circuito en logisim-evolution.



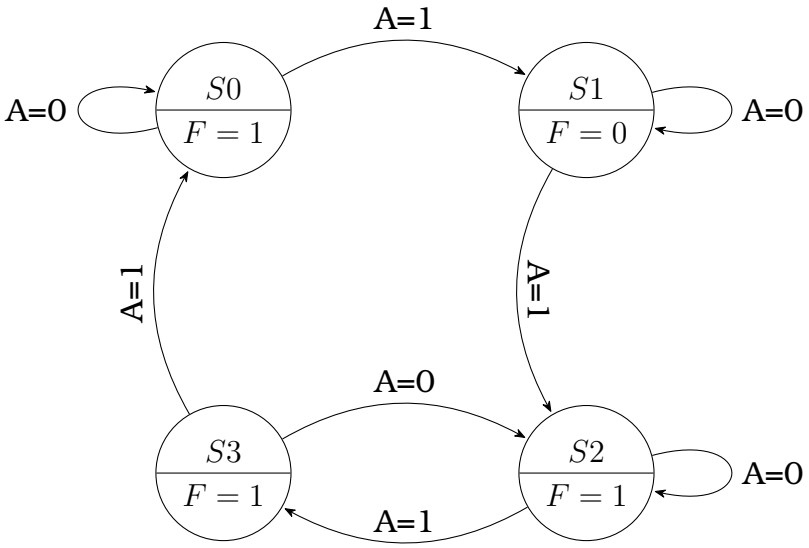
S_1	S_0	A	S_1'	S_0'	F
0	0	0			0
0	0	1			0
0	1	0			0
0	1	1			0
1	0	0			1
1	0	1			1
1	1	0			0
1	1	1			0





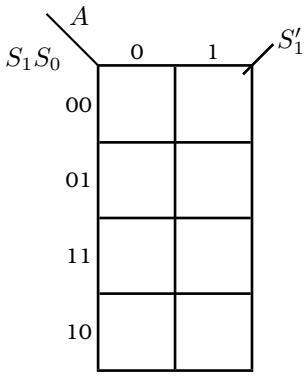


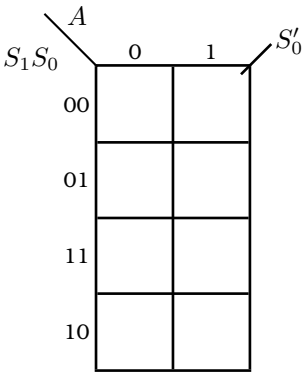
F.6 Repita los pasos del punto anterior pero para la siguiente máquina de estados.

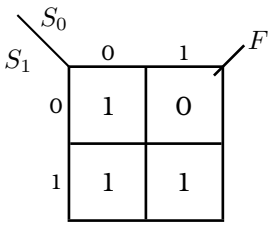


S_1	S_0	A	S_1'	S_0'	F
0	0	0			1
0	0	1			1
0	1	0			0
0	1	1			0
1	0	0			1
1	0	1			1
1	1	0			1
1	1	1			1

Simplifique







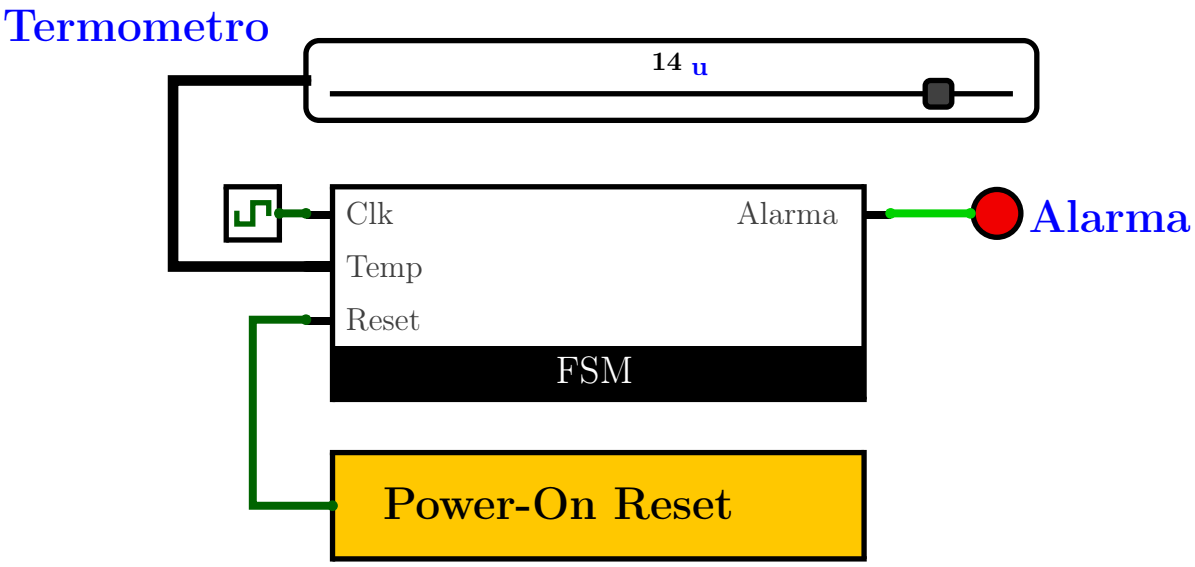
F.7 Dada la FSM de la máquina de gaseosas de la teoría. Se modifica el detector de billetes para que soporte billetes de $200 generando la salida 11. El precio de la gaseosa sigue siendo $150. Modifique la FSM para que soporte pagar con $200. Recuerda que sigue sin dar vuelto.

F.8 Utilizando FF tipo D (y lógica combinacional) implemente un contador sincrónico binario descendente de 3 bits. Plantee la tabla de verdad de las transiciones.

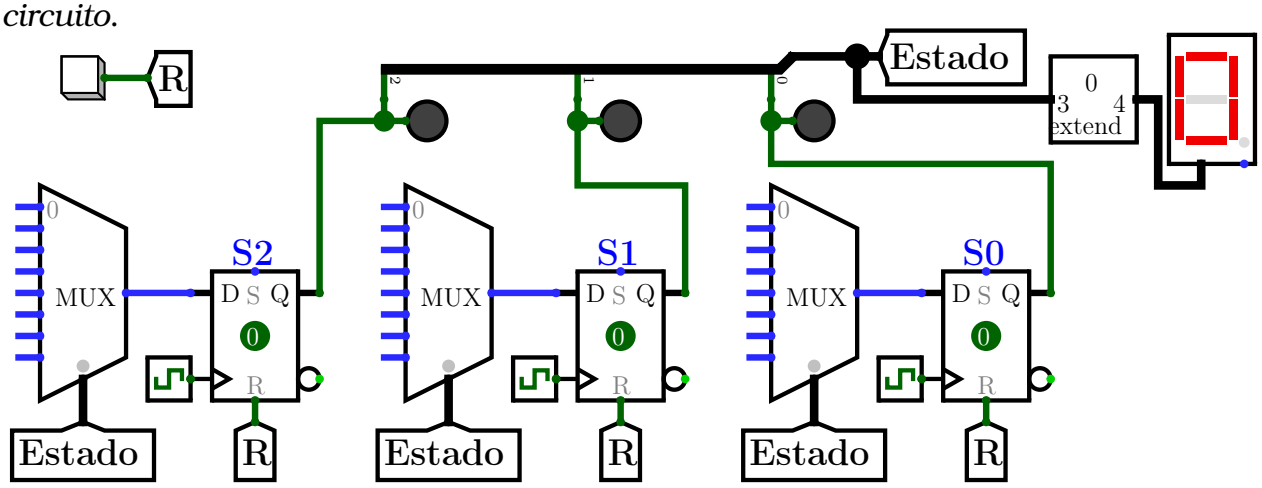
F.9 Modifique la FSM **SemaforoProgramable** para que los tiempos sean: 12 en S0, 5 en S1, 14 en S2 y 2 en S3. (Ver archivo en MIeL)

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

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 MUX funcionan como LookUp Tables en donde la selección es el estado actual (que sale de los FF D) y las entradas de datos generan el estado futuro. Indique los valores que deben conectarse en las entradas de datos de los MUXes para que se comporte como un contador binario sincrónico ascendente. *Nota: Solo debe poner unos y ceros en las entradas de datos , no debe realizar ninguna otra modificación sobre el circuito.*



Recomendamos estudiar la siguiente tabla:

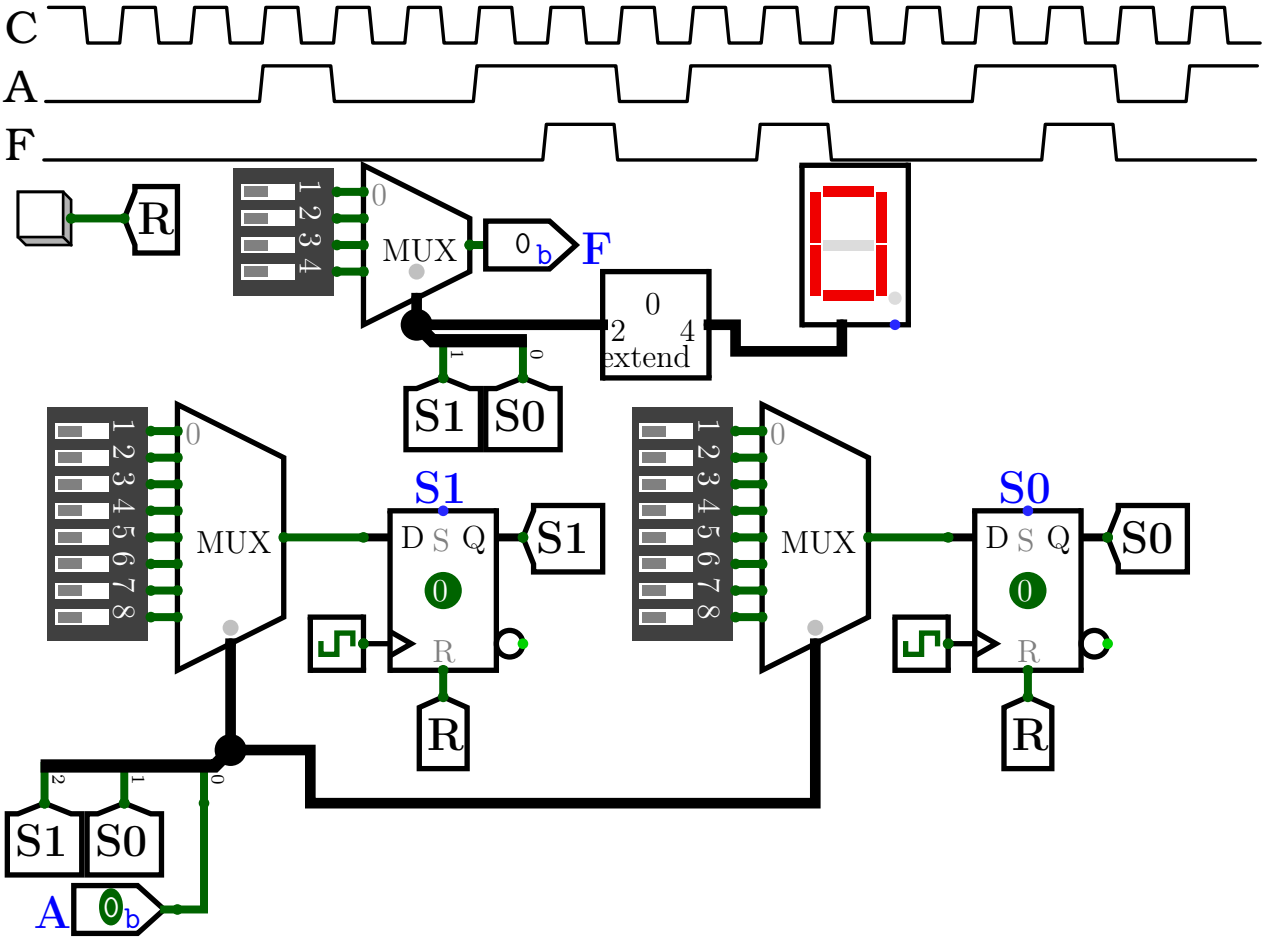
S_2	S_1	S_0	S_2'	S_1'	S_0'
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

F.13 Utilizando el circuito anterior, genere un contador binario sincrónico descendente.

F.14 Utilizando el circuito anterior, genere un contador binario sincrónico ascen-

dente módulo 5, es decir, debe contar 000,001,010,011,100 y repite.

F.15 Dada la siguiente FSM programable modifique los valores en los DIP switches para que genere un uno en la salida F cuando se detecta la secuencia de entrada 011 en A.*Nota: recomendamos comenzar realizando el diagrama de transición de estados, luego armar la tabla de verdad y de la misma tomar los valores para los DIP switches. Por ultimo pruebe como entrada 0001001101100l101, y verifique que F=1 siempre que las ultimas 3 entradas sean 011.*



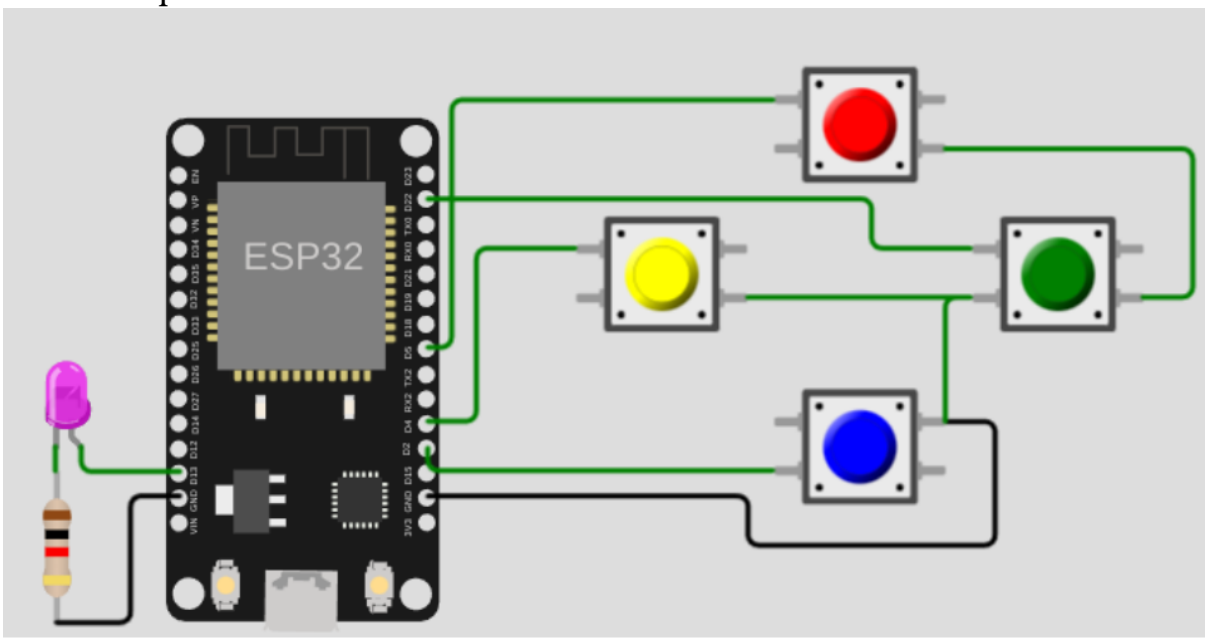
G- Máquinas de Estado Finito (FSM) Software

G.1 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.2 Modifique el diseño SemaforoFSMIntermitente para que tenga 6 segundos de verde titilante antes de pasar a amarillo.

G.3 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.

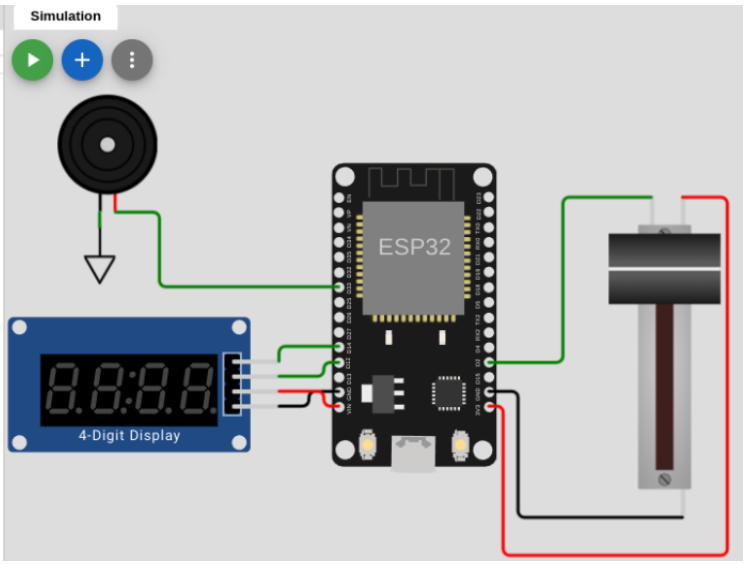
G.4 Dado el siguiente diseño con 4 botones (arriba-Rojo , abajo-Azul, izquierda-Amarillo, derecha-Verde) y un led de salida, diseñe una maquina de estados que encienda el led si el usuario ingresa la secuencia : Arriba, Arriba, Abajo, Abajo, Izquierda, Derecha, Izquierda, Derecha. Una vez ingresada la 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.5 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).

main.py | diagram.json | tm1637.py

```
from machine import ADC,Pin,PWM
import time
import tm1637
from tm1637 import TM1637
#Definimos la entrada analogica (entre 0 y 4095)
adc = ADC(2)
display = TM1637(clk=Pin(14),dio=Pin(12))
parlante = PWM(Pin(33), freq=440, duty_u16=32768)
parlante.duty(0) #ciclo de actividad 0%
while True:
    time.sleep_ms(100)
    val = adc.read()
    display.number(val)
    if val>3000:
        parlante.duty(50)
    else:
        parlante.duty(0)
```



G.6 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)
```