



Taller de Sistemas Embebidos Máquina de Estados



Información relevante

Taller de Sistemas Embebidos

Asignatura correspondiente a la **actualización 2023** del Plan de Estudios 2020 y resoluciones modificatorias, de Ingeniería Electrónica de FIUBA

Estructura Curricular de la Carrera

El **Proyecto Intermedio** se desarrolla en la asignatura **Taller de Sistemas Embebidos**, la cual tiene un enfoque centrado en la **práctica propia de la carrera** más que en el desarrollo teórico disciplinar, con eje en la **participación de las y los estudiantes**

Más información . . .

. . . sobre la **actualización 2023** . . . <https://www.fi.uba.ar/grado/carreras/ingenieria-electronica/plan-de-estudios>

. . . sobre el **Taller de Sistemas Embebidos** . . . <https://campusgrado.fi.uba.ar/course/view.php?id=1217>

Por Ing. Juan Manuel Cruz, partiendo de la platilla Salerio de Slides Carnival

Este documento es de uso gratuito bajo Creative Commons Attribution license (<https://creativecommons.org/licenses/by-sa/4.0/>)

You can keep the Credits slide or mention SlidesCarnival (<http://www.slidescarnival.com>), Startup Stock Photos (<https://startupstockphotos.com/>), Ing. Juan Manuel Cruz and other resources used in a slide footer



¡Hola!

Soy Juan Manuel Cruz
Taller de Sistemas Embebidos
Consultas a: jcruz@fi.uba.ar

1

Introducción

Actualización 2023 del Plan de Estudios 2020 y resoluciones . . .



Conceptos básicos

Referencia:

- ▶ Itemis CREATE (web site)
 - ▶ Utilice el poder de las máquinas de estados.
 - ▶ Con Itemis CREATE, puede crear fácilmente **sistemas complejos** de forma visual. Simule y pruebe el comportamiento de su sistema mientras modela.
 - ▶ Los generadores de código traducen su máquina de estados en código fuente de alta calidad para diferentes plataformas de destino.



Conceptos básicos

- ▷ Documentation
 - ▷ User Guide
 - ▷ Obtenga más información sobre las **máquinas de estados** en general, cómo instalar **itemis CREATE** y cómo utilizar las funciones de la herramienta para obtener el **máximo beneficio** para su **trabajo**.
 - ▷ What is a state machine?
 - ▷ Una **máquina de estados** es un **modelo de comportamiento**.
 - ▷ Consta de un **número finito de estados** y, por lo tanto, también se denomina **máquina de estados finitos (FSM)**
 - ▷ Según el **estado actual** y una **entrada determinada**, la **máquina** realiza **transiciones de estado** y **produce resultados**.



Solución Adecuada

... lo más **simple** posible, previa determinación del objetivo de **excelencia** a cumplir, obviamente contando con la **documentación debida** y recurriendo a la **metodología de trabajo adecuada**



2

Documentación debida

1er Cuatrimestre de 2024, dictado por primera vez . . .

“

Itemis CREATE (web site)

Documentation

User Guide

What is a state machine?



What is a state machine?

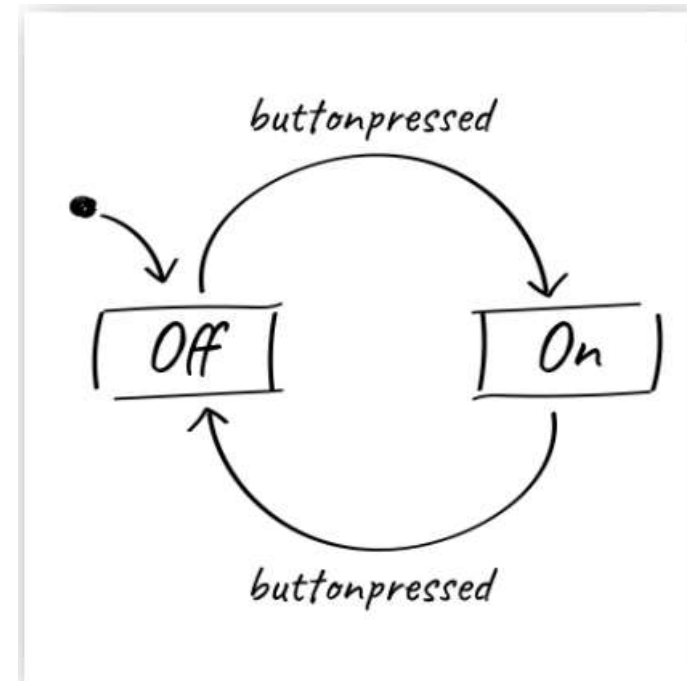
■ What is a state machine?

- ▶ Una máquina de estados es un modelo de comportamiento.
- ▶ Consta de un número finito de estados y, por lo tanto, también se denomina máquina de estados finitos (FSM)
- ▶ Según el estado actual y una entrada determinada, la máquina realiza transiciones de estado y produce resultados.
- ▶ Hay tipos básicos como las máquinas Mealy y Moore y tipos más complejos como los diagramas de estado Harel y UML.



What is a state machine?

- ▶ Los componentes básicos de una máquina de estados son los estados y las transiciones.
- ▶ Un estado es una situación de un sistema que depende de entradas anteriores y provoca una reacción a las entradas siguientes.
- ▶ Un estado está marcado como estado inicial; aquí es donde comienza la ejecución de la máquina.
- ▶ Una transición de estados define para qué entrada se cambia un estado de uno a otro.
- ▶ Dependiendo del tipo de máquina de estados, los estados y/o transiciones producen salidas.

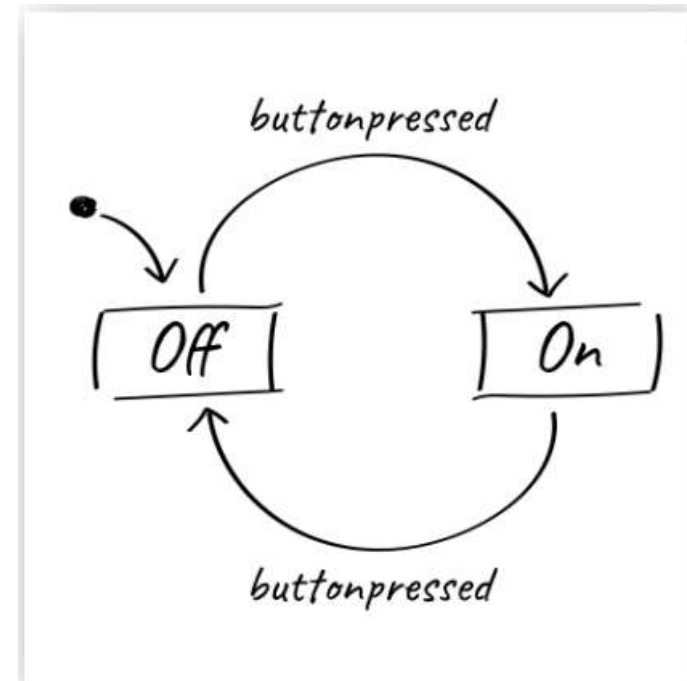


Simple state machine



What is a state machine?

- ▶ Considere la máquina de estados simple anterior.
- ▶ Consta de dos estados, apagado y encendido.
 - ▶ Off es el estado inicial aquí; se activa cuando se ejecuta la máquina de estados.
 - ▶ Las flechas entre los estados indican las posibles transiciones de estado.
 - ▶ Ellas definen para qué entrada se produce un cambio de estado.
 - ▶ Aquí, el estado activo cambia de On a Off por la entrada *buttonpressed*, y nuevamente a On por la misma entrada.

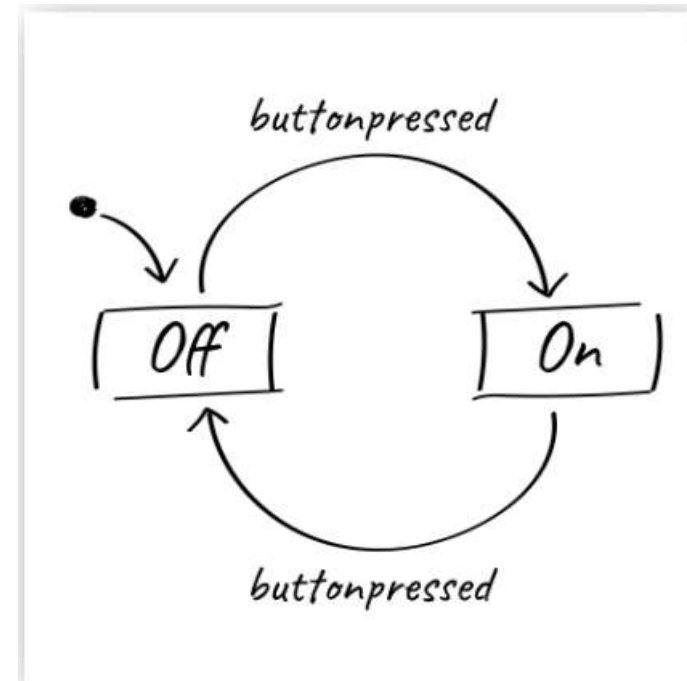


Simple state machine



What is a state machine?

- ▶ Tenga en cuenta: en la teoría de los autómatas, un autómata reacciona a las entradas y produce salidas.
 - ▶ Allí se suelen utilizar los términos **entrada** y **salida** para símbolos que pertenecen a un alfabeto.
 - ▶ Las **máquinas de estados** modernas utilizan una definición ampliada de **entradas** y **salidas**.
 - ▶ Las **entradas** pueden ser **eventos** como un clic en un botón o un disparador de **tiempo** (**time trigger**), mientras que las **salidas** son **acciones** como una **llamada** a una **operación** o una **asignación de variable**.
 - ▶ A continuación, ampliaremos el ejemplo de cambio simple para explicar las **diferencias** entre las **máquinas Mealy** y **Moore**, así como los diagramas de estado de **Harel** y las **máquinas de estado UML**.

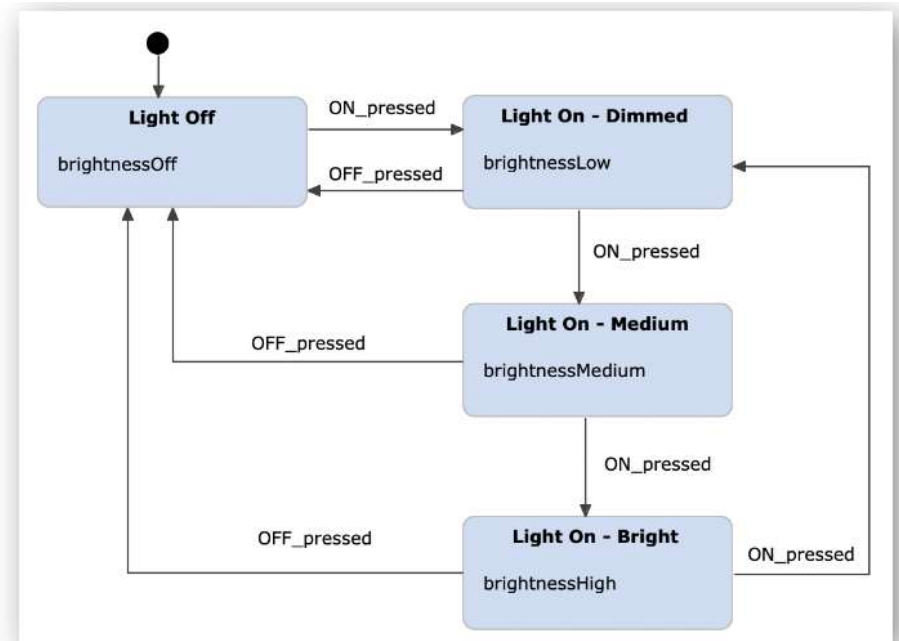


Simple state machine



Máquinas de Moore

- En la teoría de los **autómatas**, existen dos tipos básicos de **máquinas de estados finitos (FSM)**.
- Una de ellas se llama **máquina de Moore**, que lleva el nombre de su inventor **Edward Moore**, (introdujo el concepto en 1956).
- Las máquinas de Moore constan de **estados y transiciones**.
- Los **estados** pueden producir **resultados**, y los resultados están **determinados** únicamente por el **estado actual**, no por ninguna **entrada**.



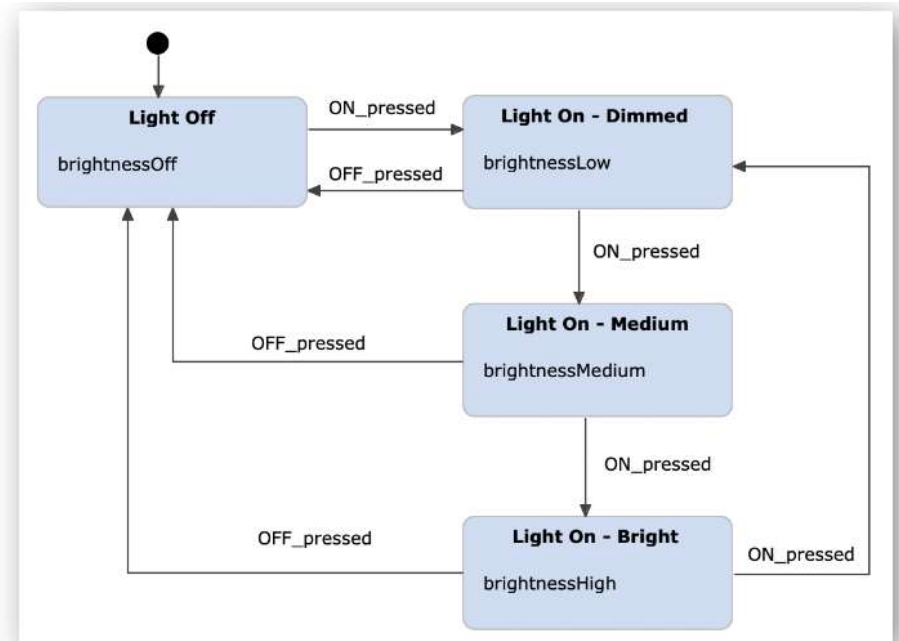
Light switch example as a Moore machine, modeled with [itemis CREATE](#)



Máquinas de Moore

Ampliamos el ejemplo del interruptor anterior a un interruptor de luz con diferentes niveles de brillo.

- El interruptor de la luz tiene dos botones, un botón de ON y un botón de OFF.
- Al presionar el botón ON se enciende la luz y se alterna entre los diferentes niveles de brillo.
- Este comportamiento está modelado por la siguiente máquina de estados.

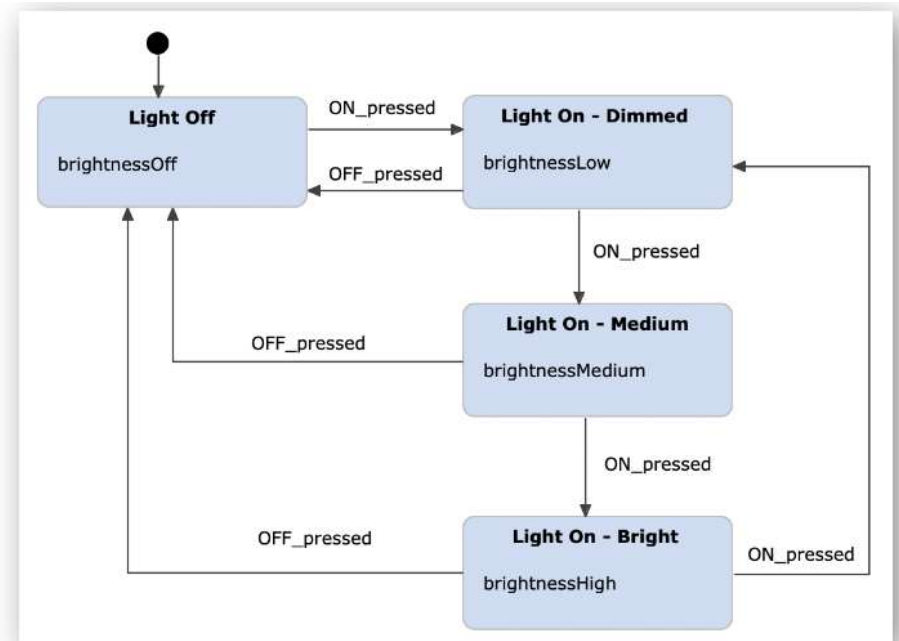


Light switch example as a Moore machine, modeled with [itemis CREATE](#)



Máquinas de Moore

- ▶ Al presionar un botón se genera un evento correspondiente (ON_pressed o OFF_pressed) ante el cual la máquina reacciona con un cambio de estado y las salidas correspondientes.
- ▶ La salida de la máquina de estados es simplemente el nivel de brillo
- ▶ Como en las máquinas Moore solo los estados producen resultados, necesitamos un estado dedicado por nivel de brillo.

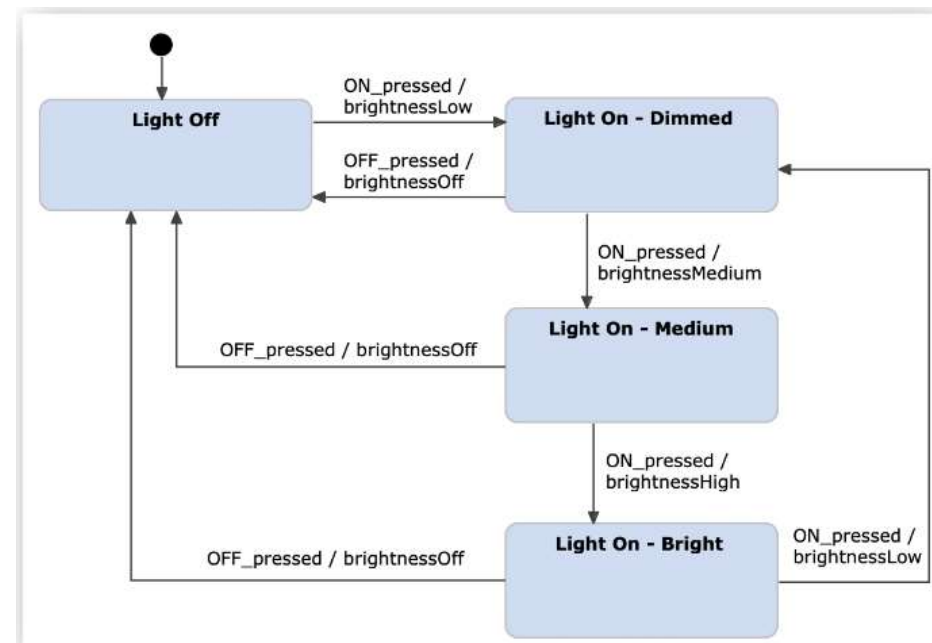


Light switch example as a Moore machine, modeled with [itemis CREATE](#)



Máquinas de Mealy

- Las máquinas **Mealy** fueron inventadas por George H. Mealy en 1955.
- En comparación con las máquinas de Moore, las máquinas Mealy producen resultados sólo en transiciones y no en estados.
- Esto a menudo da como resultado diagramas de estados con menos estados porque se puede poner más lógica en las transiciones.
- Tenga en cuenta que ambos diagramas de estado, la máquina de Moore anterior y la de Mealy, describen exactamente el mismo sistema. De hecho, la teoría de los autómatas afirma que siempre se puede traducir una máquina de Moore a una máquina de Mealy y viceversa, sin perder expresividad.



Light switch example as a Mealy machine



Diagramas de Estado de Harel

- Aunque las máquinas Mealy ya pueden reducir el número de estados requeridos, en el caso de sistemas complejos, estos autómatas se vuelven fácilmente inmanejables. O para decirlo en palabras de David Harel:
 - "Un sistema complejo no puede describirse beneficiosamente de esta manera ingenua, debido a la multitud de estados inmanejables y en crecimiento exponencial, todos los cuales tienen que organizarse de una manera plana y no estratificada, lo que resulta en un diagrama de estados desestructurado, poco realista y caótico."
- Harel concluyó que "un enfoque de estados debe ser modular, jerárquico y bien estructurado" e introdujo conceptos adicionales como estados compuestos y ortogonalidad (paralelismo).



Diagramas de Estado de Harel

- Acuñó el término “statechart” y lo definió como:
 - ▷ “statecharts = state-diagrams + depth + orthogonality + broadcast communication”
- Básicamente, los diagramas de estado de Harel son máquinas de Mealy/Moore ampliadas con conceptos adicionales que nos permiten modelar sistemas complejos de una manera práctica.
- Al utilizar estados compuestos y subdiagramas, podemos aportar más profundidad a los diagramas de estado, manteniendo al mismo tiempo los diagramas claros y bien estructurados.



Diagramas de Estado de Harel

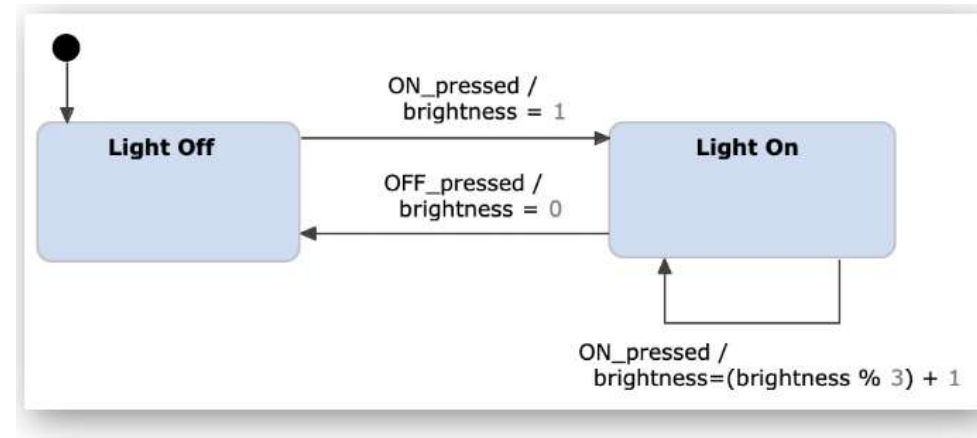
- Las **regiones** nos ayudan a expresar la **ortogonalidad**: diferentes máquinas de subestados que se pueden **ejecutar** una junto de la otra.
- Los **eventos** nos permiten lograr una **comunicación en broadcast** y nos brindan un medio sólido para describir **comportamientos complejos**.
- Usando **guardas**, podemos afirmar que un determinado **evento desencadena una transición** solo si se **cumple** una **condición** determinada.
- Las **transiciones** entre niveles, los **estados históricos**, la **lógica temporal**, así como las **acciones al entrar, salir y permanecer** en un estado son otros elementos del diagrama de estado de Harel.



Diagramas de Estado de Harel

Los diagramas de estado de Harel pueden definir variables que se pueden utilizar en expresiones de entrada y salida.

- ▶ En cuanto al ejemplo del interruptor de luz, esto nos permite almacenar el nivel de brillo en una variable en lugar de en varios estados.
- ▶ De esa manera, podemos simplificar el diagrama de estados fusionando todos los estados de Light On en uno y ejecutando las acciones de salida en una autotransición.
- ▶ Aquí simplemente incrementamos el valor de brillo cada vez que se realiza la transición.

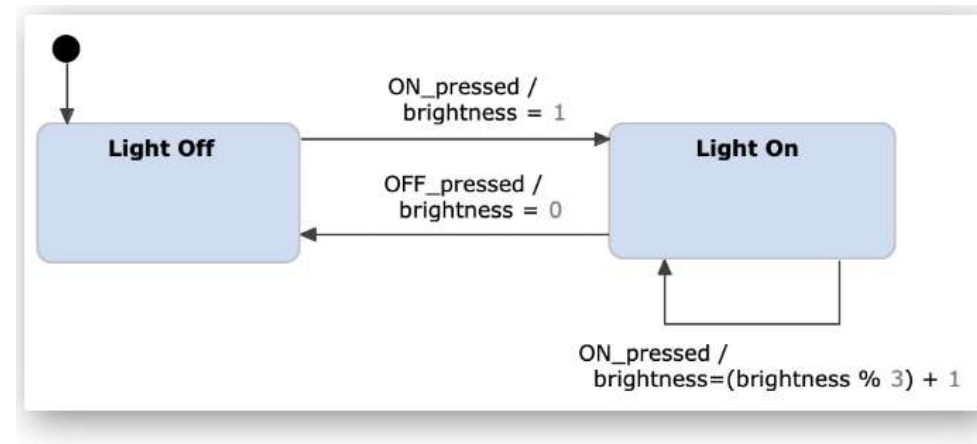


Light switch example as a Harel statechart



Diagramas de Estado de Harel

- ▶ Usamos el operador módulo para asegurar que el valor de brillo permanezca entre 1 y 3.
- ▶ Esto tiene la ventaja de que podemos cambiar la cantidad de niveles de brillo sin agregar nuevos estados.



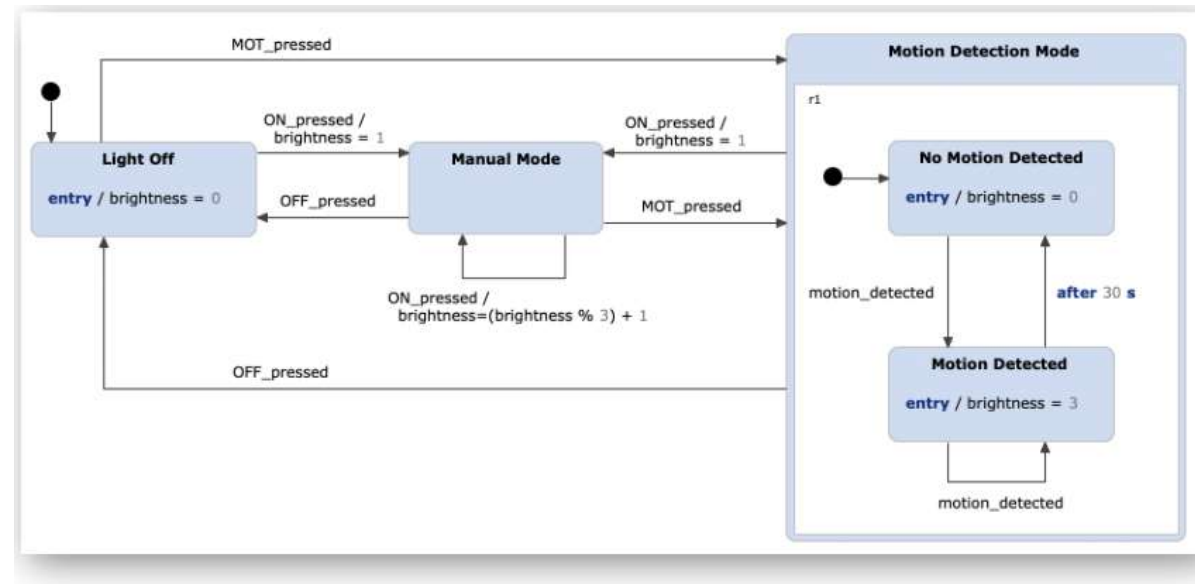
Light switch example as a Harel statechart



Diagramas de Estado de Harel

Para mostrar el uso de **estados compuestos** ampliamos el ejemplo del interruptor de luz con un modo de detección de movimiento.

- ▶ Cuando se presiona el botón MOT, se activa el sensor de movimiento.
- ▶ Una vez que el sensor detecta cualquier movimiento (evento `motion_detected`), la luz se enciende al nivel de brillo más alto (brillo = 3).



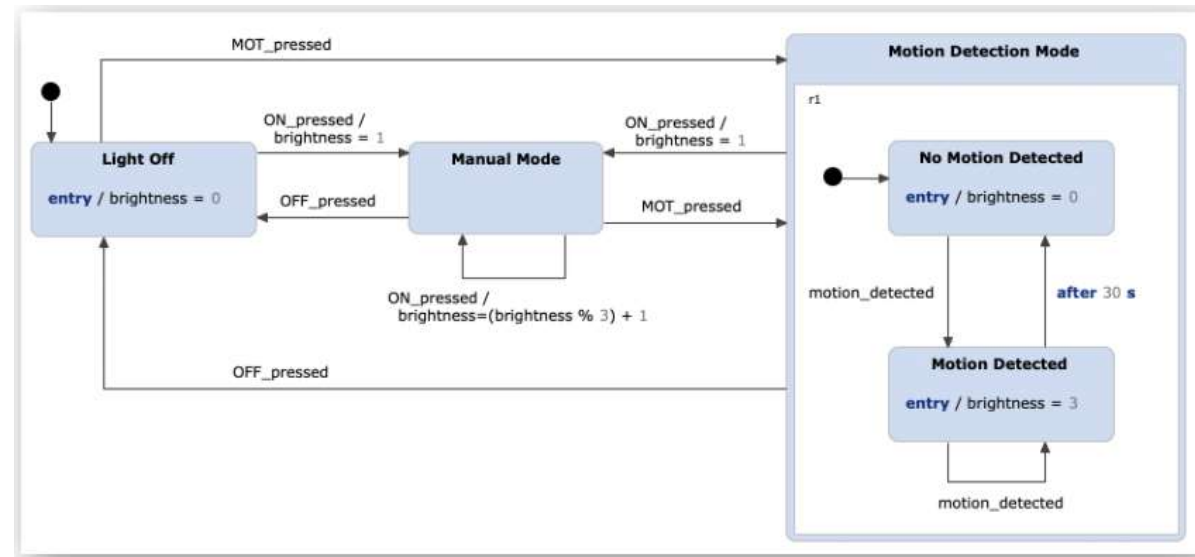
Extended light switch example as a Harel statechart with composite states



Diagramas de Estado de Harel

- Esto se puede modelar con un estado compuesto que agrupe los dos estados Motion Detected y No Motion Detected.

Tenga en cuenta también que los diagramas de estados de Harel combinan las características de las máquinas de Mealy y Moore, por lo que los resultados pueden producirse por estados y transiciones como se indica en el diagrama de estados anterior.

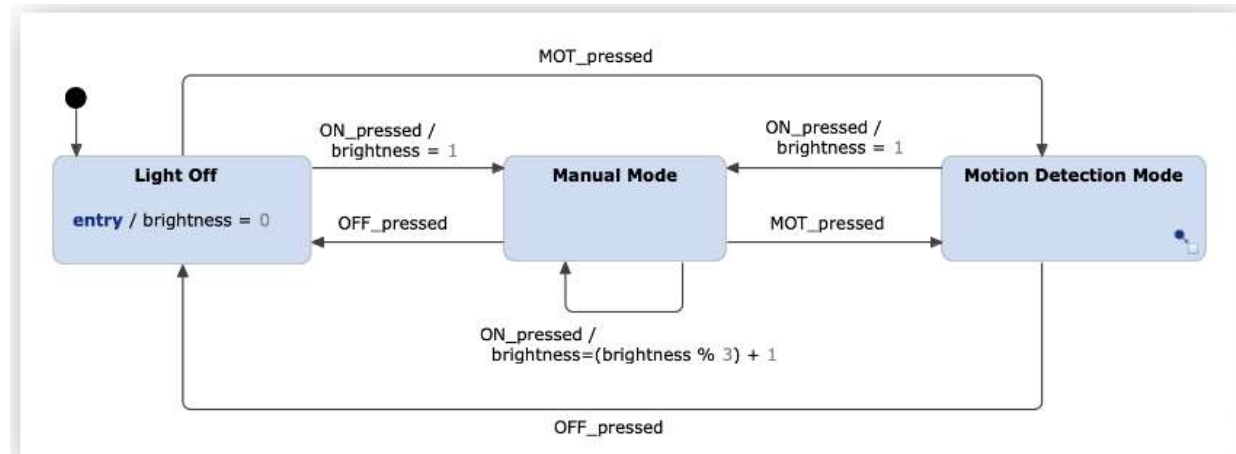


Extended light switch example as a Harel statechart with composite states



Diagramas de Estado de Harel

- Incluso podemos ir un paso más allá y **ocultar** la lógica del modo de **detección de movimiento** en un subdiagrama.
- De esa manera, el **sistema** se vuelve **más comprensivo** ya que se pueden **ver directamente los diferentes modos** y cómo **cambiar** entre ellos.
- Otros conceptos como **ortogonalidad** o **estados históricos** se omiten aquí por motivos de brevedad. Leer quick reference =>
https://www.itemis.com/en/products/item-is-create/documentation/user-guide/quick_ref#quick_ref



Extended light switch example as a Harel statechart with sub diagrams



The present age: UML state machines

Las máquinas de estados UML se basan en la notación de diagrama de estados introducida por David Harel.

Además, UML amplía la notación de los diagramas de estado de Harel mediante principios orientados a objetos.

- Asignando esto a nuestro ejemplo de interruptor de luz, en UML podemos modelar las posibles acciones del interruptor de luz como un tipo con operaciones `turnOn()`, `turnOff()`, `setBrightness(value)` y así sucesivamente.

La siguiente tabla ilustra de un vistazo las diferencias entre los tipos descritos anteriormente.

	Mealy	Moore	Harel	UML
States and transitions	✓	✓	✓	✓
Transitions produce output	✓		✓	✓
States produce output		✓	✓	✓
Depth (hierarchies, composite states)			✓	✓
Orthogonality (parallel substatemachines)			✓	✓
Broadcast communication (events)			✓	✓
History, actions, delays, timeouts, conditions			✓	✓

Differences between the state machine types



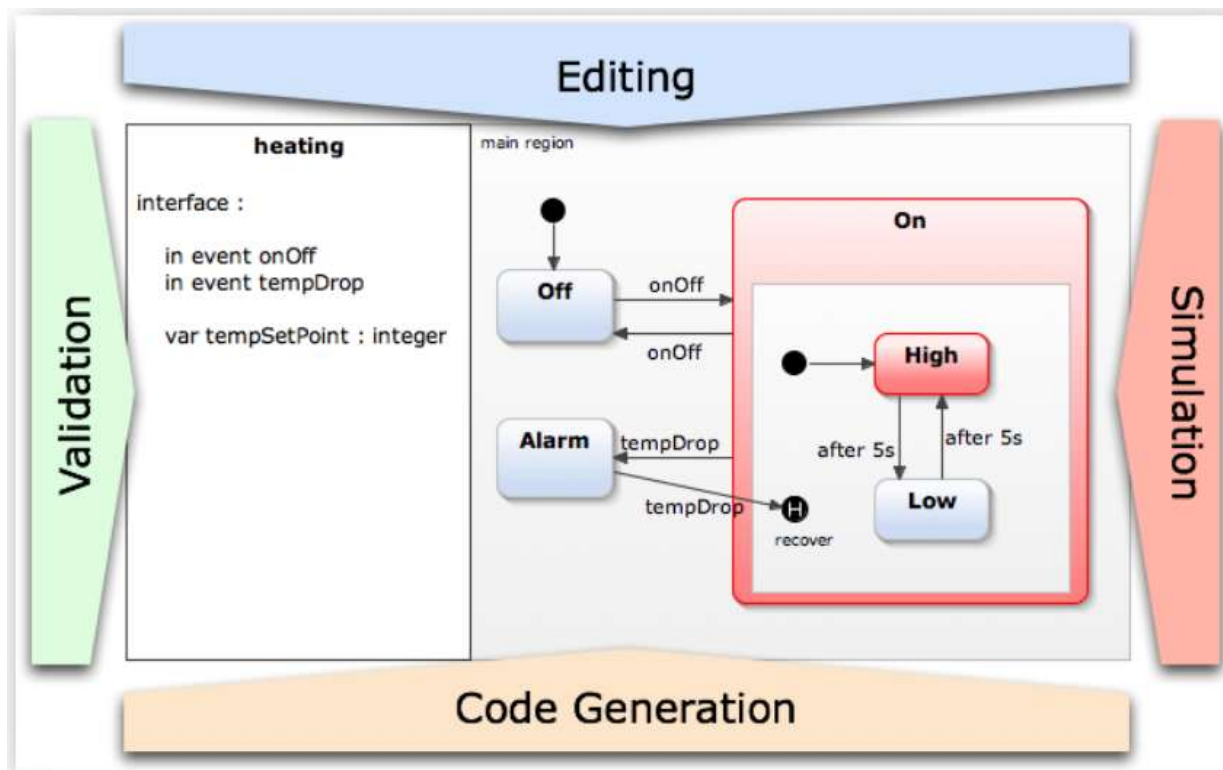
The present age: UML state machines

■ Obtenga más información sobre el modelado de sistemas con máquinas de estados en nuestro documento técnico gratuito:

- ▶ Los ejemplos de gráficos de estado de este artículo fueron diseñados con **Itemis CREATE**: <https://www.itemis.com/en/yakindu/state-machine/>, cuya documentación está leyendo ahora mismo.
- ▶ Los **diagramas de estado** creados se basan en los gráficos de estado de **Harel** y son muy cercanos, pero no idéntico a las **máquinas de estado UML**. Las diferencias concretas se explican en la documentación disponible.
- ▶ **Itemis CREATE** viene con un **simulador de diagrama de estado** que permite ejecutar directamente los **diagramas de estado modelados**. Varios **generadores de código** traducen el diagrama de estado al código fuente.



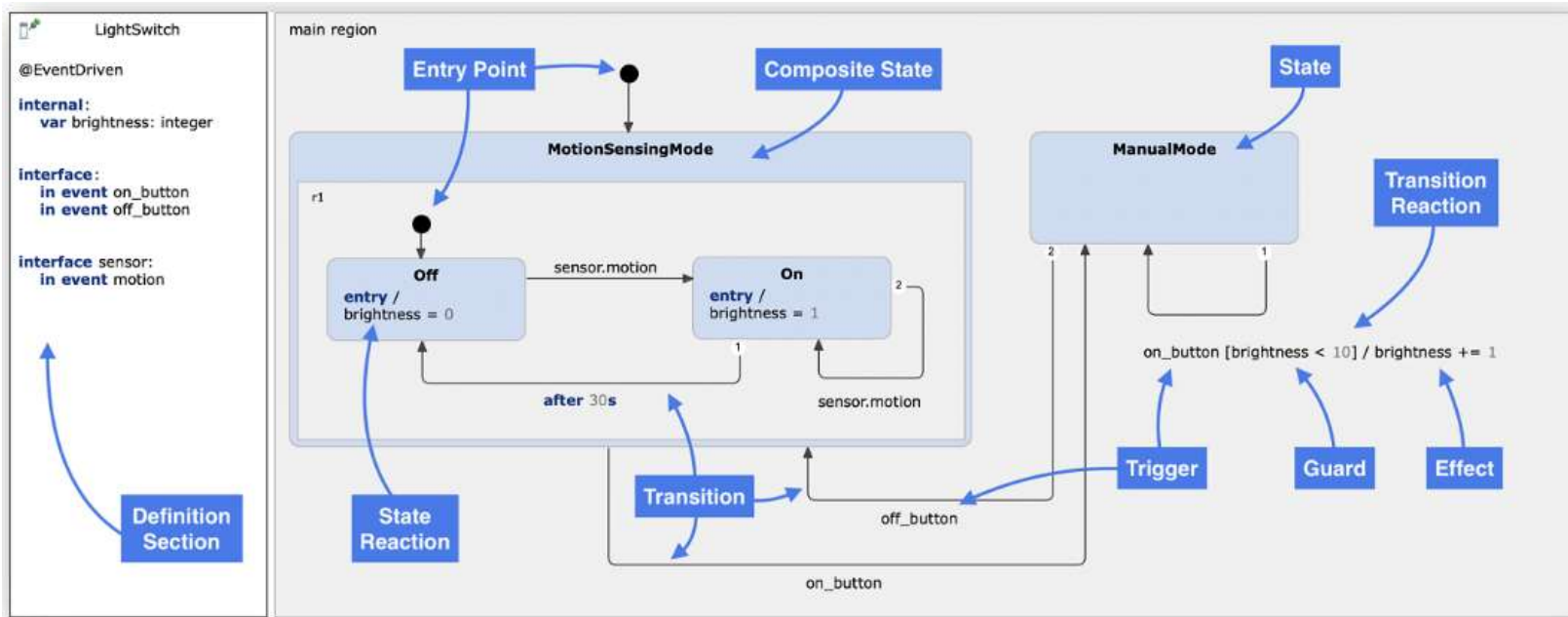
Itemis CREATE



Features of itemis CREATE



Itemis CREATE



Overview of a statechart's basic building blocks



Diseño Paso a Paso

- A partir de las especificaciones del **sistema**:
 - ▷ Primer paso Identificar los **eventos** y las **acciones**
 - ▷ Segundo paso Identificar los **estados**
 - ▷ Tercer paso Agrupar por **jerarquías**
 - ▷ Cuarto paso Agrupar por **paralelismo**
 - ▷ Quinto paso Añadir las **transiciones**
 - ▷ Sexto paso Añadir las **sincronizaciones**
- Elegida una herramienta de software podremos: **Editar, Verificar y Validar** (Simular) el **diagrama de estados**
- Culminando con la **generación del código** (opción posible dependiendo de la herramienta de software)



Convención de identificadores

■ A fin de no confundir los elementos del diagrama de estados es recomendable adoptar una convención para identificarlos.

▷ Por ejemplo, recurriendo a prefijos/sufijos para identificar algunos elementos como:

▷ Evento	ev_	Acción/Actividad	op_
▷ Máquina de Estados	sm_		
▷ Variable de Estado	_st	Variable de Evento	_ev

■ Cuidado, hay palabras reservadas:

- ▷ internal, interface, state, event
- ▷ entry, exit, do, after, always, oncycle, else, default, etc.



Codificación en C

Tenemos múltiples opciones para codificar en C diagramas de de estado finito, a saber:

- ▶ Mediante combinaciones de **switch** & múltiples **if** o **punteros a función** (variables de control: state & event)
- ▶ Tablas de Estados de una o dos dimensiones
- ▶ Patrones de diseño de estado orientados a objetos
- ▶ Otras técnicas que combinan a las anteriores (**frameworks**)

Table 3.1: Two-dimensional state table for the time bomb

		Events →			
		UP	DOWN	ARM	TICK
States ↓	Setting	setting_UP(), setting	setting_DOWN(), setting	setting_ARM(), timing	empty(), setting
	Timing	timing_UP(), timing	timing_DOWN(), timing	timing_ARM(), setting(*)	timing_TICK(), timing(**)

Notes:

(*) The transition to “setting” is taken only when (me->code == me->defuse).

(**) The self-transition to “timing” is taken only when (e->fine_time == 0) and (me->timeout != 0).



Codificación en C

Table 3.2: One-dimensional state transition table for the time bomb

Current State	Event (Parameters)	[Guard]	Next State	Actions
setting	UP	[me->timeout < 60]	setting	++me->timeout; BSP_display (me->timeout);
	DOWN	[me->timeout > 1]	setting	--me->timeout; BSP_display (me->timeout);
	ARM		timing	me->code = 0;
	TICK		setting	
timing	UP		timing	me->code <=&= 1; me->code = 1;
	DOWN		timing	me->code <=&= 1;
	ARM	[me->code == me->defuse]	setting	
	TICK (fine_time)	[e->fine_time == 0]	choice	--me->timeout; BSP_display (me->timeout);
		[me->timeout == 0]	final	BSP_boom();
		[else]	timing	



Referencias

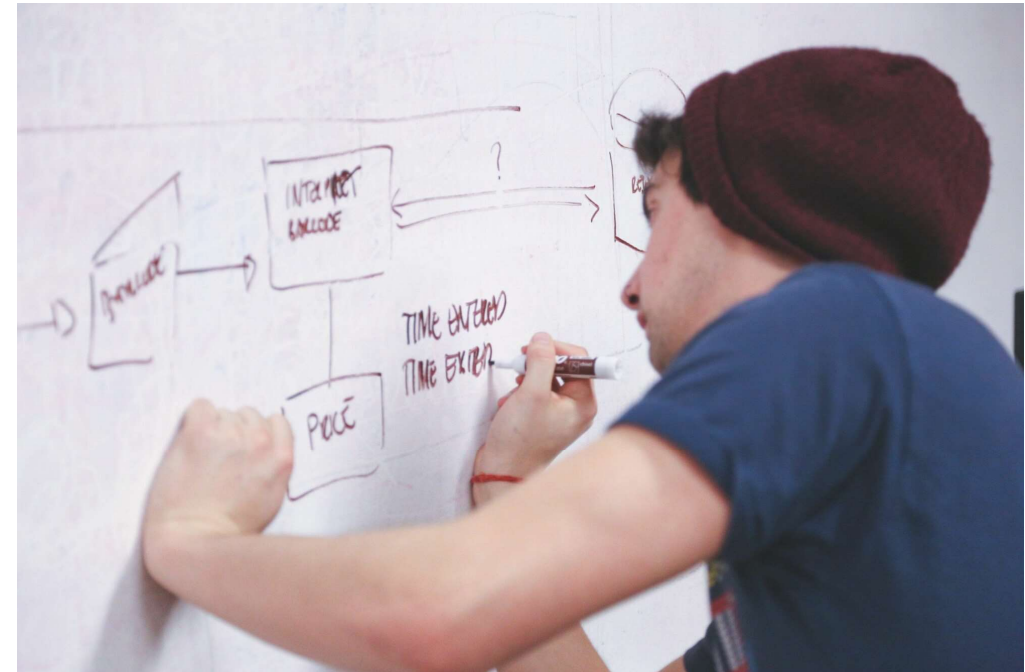
- Itemis CREATE - <https://www.itemis.com/en/products/itemis-create/>
- Itemis CREATE, What is a state machine? - https://www.itemis.com/en/products/itemis-create/documentation/user-guide/overview_what_are_state_machines?hsLang=en
- Itemis CREATE, Quick reference - https://www.itemis.com/en/products/itemis-create/documentation/user-guide/quick_ref?hsLang=en
- Itemis CREATE, Statechart language reference - https://www.itemis.com/en/products/itemis-create/documentation/user-guide/sclang_statechart_language_reference?hsLang=en
- Practical UML Statecharts in C/C++, 2nd Ed, Event-Driven Programming for Embedded Systems by Miro Samek, Ph.D. - <https://www.state-machine.com/psicc>
- Reactive System Framework by Leandro Francucci - <http://sourceforge.net/projects/rkh-reactivesys/>



Manos a la obra con el . . .

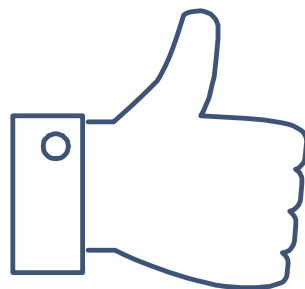
. . . Proyecto Intermedio

. . . un enfoque centrado en la práctica propia de la carrera más que en el desarrollo teórico disciplinar, con eje en la participación de las y los estudiantes



A person with short dark hair, seen from the back, is looking at a wall covered in various design sketches, photos, and notes. The wall is a collage of creative work, including wireframes, hand-drawn diagrams, and photographs of people and objects. The person is wearing a grey and black striped sweater. The overall scene suggests a creative or design studio environment.

Las y los estudiantes preguntarán:
¿en qué lío nos metimos?



¡Muchas gracias!

¿Preguntas?

...

Consultas a: jcruz@fi.uba.ar