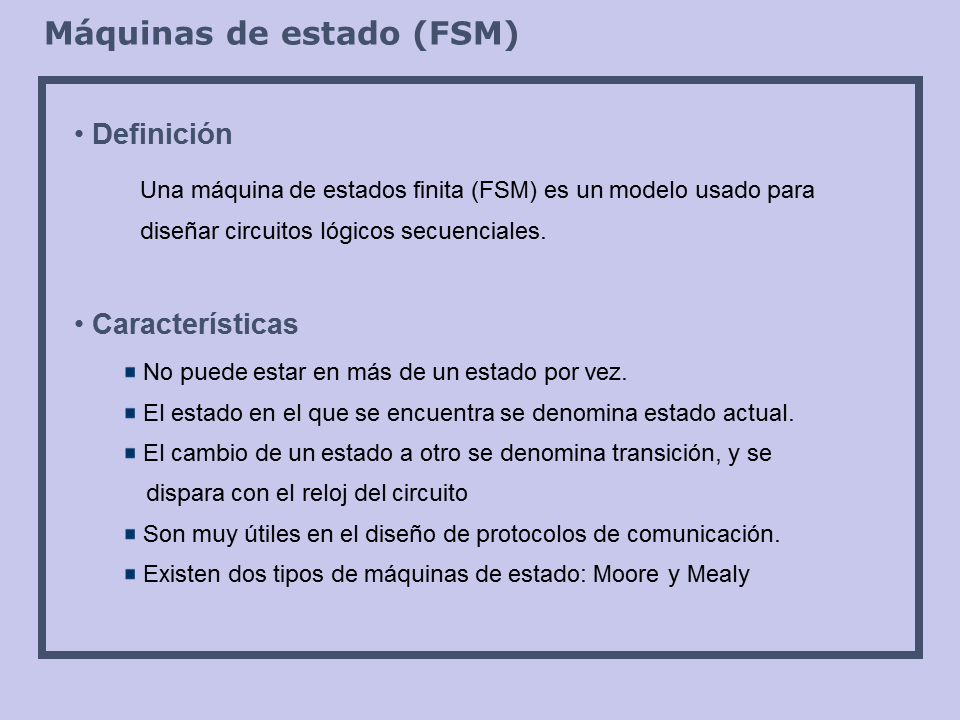
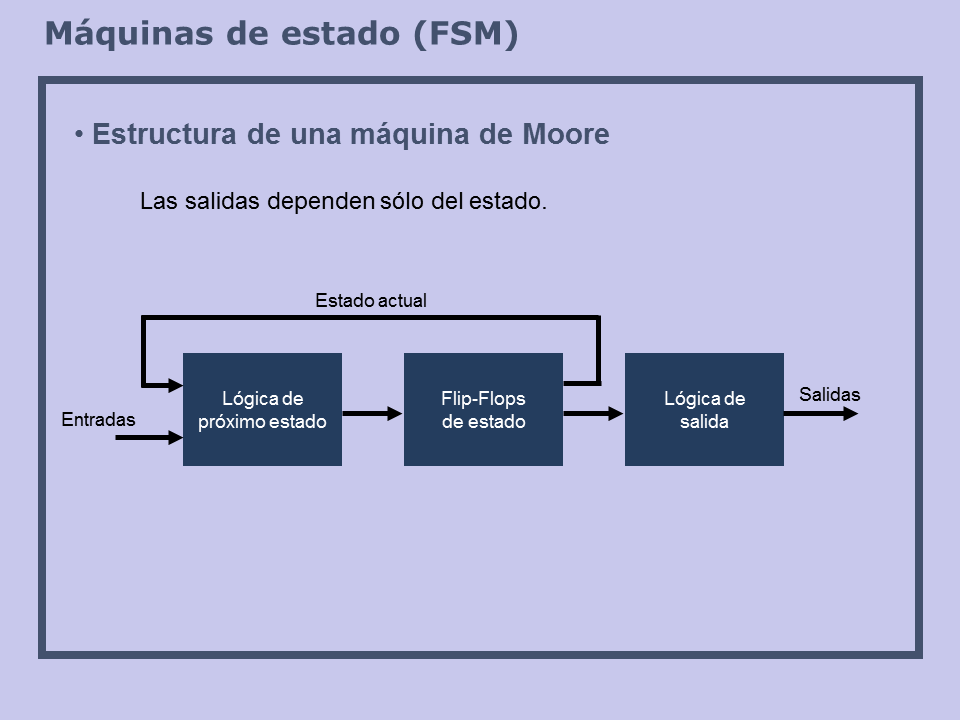
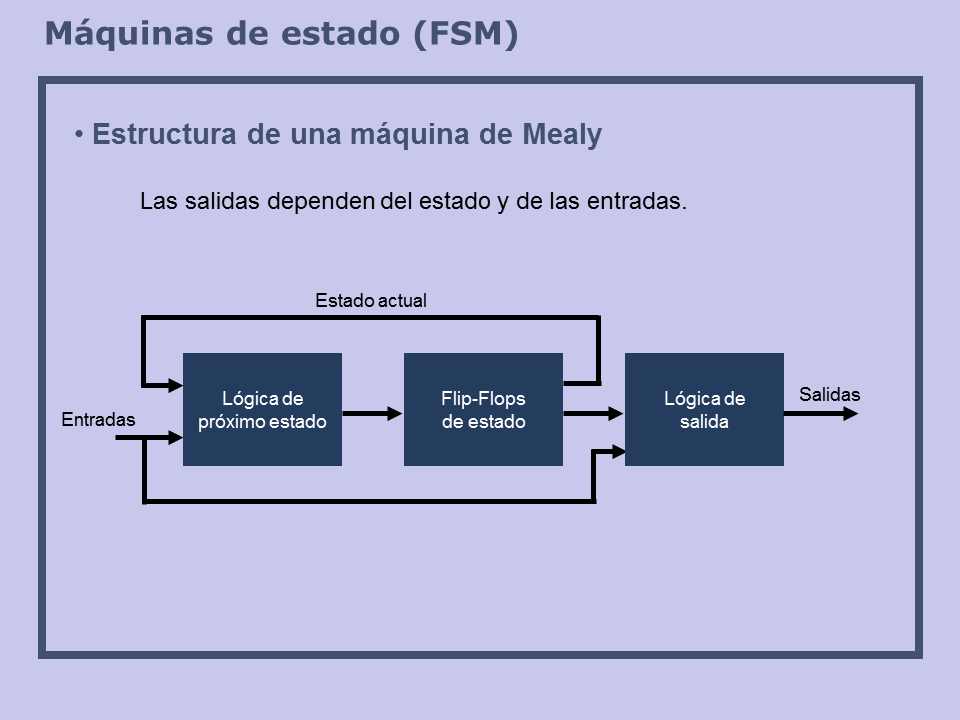
FSM LED ON – OFF

Breve descripción de una FSM (Machine state finite).



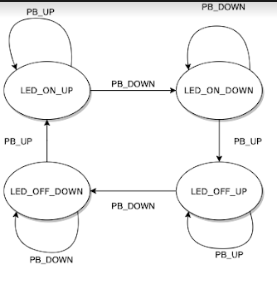




**Implementación de una máquina de estados**

**Problema:** Se desea implementar una FSM, dicha máquina de estados debe contar con un pulsador el cual facilitara una entrada de forma digital, este permitirá que cada vez que sea presionado se activara una salida, en este caso un led, este led se enciende o apagua dependiendo de su estado anterior.

1. **Esquema de la FSM tipo Moore:**



1. **Generación de código en lenguaje C.**

**2ª) Definición de variables y de funciones que vamos a necesitar a lo largo del proceso.**

**enum** states {*LED\_ON\_DOWN*, *LED\_ON\_UP*, *LED\_OFF\_DOWN*, *LED\_OFF\_UP*} current\_state;

**enum** inputs {*PB\_DOWN*,*PB\_UP*} current\_input;

**enum:** Es una manera de asociar nombres a números, y por consiguiente de ofrecer más significado a alguien que lea el código. La palabra reservada enum (en lenguaje C) enumera automáticamente cualquier lista de identificadores que se le pase, asignándoles valores de 0, 1, 2, etc.

Con ayuda del comando enum se generan dos vectores, un vector para los estados actuales y otro para las entradas actuales.

*LED\_ON\_DOWN =* Led en estado ON y botón presionado.

*LED\_ON\_UP =* Led en estado ON y botón sin presionar.

*LED\_OFF\_DOWN =* Led en estado OFF y botón presionado.

*LED\_OFF\_UP =* Led en estado OFF y botón sin presionar.

int8\_t **get\_input**(**void**);

**void** **set\_output**(int8\_t current\_state);

uint8\_t cuenta = 0;

current\_state = *LED\_OFF\_UP*;

**int8\_t** = Genera una función de nombre “get\_input” que me devuelve un numero entero de 8bits.

**Void set\_output** = Me genera una función la cual hace uso de un parámetro de entrada llamado “current\_state”, dicho parámetro es un entero de 8bits.

**Nota:** Más adelante se explicará el funcionamiento de estas funciones.

En las siguientes líneas de código se crea una varíale llamada “cuenta” la cual nos ayudara a inspeccionar el funcionamiento del código más adelante, además a la variable “current\_state” se le asigna un valor inician llamado “LED\_OFF\_UP”.

int8\_t **get\_input**(**void**)

{

**if** (!HAL\_GPIO\_ReadPin(B1\_GPIO\_Port, B1\_Pin))

**return** *PB\_DOWN*;

**else**

**return** *PB\_UP*;

}

 La anterior función cuando es llamada es la encargada de preguntar el estado actual del pulsador, si dicho pulsador se encuentra pulsado retorna “PB\_DOWN” y de lo contrario “PB\_UP”.

{

**switch**(current\_state)

{

**case** *LED\_ON\_UP*:

**case** *LED\_ON\_DOWN*:

HAL\_GPIO\_WritePin(LD2\_GPIO\_Port, LD2\_Pin, *GPIO\_PIN\_SET*);

**break**;

**case** *LED\_OFF\_UP*:

**case** *LED\_OFF\_DOWN*:

HAL\_GPIO\_WritePin(LD2\_GPIO\_Port, LD2\_Pin, *GPIO\_PIN\_RESET*);

**break**;

}

}

La función anteriormente ilustrada al ser llamada es la encargada de identificar el estado actual de la máquina de estados, y así lograr asignarle un estado ya sea alto o bajo a la salida del led según sea el caso.

2b) Desarrollo de la máquina de estados finitos dentro del bucle “While”.

 current\_input = get\_input();

**switch**(current\_state)

    {

**case** *LED\_ON\_UP*:

**switch**(current\_input)

    {

**case** *PB\_DOWN*:

    current\_state = *LED\_ON\_DOWN*;

    cuenta++;

**break**;

**default**:

    current\_state = *LED\_ON\_UP*;

    }

**break**;

**case** *LED\_ON\_DOWN*:

**switch**(current\_input)

    {

**case** *PB\_DOWN*:

    current\_state = *LED\_ON\_DOWN*;

**break**;

**default**:

    current\_state = *LED\_OFF\_UP*;

    }

**break**;

**case** *LED\_OFF\_UP*:

**switch**(current\_input)

    {

**case** *PB\_DOWN*:

    current\_state = *LED\_OFF\_DOWN*;

    cuenta++;

**break**;

**default**:

    current\_state = *LED\_OFF\_UP*;

    }

**break**;

**case** *LED\_OFF\_DOWN*:

**switch**(current\_input)

    {

**case** *PB\_DOWN*:

    current\_state = *LED\_OFF\_DOWN*;

**break**;

**default**:

    current\_state = *LED\_ON\_UP*;

    }

**break**;

    }

    set\_output(current\_state);

En la primera línea de código de hace un llamado a la función “get\_input” y el valor que esta nos arroja es almacenado en la variable anteriormente creada “current\_input”, esto me permitirá estar constantemente verificando el estado del pulsador como ya se había explicado anteriormente, arrojando los estados “PB\_DOWN” y “PB\_UP”.

Siguiendo el código encontramos la implementación de un “Switch Case” el cual toma la variable “current\_state” y dependiendo del valor que dicha variable obtenga durante la ejecución del código el “Switch Case” ejecutara diferentes acciones.

Para no hacer muy extensa esta interpretación del código se optará por explicar a profundidad un solo caso de los 4 casos posibles, ya que como se puede observar en el código todos tienen la misma estructura.

Caso donde la variable “current\_state” toma el valor de “LED\_ON\_UP”, que como describimos en la primera página de este documento significa que el led está en estado ON y el pulsador no está presionado, dentro de este caso se puede observar que se implementa otro “Switch Case”, pero en este momento se toma como referencia la variable “current\_input”, ya que se debe de verificar el estado actual de la FSM y la posición actual del pulsador, para así poder cambiar de estado. Ya que estamos en el estado “LED\_ON\_UP” la variable “current\_input” será la encargada de hacer que la FSM pase al siguiente estado “LED\_OFF\_DOWN” o se quede en el actual estado “LED\_ON\_UP”, esto lo hace asignándole a la variable “current\_state” el valor del estado siguiente o del actual ya sea el caso, dicha decisión de asignar un valor u otro está determinada directamente por la variable “current\_input” y el valor que esta toma que puede ser “PB\_DOWN” o “PB\_UP” que significa pulsador oprimido o sin oprimir respectivamente, así se permite avanzar de estado o no.

La variable “cuenta” que vemos en la porción de código anteriormente descrito como ya se había especificado su función es para verificar que el código se esté ejecutando correctamente, dicha variable incrementara su valor hasta llegar a 255 y se reiniciara, esto ocurrirá desde que se permanezca el mismo estado “LED\_ON\_UP”.