Diseño de interacción

Eloi Maduell i García
Santiago Vilanova Ángeles

PID_00184751

Material docente de la UOC

UOC
Universitat Oberta de Catalunya

Eloi Maduell i García

Formado como diseñador gráfico, alterna el desarrollo de proyectos artísticos basados en la programación de software, el diseño interactivo y la investigación sonora con su actividad como grafista e ilustrador. Miembro del colectivo Telenoika, ha participado como creativo en numerosos proyectos relacionados con el mapping audiovisual; la aplicación de tecnologías interactivas en el teatro con la compañía Playmodes; el reciclaje creativo de deshechos tecnológicos, como miembro de los Luthiers Drapaires; la robótica, la música experimental y la docencia en estos campos. Sus trabajos se han expuesto en salas de teatro, exposiciones y festivales como Sónar (2006 y 2009), Palau Euskalduna de Bilbao, Kunst Akademie de Berlín, Mapping Festival de Ginebra, Primavera Sound, CCCB, CaixaFòrum, Palau de la Música, Laboral, MIAC, Festival VAD o Mercat de les Flors, entre otros. Actualmente es profesor en distintas universidades catalanas en materias relacionadas con el diseño interactivo y la creatividad audiovisual. Parte de su trabajo se puede consultar en las siguientes direcciones:
http://www.playmodes.com
http://www.telenoika.net/drapaires
http://vimeo.com/ox

Santiago Vilanova Ángeles

Ingeniero informático especialista en la creación audiovisual en directo. Desde hace más de 12 años explora e investiga las posibilidades de la creación digital en directo. Miembro fundador de la Asociación Cultural Telenoika, que desde hace más de 10 años dinamiza la escena audiovisual más independiente de nuestro país a través de sus actividades en el campo de las artes audiovisuales y las nuevas tecnologías de la comunicación. Con el equipo de Telenoika Mapping ha desarrollado proyectos de vídeo-mapping desde el año 2009. Coordinador del Festival de Audiovisuales y Nuevas Tecnologías: VideA, durante los años 2000, 2001 y 2002, uno de los festivales pioneros a nivel mundial en difundir explícitamente el trabajo de los Video-Jockeys más conocidos actualmente. Realizador de cortometrajes de creación audiovisual, como Telenoia (1998), Eco (2000) o Somni de Terra i Aigua (2003). Con el sobrenombre de 'pause vj' ha actuado en todo tipo de festivales de creación audiovisual por todo el territorio, a nivel nacional e internacional desde el año 2002 al año 2006. Ganó en el 2005 el Premio Vjology en Holanda. Director técnico del Festival VAD desde el año 2006, de cuya organización técnica se encarga actualmente y lleva la programación más experimental y joven.Desde hace unos años, imparte cursos y talleres de creación audiovisual en directo y vídeo-mapping en centros cívicos, instituciones, festivales, universidades y posgrados. Desarrollador dentro del colectivo Playmodes juntamente con Santiago Vilanova, con quien desarrolla aplicaciones interactivas enfocadas a la escena.

El encargo y la creación de este material docente han sido coordinados por el profesor: Quelic Berga Carreras (2012)

Primera edición: febrero 2012

Avda. Tibidabo, 39-43, 08035 Barcelona
Diseño: Manel Andreu
Realización editorial: Eureca Media, SL
Depósito legal: B-5.145-2012

Introducción

1) Introducción al diseño de interacción

La **explosión de la electrónica** de consumo iniciada en los años cincuenta, que ha ido en progresión exponencial hasta hoy en día, ha inundado nuestras vidas de un gran número de aparatos electrónicos con numerosas funciones. Los primeros dispositivos electrónicos de consumo, como las radios o los televisores, ya incorporaban, de una manera u otra, interfaces de control mediante botones y potenciómetros que facilitaban a los usuarios el manejo de los aparatos.

Fuente: Wikipedia (cc)

La revolución **digital** iniciada en los años ochenta representa un paso más en la sofisticación y las posibilidades de estos artefactos electrónicos, que actualmente se han convertido en elementos imprescindibles de nuestras actividades cotidianas.

La gran proliferación de ordenadores, videoconsolas, televisores, calculadoras, teléfonos móviles, teclados electrónicos, etc., y la creciente incorporación de funcionalidades a estos aparatos han obligado a los diseñadores y a los fabricantes a hacer estudios orientados a la usabilidad de los productos para facilitar a los usuarios la interacción con los dispositivos.

En este contexto emerge la disciplina del **diseño de interacción**, que tiene el objetivo de hacer más **amigable** el uso de los aparatos.

Hoy en día, con la extensa proliferación de aparatos digitales de última generación, como asistentes digitales personales o PDA, teléfonos inteligentes o *smartphones* y ordenadores portátiles, la aplicación de conceptos del diseño de interacción se hace imprescindible para cualquier fabricante de productos electrónicos digitales. Elementos como el ratón, el teclado o el mando a distancia surgen como respuesta a esta necesidad de usabilidad de los nuevos productos electrónicos y han pasado a ser de gran utilidad para los usuarios. El abaratamiento progresivo de la microelectrónica digital y los microprocesadores ha favorecido la generalización de tecnologías que solo hace unos años eran patrimonio de un grupo reducido de especialistas. Actualmente, dispositivos como las redes Ethernet, el Wi-Fi, las videocámaras o los sistemas de microfonía no solo son accesibles a casi todo el mundo, sino que además a menudo los encontramos integrados en los productos electrónicos, lo que permite el diseño de interacciones cada vez más sofisticadas.

Fuera de la esfera de los productos de mercado, en el campo de la experimentación artística con las nuevas tecnologías, se emplean estas técnicas de diseño interactivo para hacer partícipes a los espectadores, convirtiéndolos así en su-

jetos activos de la composición de las obras, más allá del paradigma de sujeto contemplativo. Es interesante fijarse en la proliferación actual de videoinstalaciones interactivas, arte sonoro participativo o arte en línea que usan de una manera u otra estas técnicas de diseño interactivo, invitando a los espectadores/usuarios a participar como sujetos activos de las propuestas artísticas.

Así pues, el diseño de interacción es la disciplina que define el comportamiento de los productos y los sistemas con los que interactúa el usuario.

Ciertos principios básicos de la psicología cognitiva ofrecen la base para el diseño de las interacciones, y en este contexto se desarrollan interacciones que obedecen a principios como los del mapa mental (asociación de conceptos) o la metáfora de la interfaz (como el escritorio). Los productos del diseño de interacción son típicamente desarrollados mediante análisis y pruebas con usuarios, y su diseño se evalúa en términos de usabilidad e influencia afectiva.

Hemos de tener presente en todo momento que la disciplina del diseño interactivo no es un fin en sí mismo, sino que debe ayudar a los usuarios a interactuar con un sistema determinado, y que, por lo tanto, tendría que reducir al mínimo su complejidad de uso, sin eliminar funcionalidades importantes. Es clave que entendamos que la interactividad por sí misma no tiene ningún valor si no ayuda a llevar a cabo acciones complejas de manera simple y eficaz.

A lo largo de este documento, repasaremos los distintos conceptos y las prácticas más habituales en el campo del diseño interactivo. Profundizaremos en nociones del ámbito del análisis de audio, la visión por ordenador, el diseño electrónico interactivo o la interactividad mediante ratón y teclado. Estos materiales os servirán para hacer un repaso fundamental y os permitirán tener una idea general de las posibilidades que ofrece este sector. La asignatura contiene otros materiales, como los casos de estudio *Mosaic*, los códigos de ejemplo con comentarios, las propuestas prácticas, los vídeos tutoriales y los enlaces externos de referencia. Por medio del análisis de casos reales, podremos ver cómo se orientan proyectos de diseño interactivo en la esfera del mercado, la práctica artística o el entorno industrial. El objetivo principal es que dispongáis de una visión lo bastante amplia y completa como para poder hacer propuestas críticas y coherentes en cada contexto y aprender a analizar las ya existentes.

2) Entrada (*input*) - proceso - salida (*output*)

Podríamos resumir el proceso de la interacción con la secuencia clásica (aplicable a todos los procesos de computación) entrada (*input*) - proceso - salida (*output*).

A grandes rasgos, cuando interactuamos con un sistema, lo hacemos introduciendo datos en este mediante periféricos de entrada (el ratón, el teclado, la palanca de control o *joystick*, cámaras, sensores...). Estos datos son procesados y analizados por el sistema de software en la CPU. Una vez analizados estos datos de entrada, el sistema interactivo responde provocando una acción determinada en algún periférico de salida (el monitor, la impresora, un videoproyector, los altavoces...).

3) Los módulos

Hemos estructurado esta documentación atendiendo a seis grandes bloques temáticos que consideramos fundamentales en el ámbito del diseño de interacción.

El primer módulo, "**Teclados**", hace referencia a las características, los usos y las potencialidades de las interfaces basadas en teclados alfanuméricos y musicales.

El segundo módulo, "**Dispositivos apuntadores**", analiza toda una serie de dispositivos físicos que nos permiten obtener datos de posicionamiento en el plano bidimensional, como por ejemplo el ratón, la palanca de control o los mandos de juego o *gamepads*. Se incluyen también en este módulo referencias a pantallas táctiles y multitáctiles.

El tercer módulo, "**Análisis de audio**", se centra en toda una serie de conceptos teóricos y de recursos técnicos que nos permitirán afrontar el diseño de interacciones basadas en el reconocimiento de características sonoras como la amplitud o la frecuencia.

El cuarto módulo, "**Dispositivos electrónicos**", es un compendio de conceptos teóricos relacionados con el prototipado de dispositivos electrónicos interactivos. Tomando la plataforma Arduino como punto de partida, se revisan una serie de componentes electrónicos (sensores y actuadores) imprescindibles de cara al diseño de interacciones.

El quinto módulo, "**Visión artificial**", ofrece la base conceptual y analiza las distintas técnicas más habituales en el contexto del diseño de interacciones basadas en la visión artificial.

El sexto módulo, "**Comunicación y tratamiento de datos**", a modo de anexo, ofrece algunos recursos referentes a redes y protocolos de comunicación, y varios procedimientos matemáticos habituales en el tratamiento de datos.

4) Objetivos

El objetivo fundamental de este material es ofrecer una base de conocimientos para que podáis desarrollar vuestros propios diseños interactivos atendiendo a principios como la funcionalidad, la usabilidad y la ergonomía. A pesar de que hemos pretendido hacer un análisis exhaustivo de cada una de las áreas de conocimiento de las que tratan los módulos, no hemos querido olvidar el aspecto inherentemente creativo del diseño de interactividad, y mediante reflexiones a lo largo de todos los módulos os invitamos a repensar los usos de cada uno de los dispositivos orientados a la interacción que se presentan en la documentación de la asignatura.

La meta final de la asignatura *Diseño de interacción* que cursáis es que, al final del proceso, seáis capaces de proyectar y hacer interfaces interactivas basadas en los diversos dispositivos y técnicas que proponemos: teclados, dispositivos apuntadores, análisis de audio, dispositivos electrónicos y visión por computadora.

5) Entornos de programación interactiva

Para afrontar la praxis del diseño interactivo, usaremos herramientas libres (Open Source) de programación interactiva, y también entornos de hardware libres (plataforma Arduino).

Estas herramientas permiten el prototipado rápido de diseños interactivos y en la mayoría de casos no requieren grandes conocimientos previos del área de la informática, la programación o la electrónica. Sin embargo, se trata de herramientas orientadas mayoritariamente al diseño de interfaces interactivas para el contexto artístico o experimental, puesto que no cumplen todos los estándares de fiabilidad y robustez que pide un contexto de fabricación industrial.

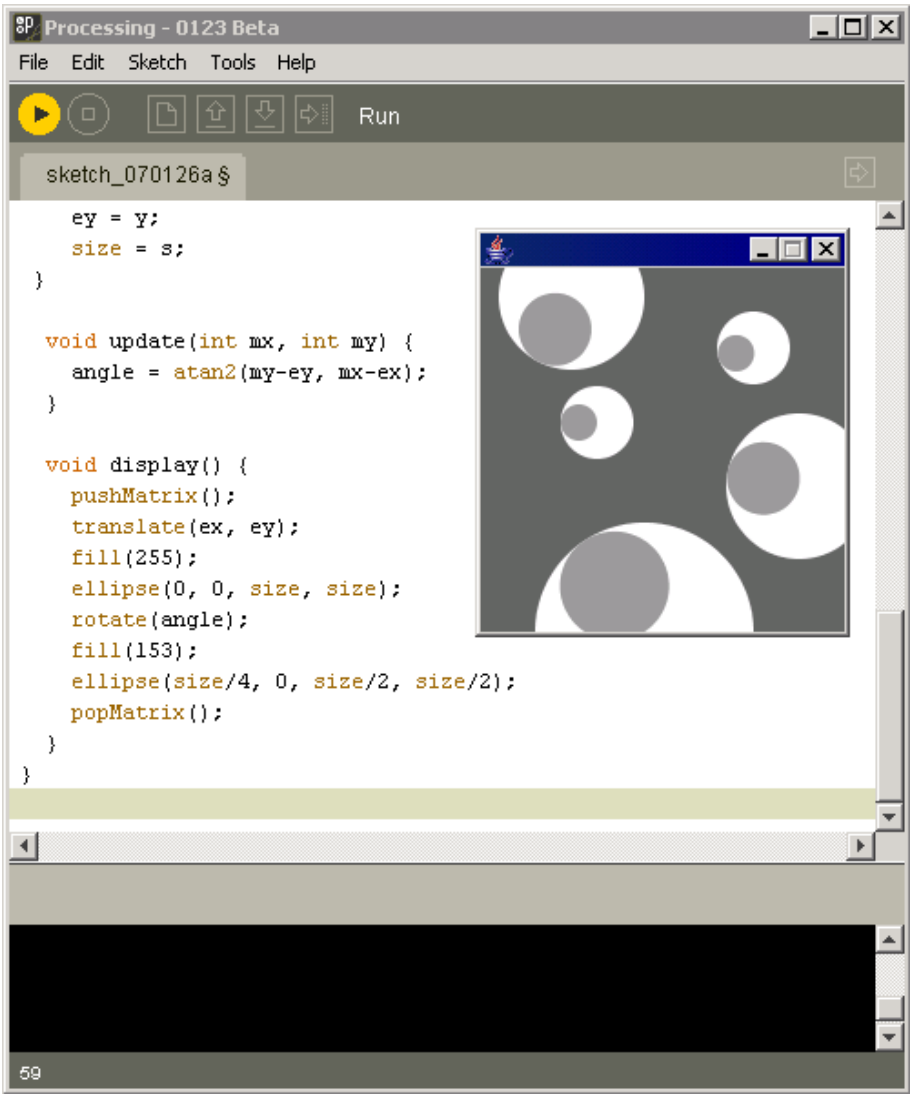
El diseño interactivo implica, en la mayoría de proyectos, la realización de una serie de prototipos de las interfaces para probar su usabilidad antes de que sean desarrollados de manera definitiva mediante sistemas profesionales.

Hay un gran número de herramientas de programación de diseño interactivo que permiten prototipar interfaces y sistemas complejos con relativa simplicidad. Vale la pena distinguir dos grandes familias de software en función del sistema de programación.

a) Herramientas de prototipado mediante código

Estas herramientas, entre las que podemos destacar Processing, openFrameworks o el entorno de desarrollo de software de Arduino, permiten prototipar las distintas funcionalidades del software con la programación mediante el código simplificado de lenguajes como Java, C o C++.

Habitualmente, se trata de sistemas que permiten acabados más robustos y fiables, a pesar de que requieren ciertos conocimientos previos de programación informática o una buena disposición a su aprendizaje.

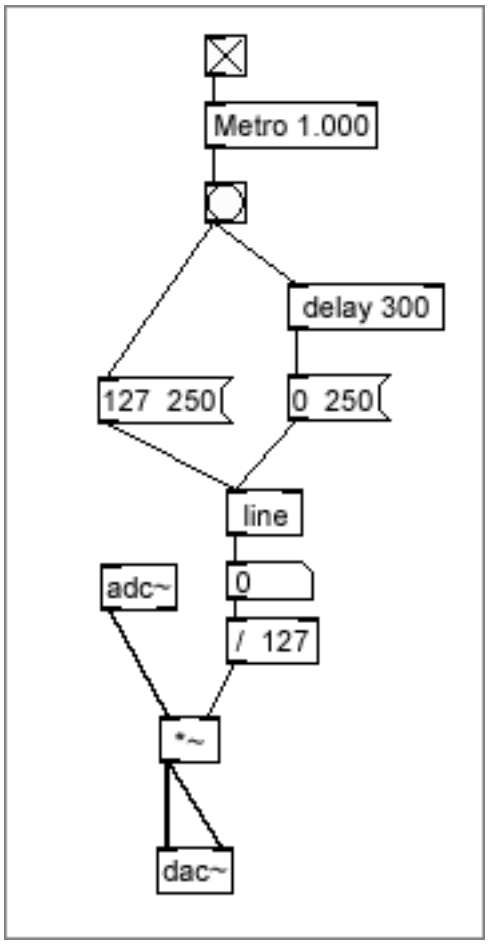


Fuente: processing.cc

b) Herramientas de prototipado mediante programación diagramática

El sistema diagramático, en un principio orientado a usuarios no iniciados en el mundo de la programación de software, se dirige a perfiles más cercanos a los músicos, los diseñadores o los artistas. Estas herramientas permiten el prototipado de sistemas interactivos mediante la conexión de objetos gráficos que llevan a cabo distintas operaciones, y ofrecen un tipo de programación muy similar a los diagramas de flujo.

Dentro de esta familia, vale la pena destacar entornos como Pure Data, Max MSP o Quartz Composer. Su uso es muy popular en entornos de creación artística, puesto que son muy efectivos, y están orientados al diseño de aplicaciones relacionadas con el arte sonoro, las videoinstalaciones y el arte mediático en general.



Fuente: flossmanuals (cc)

6) Conclusiones

Así pues, como hemos visto, en esta primera síntesis fundamentaremos la teoría y la práctica del diseño de interacción en un conjunto de materiales, teóricos y prácticos, basados en el uso de herramientas de software y hardware libre, atendiendo siempre a los principios de usabilidad, ergonomía y búsqueda creativa.

Esperamos que, con el objetivo de llevar a cabo vuestros propios proyectos de diseño interactivo, os sean de utilidad los materiales que a continuación os presentamos.

Contenidos

3. Diseñando interacciones
4. Más allá. Recursos específicos

Bibliografía

Bibliografía y recursos específicos

A pesar de que a lo largo de los módulos siguientes trataremos de hacer una introducción exhaustiva a los diferentes usos y técnicas relacionados con el diseño de interacción, recomendamos que uséis como material complementario los recursos en línea que especificamos a continuación:

Tom Igoe, recursos sobre computación física
http://tigoe.net/pcomp/index.shtml

Arduino Playground
http://www.arduino.cc/playground/

Tutoriales Processing
http://processing.org/learning/

Recursos en línea sobre conceptos relacionados con el diseño de interacción
http://www.interaction-design.org/

Revista gratuita en línea sobre diseño de interacción
http://www.revistafaz.org/

Recomendamos también la lectura de los libros y los manuales de referencia siguientes:

Chris Crawford (2002). *The art of interactive design*. No Starch Press

Programming Interactivity. A Designer's Guide to Processing, Arduino, and openFramework. O'Reilly Media. 15 de julio de 2009.

Teclados

Santiago Vilanova Ángeles

PID_00184753

Universitat Oberta de Catalunya

Índice

Introducción

Hay muchos tipos de teclados, desde el típico teclado de ordenador hasta la botonera del ascensor, pasando por los teclados de los teléfonos inteligentes o *smartphones* o las calculadoras. Todos estos dispositivos interactivos requieren una atención específica, y a lo largo de este módulo haremos un repaso de ellos, fijándonos en sus características específicas y en sus potencialidades a la hora de afrontar un diseño de interacción.

> Un teclado, en su definición más genérica, consiste en un conjunto de teclas que tienen como función servir de interfaz de interacción con un instrumento, aparato o mecanismo.

Hemos de entender los teclados alfanuméricos modernos como "metáforas" de la máquina de escribir, siendo conscientes de la flexibilidad inherente a los dispositivos digitales. Al contrario de lo que sucede en las antiguas máquinas de escribir, los teclados digitales nos permiten el remapeo de cada una de las teclas y, por lo tanto, tenemos todo un campo abierto de posibilidades creativas a la hora de asignar funcionalidades a las teclas o a las combinaciones entre ellas.

Además de los teclados alfanuméricos, en esta definición amplia del concepto *teclado*, hemos querido incluir los teclados musicales MIDI, de manera que haremos también un repaso rápido de este protocolo de comunicación, muy extendido en el mundo de la interactividad relacionada con la música y las artes visuales.

Así pues, hemos dividido este módulo en dos partes: la primera hará referencia a los teclados alfanuméricos, y en la segunda estudiaremos las características de los teclados y el protocolo MIDI.

Objetivos

1. Aprender de las diferentes características de los teclados más habituales.

2. Analizar sus sistemas de comunicación y la manera en que codifican los datos.

3. Orientar diseños de interacción basados en teclados que tengan en cuenta factores como la usabilidad o el mapeado.

1. Teclados alfanuméricos

1.1. Conceptos teóricos

Hoy en día estamos rodeados de centenares de teclados alfanuméricos, desde los de los teléfonos móviles o las calculadoras, hasta los teclados **QWERTY** de los ordenadores. Cada uno de ellos presenta funciones muy definidas, como por ejemplo la marcación de números de teléfono, la introducción de dígitos en un cajero automático o la escritura en un procesador de textos para programar.

En el teclado más común para nosotros, el de los ordenadores, la disposición de las teclas se remonta a las primeras máquinas de escribir, que eran completamente mecánicas. Al pulsar una letra en el teclado, se movía un pequeño martillo mecánico que golpeaba el papel a través de una cinta impregnada de tinta. Sobre la distribución de los caracteres en el teclado, surgieron dos variantes principales: la francesa **AZERTY** y la alemana **QWERTZ**. Ambas respondían a cambios en la disposición, según las teclas más frecuentemente usadas en cada idioma. Para evitar el encallamiento de teclas en las antiguas máquinas de escribir, se eligió una disposición que permitiera que las teclas más frecuentes se encontraran tan alejadas como fuera posible, lo que provocó también una gran alternancia entre el uso de ambas manos.

A los teclados en la versión para el idioma español, los más habituales para nosotros, además de la *ñ*, se les añadieron los caracteres de acento abierto (`), acento cerrado (´), diéresis (¨) y acento circunflejo (^), aparte de la cedilla (*ç*), aunque estos caracteres se usan más en francés, portugués o catalán.

Actualmente, los teclados de ordenador convencional, que normalmente conectamos por **USB** al ordenador, disponen de un **microcontrolador interno** que se encarga de traducir cada pulsación de teclado a un mensaje numérico codificado según el **protocolo ASCII**.

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

Fuente: Wikipedia (cc)

Para cada pulsación o liberación de una tecla, el microcontrolador envía un código identificativo que se llama **código de escaneo** o ***scan code***. Con el fin de permitir la pulsación simultánea de varias teclas, el teclado genera un código diferente cuando una tecla es pulsada y cuando es liberada.

1.2. Las herramientas

1.2.1. QWERTY

En cuanto a estructura, los teclados clásicos de ordenador están divididos en **cuatro bloques diferenciados**, cada uno de los cuales presenta funcionalidades definidas. Es importante que tengamos en cuenta esta estructuración a la hora de plantear un diseño de interacción con teclado, tratando de seguir las convenciones para facilitar la intuitividad del sistema y su proximidad a los usuarios o, al contrario, saltándonos completamente estas convenciones de modo justificado. En general, podríamos estructurar un teclado QWERTY en las siguientes partes:

1) Bloque de funciones. Va desde la tecla F1 hasta la tecla F12, y está estructurado en tres bloques de cuatro teclas: de F1 a F4, de F5 a F8, y de F9 a F12. Estas teclas funcionan de acuerdo con el programa que esté abierto.

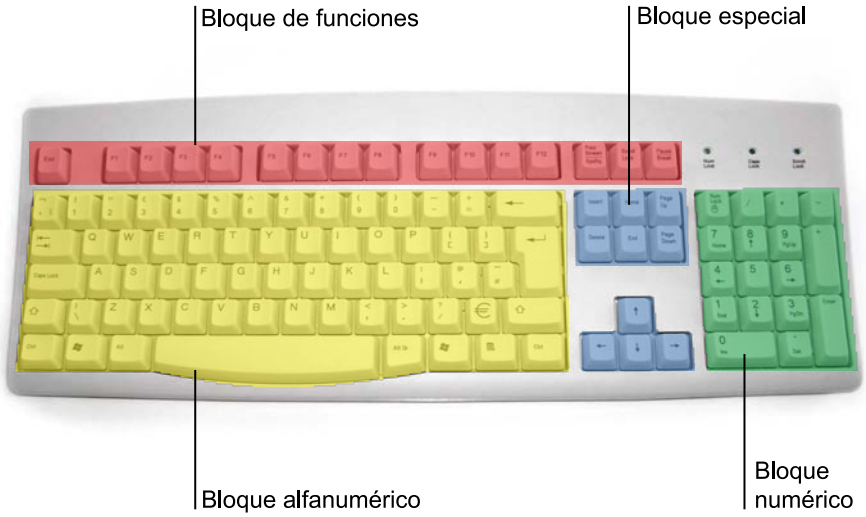
Ejemplo

Por ejemplo, en muchos programas, pulsando la tecla F1 se accede a la ayuda asociada a ese programa.

2) Bloque alfanumérico. Está situado en la parte inferior del bloque de funciones, y contiene los números arábigos del 1 al 0 y el alfabeto, organizado como en una máquina de escribir, aparte de algunas teclas especiales.

3) Bloque especial. Se halla a la derecha del bloque alfanumérico y contiene algunas teclas especiales, como Imp Pant, Bloq, teclas de desplazamiento, pausa, inicio, fin, insertar, suprimir, RePág y AvPág, y las flechas direccionales, que permiten mover el punto de inserción en las cuatro direcciones.

4) Bloque numérico. Se encuentra a la derecha del bloque especial, se activa pulsando la tecla Bloq Núm y contiene los números arábigos organizados como en una calculadora para facilitar la digitación de cifras. Además, recoge los signos de las cuatro operaciones básicas: suma (+), resta (–), multiplicación (*) y división (/); también contiene una tecla Retorno o Enter.



1.2.2. Teclados de teléfonos móviles (*smartphones*)

La mayoría de teléfonos móviles de hoy en día disponen de teclados alfanuméricos para marcar números de teléfono, escribir SMS o acceder a las funcionalidades de los menús internos. A pesar de que existen tantos tipos de teclados como teléfonos móviles (desde teclados físicos hasta teclados táctiles integrados en las pantallas gráficas o *graphic displays*), la mayoría de teclados utilizan un sistema que permite que cada tecla pueda alojar diferentes caracteres alfanuméricos o funciones, normalmente con el análisis del tiempo de pulsación o la doble/triple pulsación.

Ejemplo

Por ejemplo, si mantenemos presionado un botón durante más tiempo, alternamos los caracteres numéricos y los alfabéticos.

Hay que decir que en ciertos países, como Estados Unidos, la telefonía fija también dispone de este sistema que permite escribir textos a partir de la repetición de las pulsaciones de teclado.

Fuente: Wikipedia (cc)

1.2.3. Teclados numéricos

Los teclados numéricos sencillos de las máquinas calculadoras o de las teclas de cursor o *keypads* pueden ser útiles en determinadas situaciones. Muchos sistemas de alarma, intercomunicadores o artículos electrónicos de consumo incorporan estos teclados sencillos para ofrecer una vía de interacción con los usuarios.

Vale la pena remarcar que la mayoría de estos dispositivos numéricos son sencillamente conectables a Arduino y pueden ser útiles en la confección de objetos interactivos.

Fuente: Wikipedia (cc)

1.2.4. Botoneras

Más extensamente, podríamos considerar mandos a distancia, botoneras de ascensor o intercomunicadores de bloques de pisos como casos muy específicos de teclado.

La popularización de los sistemas DIY

La popularización de los sistemas DIY (*do it yourself*, 'hágalo usted mismo') en el campo de la electrónica ha llevado a un gran número de usuarios al diseño de sus propios artefactos. Entre los compositores de música electrónica, son muy populares estos sistemas personalizados, y muchos músicos construyen sus propios instrumentos adaptando los diseños de teclados a sus necesidades interpretativas.

Fuente: Wikipedia (cc)

1.3. Diseñando interacciones

1.3.1. Tecla a tecla

La interacción más básica que podemos diseñar usando un teclado es la **detección de la tecla pulsada**.

Como hemos visto, cada tecla o combinación de teclas tiene asociado un valor ASCII diferente que es enviado cada vez que se pulsa una tecla.

Dependiendo del entorno de diseño interactivo en el que trabajemos, también podemos recibir datos cuando la tecla se haya dejado de pulsar; por lo tanto, cada acción de pulsación de una tecla nos da tres parámetros:

- Inicio de la pulsación
- Valor ASCII de la tecla
- Fin de la pulsación

A partir de estos datos, se pueden generar diversas interacciones, asignando distintas acciones a las pulsaciones / finales de pulsación de cada tecla.

1.3.2. Tiempo de pulsación

Otro método sencillo de diseño interactivo basado en el teclado es el **cálculo del tiempo de pulsación de las teclas**, que nos puede ayudar a distinguir entre toques cortos y toques largos de las mismas.

1.3.3. Combinaciones de teclas

Asimismo, la **pulsación combinada de distintas teclas** puede ser un recurso más a la hora de plantear una interacción mediante el teclado. De hecho, es común encontrar este tipo de interacción basada en combinaciones de teclas en todos los sistemas operativos.

Ejemplo

Como ejemplo más representativo, podríamos mencionar la funcionalidad de "copiar y pegar", que se hace normalmente con la combinación de Ctrl + C y Ctrl + V.

1.3.4. Palabras

Otro método interesante de interactividad con los usuarios puede ser la introducción de palabras o cadenas de caracteres.

Mediante la confección previa de una base de datos con palabras clave (por ejemplo, izquierda, derecha, grande, pequeño...), podemos diseñar un sistema interactivo que "escuche" las peticiones textuales de los usuarios y actúe en consecuencia.

Un diseño de este tipo requiere la construcción de un espacio de memoria en el que almacenar los caracteres y la comparación de los caracteres introducidos en la base de datos.

Frecuentemente se utiliza la tecla Enter como indicador de que se ha acabado de introducir la secuencia de caracteres pertinentes.

2. Teclados MIDI

2.1. Conceptos teóricos

MIDI

Siglas de *musical instrument digital interface*, 'lenguaje de control y comunicación entre instrumentos musicales digitales'.

MIDI se creó en los años ochenta, en plena eclosión de la microelectrónica digital, para centralizar y secuenciar diversos aparatos generadores de sonidos en un controlador principal. Si queremos secuenciar una canción que conste de ocho instrumentos diferentes, necesitaremos un secuenciador que se entienda con los ocho instrumentos por medio de un lenguaje estándar. Esto era MIDI originariamente. Actualmente, la mayoría de sistemas de informática musical incorporan puertos MIDI (con conectores DIN de cinco pines) para la comunicación con sintetizadores externos o el control mediante controladores. Hoy en día hay muchos aparatos que no son estrictamente musicales y que permiten el control MIDI: mesas de luces, controladores de servomotores, robots, etc. MIDI no transmite datos de audio, solo datos de control, organizados tal como sigue:

- **Notas**. Es la información que se genera cuando presionamos una tecla del piano. El tipo de mensaje puede ser Note ON o Note OFF, y el valor de estas notas varía en función de la nota que toquemos. Por ejemplo, C5 (do de la quinta octava) es la nota 60, C # 5 = 61, D5 = 62, D # 5 = 63... Hay ciento veintiocho notas diferentes (0-127).

- **Controladores continuos (CC)**. Es la información que se genera cuando movemos un potenciómetro o *fader*. A diferencia de los datos de nota, que son binarios (puesta en marcha / apagado), estos datos son continuos y trabajan en un rango de 0 a 127. Esto quiere decir que si tenemos el potenciómetro al mínimo, enviará un valor 0; si lo tenemos a la mitad, enviará un valor 64, y si lo tenemos al máximo, enviará un valor 127.

Además de estos dos tipos de mensaje, hay unos cuantos más (SysEx, Program Change, NRPN, Pitch Bend), con unas propiedades diferentes y que, de momento, no nos serán útiles.

Todos estos datos corren por un canal MIDI (1-16).

2.2. Las herramientas

2.2.1. Teclado o controlador MIDI

Para prototipar nuestros primeros diseños interactivos basados en MIDI, recomendamos usar un teclado MIDI (que suele reproducir la estructura de un piano tradicional, en pequeño tamaño y con menos octavas) o un controlador (constituido normalmente por una caja con botones y potenciómetros). La configuración o *setup* más común es aquella en la que se utiliza el teclado o el controlador MIDI como dispositivo de mando de los sistemas de software o electrónicos.

Actualmente, la mayoría de teclados y controladores MIDI pueden conectarse a los ordenadores mediante USB, en vez de hacerlo mediante el cable MIDI, lo que nos ahorra el uso de una interfaz MIDI-USB.

Fuente: audiomidicontroller.com

2.2.2. Interfaz MIDI

Pieza de hardware que actúa de mediadora entre un sistema de secuenciación MIDI (normalmente un PC) y los instrumentos secuenciados, o entre un controlador y los sintetizadores, muestreadores (*samplers*), etc.

Normalmente, dispone de varias entradas y salidas.

La mayoría de tarjetas de audio profesional incorporan puertos de entrada y salida MIDI, y actúan como interfaces MIDI.

Fuente: foromusicos.es

2.2.3. Controladores o *drivers* MIDI virtuales

"Tubos" de comunicación interna dentro de un mismo ordenador.

Son útiles para comunicar dos programas MIDI, porque los configuran a uno como maestro y al otro como esclavo. La compañía MIDI-OX desarrolla controladores virtuales (MIDI Yoke) y gestores (MIDI-OX) que permiten comunicación interna y múltiples transformaciones en los datos de control.

2.2.4. MIDI por Ethernet

Hasta hace muy poco, el modo de transmitir datos MIDI era por cable MIDI. Con la llegada de las aplicaciones de MIDI por Ethernet (UDP> TCP> UDP), se ha abierto un gran número de posibilidades impensables hasta ahora. Podemos enviar datos de control MIDI en tiempo real a otra parte de la red local, y crear una matriz de ordenadores interconectados sin necesidad de disponer de interfaces MIDI tan caras. Y todo ello lo podemos hacer a través del cable de red y direccionadores o *routers*. Además, los datos TCP viajan mucho más rápido que los datos MIDI, de modo que se recortan muchísimo las latencias[1] (inferiores a 1 ms).

(1)La latencia es el tiempo de espera (normalmente definido en milésimas de segundo) desde que se envía un dato hasta que es recibido por el receptor.

2.2.5. Software de monitorización

Midi-OX (PC), MIDI Monitor (Mac)

Una herramienta útil a la hora de visualizar los datos MIDI que enviamos o recibimos son los monitores MIDI. Existen varios softwares que cumplen esta función, la mayoría de los cuales son gratuitos bajo licencia *freeware* o *shareware*.

2.2.6. Hardware compatible con MIDI

Actualmente, podemos encontrar en el mercado diferentes dispositivos compatibles con MIDI. Estos se pueden manejar por medio de mensajes MIDI de modo remoto. Es el caso de un gran número de sintetizadores de sonido, mesas de control de iluminación espectacular o mezcladores de vídeo, entre muchos otros.

Fuente: Wikipedia (cc)

2.2.7. Software compatible con MIDI

Asimismo, hay una gran cantidad de software que permite la automatización mediante mensajes MIDI. La mayoría de softwares orientados a la creación o la manipulación musical ofrecen esta opción. Sin embargo, la mayoría de softwares de manipulación de vídeo en tiempo real, softwares de control de iluminación y casi todos los entornos de programación interactiva incluyen la posibilidad de entrada de datos MIDI.

Fuente: Wikipedia (cc)

2.3. Diseñando interacciones

2.3.1. Mapeo

Entendemos por *mapeo* la asignación de teclas o valores de los potenciómetros a parámetros concretos de un sistema de software.

Pongamos por caso que queremos asignar el valor del potenciómetro 1 de nuestro controlador MIDI a la velocidad de reproducción de un vídeo o a la medida de un círculo. Al proceso de asignación del valor del uno al otro lo denominaremos *mapeo*.

Para poder mapar un parámetro, necesitaremos tener habilitada en el software receptor la entrada MIDI y estar seguros de que está recibiendo los datos correctamente. Para ello, podemos usar algún software de monitorización de datos MIDI (MIDI Monitor, Midi-Ox).

Una vez habilitada la entrada de datos MIDI en el software receptor, solo se trata de asignar el tipo (Nota o CC) y el número de mensaje MIDI (0-127) al parámetro deseado.

Dependiendo del entorno en el que estemos trabajando, este procedimiento se hará de diferentes maneras.

Muchos entornos comerciales de composición y directo de audio, así como softwares de manipulación de vídeo en tiempo real, incorporan una funcionalidad llamada MIDI Learn que facilita mucho este proceso de mapeo de parámetros. En la mayoría de casos, solo se trata de activar el sistema de aprendizaje (MIDI Learn), seleccionar el parámetro receptor y tocar la tecla o el potenciómetro que queramos asignar a ese parámetro. Así podemos reasignar muy rápidamente las funciones del teclado o dispositivo físico.

2.3.2. Fórmulas y transformación de datos entrantes

En determinadas situaciones, nos puede convenir transformar los datos de control para invertirlos, asignarles determinados valores o convertirlos en curvas exponenciales; por ejemplo, mediante el uso de distintas fórmulas y algoritmos matemáticos, podemos transformar los datos entrantes según nuestras necesidades.

3. Más allá. Recursos y bibliografía específica

3.1. Secuenciadores y partituras: líneas de tiempo (*timelines*)

Más allá del uso directo de controladores y teclados, el estándar MIDI abre un abanico de posibilidades que es muy interesante explorar.

Una de estas posibilidades es la de la escritura de partituras de acontecimientos o *events*. Como el lenguaje MIDI se emplea mayoritariamente en plataformas de composición musical basadas en líneas de tiempo, disponemos de toda una serie de herramientas, pensadas para la escritura de acontecimientos en el tiempo, que podemos utilizar para secuenciar cualquier tipo de acontecimiento con una dinámica temporal.

Este tipo de escritura es muy útil en contextos como el teatro, la automatización de proyectos museísticos o el diseño de artefactos electrónicos (escultura cinética, autómatas musicales...).

Si os interesa seguir explorando las posibilidades de este tipo de escritura temporal, os recomendamos que echéis una ojeada a entornos de composición musical asistida por ordenador, como Ableton Live, Cubase o IanniX.

3.2. Bibliografía

Sergi Jordà Puig (1997). *Audio digital y MIDI*. Madrid: Anaya Multimedia.

Dispositivos apuntadores

Santiago Vilanova Ángeles

PID_00184752

UOC
Universitat Oberta de Catalunya

Índice

Introducción

En determinadas situaciones, necesitamos diseñar un tipo de interacción que nos aporte información sobre una posición concreta en un plano bidimensional. Por regla general, cuando hablamos de la interacción de los usuarios con una **GUI** (interfaz gráfica de usuario, en inglés *graphical user interface*), requerimos este tipo de información de posicionamiento del cursor en la pantalla. El mismo caso es aplicable a numerosos videojuegos o sistemas de navegación que también trabajan a partir de los datos de coordenadas bidimensionales.

Además, la mayoría de dispositivos de posicionamiento XY también incorporan algún tipo de intercambiador digital, como los botones derecho e izquierdo de un ratón de PC o los de función de los mandos de juego (*gamepads*).

Objetivos

1. Profundizar en el uso de dispositivos apuntadores como integrantes de un diseño de interacción.

2. Analizar las características de los diferentes dispositivos apuntadores existentes.

3. Analizar algunos de los algoritmos posibles de gestión de los datos de posicionamiento.

4. Definir estrategias de cara a un diseño interactivo usable y eficiente.

1. Conceptos teóricos

Cuando hablamos de un dispositivo apuntador, nos referimos a un **periférico** que tiene la función de generar datos de posicionamiento bidimensional o tridimensional.

En este módulo nos centraremos en los **dispositivos de posicionamiento bidimensional**.

Más allá de las características técnicas que permiten el funcionamiento de estos periféricos, vale la pena remarcar que para afrontar un diseño de interacciones basado en la posición XY de un dispositivo apuntador, es muy útil tener en todo momento una buena referencia sobre matemáticas orientadas a la geometría plana: cálculo vectorial, trigonometría, geometría...

Con esta idea en mente, hemos recopilado una serie de fórmulas y algoritmos útiles para el procesamiento de datos de posicionamiento en el espacio que podéis consultar en uno de los materiales anexos a este documento.

De hecho, como veréis más adelante, se trata de un recurso transversal que os será útil en otros módulos de este material docente.

2. Las herramientas

2.1. Ratón

El ratón o *mouse* es un **dispositivo apuntador** que se usa para facilitar la interacción con un entorno gráfico en un computador.

Por lo general, está fabricado en plástico y se utiliza con una de las manos. Detecta el movimiento relativo en dos dimensiones por la superficie plana en la que se desplaza y se refleja habitualmente como un puntero o flecha en el monitor.

Fuente: Wikipedia (cc)

2.2. Palanca de mando

Una palanca de mando o *joystick* (del inglés *joy*, 'alegría', y *stick*, 'palo') es un **dispositivo de control** de dos o tres ejes que se usa desde en una computadora o una videoconsola hasta en un transbordador espacial o en los aviones de caza, pasando por las grúas.

Solemos distinguir entre **palancas digitales** (que leen cuatro interruptores *on/off* más sus combinaciones y los botones de acción) y **analógicas** (que usan potenciómetros para leer continuamente el estado de cada eje); estas últimas son más precisas.

Fuente: Wikipedia (cc)

2.3. Mando de juego

Un mando de juego o *gamepad* es un **dispositivo de entrada** que se emplea para interactuar con un videojuego y que permite moverse e interactuar con los elementos del juego para hacer las diversas acciones necesarias.

Un mando de juego se caracteriza porque es un tablero con una o varias palancas o cruces, que pueden ser analógicas o digitales, diseñadas para usarse con el dedo pulgar, y una serie de botones, generalmente colocados en el lado derecho, cada uno de los cuales tiene una función específica.

Fuente: Wikipedia (cc)

2.4. Pantalla táctil

Una pantalla táctil es una pantalla que permite la entrada de datos en el dispositivo mediante un toque directo sobre su superficie.

Fuente: Wikipedia (cc)

Además, gracias a que la pantalla actúa como periférico de salida, muestra los resultados introducidos previamente. Así pues, la pantalla táctil puede actuar como **periférico de entrada** y como **periférico de salida** de datos.

Ejemplo

Han llegado a ser muy comunes en terminales de venta al público, en cajeros automáticos y en asistentes digitales personales o PDA.

2.5. Pantallas multitáctiles

En los últimos años, se han introducido en el mercado diversos dispositivos que incorporan funcionalidades multitáctiles. Productos como el iPad, el iPod touch o el iPhone de Apple han hecho un uso extensivo de esta nueva tecnología, una tecnología que se ha ido imponiendo gracias a su inmediatez y a un diseño interactivo intuitivo y sencillo.

Desde el punto de vista analítico, un **sistema de interacción multitáctil** consiste en un sistema de posicionamiento de múltiples cursores dinámicos (los dedos).

Cuando colocamos los dedos sobre una de estas superficies (que habitualmente se integran con la propia pantalla del dispositivo), un sistema de captación detecta su posición en la pantalla o *display*. Este sistema de captación se puede basar en técnicas de visión artificial o en sistemas de sensores capacitivos.

Mediante el uso de diversos algoritmos matemáticos, se han diseñado interacciones basadas en la distancia entre dos dedos (para cambiar la medida de un objeto o el nivel de zoom de la pantalla), el ángulo que forman respecto al eje *x* (para rotar objetos), el número de dedos situados sobre la pantalla, la velocidad del movimiento, etc.

Fuente: Wikipedia (cc)

2.6. GUI

La **interfaz gráfica de usuario**, conocida también como GUI (del inglés *graphical user interface*), es un programa informático que actúa como interfaz de usuario utilizando un conjunto de imágenes y objetos gráficos para representar la información y las acciones disponibles en la interfaz.

Su principal uso consiste en proporcionar un entorno visual sencillo que permite la comunicación con el sistema operativo de una máquina u ordenador.

Surge como evolución de las interfaces de consola de mandos que se empleaban para hacer trabajar a los primeros sistemas operativos y es una pieza fundamental en un entorno gráfico.

Ejemplo

Como ejemplos de interfaces gráficas de usuario, hay que mencionar los entornos de escritorio Mac OS, Aqua, Windows y X-Window, de GNU/Linux.

En el contexto del proceso de interacción persona-ordenador, la interfaz gráfica de usuario es el artefacto tecnológico de un sistema interactivo que posibilita, mediante el uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.

Fuente: Wikipedia (cc)

3. Diseñando interacciones

3.1. Interacción con GUI

1) Posición del cursor

Cuando extraemos los datos de posición del ratón en pantalla, obtenemos sus coordenadas en la matriz de píxeles de la pantalla gráfica (*graphic display*). Estas dos coordenadas *x* e *y* obtendrán valores dentro del rango determinado por la resolución de nuestra pantalla. Si nuestro ordenador está configurado para hacer una pantalla de 1.024 × 768 píxeles, los valores de las coordenadas de posición del cursor estarán dentro del rango 0-1.024 para la coordenada *x* y dentro del rango 0-768 para la coordenada *y*. Hay que tener en cuenta que generalmente la posición (0,0) se encuentra en la esquina superior izquierda de la pantalla y que, por lo tanto, el valor de *x* se incrementa a medida que desplazamos el cursor hacia la derecha y que el valor de *y* se incrementa a medida que lo desplazamos hacia abajo.

A partir de estos datos de posición del cursor, se pueden generar una gran cantidad de interacciones basadas en cálculos relativos a la posición o a su evolución en el tiempo:

- distancia a un elemento gráfico o contacto con este,
- ángulo respecto a un eje y
- velocidad del movimiento.

Observación

Tened siempre en cuenta que los datos que nos ofrece el ratón son relativos respecto a su ubicación física. Es decir, la posición física del ratón sobre la mesa no representa exactamente la posición del cursor en la pantalla. Tanto es así que, cuando usamos el ratón, en cualquier momento podemos levantarlo y llevarlo a una parte de la mesa que nos proporcione más comodidad para seguir orientando el cursor.

2) Posición + clic

Aparte de la posición *xy*, la mayoría de dispositivos apuntadores nos ofrecen otra entrada de información basada en la **pulsación de conmutadores** o *switchs* digitales. Estos conmutadores son una fuente inestimable de recursos a la hora de diseñar interacciones para uno de estos dispositivos, de manera que se pueden generar una gran cantidad de interacciones basadas en los clics:

- clic simple,
- doble clic,
- triple clic, cuádruple clic, etc.,
- tiempo entre clics (velocidad de pulsación),

- alternancia entre clic derecho y clic izquierdo y
- simultaneidad de clics.

Es muy frecuente el uso combinado de los datos de posición y el clic.

Ejemplo

Un claro ejemplo de ello es el botón virtual en una GUI, cuyo modo habitual de funcionamiento se basa en colocar el cursor encima del botón y hacer clic sobre este.

3) Arrastrar y soltar (*drag and drop*)

Un caso remarcable de uso combinado de posicionamiento y clic es la función de arrastrar y soltar. Este algoritmo se basa en una secuencia determinada de acciones:

Arrastrar y soltar

El procedimiento de arrastrar y soltar es muy habitual en la mayoría de sistemas operativos actuales, puesto que resulta una metáfora extremadamente próxima al modo de hacer humano.

- Presionar y mantener presionado el botón del ratón u otro dispositivo apuntador para coger el objeto.
- Arrastrar el objeto/cursor/dispositivo apuntador a la ubicación deseada.
- Soltar el objeto soltando el botón.

4) Combinaciones con el teclado

Otro sistema muy frecuente de interacción basado en los ratones es su combinación con las teclas del teclado.

Es habitual el uso combinado de las teclas Mayús, Alt o Ctrl con el clic del ratón o el procedimiento de arrastrar y soltar para acceder a funciones concretas de diversos softwares.

Estas posibilidades de combinación entre ambos dispositivos, **teclado** y **ratón**, ofrecen grandes ventajas de usabilidad en el software más complejo, con acceso rápido a funciones que, de otro modo, serían menos accesibles.

Reflexión

Estamos acostumbrados a un diseño muy concreto de la interactividad mediante ratón y teclado, pero, como sabemos, estos diseños son flexibles y podemos proyectar nuevas maneras de usar estos dispositivos.

Proponemos un ejercicio/reflexión que nos puede ayudar a repensar estas funcionalidades: ¿cómo deberíamos diseñar un ratón para que fuera útil para personas invidentes?

3.2. Interacción con otros dispositivos

Hasta ahora, hemos hecho un repaso de los diferentes métodos de diseño interactivo relacionados con las interfaces gráficas. Sin embargo, el uso de los dispositivos de posicionamiento no se limita solo a los entornos gráficos, puesto que es útil en una gran cantidad de aplicaciones industriales, de automoción e incluso aeronáuticas.

Reflexión

A lo largo de esta asignatura, aprenderemos a diseñar una interacción que relacione un sistema de actuadores (motores, LED, etc.) controlados mediante Arduino con un sistema interactivo formado por un ratón o un mando de juego.

Es habitual poner en relación el sistema de dirección de un vehículo motorizado o de una aeronave con un procedimiento interactivo basado en sistemas de posicionamiento como el mando de juego.

Análisis de audio

Santiago Vilanova Ángeles

PID_00184754

Universitat Oberta de Catalunya

Índice

Introducción

En este módulo haremos referencia a todo aquello relacionado con el diseño de interacción mediante el sonido. Gracias a dispositivos de captura de audio como los micrófonos, hoy en día incorporados a casi todos los aparatos electrónicos de comunicación y ocio (teléfonos, ordenadores, videoconsolas), podemos diseñar interacciones persona-ordenador sencillas y efectivas, con el canal de nuestra propia voz como interactuador o mediante el análisis de cualquier otra fuente sonora (instrumentos musicales, sonido de ambiente, etc.).

Pensamos, por ejemplo, en sistemas de iluminación domótica que permiten la activación de la luz mediante dos palmadas seguidas, en sistemas de visualización musical que posibilitan la generación automática de imágenes a partir del análisis frecuencial de la música, o en sistemas de análisis del ruido de ambiente para detectar el nivel de contaminación acústica.

Objetivos

1. Repasar conceptos fundamentales de la física ondulatoria, como la frecuencia y la amplitud.

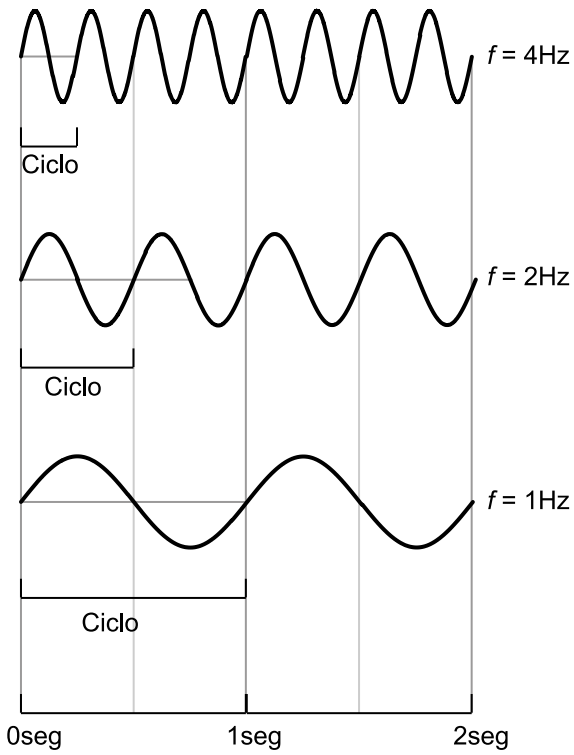
2. Sedimentar los conceptos y características fundamentales del audio digital.

3. Aprender los algoritmos más importantes relacionados con el análisis de audio.

4. Estudiar las herramientas existentes para la captura del sonido y su posterior digitalización.

1. Conceptos teóricos

1.1. Frecuencia

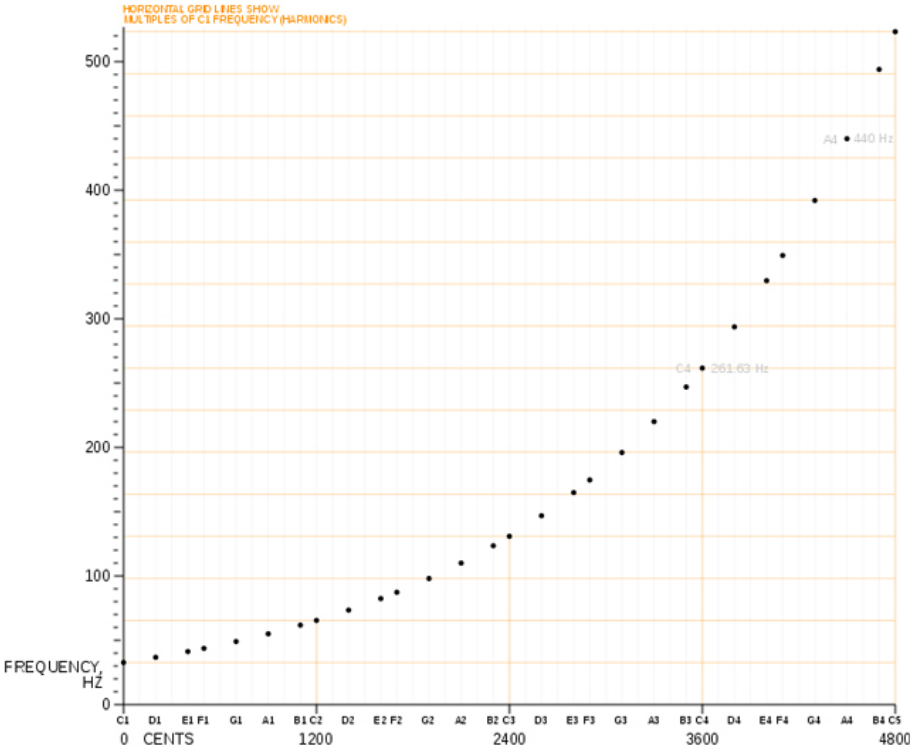


La frecuencia del sonido hace referencia a la cantidad de veces que vibra el aire que transmite ese sonido en un segundo. La unidad de medida de la frecuencia es el **hercio** (Hz). La medición de la onda puede empezar en cualquiera de sus puntos.

Para que el ser humano pueda oír un determinado sonido, su frecuencia debe estar comprendida entre los 20 y los 20.000 Hz. La mayoría de sistemas de captación convencional no superan este rango de frecuencias.

Los sonidos que percibimos como **agudos** tienen una frecuencia mayor que los sonidos **graves**. La frecuencia de los tonos agudos oscila entre los 2.000 y los 8.000 Hz, mientras que la de los graves varía entre los 20 y los 250 Hz. Los tonos **medios** tienen una frecuencia de oscilación de entre 500 y 1.000 Hz.

1.2. Frecuencia y tonos musicales



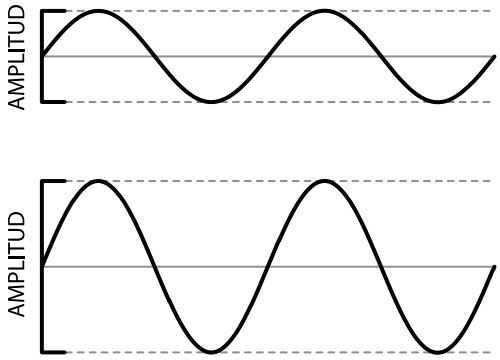
Fuente: Wikipedia

En la notación musical occidental, se toma como referencia la frecuencia de 440 Hz, que es asignada a la nota A4 (la de la 4.ª octava). Entre una nota y su octava superior, que es del doble de frecuencia, hay doce subdivisiones o semitonos.

La ratio numérica entre las frecuencias de dos semitonos sucesivos es exactamente la raíz duodécima de 2 (un factor de aproximadamente 1,05946).

Este sistema de subdivisión de la escala musical en doce tonos se denomina ***temperamento igual*** y, a pesar de que es el sistema de afinación más extendido en la música occidental, no es el único existente, de modo que pueden construirse sistemas de división infinitos de una escala musical, con distintas ratios de frecuencia entre notas correlativas y distinto número de subdivisiones.

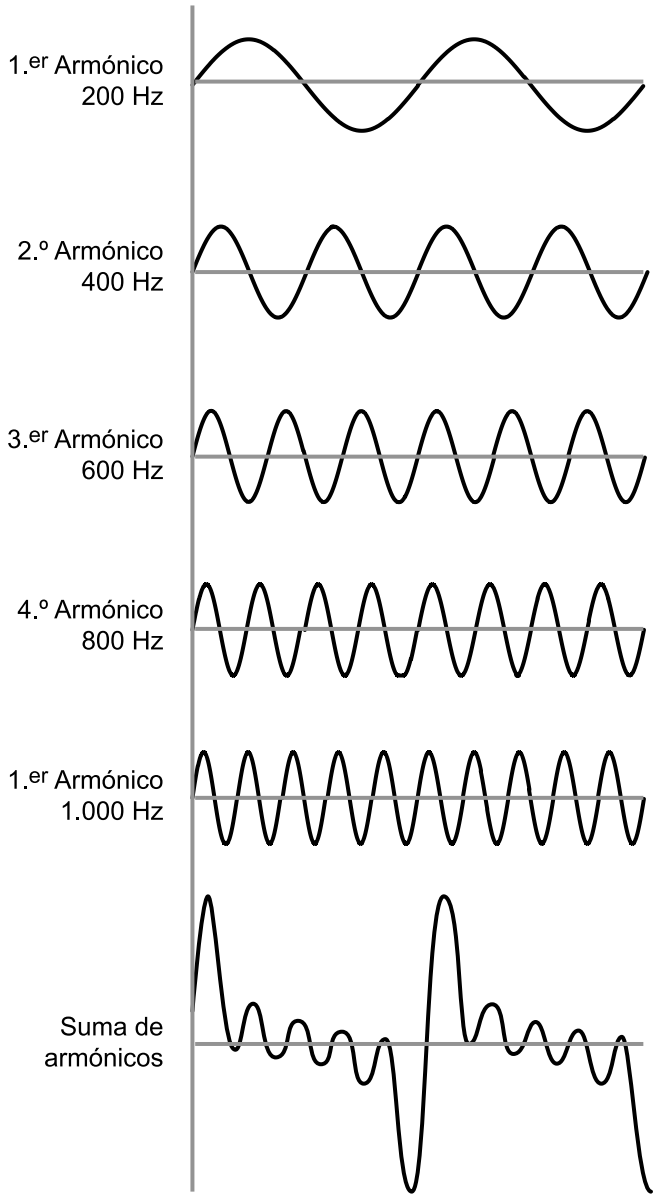
1.3. Amplitud



La amplitud de una onda sonora es el grado de movimiento de las moléculas de aire en la onda, y corresponde a la intensidad de la expansión y la compresión atmosférica que la acompañan. Cuanto más elevada es la amplitud de la onda, más intensamente golpean las moléculas el tímpano y más **fuerte** es el sonido percibido.

La amplitud de una onda sonora puede expresarse en unidades absolutas midiendo la distancia de desplazamiento de las moléculas del aire o la diferencia de presión atmosférica entre la compresión y la expansión. Habitualmente nos referiremos a la amplitud de una onda sonora empleando su medida en decibelios (dB), aunque también puede venir expresada en pascales y milibares.

1.4. Señales complejas: armónicos



La mayoría de osciladores, incluyendo la voz humana, un violín o una estrella cefeida, son más o menos periódicos y están formados por armónicos.

La mayoría de osciladores pasivos, como las cuerdas de una guitarra, la membrana de un tambor o una campana, oscilan de modo natural a diferentes frecuencias simultáneas, conocidas como parciales. Cuando el oscilador es largo y fino, como una cuerda de guitarra o la columna de aire en una trompeta, la mayoría de los parciales son múltiplos enteros de la frecuencia fundamental; estos parciales se denominan **armónicos**.

Los armónicos son las diferentes frecuencias presentes en el movimiento de un oscilador, múltiplos íntegros de la frecuencia fundamental.

Los parciales con frecuencias que no son múltiplos enteros de la frecuencia fundamental se denominan **inarmónicos**, y generalmente son percibidos como desagradables. El sonido de las campanas, por ejemplo, contiene muchos inarmónicos.

Percepción del oído humano

El oído humano no percibe los armónicos como notas separadas. De hecho, una nota musical formada por muchas frecuencias armónicamente relacionadas se percibe como un sonido único, cuya calidad o timbre es el resultado de la amplitud relativa de cada una de las frecuencias armónicas. El sonido de una nota de piano, por ejemplo, está formado por una combinación compleja de armónicos, aunque nuestro cerebro interpreta como definitorio de la altura tonal solo el armónico fundamental. El resto de armónicos presentes en el sonido del piano contribuyen a su carácter tímbrico y nos hacen percibir su sonido característico, diferenciable tímbricamente del sonido de la guitarra, por ejemplo.

1.5. Monofonía/polifonía

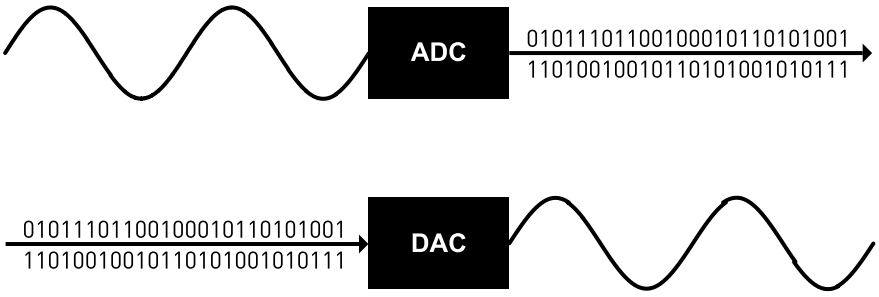
A la hora de plantear el diseño de nuestra interacción, hemos de tener muy en cuenta si vamos a trabajar con señales **polifónicas** o **monofónicas**. Una señal monofónica está formada por un único armónico fundamental (la voz humana o una trompeta) y es mucho más sencilla de analizar (como tono puro o *pure pitch*, como veremos más adelante) que una señal polifónica, que está compuesta por un conjunto de sonidos simultáneos a distintas frecuencias (una orquesta sinfónica, acordes de piano, rumor de la calle).

Aun así, como veremos más adelante, el análisis de señales mediante FFT nos puede ofrecer información muy útil, incluso en el caso de señales de audio polifónico.

Las características más importantes del sonido son la **frecuencia** (número de oscilaciones por segundo, medida en hercios) y la **amplitud** (intensidad de la onda sonora, medida en decibelios). Además, todos los sonidos están formados por un número de frecuencias que se producen simultáneamente a la frecuencia fundamental, que denominamos **armónicos** o **inarmónicos** dependiendo de si sus frecuencias son múltiplos enteros de la frecuencia fundamental o no lo son, y que aportan a cada sonido su carácter tímbrico.

1.6. ADC/DAC

ADC/DAC: convertidor analógico-digital / convertidor digital-analógico.



Dentro del dominio digital, los datos se almacenan en formato binario y con una resolución determinada por las capacidades de memoria y poder de computación de los procesadores. Del mismo modo que una imagen se almacena como una matriz de un número finito de píxeles, una onda sonora se almacena y se procesa como una lista finita de valores de amplitud en el tiempo.

La conversión de las vibraciones sonoras del mundo real, de "resolución infinita", en valores digitales comprensibles por un microprocesador se hacen gracias a un conjunto de componentes y microcontroladores electrónicos llamados **convertidores analógico-digital** (en inglés *analog to digital converters*, ADC).

Mediante transductores electroacústicos (micrófonos), se convierten las vibraciones en datos de audio analógico (cambios de voltaje), y los ADC se encargan de traducir estos valores de voltaje en señales binarias procesables por los ordenadores.

En el caso inverso, el de los **convertidores digital-analógico** (en inglés *digital to analog converters*, DAC), se convierten datos digitales en valores de voltaje que, cuando se aplican a un sistema de amplificador y altavoces, provocan el movimiento de las membranas de estos últimos, que a su vez provocan el movimiento del aire en frecuencias audibles.

1.7. *Sampling rate*

Frecuencia de muestreo

> La **frecuencia de muestreo**, normalmente medida en hercios (valores por segundo), es la resolución en el dominio temporal del que se compone una determinada muestra de audio.

Cuanto más alta sea la frecuencia de muestreo, más valores de amplitud por segundo tendrá la muestra y, por tanto, más resolución. Por poner varios ejemplos, la frecuencia de muestreo de una música comercial distribuida en CD o

MP3 suele ser de 44.100 Hz; la frecuencia de muestreo de la línea de telefonía fija es de 11.000 Hz, y las grabaciones profesionales de audio en estudio se pueden llegar a hacer a 192.000 Hz.

1.8. *Bit depth*

Bit depth: profundidad de bits.

Cada uno de los valores de amplitud (volumen) de una muestra de audio se entrega con un número entre –1 y 1, siendo 0 el valor neutro. Así pues, estamos hablando de números decimales.

Dependiendo de la profundidad de bits, estos números decimales entre –1 y 1 serán más o menos precisos.

El estándar actual está fijado en 16 bits para las grabaciones de audio en CD o MP3, aunque todavía encontramos audio a 8 bits en las videoconsolas portátiles y en pequeños dispositivos electrónicos. Las grabaciones de audio profesional suelen hacerse a 24 o 32 bits.

Así pues, la profundidad de bits define la resolución de los números decimales de los valores de amplitud de una onda sonora.

2. Las herramientas

2.1. Microfonía

La herramienta básica del diseño de interacciones mediante el sonido es el micrófono.

El **micrófono** es un transductor electroacústico. Su función es la de traducir las vibraciones debidas a la presión acústica ejercida sobre su cápsula por las ondas sonoras en energía eléctrica, lo que permite grabar sonidos de cualquier lugar o elemento.

Podemos hallar micrófonos de diferentes formas y sistemas de operación, de los que nos fijaremos en tres: direccionales, omnidireccionales y por contacto.

1) Direccionales

Los micrófonos unidireccionales o direccionales son aquellos micrófonos muy sensibles a una única dirección y relativamente sordos a las restantes. Su principal inconveniente es que no dan una respuesta constante: son más direccionales si se trata de frecuencias altas (agudos) que si son de frecuencias bajas (graves), puesto que la direccionalidad del sonido, como de todo tipo de ondas (ya sea mecánicas o electromagnéticas), depende de su frecuencia. Su principal ventaja es que permiten una captación localizada del sonido. Normalmente, se utilizan acoplados a jirafas de sonido.

2) Omnidireccionales

Los micrófonos omnidireccionales tienen un diagrama polar de 360° (la circunferencia completa).

Este tipo de micrófonos tienen una respuesta de sensibilidad constante, lo que significa que captan todos los sonidos, independientemente de la dirección desde la que lleguen.

Su principal inconveniente es que, al captarlo todo, captan tanto lo que queremos que capten como lo que no: ruido del entorno, reflexiones acústicas, etc.

3) De contacto

Toman el sonido porque están en contacto físico con el instrumento. Se utilizan también como sensores para disparar un sonido de un módulo o *sampler* por medio de un disparador o *trigger* MIDI.

2.2. Tarjetas de sonido

Estos sistemas de microfonía, en el caso del diseño de un sistema interactivo basado en software, se han de conectar a un ordenador para que procese los sonidos recogidos. Para conectar un micrófono al ordenador, usaremos una **tarjeta de audio**. Casi todos los ordenadores actuales llevan tarjetas de audio integradas y suelen incorporar un simple conector mini o *minijack* de 3,5 mm.

La mayoría de sistemas de microfonía profesional se conectan mediante XLR, mucho más estable y libre de ruidos, por lo que recomendamos que, en el caso de que se necesiten datos sonoros netos y fiables para el diseño interactivo, se usen tarjetas de audio externas con entradas y salidas con conectores balanceados y de gran formato.

2.3. Fuentes sonoras

Una decisión importante cuando diseñemos una interacción basada en audio es cuál será la fuente sonora que desencadenará las acciones diseñadas: la voz humana, un instrumento musical, palmadas...

Según cuál sea esta fuente, deberemos elegir el sistema técnico más adecuado: micrófonos de contacto, direccionales, inalámbricos...

3. Diseñando interacciones con sonido

A continuación, enumeraremos algunas de las técnicas que nos permiten interpretar datos para generar reacciones interactivas con el sonido. Como sabéis, una parte importante del diseño de una interacción es escoger qué información se recoge, cómo se analiza y cómo se definen reacciones al respeto. En este apartado, describiremos diversas formas de análisis del sonido obtenido. A partir de estas técnicas, podréis pensar más adelante en reacciones, ya sean en pantalla o mediante actuadores físicos (luces, motores, pistones, etc.).

3.1. Análisis de amplitud

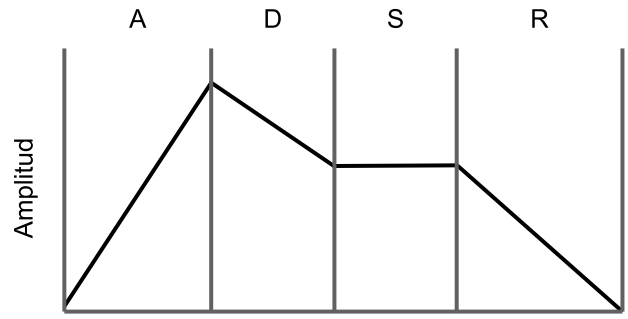
Por medio del análisis del volumen aparente de una muestra de sonido, podemos diseñar toda una serie de interacciones sencillas.

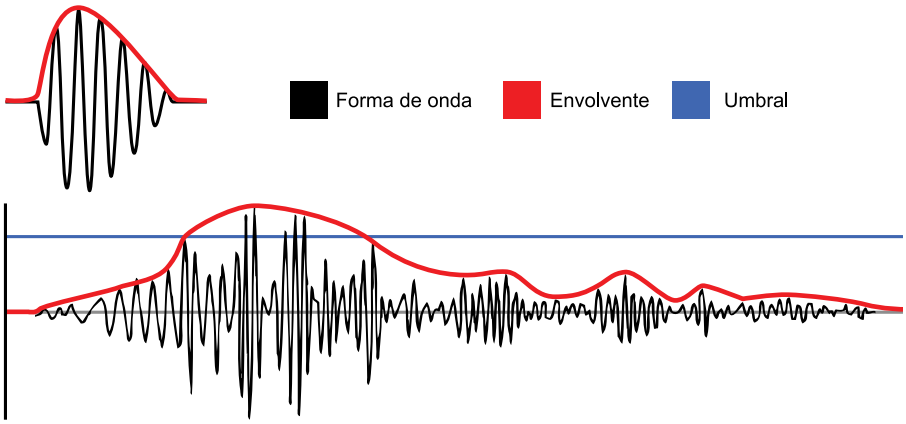
Hemos visto que, estrictamente, la amplitud de una muestra de audio cambia muchas veces en un pequeño fragmento de tiempo (la onda sonora audible más "lenta" tiene una frecuencia de 20 ciclos por segundo). Esto nos podría llevar a la conclusión de que, en realidad, la amplitud de una muestra cambia tantas veces por segundo que es imposible extraer de ella una información útil de cara al diseño de interacción.

1) Seguidores de envolvente

Para superar esta dificultad, se acostumbra a utilizar una técnica de "suavizado" de la onda sonora que permite extraer los datos de amplitud aparente. Esta técnica se conoce como **seguimiento del envolvente** (*envelope following*) y, por medio de la definición de los valores de ataque y caída de la curva de análisis, podemos obtener un muestreo suavizado de los valores de amplitud de la onda sonora.

Normalmente, las envolventes se pueden ajustar mediante cuatro parámetros: ataque, caída, sostenimiento, extinción (ADSR: *attack*, *decay, sustain, release*). Definiendo los tiempos de cada uno de estos parámetros, podemos generar envolventes de todo tipo (la envolvente de amplitud de una nota de piano, por ejemplo, tendría un ataque muy rápido, una caída y un sostenimiento muy breves, y una extinción bastante extensa).





A partir de estos datos que nos ofrece la curva de envolvente (ved imagen), podemos empezar a diseñar interacciones basadas en el volumen del sonido.

2) Definición de umbrales

Quizás el diseño de interacción más popular se base en la definición de un **umbral de amplitud** (*amplitude treshold*), definido en decibelios (dB), a partir del que se dispara un acontecimiento o *event*.

> Pongamos por caso que definimos un umbral alto y obligamos a los usuarios de nuestro diseño interactivo a emitir un sonido muy alto (gritando, por ejemplo) para que se dispare una acción determinada (diferente de un acontecimiento audiovisual, encender una luz, mover un motor, etc.).

3) Cálculo estadístico

Sin embargo, a partir del análisis estadístico y el uso de las matemáticas en general, podemos llegar a diseñar interacciones complejas que tengan en cuenta factores como el nivel de cambio de volumen a lo largo del tiempo.

> Por medio de algoritmos de programación, podríamos definir una interacción basada en el número de veces que se da una palmada, en la progresión de amplitudes (*in crescendo* o *in decrescendo*) o en la frecuencia de variación de amplitudes en el tiempo, por poner solo algunos ejemplos.

Ejemplo

Por ejemplo, para medir la expresividad de un instrumento musical o detectar alteraciones repentinas del espacio sonoro.

3.2. Análisis de frecuencia

Un algoritmo de detección de tono puro o *pure pitch* se encarga de estimar el tono o la frecuencia fundamental de una señal casi periódica o virtualmente periódica proveniente de una grabación digital.

Hay dos métodos populares de seguimiento de tono o *tracking pitch*: **ZCD** (*zero crossing detector*, 'detector de cruces por cero'), en el dominio del tiempo, y **FFT** (*fast fourier transforms*, 'transformada rápida de Fourier'), en el dominio de la frecuencia.

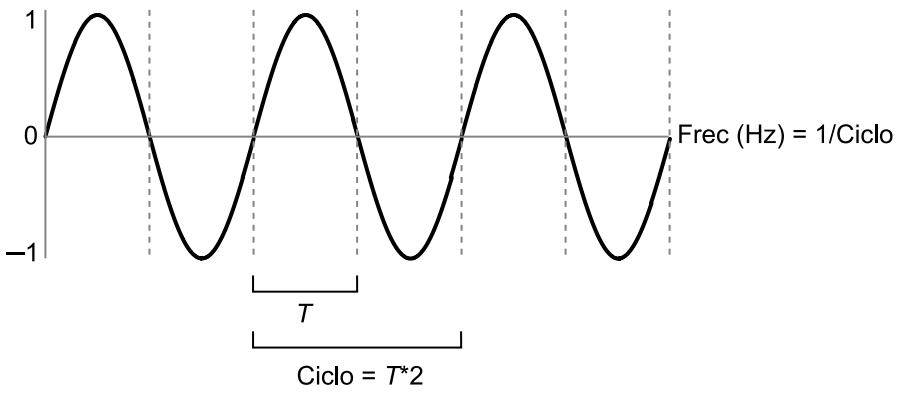
El ZCD es un algoritmo muy ligero que consume pocos recursos y que se puede implementar en la mayoría de sistemas (incluso en Arduino) con capacidad limitada de procesado de señales. Sin embargo, el ZCD funciona razonablemente bien en situaciones como el análisis de tonos telefónicos (DTMF) o en cualquier otro caso de análisis de señales sintéticas y periódicas, pero no es muy fiable cuando la señal que se ha de analizar es muy compleja o rica en armónicos, como señales de audio polifónico o procesamiento del habla.

El análisis por FFT es muy fiable y robusto, a pesar de que es muy exigente en cuanto a carga de procesador y no todos los dispositivos están preparados para hacer tareas de FFT en tiempo real.

1) ZCD

El algoritmo de cruce por cero se basa en la detección del cruce de la señal por el punto de amplitud cero y el cálculo del tiempo entre dos cruces sucesivos. Esta medida del tiempo entre los sucesivos cruces por cero nos da un dato fiable de la frecuencia de la onda sonora, que podemos aislar tal como sigue:

Tiempo entre ceros (en segundos) = T
Frecuencia en hercios = F
F = (1 / T) * 2



2) FFT

El análisis frecuencial mediante FFT (transformada rápida de Fourier) es muy versátil y nos puede dar una gran cantidad de datos útiles en el contexto del diseño de interacciones. *Grosso modo*, el método de Fourier se basa en la reducción de un fragmento (*window*) de la señal a sus componentes sinusoidales para facilitar el análisis matemático. Una vez hecha esta reducción sinusoidal, el algoritmo da información sobre las diferentes frecuencias presentes en la señal y sus valores de amplitud.

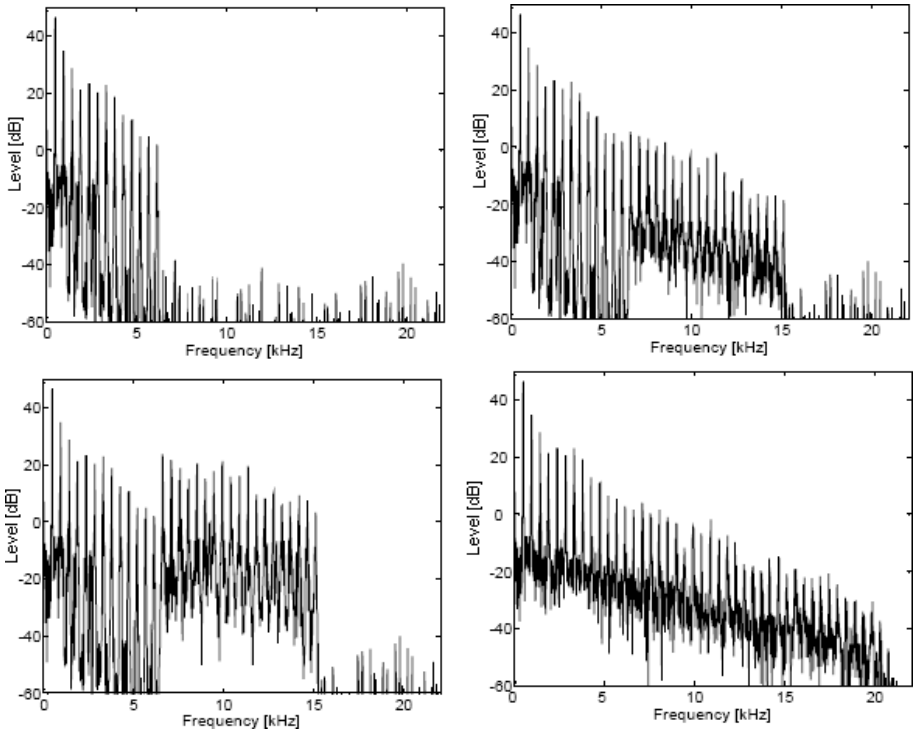
Este método, por tanto, además de darnos información sobre el tono fundamental de una señal de audio monofónico (una trompeta, por ejemplo), es capaz de facilitar una relación de energía para cada banda de frecuencias dentro del espectro sonoro.

Con esta información sobre cada banda del espectro, podemos diseñar un algoritmo de interacción sonora que mida la amplitud de tres bandas frecuenciales (graves, medios y agudos, por ejemplo) y asignar diferentes acontecimientos interactivos a la energía presente en cada una de esas bandas.

Ejemplo

Un algoritmo como este podría ser efectivo en una situación de visualización musical, por ejemplo.

Del mismo modo que con el análisis de amplitud, aplicando algoritmos matemáticos sobre la señal obtenida tras el análisis, podríamos diseñar interacciones más complejas: detección de escalas o patrones musicales, reconocimiento de bandas de formantes (fonemas), detección de estados de ánimos del habla poniendo en relación frecuencias y amplitudes y su evolución en el tiempo...



4. Más allá. Recursos y bibliografía específica

4.1. Otras estrategias de análisis

Reconocimiento del habla: http://en.wikipedia.org/wiki/Speech_recognition

4.2. Música visual

A lo largo de la historia del arte, podríamos encontrar numerosas analogías entre música y artes visuales gracias a artistas que incorporan un cierto sentido **sinestésico** a sus obras, trabajando en el análisis y la generación de pinturas o de música que evocan conceptos o sensaciones extravisuales o extramusicales. Casos de artistas visuales como Wassily Kandinsky son ilustrativos de esta corriente y representan una analogía clara con los métodos modernos de **visualización sonora** ejecutada mediante algoritmos de análisis de audio.

En esta línea, recomendamos que consultéis la obra de artistas como Oskar Fischinger, Norman McLaren o Walter Ruttman, puesto que su trabajo tiene mucho que ver con el contenido de este módulo y ofrece un punto de vista original acerca del análisis y la comprensión de la música como portadora de valores extramusicales.

http://en.wikipedia.org/wiki/Visual_music

4.3. Instituciones y centros de investigación

IRCAM: http://www.ircam.fr

MTG: http://mtg.upf.edu/

4.4. Herramientas especializadas de software

SuperCollider: http://www.audiosynth.com/

ChucK: http://chuck.cs.princeton.edu/

Csound: http://www.csounds.com/

Pure Data: http://puredata.info/

Max/MSP: http://cycling74.com/

Reaktor: http://www.native-instruments.com/

4.5. Bibliografía y recursos en línea

Miller Pucket, *The theory and technique of electronic music*

http://crca.ucsd.edu/~msp/techniques.htm

http://es.wikipedia.org/wiki/Transformada_r%C3%A1pida_de_Fourier

http://en.wikipedia.org/wiki/Pitch_detection_algorithm

http://en.wikipedia.org/wiki/Envelope_detector

Dispositivos electrónicos

Santiago Vilanova Ángeles

PID_00184755

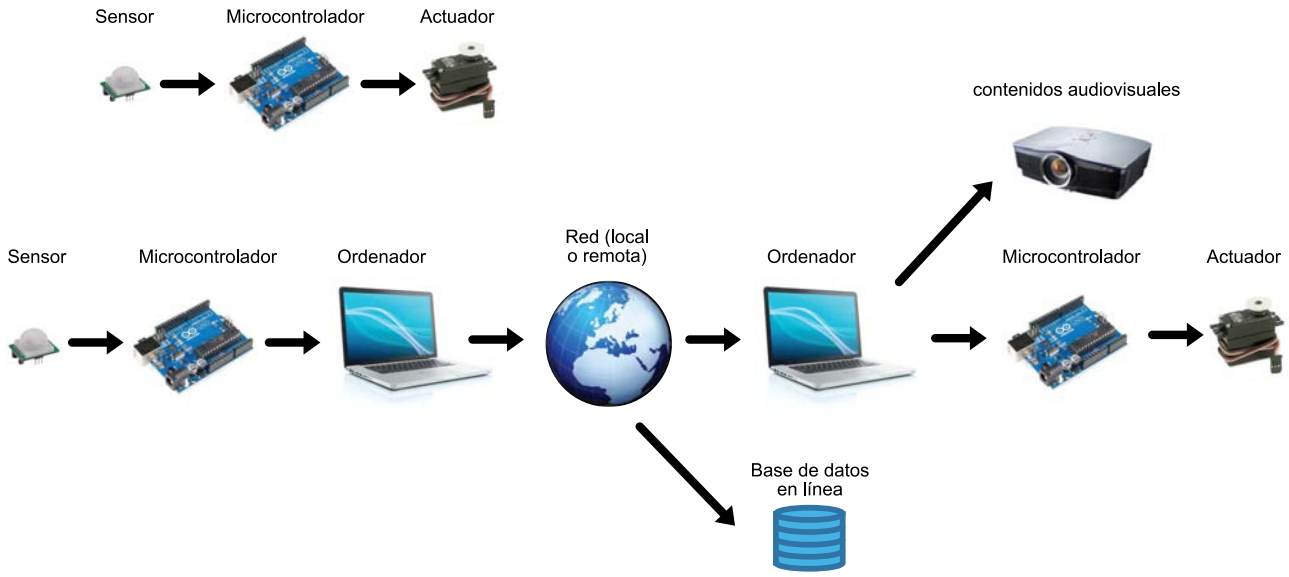
UOC
Universitat Oberta de Catalunya

Índice

Introducción

Un área muy extensa dentro del campo del diseño de interacciones es la del prototipado y el diseño de circuitos y artefactos electrónicos. Con el uso de redes de sensores y actuadores regulados por microcontroladores digitales, se pueden diseñar interacciones basadas en los datos recibidos por los sensores (de humedad, de temperatura, de proximidad, de presión, de flexión, etc.). Estos datos recopilados por los sensores e interpretados por los microchips digitales pueden, a su vez, incidir en la activación de otros componentes electrónicos, como motores, pistones electromagnéticos, LED, etc., y dar lugar a un sistema interactivo electrónico autónomo. Del mismo modo que pasa con los actuadores, los datos provenientes de los sensores se pueden convertir en datos digitales que administren un sistema de software y que den lugar a distintos acontecimientos (*events*) controlables mediante ordenadores (disparos de audio o de sonido, conexiones a Internet...).

Observad el esquema que encontraréis a continuación para entender algunas de las posibilidades de conectividad de estos sistemas electrónicos sumados a un sistema informático:



A lo largo de este módulo, estudiaremos las distintas estrategias aplicables al diseño de interacciones basadas en sistemas electrónicos. Tomaremos como plataforma de estudio el sistema **Arduino**, una plataforma de Open Hardware que ha significado una revolución en el campo de las tecnologías DIY (*do it yourself*, 'hágalo usted mismo'). Se trata de una placa con un microchip sencillo pero a la vez fácil de utilizar y con muchas posibilidades. Utilizando sensores o actuadores junto con Arduino, se pueden hacer prototipos interactivos de modo económico y eficaz. Gracias a su facilidad de uso y al gran número de usuarios que se han volcado en Internet, creando una comunidad abierta de

personas que intercambian conocimientos en forma de esquemas y diagramas electrónicos, códigos de software y estrategias de comunicación, Arduino se ha convertido en una de las herramientas más populares para estudiar e investigar la interactividad con la electrónica y los dispositivos físicos.

Objetivos

1. Introducir los principios fundamentales de la electrónica.

2. Proporcionar las claves para el aprendizaje de los conceptos básicos del prototipado.

3. Enumerar algunos de los componentes principales de la electrónica.

4. Dar a conocer algunas de las posibilidades del prototipado electrónico en el ámbito interactivo.

5. Entender cómo funciona un circuito sencillo.

6. Aportar el conocimiento necesario para que podáis diseñar vuestro propio circuito electrónico interactivo, trabajado con sensores y actuadores, confeccionado mediante la plataforma Arduino.

1. Conceptos teóricos

1.1. Breve introducción a la electrónica

Antes de entrar en profundidad en la materia propia del diseño de interacción mediante el prototipado electrónico, hemos de hacer una breve introducción a los conceptos básicos de la electrónica moderna.

Hoy en día, definimos la **electrónica** como la parte de la ciencia y de la técnica que trata del estudio de los electrones y de sus aplicaciones en el tratamiento y la transmisión de información.

A continuación, expondremos diversos conceptos clave para entender los principios de la electrónica. Aunque no pretendemos hacer una explicación exhaustiva, sí que intentaremos que sea la información necesaria para que podáis desarrollar después vuestros prototipos. Se trata de los fundamentos clave y, por tanto, tenéis que asimilarlos poco a poco. A lo largo del módulo, haremos prácticas y analizaremos diferentes ejemplos, de manera que esta lista os ha de servir como diccionario de referencia.

1.1.1. Corriente continua (DC) y corriente alterna (AC)

Podemos considerar básicamente dos tipos de corriente eléctrica: la **corriente continua** (DC) y la **corriente alterna** (AC).

- La corriente continua se caracteriza por el hecho de que en un circuito eléctrico el desplazamiento de electrones se hace siempre en el mismo sentido, con una tensión y una intensidad constantes en el tiempo.

Ejemplo

Es la corriente que proporcionan, por ejemplo, las pilas y las baterías.

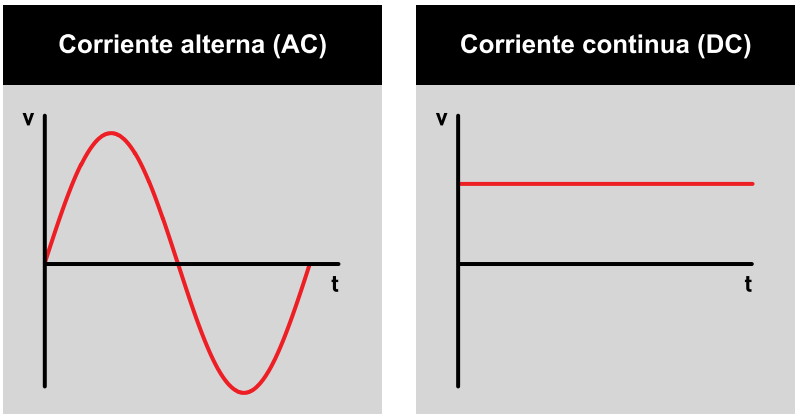
- En cambio, la corriente alterna se distingue por el hecho de ser una corriente variable, en la que las principales magnitudes que la definen, tensión e intensidad, cambian continuamente de valor y de sentido (la corriente de la red eléctrica doméstica es una onda sinusoidal a 50 Hz).

Ejemplo

Es la corriente que se usa mayoritariamente, tanto en las viviendas como en las industrias. Cuando conectamos un aparato a un enchufe, le estamos suministrando corriente alterna.

Los circuitos electrónicos (teléfonos, mandos a distancia, calculadoras, etc.) suelen funcionar casi siempre con corriente continua, puesto que, como hemos visto, su función es la del tratamiento y la transmisión de información, más que el suministro y la gestión de grandes cantidades de energía eléctrica. La corriente alterna permite mover grandes cantidades de energía en circuitos

de largas distancias y es la que se utiliza, por ejemplo, en los electrodomésticos y en las fábricas, porque permite activar motores potentes, iluminar grandes superficies, etc.



1.1.2. Intensidad, voltaje y resistencia. Ley de Ohm

- **Intensidad (o corriente)**. Medida en amperios (A), se define como el flujo de carga eléctrica por unidad de tiempo que recorre un material.

- **Voltaje**. Es la diferencia de potencial eléctrico entre dos puntos. Se mide en voltios (V).

- **Resistencia**. La resistencia eléctrica de un objeto es una medida de su oposición al paso de corriente.

La **ley de Ohm** establece que la corriente que atraviesa un circuito eléctrico es directamente proporcional a la diferencia de potencial que hay entre sus extremos e inversamente proporcional a la resistencia del circuito.

En términos matemáticos, la ley se expresa por medio de la ecuación $\boldsymbol{I = V / R}$.

Donde V es la caída de voltaje o diferencia de potencial e I es la corriente. La ecuación da como resultado la constante de proporcionalidad R, que es la **resistencia eléctrica del circuito**.

1.1.3. Resistencias

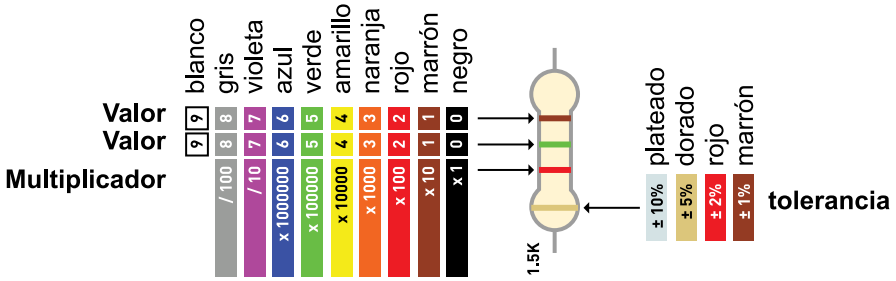
Los resistores son componentes que ofrecen una determinada resistencia al paso de la corriente eléctrica.

Es muy habitual encontrarlos en circuitos electrónicos y se usan sobre todo para limitar la intensidad de la corriente eléctrica en un punto determinado del circuito o para dividir el valor total de la tensión.

Como ejemplo para entender una de las aplicaciones de los resistores, diremos que para conectar un LED a una salida de Arduino, necesitaremos hacerlo a través de una resistencia de 220 ohmios para reducir la tensión de salida de Arduino desde 5 V hasta 3,3 V, que es el voltaje máximo al que puede trabajar un LED estándar. Así pues, una resistencia sirve, en este caso, para atenuar el voltaje de una fuente de alimentación hasta el LED.

Fuente: Steve Lodefink (cc)

Su valor, que se mide en ohmios, puede estar escrito directamente en la cara exterior del componente, a pesar de que, generalmente, se determina a partir de un código internacional de colores.

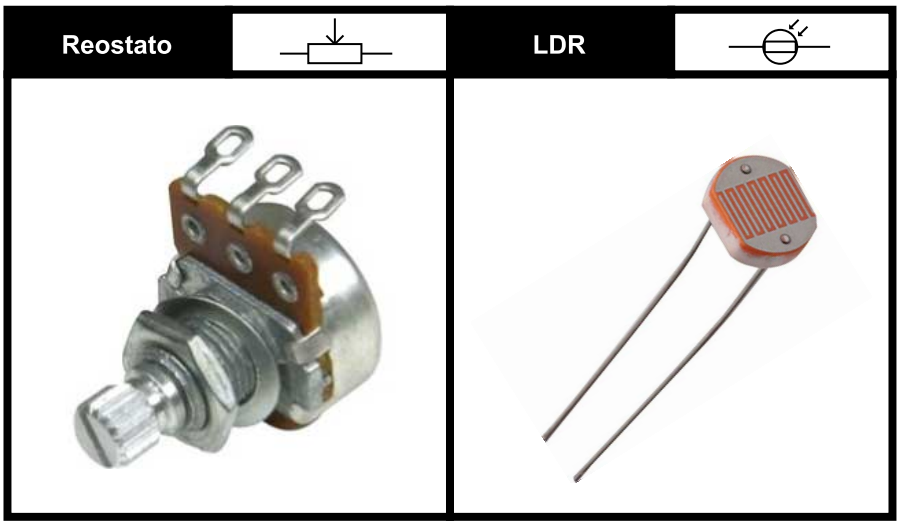


1.1.4. Resistencias variables

Los potenciómetros o reóstatos son resistores variables que se pueden graduar manualmente.

Sirven, por ejemplo, para variar el volumen de un aparato de música, la intensidad de luz de una bombilla o la velocidad de un motor.

Hay otros resistores variables, como las células fotoeléctricas, que gradúan su nivel de resistencia en función de la luz que captan.



1.1.5. Condensadores

En electrónica, en ocasiones es necesario disponer de componentes capaces de almacenar electricidad temporalmente y de descargarla de golpe en un determinado instante; por ejemplo, el flash de una máquina fotográfica. Estos componentes son los condensadores.

Así pues, el condensador es un componente que sirve para **almacenar temporalmente cargas eléctricas** sobre una superficie relativamente pequeña. La capacidad de los condensadores se mide en faradios (F).

Los condensadores son, después de los resistores, los componentes electrónicos más utilizados. Se usan, entre otras aplicaciones, en fuentes de alimentación, en filtros electrónicos y en circuitos de sintonización de señales de radiofrecuencia. Prácticamente todos los aparatos electrónicos emplean condensadores: ordenadores, teléfonos móviles, televisores, casetes de bolsillo, etc.

Fuente: Wikipedia (cc)

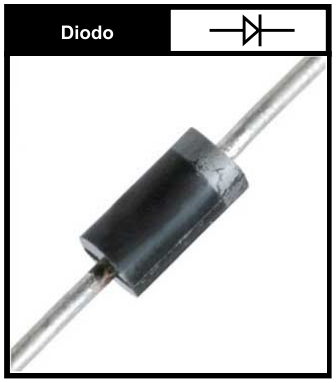
1.1.6. Diodos

Los diodos son unos componentes electrónicos activos que permiten el paso de la corriente en un único sentido.

El diodo es seguramente el componente semiconductor más sencillo. Tiene dos terminales, denominados ánodo y cátodo.

Si se conecta el borne positivo de una pila o fuente de alimentación al ánodo y el negativo al cátodo, el diodo conduce (estado de conducción) y permite el paso de la corriente a su través. Cuando se encuentra en este estado, el diodo tiene polarización directa y podemos decir que se comporta como un interruptor cerrado. Si invertimos la polaridad, el positivo lo conectamos al cátodo y el negativo al ánodo, el diodo no conduce (estado bloqueado) y no permite el paso de corriente a su través. El diodo se encuentra en polarización inversa y se comporta como un interruptor abierto (un interruptor cerrado hace que el circuito funcione, ya que conecta el circuito y permite su circulación; en cambio, un interruptor abierto no hace contacto y no permite el paso de la electricidad).

Los diodos son muy utilizados en fuentes de alimentación como rectificadores, es decir, para convertir en corriente continua la corriente alterna de la red eléctrica. También se utilizan en circuitos limitadores, en funciones lógicas y como elementos de protección.



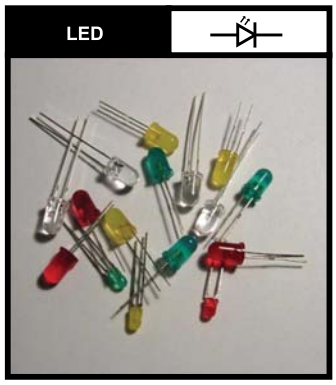
Fuente: Wikipedia (cc)

1.1.7. LED

Hay un tipo especial de diodo, llamado **LED** (*light emitting diode*), muy popularizado y utilizado como indicador luminoso del estado de un aparato (encendido, apagado, en espera, etc.) y que tiene como característica principal la emisión de luz cuando conduce.

El LED es un componente electrónico que **emite luz** cuando es atravesado por una corriente eléctrica.

Se trata de un diodo semiconductor parecido, desde el punto de vista electrónico, al que habéis estudiado, pero que tiene la propiedad de transformar la energía eléctrica en energía luminosa. Las ventajas más importantes que presentan los LED respecto de las bombillas piloto de filamento son alto rendimiento energético, poca producción de calor, vida útil muy elevada, tamaño reducido, carcasa resistente, disponibilidad en varios colores y bajo consumo.



Fuente: Wikipedia (cc)

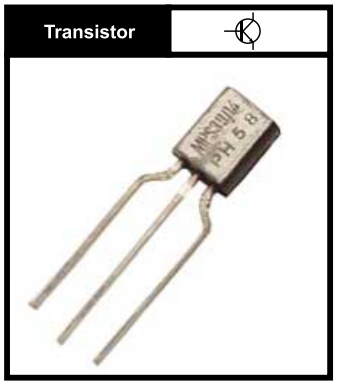
1.1.8. Transistores

Aunque existen decenas de tecnologías de transistores (MOSFET, bipolar, CMOS...), con diferentes estructuras internas, a grandes rasgos podríamos definir el transistor como un componente electrónico formado internamente por tres capas de material semiconductor y que consta de tres partes bien diferenciadas: emisor (E), base (B) y colector (C). Físicamente, la base siempre está entre el emisor y el colector, de manera que el emisor y el colector quedan en los extremos.

El transistor, por tanto, es una especie de sándwich entre capas de material semiconductor de signo opuesto (P –positivo– o N –negativo–).

La combinación de estas partes de material semiconductor de clase P o N da lugar a dos tipos de transistores: el transistor PNP y el transistor NPN; dependiendo de la funcionalidad o el tipo de componente que necesitemos asociar al transistor, requeriremos uno u otro (en una tira de LED comercial, por ejemplo, la regulación de la intensidad de los colores RGB se hace por medio de los terminales negativos, puesto que suele tratarse de diodos LED de ánodo común y, por lo tanto, necesitaremos usar un transistor NPN).

Con la aplicación de una pequeña corriente a través de la unión base-emisor, se establece una corriente mucho mayor entre la unión colector-emisor.



Fuente: Wikipedia (cc)

Así pues, el transistor es un componente electrónico que nos deja "amplificar" voltajes eléctricos y que permite convertir pequeñas tensiones eléctricas en tensiones más grandes.

Este componente será útil si, por ejemplo, queremos activar un motor de 12 V mediante las salidas analógicas de Arduino, que trabajan a un máximo de 5 V.

1.1.9. Relés

El relé es un **interruptor** eléctrico que se acciona por medio de un **electroimán**.

Está formado por una bobina que, cuando circula una corriente eléctrica, atrae una lámina metálica que acciona un contacto que se abre o se cierra. Cuando la corriente deja de circular por la bobina del electroimán, un muelle hace volver a la lámina metálica y al contacto a su posición original. Los relés son muy útiles y se usan mucho, porque con corrientes de poca intensidad permiten controlar otros circuitos de intensidad mucho mayor, y también porque pueden ser gobernados a distancia. La corriente que circula por la bobina del relé recibe el nombre de corriente de maniobra o de mando, mientras que la que circula por el segundo circuito (es decir, por los contactos) recibe el nombre de corriente principal o de potencia.

Fuente: Wikipedia (cc)

Ejemplo

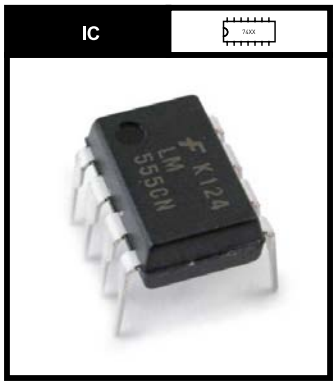
Por ejemplo, un relé de 9 voltios de corriente continua lo podemos accionar usando una pila de 9 voltios para el circuito de maniobra; en cambio, en el circuito de potencia, a través de los contactos del relé, podemos conectar una bombilla de 220 voltios de corriente alterna que se encenderá cuando demos corriente al relé de 9 voltios. Podemos manipular así aparatos de corriente de 220 V mediante un circuito que funcione con pilas de 9 V.

1.1.10. Circuito integrado (IC, *integrated circuit*)

Un circuito integrado o chip es un **dispositivo electrónico de pequeñas dimensiones** que consiste en un conjunto de elementos, como diodos, transistores, resistores y condensadores, permanente e íntimamente conectados a un material semiconductor (generalmente, silicio) y que forman un **circuito miniaturizado**.

Sin duda, el circuito integrado ha sido el elemento básico de la revolución tecnológica de la segunda mitad del siglo XX. Lo encontramos presente en muchos aparatos de uso cotidiano: ordenadores, televisores, teléfonos, aparatos de audio y vídeo, equipos de música, electrodomésticos, automóviles, relojes... E incluso encontramos pequeños chips (microchips) en algunas tarjetas inteligentes (como tarjetas de crédito, tarjetas monedero, tarjetas telefónicas), pasaportes, etc.

De hecho, el corazón de Arduino es un microchip, el Atmega328. Este microchip, responsable de la gestión de entradas y salidas de la placa, es reprogramable mediante el lenguaje propio de Arduino, lo que permite configurar el comportamiento del microchip según la lectura de los sensores y decidir qué acción han de hacer los actuadores en consecuencia. Veremos ejemplos con Arduino más adelante en este documento.

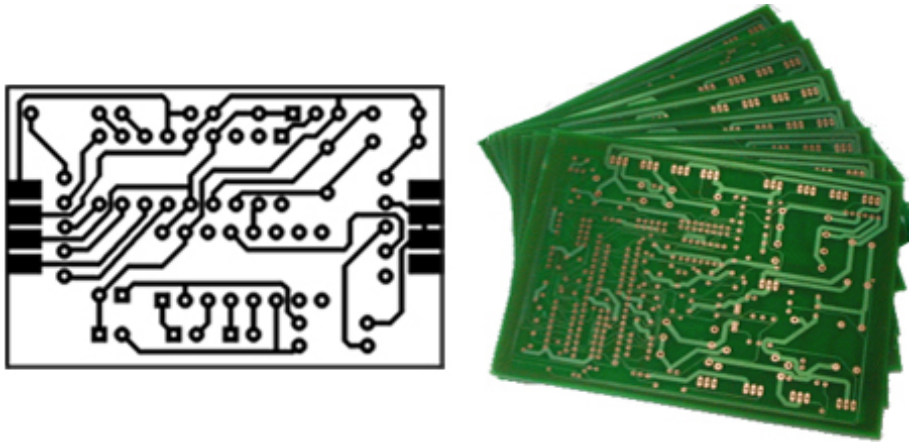


Fuente: Wikipedia (cc)

1.1.11. Placa de circuito impreso (PCB, *printed circuit board*)

Una placa de circuito impreso (PCB, *printed circuit board*) es una placa diseñada para acoger circuitos electrónicos con múltiples componentes.

Estas placas disponen de pistas de material conductor (generalmente estaño o cobre) que conectan los diferentes componentes electrónicos. Actualmente, gracias a las herramientas de diseño CAD, como Eagle, es relativamente sencillo diseñar placas de circuito impreso en casa, placas que después se pueden fabricar mediante sistemas industriales o de fabricación casera.



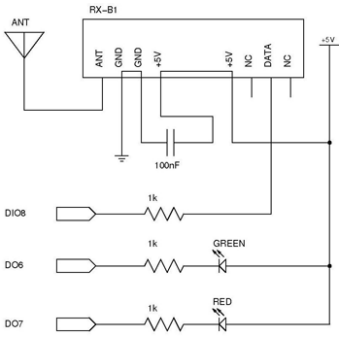
Diseño previo de PCB realizado mediante herramientas CAD y placa impresa final
Fuente: Wikipedia (CC)

1.1.12. Esquemáticas

Un **diagrama electrónico**, también conocido como *esquema eléctrico* o *esquemática*, es una representación pictórica de un circuito eléctrico.

Muestra los diferentes componentes del circuito de modo simple y con **pictogramas** estandarizados de acuerdo con normas, y las conexiones de alimentación y de señal entre los distintos dispositivos. La disposición de los componentes y las interconexiones en el esquema generalmente no corresponden a sus ubicaciones físicas en el dispositivo acabado.

Por regla general, las mismas herramientas para el diseño de placas de circuito impreso permiten confeccionar esquemáticas, ya que incorporan bibliotecas con los símbolos de los componentes electrónicos y se adaptan a la normativa de dibujo de este tipo de diagramas.



Fuente: Wikipedia (cc)

Recomendamos el uso del software libre Fritzing para el diseño de vuestros diagramas y placas de circuito impreso, puesto que este programa se integra muy bien con la plataforma Arduino y es una herramienta sencilla y versátil.

En el wiki de la comunidad Arduino y en el del aula, podréis encontrar también la documentación necesaria para poder interpretar de un simple vistazo las esquemáticas que necesitéis consultar en la red.

Además de este resumen general, recomendamos la lectura de los diferentes materiales sobre electrónica que encontraréis referenciados en el aula y en la bibliografía.

Historia de la electrónica

El origen de la electrónica se suele situar a finales del siglo XIX, cuando en el año 1883, Thomas A. Edison descubrió el efecto termoiónico, al observar que cuando se calentaba un material metálico se producía una emisión de electrones. J. A. Fleming aprovechó este descubrimiento para construir, en 1904, la válvula de vacío, con la que detectó señales de radio, y que es considerada el primer componente electrónico.

En 1948 las válvulas de vacío empiezan a ser sustituidas por diodos y transistores fabricados con materiales semiconductores, como el silicio o el germanio, lo que representó un salto cualitativo importante en la electrónica, dado que estos materiales gozaban de ventajas considerables: eran más sólidos y robustos, más resistentes a los golpes; además, tenían un volumen mucho más reducido, una vida útil más larga, y mejoraban el tratamiento de la señal.

En **1960** apareció el primer **circuito integrado (chip)**, lo que permitió miniaturizar todavía más los equipos electrónicos. En 1971 la empresa Intel fabricó el primer chip microprocesador y dio un nuevo impulso al progreso tecnológico y a la investigación en electrónica. Todos estos avances de la electrónica han sido fundamentales para el desarrollo de diferentes campos de aplicación industrial y doméstica: automatización, control y regulación de procesos, informática, robótica, telecomunicaciones, transportes, electromedicina, investigación científica y espacial, láser, electrónica de consumo, electroacústica, etc. El ámbito de investigación y de desarrollo de la electrónica consiste en diseñar nuevos circuitos basados en el comportamiento de los electrones en los materiales. Por eso su evolución práctica va ligada al conocimiento tecnológico de los materiales.

Hoy en día estamos rodeados de componentes electrónicos. Uno de los campos de investigación más emergentes es la nanotecnología, en la que se investiga la posibilidad de seguir reduciendo el tamaño de los chips hasta medidas nanométricas y la construcción de circuitos moleculares.

Fuente: Wikipedia

2. Las herramientas

2.1. Arduino

Arduino es una plataforma de hardware libre basada en una **placa** con un **microcontrolador** y un entorno de desarrollo, y diseñada para **facilitar el uso de la electrónica en proyectos multidisciplinares**.

El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. El software se basa en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (*boot loader*) que corre en la placa.

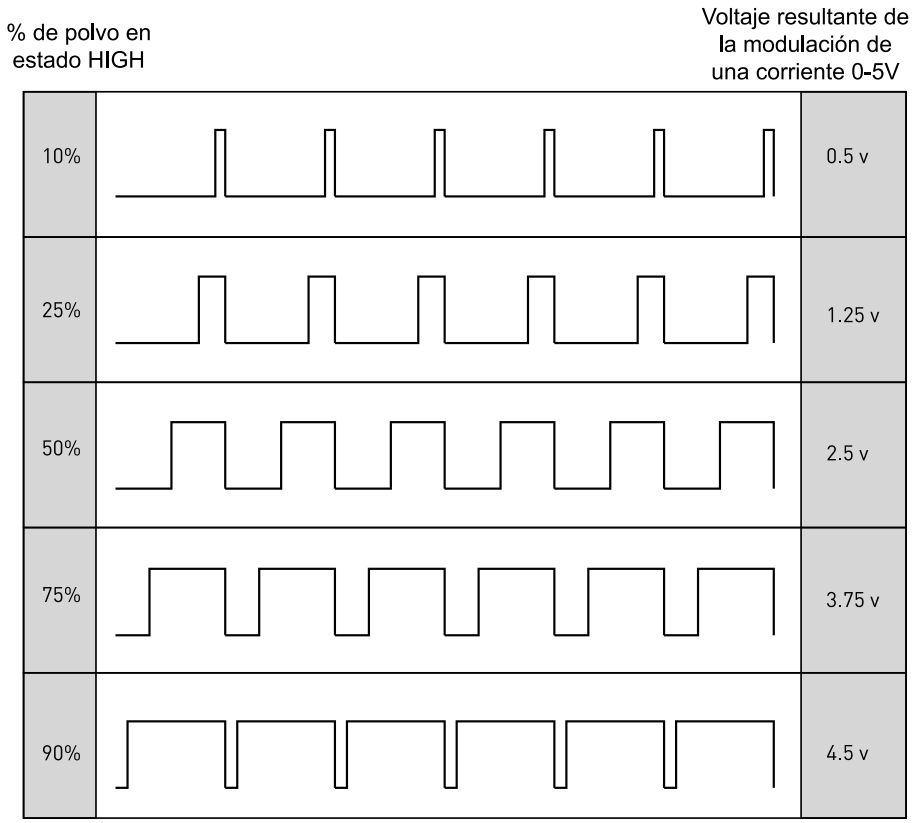
Arduino se puede utilizar para desarrollar objetos interactivos autónomos o conectarse a software del ordenador (por ejemplo, Macromedia Flash, Processing, Max/MSP, Pure Data). Las placas pueden montarse a mano o adquirirse. El entorno de desarrollo integrado libre puede descargarse gratuitamente.

Al tratarse de **Open Hardware**, son libres tanto su diseño como su distribución; es decir, puede utilizarse libremente para el desarrollo de cualquier tipo de proyecto sin tener que pagar por una licencia, puesto que es una licencia abierta (Creative Commons).

Consta de catorce entradas digitales configurables como entrada o salida que actúan a 5 voltios. Cada pin puede proporcionar o recibir como máximo 40 mA. Estas entradas y salidas digitales permiten la conexión de sensores y actuadores, y operar con ellos en modo binario, es decir, alternando entre dos estados (en el caso de la conexión de un LED como actuador, por ejemplo, solo podríamos alternar entre el 0% y el 100% de luminosidad y, obviamente, todos los valores de luminosidad intermedios).

Los pines 3, 5, 6, 8, 10 y 11 pueden proporcionar una salida PWM (*pulse width modulation*). La técnica de la modulación de ancho de impulsos permite simular el comportamiento de una corriente analógica con la modulación de una señal de voltaje digital mediante la variación de la simetría de las ondas cuadradas. A efectos prácticos, mediante la PWM podemos obtener una corriente regulada entre 0 y 5 V obteniendo todos los valores intermedios (por ejemplo, 3,3 V), a diferencia del comportamiento de las salidas digitales. La escritura de valores PWM en un actuador conectado a Arduino oscilará entre valores numéricos dentro del rango 0-255. (En el caso de la conexión de un LED co-

mo actuador, por ejemplo, podríamos generar rampas de interpolación de la luminosidad desde 0% hasta 100%, pasando por todos los valores de luminosidad intermedios.)



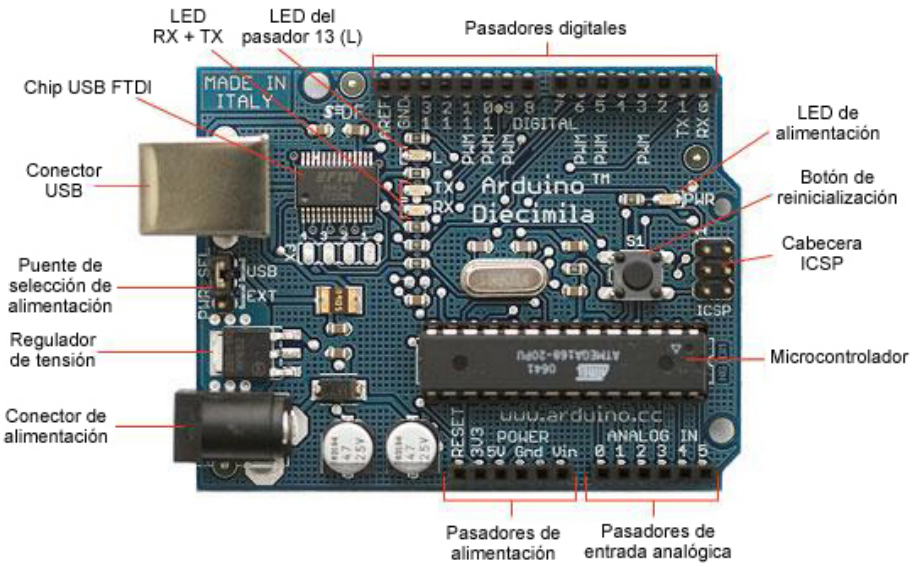
Si se conecta cualquier sensor o actuador en los pines 0 y 1, interferirá con la comunicación USB, puesto que mediante estos dos pines se gestiona la comunicación serial. Así pues, debemos ir con cuidado y ser conscientes de que si queremos usar estos puertos como parte de nuestro diseño, tendremos que desconectar cualquier cosa que tengamos conectada cada vez que queramos actualizar el firmware, o si queremos usar las capacidades de comunicación serial con un ordenador.

El Arduino también tiene seis entradas analógicas que proporcionan una resolución de 10 bits. Por defecto miden de 0 voltios (masa) hasta 5 voltios. La lectura de los valores de un sensor en Arduino oscilará entre valores numéricos dentro del rango 0-1.024.

Uno de los puntos fuertes de la plataforma Arduino es su comunidad en línea.

Gracias a un sistema de foros (arduino.cc/forums), tutoriales y ejemplos (arduino.cc/playground) en línea muy organizados, Arduino consigue resolver las necesidades de aquellos que, sin conocimientos previos de electrónica y programación, quieren introducirse en el mundo del diseño de interacciones mediante el prototipado electrónico.

Recomendamos de manera entusiasta que consultéis los foros y el *playground* de Arduino, puesto que constituyen una fuente inacabable de inspiración, conocimientos y apoyo durante el proceso de aprendizaje.



Fuente: sparkfun (cc)

2.2. Sensores

Un sensor es un dispositivo que detecta magnitudes físicas o químicas y que las transforma en variables eléctricas. Las magnitudes que mide un sensor pueden ser, por ejemplo, humedad, temperatura, proximidad, presión, flexión, vibración, intensidad lumínica...

En general, según su modo de funcionamiento, hallaremos dos tipos de sensores: analógicos y digitales. Habitualmente, los sensores analógicos devuelven **un rango de voltajes** entre 0 y *N* voltios (en las prácticas que realizaremos, *N* = 5), dependiendo de la magnitud de lectura (la cantidad de presión que ejerzamos sobre el sensor, la temperatura que apliquemos, etc.).

Además, **los sensores digitales devuelven solo dos valores** (HIGH, LOW), generalmente 0 V para el estado LOW y *N* V para el estado HIGH. En muchos casos (por ejemplo, la lectura de los sensores de distancia por ultrasonidos), los valores intermedios se pueden obtener a partir de la **medida de frecuencia** de la alternancia entre estos dos valores LOW y HIGH.

A continuación, referenciamos la tabla de un artículo de Wikipedia con enlaces que os pueden hacer conocer algunos de los numerosos sensores que existen hoy en día.

Magnitud	Transductor
Posición lineal o angular	Potenciómetro
	Codificador (*encoder*)
	Sensor Hall
Desplazamiento y deformación	Transformador diferencial de variación lineal
	Galga extensiomètrica
	Magnetoestrictivos
	Magnetorresistivos
	LVDT
Velocidad lineal y angular	Dinamo tacométrica
	Codificador (*encoder*)
	Detector inductivo
	Servoinclinómetros
	RVDT
	Giroscopio
Aceleración	Acelerómetro
	Servoacelerómetro
Fuerza y par (deformación)	Galga extensiométrica
	Triaxiales
Presión	Membranas
	Piezoeléctricos
	Manómetros digitales
Caudal	Turbina
	Magnético
Temperatura	Termopar
	RTD
	Termistor NTC
	Termistor PTC
	[Bimetal - termostato]
Sensores de presencia	Inductivos
	Capacitivos
	Ópticos
Sensores táctiles	Matriz de contactos

Fuente: Wikipedia

Magnitud	Transductor
	Piel artificial
Sensor de proximidad	Sensor final de carrera
	Sensor capacitivo
	Sensor inductivo
	Sensor fotoeléctrico
Sensor acústico (presión sonora)	Micrófono
Sensor de luz	Fotodiodo
	Fotorresistencia
	Fototransistor
	Célula fotoeléctrica

Fuente: Wikipedia

Generalmente, conectaremos estos sensores a las entradas analógicas de Arduino, de modo directo o por medio de un circuito que manipule o ajuste la señal (si el rango de señal eléctrica que ofrece el sensor es distinto de 0-5 V o si necesitamos algún tipo de filtrado de la señal mediante resistencias).

Una vez conectados los sensores a Arduino, cargando el código adecuado, podremos proceder a la lectura de los datos recogidos por los mismos y a su posterior uso como parte integrante de un diseño interactivo.

2.3. Actuadores

inici text clau

Un actuador es un dispositivo que transforma energía eléctrica (o hidráulica o neumática) en movimiento, luz, sonido...

Ejemplo

Serían ejemplos claros de actuadores los motores, los LED y los cristales piezoeléctricos, pero también los componentes electrónicos que actúan como interfaz de potencia de otros dispositivos, como los relés o los transistores. A estos últimos elementos se les denomina generalmente *preactuadores*.

Del mismo modo que pasa con los sensores, según su modo de funcionamiento los actuadores pueden ser analógicos o digitales. Si los actuadores aceptan **rangos de tensión eléctrica** (por ejemplo, los LED), diremos que son actuadores **analógicos**. Al contrario, si solo aceptan **dos estados eléctricos (LOW y HIGH)**, diremos que son actuadores **digitales**.

Del mismo modo que pasa con los sensores, para conectar los actuadores a Arduino, deberemos tener en cuenta la tensión eléctrica que necesitan para funcionar. Si esta tensión varía entre 0 y 5 V, podemos conectar los actuadores directamente a Arduino. En caso contrario, necesitaremos un circuito acondicionador de la señal, mediante el uso de transistores, resistencias, relés, etc.

Fuente: Wikipedia (cc)

2.4. Componentes electrónicos

En la primera parte teórica de este módulo, hemos hecho un repaso de los componentes más comunes que se utilizan en el prototipado electrónico. Dependiendo de cuál sea el circuito que diseñemos, es posible que necesitemos diferentes componentes electrónicos que nos permitan acondicionar las señales eléctricas para que se comporten tal como necesita nuestro diseño.

2.5. Cableado y estaño

Para hacer las conexiones entre componentes con la placa de pruebas se usa cable unifilar de diversos colores.

En el caso de circuitos con un paso más de finalización, hechos mediante placas perforadas de baquelita, se usa estaño para hacer los caminos de soldadura.

2.6. Soldador

Aunque las prácticas hechas mediante placas de pruebas (*protoboards*) no necesitan el uso del soldador, puesto que todos los circuitos se montan sobre una placa de pruebas que no necesita soldadura, hemos de tener en cuenta que el soldador, o la estación de soldadura, es un elemento imprescindible a la hora de diseñar prototipos electrónicos funcionales, porque nos permite dibujar pistas permanentes de estaño en una placa perforada, método que da lugar a una aproximación relativamente fiel a un producto electrónico acabado.

2.7. Placa de pruebas (*protoboard*)

Una **placa de pruebas**, también conocida como ***protoboard*** o ***breadboard***, es una placa de uso genérico reutilizable o semipermanente que se emplea para construir prototipos de circuitos electrónicos con o sin soldadura.

Normalmente, se utilizan para la realización de pruebas experimentales. En una placa de pruebas común, los distintos agujeros están conectados tal como sigue:

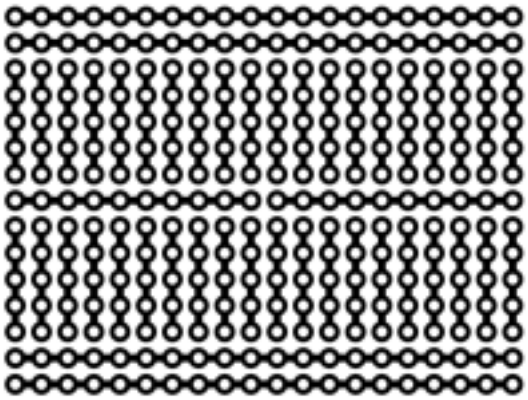
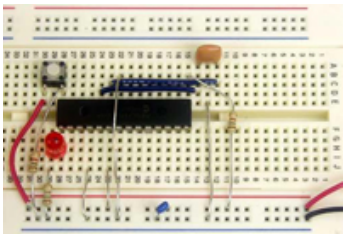


Diagrama de conexiones internas de las pistas conductivas en una placa de pruebas.
Fuente: Wikipedia (cc)

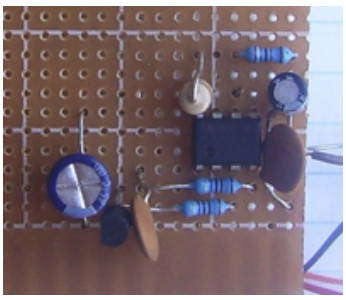


Fuente: Wikipedia (cc)

2.8. *Perfboard*

Placa de circuito perforada con agujeros rodeados de material conductor, usualmente de cobre, pero que no están interconectados.

Este tipo de placas requieren que cada componente esté soldado a la placa y que, además, las interconexiones se hagan por medio de cables o caminos de soldadura, normalmente realizados con estaño.



Fuente: Wikipedia (cc)

3. Diseñando interacciones con Arduino

3.1. Utilización de la placa de pruebas

Como hemos visto en los puntos anteriores, la placa de pruebas es muy útil para hacer pruebas rápidas de circuitos. Hemos de tener en cuenta el diagrama de conexiones internas de la placa y conectar a los laterales los polos positivo y negativo de la corriente eléctrica, que en la mayoría de casos será de +5 V cc.

Ved también

Ved el diagrama del apartado 2.7 de este mismo módulo.

Una vez conectadas a la placa de pruebas, estas señales de voltaje positivo y negativo se pueden llevar a cualquier otro punto de la misma, facilitando su diseño y reduciendo el número y la longitud de los cables que tendremos que utilizar.

3.2. Firmware

Arduino es un sistema combinado de hardware y software. El comportamiento y la relación entre las entradas y las salidas se definen por medio de códigos de software (firmwares) que se pueden cargar en la memoria de la placa. Así pues, Arduino nos permite, mediante programación, modificar el comportamiento del microchip (cambiando la información del firmware) para hacer que funcione del modo deseado según el circuito que hayamos diseñado. Arduino dispone de un entorno de desarrollo (IDE, *integrated development environment*) propio, que permite la entrada de código (basado en el lenguaje C). Dependiendo del tipo de proyecto que queramos desarrollar, necesitaremos diferentes firmwares, que podemos programar nosotros mismos usando la referencia en línea del lenguaje de programación y la ayuda de las comunidades en línea de aficionados.

3.3. Lectura de un sensor analógico

Los sensores analógicos conectados a Arduino provocan en este una lectura de valores dentro del rango numérico 0-1.024. Por medio de distintas manipulaciones numéricas o algoritmos de programación, podemos acondicionar estos datos para que se comporten del modo deseado.

3.4. Lectura de un sensor digital

Los sensores digitales conectados a Arduino provocan en este una lectura de valores binarios: HIGH o LOW. De nuevo, por medio de distintas manipulaciones numéricas o algoritmos de programación, podemos acondicionar estos datos para que se comporten del modo deseado.

3.5. Conexión de un actuador digital

Arduino dispone de numerosos pines de salida digital. Estas salidas solo tienen dos estados: HIGH o LOW. Cuando se encuentran en estado HIGH, las salidas entregan 5 V de salida, y cuando se encuentran en estado LOW entregan 0 V de salida.

Estos pines están pensados para ser utilizados con actuadores que solo necesitamos que tengan dos estados.

> Un relé, por ejemplo, es un tipo de actuador binario que encajaría con este tipo de salida. Si le conectáramos un LED, por ejemplo, solo podríamos variar la luminosidad entre dos estados (apagado, y encendido a máxima intensidad luminosa), y se perderían todos los valores de luminosidad intermedia.

3.6. Conexión de un actuador analógico

Aunque Arduino no dispone de salidas analógicas reales, ofrece una "emulación" de estas por medio del sistema PWM (modulación de ancho de pulsos). El funcionamiento es el siguiente: para dar una intensidad de 2,5 V, se modula la frecuencia en la que se envían 0 y 5 V para que la media sea de 2,5. Por tanto, estas salidas digitales permiten resultados "analógicos", ya que facilitan la regulación del voltaje de salida dentro del rango de los 0-5 V cc.

Conectando un LED, por ejemplo, y regulando el voltaje de salida, podemos controlar el nivel de intensidad lumínica que genera la luz del LED.

El rango de valores en el que funciona Arduino para el control de salidas analógicas oscila dentro del rango 0-255.

3.7. Comunicación serial

Además de leer sensores o de intervenir sobre actuadores, podemos enviar información a un ordenador o recibirla mediante el **cable USB**. Así, los datos recogidos por un sensor pueden ser analizados y enviados a una computadora para ampliar las posibilidades del diseño de interacción y hacer, por ejemplo, que estos datos recogidos formen parte de una base de datos en línea, o disparar imágenes de vídeo en función de la lectura de un sensor. Sin embargo, podemos generar comunicación en el sentido contrario, es decir, enviar datos hacia Arduino desde el ordenador para activar un sistema de actuadores (motores, LED, etc.) y poner en relación otras estrategias de diseño interactivo (análisis de audio, visión artificial o *computer vision*) con un sistema electrónico. Tanto **Processing** como la mayoría de entornos de programación interactiva (Max/MSP, Flash, PD, openFrameworks...) incorporan funcionalidades de comunicación mediante puerto en serie.

La comunicación serial[1] se hace a través de los pines digitales 0 (RX) y 1 (TX), y también con el ordenador mediante USB. Por lo tanto, si utilizáis estas funciones, no podéis usar los pines 0 y 1 como entrada o salida digital. La transferencia de datos por comunicación serial implica la transferencia de información bit a bit.

(1)La palabra *serial* significa "uno después de otro".

La información se pasa bidireccionalmente entre el ordenador y Arduino mediante la alternancia de estados HIGH y LOW (1 y 0) de un pin. Del mismo modo que ponemos en marcha y apagamos un LED, podemos también enviar datos codificados. Podríamos hacer una analogía con el código Morse, en el que usamos puntos y rayas para enviar mensajes telegráficos.

Por suerte, existen bibliotecas de comunicación que nos permiten llevar a cabo todas estas acciones sin tener que programar desde cero.

Un bit puede tener dos valores: 0 y 1.

Se pueden agrupar ocho bits, lo que da lugar a un **byte**.

Un byte puede almacenar valores numéricos hasta el número 256.

Para enviar datos numéricos mayores de 256 por comunicación serial, necesitamos diseñar un protocolo que nos permita dividir un único número en dos paquetes de mensajes distintos, que después serán reensamblados, mediante operaciones matemáticas o enviar cada cifra del número como un paquete separado mediante el código ASCII.

4. Más allá. Recursos y bibliografía específica

El mundo de la electrónica es muy extenso, y no en vano hay estudios universitarios de alto nivel sobre esta disciplina. Por tanto, hay un volumen enorme de documentación al respecto que, obviamente, no podemos citar en este documento. Recomendamos que hagáis búsquedas específicas si os encontráis con problemas o necesidades concretas a la hora de afrontar vuestros diseños interactivos basados en electrónica. Sin embargo, nos gustaría hacer referencia a varias técnicas generalizadas en la comunidad de diseñadores de interacción y que pueden estar a nuestro alcance con nuestros conocimientos actuales:

4.1. XBee

XBee es un sistema inalámbrico de transmisión de datos compatible con Arduino que puede ayudarnos a diseñar sistemas que se comunican sin cables.

http://www.arduino.cc/en/Main/ArduinoXbeeShield

4.2. Eagle y Fritzing

Tanto Eagle como Fritzing son herramientas de diseño de diagramas y circuitos electrónicos asistidos por ordenador que nos ayudan en el cálculo de posicionamiento de las pistas conductivas. Eagle es el estándar industrial en este tipo de sistemas CAD, mientras que Fritzing es un proyecto Open Source pensado específicamente para Arduino y orientado a usuarios sin conocimientos previos de electrónica.

http://fritzing.org/

http://www.cadsoftusa.com/

4.3. Robótica

La robótica es una disciplina muy atractiva que hibrida diferentes áreas de conocimiento y que está a nuestro alcance con los conocimientos de los que disponemos. La robótica mezcla el diseño de sistemas electrónicos, la mecánica y la inteligencia artificial (software). Hay numerosas plataformas de robótica educativa a nuestro alcance que pueden servir de punto de inicio para introducirnos en esta disciplina.

http://mindstorms.lego.com/en-us/Default.aspx

http://www.robotis.com/xe/bioloid_en

4.4. Bibliografía

Además, nos gustaría recomendar los libros siguientes, que en mayor o menor medida ofrecen conocimientos relacionados con el prototipado electrónico y Arduino:

Practical Arduino: Cool Projects for Open Source Hardware (1.ª edición). Apress. 28 de diciembre de 2009.

Programming Interactivity: A Designer's Guide to Processing, Arduino, and openFramework (1.ª edición). O'Reilly Media. 15 de julio de 2009.

Getting Started with Arduino (1.ª edición). Make Books. 24 de marzo de 2009.

Visión artificial

Eloi Maduell i García

PID_00184756

UOC
Universitat Oberta de Catalunya

Índice

Introducción

La visión artificial o visión por computador es la ciencia y la tecnología que **permite a las "máquinas" ver**, extraer información de las imágenes digitales, resolver alguna tarea o entender la escena que están visionando.

Actualmente, las aplicaciones de la visión artificial están muy extendidas y van desde el campo de la industria (contar botellas, comprobar defectos en una cadena de montaje, interpretar un TAC médico...) y el campo de la medicina (recuento y búsqueda de células), hasta los sistemas más complejos, que permiten a los robots orientarse en un entorno desconocido, pasando por el reconocimiento de patrones de la realidad aumentada, entre otras muchas aplicaciones.

Hoy en día se utilizan cada vez más las técnicas de visión artificial en el campo del diseño interactivo mediante la interacción con superficies multitáctiles (*multitouch*), la interacción con tangibles (objetos) y el reconocimiento de gestos corporales.

Todos estos ejemplos incorporan técnicas de visión artificial.

De alguna manera, en el contexto en el que estamos, este **conjunto de técnicas** nos permite **diseñar interacciones**, de modo que el usuario utiliza su movimiento o su manipulación de objetos para interactuar con la aplicación.

Objetivos

1. Analizar las diferentes herramientas, procesos y proyectos que tienen que ver con la disciplina de la visión por computadora, aplicada al contexto de la interacción.

2. Entender la metodología de un proyecto de visión por computadora para que, mediante la aplicación de estos conceptos, podáis diseñar vuestras propias interacciones, basadas en visión artificial.

1. Conceptos teóricos

1.1. Luz visible, luz invisible

El ojo humano ve una parte del espectro de toda la luz que ilumina el universo. El rango de luz que podemos ver lo denominaremos *luz visible*.

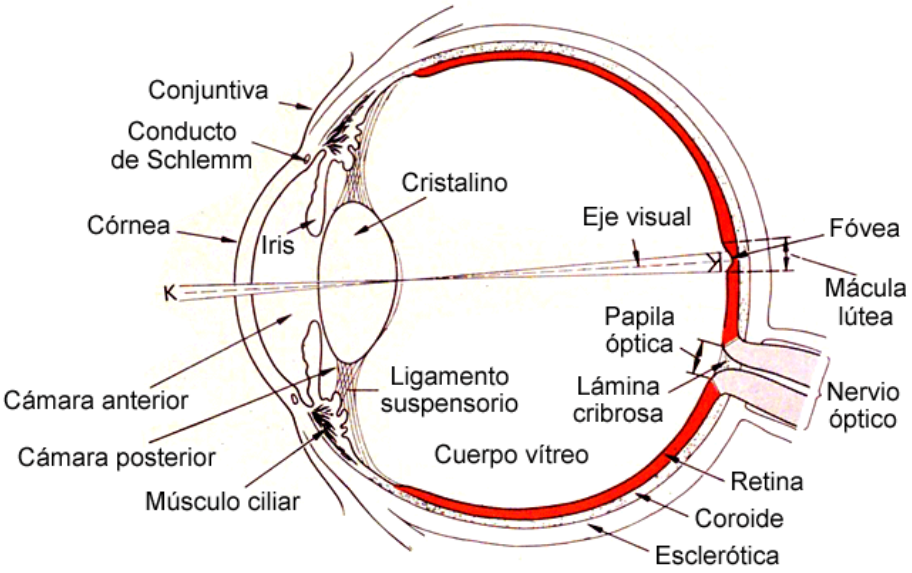
Esto quiere decir que hay frecuencias de luz que no podemos ver pero que existen, como por ejemplo, los infrarrojos y los ultravioletas.

- Los **infrarrojos** son los "colores" no visibles al ojo humano que están por debajo del rojo desde el punto de vista frecuencial.
- Los **ultravioletas** son los "colores" no visibles al ojo humano que están por encima del violeta.

Ved también

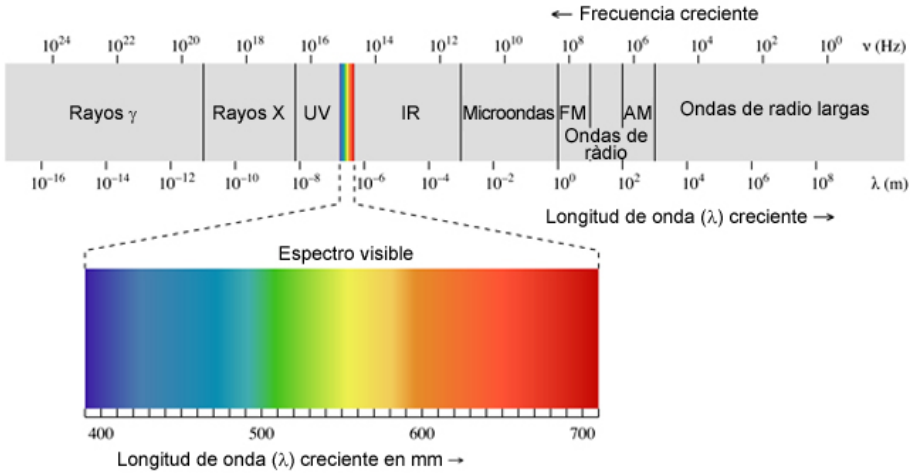
Las imágenes impresas en este documento se pueden consultar en color en el aula virtual de la asignatura en HMTL o PDF.

Esquema de la morfología de un ojo humano



Fuente: Wikipedia

Espectrograma de la luz visible y no visible



Fuente: Wikipedia (http://en.wikipedia.org/wiki/Electromagnetic_radiation).

Color	Longitud de onda
Ultravioleta	< 380 nm
Violeta	380-450 nm
Azul	450-495 nm
Verde	495-570 nm
Amarillo	570-590 nm
Naranja	590-620 nm
Rojo	620-750 nm
Infrarrojo	> 750 nm

Abreviatura: **nm** = nanómetro (unidad de medida del sistema métrico, igual a una billonésima parte de un metro.

El hecho es que la mayoría de sensores de cámaras digitales son sensibles a la luz visible, pero también son sensibles (en diferente medida) a la luz infrarroja y/o a la ultravioleta. Ahora bien, la luz infrarroja no nos es útil para construir una imagen digital, puesto que nos da información de una frecuencia que no podemos ver y que, por lo tanto, no tiene una representación posible en un color.

Por esta razón, la mayoría de cámaras digitales enfocadas a hacer fotografías o fotogramas llevan un filtro anti-IR, o sea antiinfrarojos, para cortar todas las frecuencias por debajo del espectro visible que nos resultarían un ruido innecesario, y solo dejan pasar el rango de luz visible que queremos plasmar con la cámara.

1.2. La imagen en movimiento

Los procesos de visión artificial son posibles gracias a tecnologías basadas en la captura de la imagen (cámaras de vídeo, cámaras web), sumadas a la capacidad de procesamiento de los ordenadores actuales.

En realidad, la tecnología de captación de la imagen se empieza a gestar en el siglo XIX, con la invención de los **daguerrotipos** (o incluso antes, con el descubrimiento del principio de la **cámara oscura**) y la posterior invención de la **fotografía**. Descubrimientos posteriores, como la cronofotografía y otros precursores del cine, acaban desembocando en la invención de las tecnologías de captación de la imagen en movimiento:

> Toda una serie de fotografías disparadas a una gran frecuencia que, reproducidas después a esa misma velocidad, provocan en los espectadores la ilusión del movimiento.

Actualmente, la mayoría de tecnologías de captación de imagen en movimiento (videocámaras) funcionan mediante el uso de sensores electrónicos (CCD y CMOS). Durante la primera década del siglo XXI, se ha estandarizado el uso de la fotografía y el vídeo digitales, lo que permite la grabación de imágenes en alta resolución a un relativo bajo coste y con un elevado número de imágenes por segundo (FPS).

1.3. La imagen digital

1.3.1. Matrices de píxeles

En el mundo digital, las imágenes se representan como una matriz bidimensional de píxeles en la que cada píxel puede adquirir **valores de color codificados** con tres parámetros (**R**: *red*, **G**: *green*, **B**: *blue*).

A pesar de que ello se ha heredado del mundo de la imagen analógica, normalmente trabajaremos con imágenes de proporción 4 × 3 (cuatro unidades de anchura por tres unidades de altura); es el caso de resoluciones estándar como 640 × 480 px, 800 × 600 px y 1.024 × 768 px.

16 × 9

En algunos casos, podemos encontrar imágenes en el formato 16 × 9, puesto que hoy en día se han popularizado las videocámaras de alta definición en formato panorámico.

Frecuentemente, utilizaremos imágenes de baja resolución (entre 160 × 120 y 640 × 480 píxeles) como fuente de análisis de los procesos de visión artificial, ya que en la mayoría de casos no necesitaremos excesivo detalle en las imágenes para efectuar un análisis orientado a la visión por computador.

1.3.2. Bytes, bits y colores

Un píxel normalmente se expresa mediante tres números enteros (R, G, B), que representan los componentes rojo, verde y azul de todo color. Estos valores de R, G y B se suelen expresar en un rango de 8 bits, o sea, de valores entre 0 y 255.

Ejemplo

Por ejemplo, un píxel que tenga unos valores RGB de (255, 0, 0) nos indica un color rojo puro. Un píxel que tenga unos valores RGB (255, 0, 255) nos indica un color producido por una mezcla del rojo puro y el azul puro; en este caso, por lo tanto, obtendremos un color lila intenso.

1.3.3. Frecuencia de imagen (*frame rate*)

La frecuencia de imagen (*frame rate*) hace referencia al número de imágenes por segundo. Es la medida de la frecuencia a la que un reproductor de imágenes muestra diferentes fotogramas (*frames*).

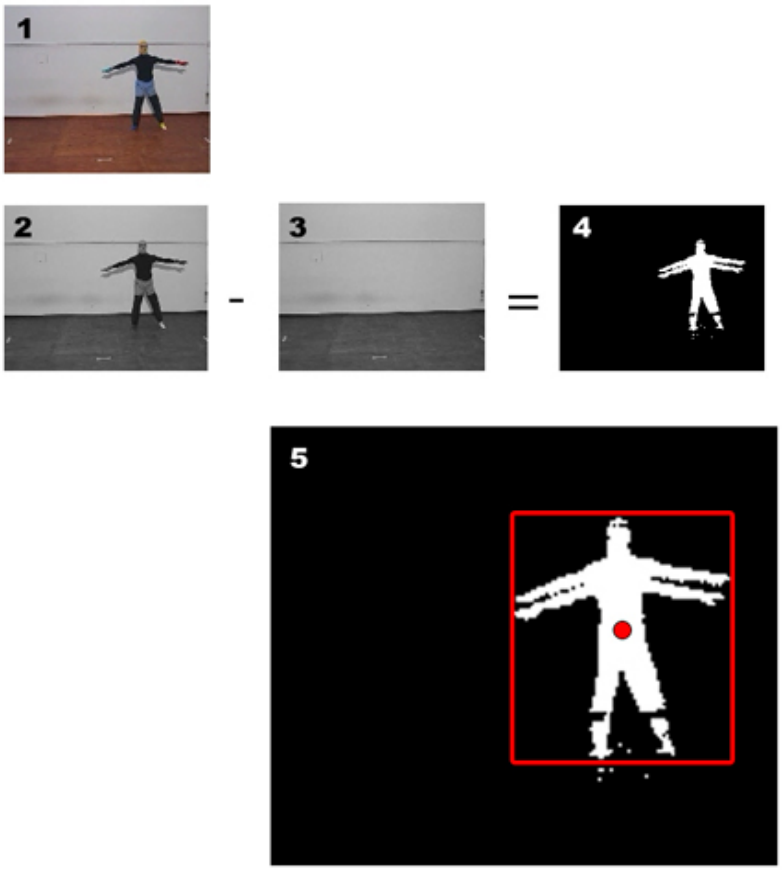
En informática estos fotogramas están constituidos por un número determinado de píxeles que se distribuyen a lo largo de una red de texturas. La frecuencia de los fotogramas es **proporcional** al número de píxeles que se han de generar y que inciden en el rendimiento del ordenador que los reproduce.

La frecuencia de actualización de las imágenes oscila, en el entorno digital, entre los 15 y los 60 FPS (*frames* por segundo). El rango entre 25 y 30 FPS es el más común.

Según nuestras necesidades, deberemos elegir tecnologías de captación de la imagen con una frecuencia de imagen específica. Si queremos analizar y extraer datos de objetos o usuarios que se mueven a grandes velocidades (coches, pájaros, corredores de fútbol...), seguramente necesitaremos un sistema de captación de vídeo con más resolución temporal (una frecuencia de imagen más alta: más fotogramas por segundo).

1.4. Un ejemplo básico

Aunque entraremos en profundidad en la materia en los apartados siguientes, es conveniente hacer un breve análisis de un típico proceso de visión artificial para que, de ahora en adelante, podáis analizar con una cierta perspectiva los conceptos que iremos introduciendo. Fijaos en este ejemplo esquema:



1) Imagen original
2) Imagen procesada en escala de grises
3) Imagen neta del fondo
4) Imagen resultante de la sustracción de fondo y la binarización
5) Extracción de datos (área del *blob* y centro de coordenadas)
Fuente: www.eyesweb.org

En este ejemplo, podéis observar algunos de los procesos típicos de la visión artificial.

A menudo necesitamos conseguir una imagen muy clara y simple del objeto o el usuario del que queremos efectuar un seguimiento. Para no cargar el procesador de los ordenadores, es habitual aplicar los algoritmos de análisis sobre imágenes de color binario (un bit: blanco y negro), de las que se puede extraer el centro de masas del área de píxeles blancos, los contornos...

Para conseguir esta imagen simple y clara, efectuaremos diferentes procesos, como podéis observar en este ejemplo: la sustracción de fondo o la binarización, entre otros.

Una vez conseguida esta imagen binaria final, los algoritmos de visión nos permiten extraer una serie de datos, como veremos a continuación. En este caso particular, se extraen las coordenadas del centro de la masa blanca de píxeles y el área que la rodea (punto y líneas rojas).

2. El espacio y las herramientas

2.1. La cámara

Para empezar a trabajar con la visión artificial, necesitamos un **dispositivo sensible a la luz visible** que nos permita almacenar las imágenes en formato digital. En otras palabras, necesitamos una cámara web, una cámara de vídeo o una capturadora analógica.

Ejemplo de cámara de visión artificial del sector industrial fabricada por Point-Grey
Fuente: http://doc.instantreality.org/

Las cámaras web y la mayoría de cámaras de vídeo se pueden conectar directamente a los ordenadores por medio de los puertos USB, FireWire o Thunderbolt, y podemos capturar sus fotogramas en **tiempo real**. En todo caso, también podemos utilizar una capturadora analógica, que a partir de una señal de vídeo analógico, nos permitirá capturar imágenes y procesarlas.

Ejemplo de cámara web fabricada por Logitech
Fuente: http://doc.instantreality.org/

Una vez que ya tengamos el dispositivo de captura en funcionamiento, hemos de considerar toda una serie de aspectos referentes al entorno en el que se hará la interacción, puesto que la **variación en la iluminación**, la complejidad de la imagen u otros factores pueden dificultar mucho el proceso de visión artificial.

Pensad que el proceso de "enseñar" a un ordenador a tomar decisiones por medio de la "visión" implica muchas dificultades y, por lo tanto, es muy importante trabajar en entornos en los que la luz y el escenario que se quieran analizar sean cuanto más sencillos y estables mejor.

2.2. Iluminación de la escena

La adquisición de imágenes por parte de una cámara varía mucho según la iluminación de la escena. Un cambio lo bastante fuerte en el ambiente lumínico puede hacer que todo el sistema de visión artificial funcione de un modo muy diferente.

Por tanto, siempre que sea posible trabajaremos en entornos en los que el ambiente lumínico sea **cuanto más estable** mejor.

En casos en los que no se pueda conseguir un ambiente estable (como por ejemplo una aplicación al aire libre), deberemos ir adaptando de alguna manera el sistema de adquisición de imágenes a las condiciones cambiantes mediante **algoritmos adaptativos** que permitan analizar periódicamente los cambios de iluminación del entorno y adaptar el sistema de visión a las nuevas condiciones.

Algunas cámaras, como la que lleva el mando de la Wii o muchas del sector industrial, son especialmente sensibles a los infrarrojos, puesto que nos permiten trabajar en un entorno sin luz visible, en la oscuridad, siempre que tengamos alguna fuente de luz infrarroja.

Ejemplo

Por ejemplo, muchos sistemas de videovigilancia incorporan, además de videocámaras sensibles a la luz infrarroja, un anillo de LED que emiten luz infrarroja, invisible al ojo pero visible para la cámara, lo que permite la visibilidad en entornos de oscuridad aparente.

Esta particularidad se aprovecha en muchos casos en el diseño de interfaces interactivas.

Por ejemplo, pongamos el caso de un teatro en el que un actor, a oscuras, describa una trayectoria por el escenario y queramos proyectar en el suelo su recorrido. En principio, con una cámara normal, como estamos a oscuras, no podríamos captar nada. Para poder trabajar en condiciones de oscuridad, lo que haríamos sería iluminar la escena con luz infrarroja, que no es visible pero que sí que es visible para la cámara sensible a los infrarrojos (por ejemplo, una cámara web a la que hubiéramos sacado el filtro anti-IR anteriormente mencionado). Así pues, una cámara de estas características, en una situación como esta, obtiene una imagen muy neta del cuerpo del actor (en blanco) sobre el fondo (negro) del escenario, una imagen perfecta para analizar con visión artificial, puesto que el cuerpo no interfiere con el fondo.

La luz infrarroja se puede generar de varias maneras: hay focos de LED infrarrojos que nos permiten "bombardear" un espacio con luz infrarroja. Otro sistema menos costoso es utilizar una luz de teatro incandescente, a la que pondremos tres filtros: uno rojo, uno verde y uno azul (colores puros). El efecto de poner un filtro rojo hace que la luz salga tintada de rojo, es decir, que eliminemos todos los colores excepto el rojo de la luz filtrada. Si sobre este filtro ponemos un filtro azul, estaremos eliminando todos los colores (incluyendo el rojo que nos había quedado) excepto el azul (que ya no estaba en la luz filtrada porque lo había eliminado el filtro rojo). Y sobre estos dos filtros añadimos el filtro verde, con resultados parecidos. Por lo tanto, hemos eliminado toda la luz visible con los tres filtros, pero no hemos eliminado las frecuencias infrarrojas con los filtros. Lo que hemos hecho ha sido transformar una luz incandescente en una luz infrarroja.

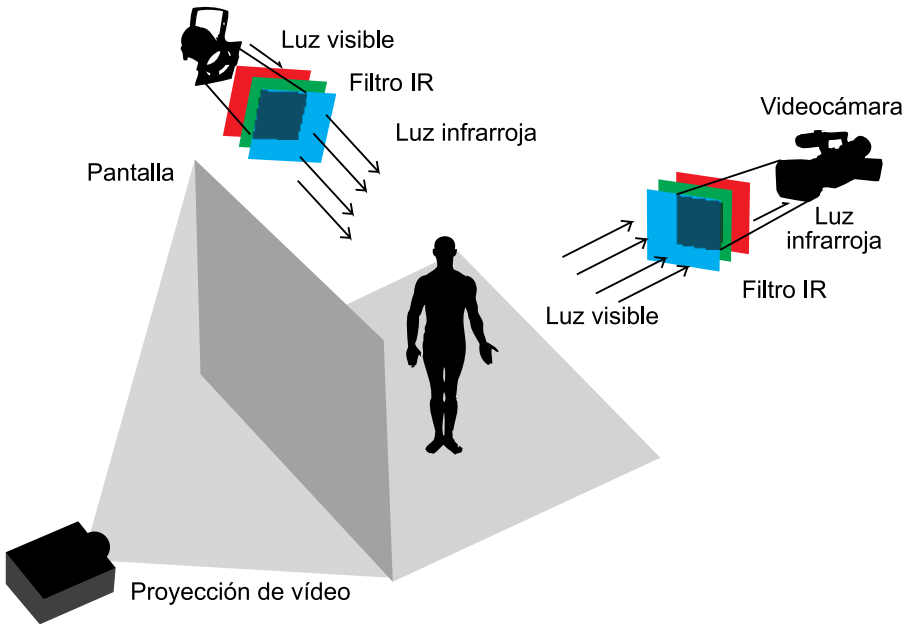
2.3. Proyecciones de vídeo y visibilidad

En un entorno interactivo nos podemos encontrar a menudo con la necesidad de tener en un mismo espacio un sistema de visión artificial y una proyección de vídeo. Como veremos algo más adelante, la complejidad de los algoritmos de visión artificial podría hacer que un sistema tenga que discernir entre una proyección de vídeo interactiva y el usuario que la manipula, algo muy dificultoso si las dos "imágenes" están integradas en un mismo espacio. Como en el apartado anterior, gracias a comprender la respuesta de la luz y sus propiedades, podemos solucionar estas dificultades.

Si no bloqueáramos la entrada de luz visible en la cámara, encontraríamos problemas en el contexto de una interacción con videoproyecciones, puesto que el sistema CV reconocería los elementos proyectados (círculos blancos) como parte integrante de los *blobs* analizados.

La luz proyectada por un proyector, evidentemente, es visible al ojo humano y/o para las cámaras digitales normales. En el caso que planteamos, queremos obtener una imagen del usuario que interactúe con la pantalla de videoproyecciones, pero no de los contenidos de la videoproyección. Para lograrlo, podemos hacer uso de la técnica descrita en el apartado anterior, referente a la luz infrarroja, usando una cámara que "vea" la luz del espectro infrarrojo.

Si a esta cámara le añadimos un filtro que bloquee la luz visible, lo que conseguiremos es que la cámara no sea sensible a la luz visible y solo capte la luz infrarroja. Por lo tanto, esta cámara podrá ver al usuario manipulando el sistema, puesto que su cuerpo hará rebotar la luz infrarroja, pero no podrá ver la proyección (que se encuentra solo dentro del rango de luz visible).



Esquema de la configuración escénica comentada, luz visible filtrada a IR, cámara de vídeo filtrada a IR y retroproyector de vídeo sobre la pantalla de fondo

Reflexión

Como corolario de este ejemplo, podemos concluir que un proyector de vídeo no genera luz infrarroja, puesto que los proyectores llevan un filtro de luz anti-IR, que no permite que "proyecten" luz en el espectro infrarrojo.

2.4. OpenCV

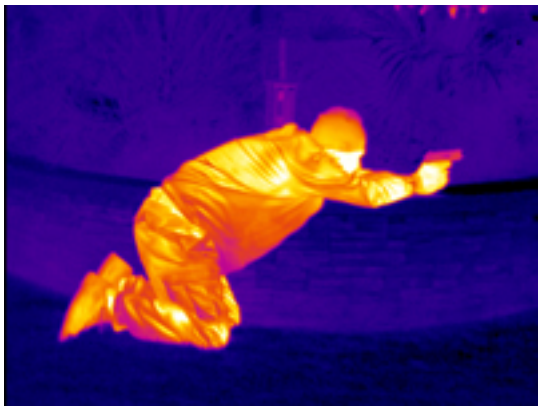
OpenCV u Open Computer Vision es un conjunto de bibliotecas de programación que nos permiten llevar a cabo todo el proceso que hemos visto anteriormente dentro de un ordenador: desde la adquisición de la imagen hasta la extracción de la información.

La gran mayoría de aplicaciones del software libre relacionadas con la visión artificial están basadas en OpenCV. Estas bibliotecas las desarrolló Intel como extensión de las bibliotecas IPL, desde 1999 como proyecto de código abierto, y en el año 2007 se llegó a publicar la versión 1.0. Estas bibliotecas son la base de algoritmos y funciones que finalmente podremos aplicar en nuestras aplicaciones.

2.5. Entornos de programación

Para desarrollar una aplicación que trabaje con visión artificial, hemos de trabajar en un entorno de programación que nos permita integrar las bibliotecas de OpenCV y combinarlas con el resto del sistema interactivo. Aquí dentro podríamos incluir entornos como Processing, openFrameworks, MAX/MSP, Pure Data... Todos estos sistemas de programación nos permiten ejecutar algoritmos y funciones de visión artificial y, a la vez, generar un sistema interactivo que trabaje con gráficos, música, vídeo, etc.

Dentro de los sectores industrial, médico y militar más profesionales, hay una gran cantidad de soluciones y alternativas, tanto de hardware (cámaras térmicas, cámaras lineales, cámaras de alta resolución o alta velocidad) como de software, a OpenCV.



Ejemplo de cámara térmica. La imagen se construye según la temperatura de cada punto
Fuente: www.x20.org

3. Diseñando interacciones: conceptos, algoritmos y funciones. Kinect

Las técnicas de visión artificial siguen normalmente un proceso desde la adquisición de la imagen hasta la toma de decisiones según la información procesada. Lo resumimos a continuación:

3.1. Adquisición de la imagen

Una imagen digital es producida por sensores digitales presentes en cámaras y otros dispositivos digitales que generan una imagen bidimensional (2D), es decir, un conjunto de $N \times M$ **píxeles** o colores o intensidades de un cierto valor que representan el **espacio** que queremos analizar.

En el mundo de la interacción, **podemos utilizar cámaras digitales de diferentes tipos** (cámaras web, DV o USB). En el fondo, cada tipo de aplicación puede necesitar un tipo u otro de cámara según las necesidades.

3.2. Procesamiento de la imagen

Antes de extraer información directamente de la imagen, se acostumbra a hacer un procesamiento previo de la misma para conseguir otra que nos permita hacer el proceso de extracción de datos más sencillo y eficiente.

Ejemplo

Por ejemplo, el procesamiento puede incluir funciones para modificar la luminosidad y el contraste, para reescalar la imagen, los niveles de color, las curvas, la binarización, el desenfoque (*blur*), etc.

Como ejemplo, el hecho de aplicar un desenfoque en una imagen nos permite, de alguna manera, "redondear" los contornos de las formas que aparecen en ella, lo nos puede ser útil en diversas ocasiones. Modificar la luminosidad para tener una imagen más contrastada nos puede ayudar a dar más importancia visual a la parte que queremos analizar. Reescalar y reencuadrar la imagen nos permite centrar el proceso de visión artificial solo sobre la porción de la imagen que nos interesa realmente, ahorrándonos así un procesamiento innecesario.

A continuación, analizaremos algunos algoritmos de procesamiento de la imagen usados frecuentemente en la visión artificial.

Esta será la imagen de ejemplo original que utilizaremos para explicar diferentes funciones del procesamiento de la imagen. Tiene una resolución de 800 píxeles horizontales por 800 píxeles verticales.

Imagen original
Fuente: www.opencv.org

3.2.1. Escala de grises

En muchos casos, el color de la imagen no nos aportará ninguna información relevante para interactuar. Por lo tanto, en estas situaciones se acostumbra a descartar la información de color de la imagen transformándola en una **imagen de tonos de grises**.

De este modo, "ahorraremos" mucha información y los cálculos serán mucho más sencillos de computar. Por ejemplo, si estamos analizando el movimiento dentro de una imagen en color, nos podemos ahorrar aproximadamente 2/3 partes de los píxeles que se han de tratar si descartamos los componentes de color y trabajamos simplemente con la imagen de intensidad luminosa o niveles de grises.

Transformamos la imagen de prueba de color (R, G, B) a tonalidad de grises, es decir, solo con una dimensión de color (L) de luminancia.

Por ejemplo, la imagen original ocupaba en la memoria, en número de bytes:

800 píxeles × 800 píxeles × 3 bytes (R, G, B) = **1.920.000 bytes**.

En cambio, la imagen transformada en blanco y negro ocupará, en número de bytes:

800 píxeles × 800 píxeles × 1 byte (L) = **640.000 bytes**.

3.2.2. Binarización por umbral (*threshold binarization*)

El proceso de binarización por umbral parte de una imagen en tonos de grises y, a partir de un valor definible de umbral de intensidad de luz llamado *threshold,* la imagen se transforma en una imagen binaria, es decir, con píxeles blancos o píxeles negros.

Si el píxel analizado tenía un valor de intensidad inferior al umbral, quedará en negro, y si su intensidad era más elevada que el umbral, quedará en blanco. De alguna manera, esto nos permite descartar los tonos medios grises y facilita mucho la computación de ciertos algoritmos, puesto que otra vez hemos transformado la imagen en algo mucho más ligero y fácil de procesar.

Transformamos la imagen de prueba de color (R, G, B) a blanco y negro, es decir, solo con dos colores: blanco absoluto y negro absoluto.

Tal como hemos visto, la imagen original pesaba 1.920.000 bytes (1 byte = 8 bits); por lo tanto, pesaba 15.360.000 bits.

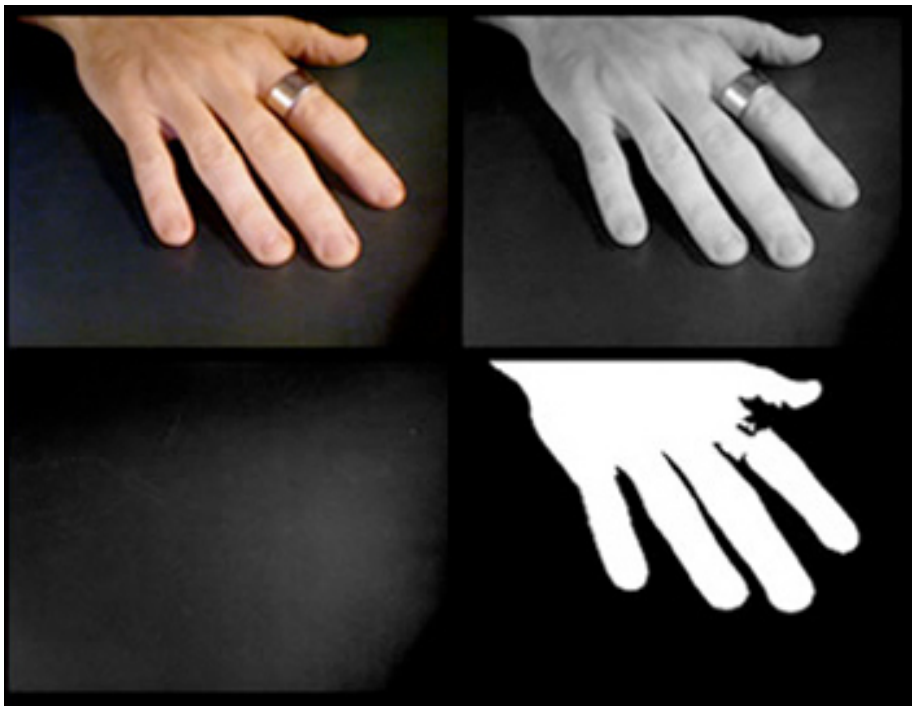
En cambio, la imagen transformada con umbral de blanco y negro ocupará, en número de bits:

800 píxeles × 800 píxeles × 1 bit = 640.000 bits, es decir, 24 veces menos en número de bits que el original.

Velocidad y resolución de la imagen

Muchos de los procesos de visión artificial se basan en trabajar por cada píxel de la imagen internamente; por lo tanto, si con la binarización conseguimos 24 veces menos de

bits que procesar, los algoritmos correrán mucho más rápido. Así pues, un factor determinante en la velocidad de los cálculos que se generan en las aplicaciones de visión artificial es la resolución de la imagen en píxeles. Una imagen de 320 × 240 píxeles será procesada mucho más rápidamente que una imagen de 1.024 × 768 píxeles.



Fuente: www.openframeworks.cc

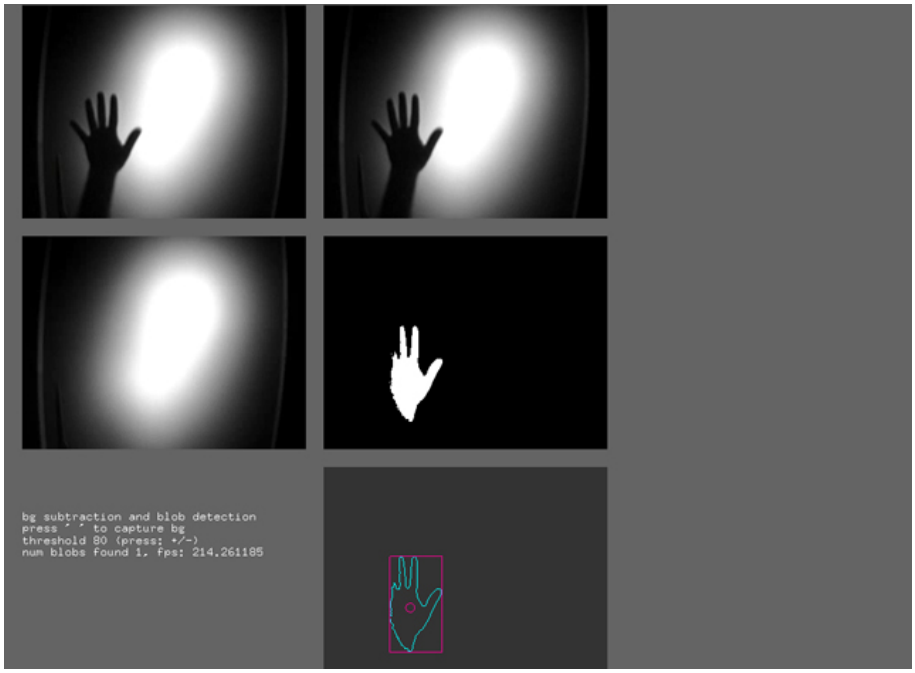
En esta imagen, podemos ver las tres representaciones hasta ahora mencionadas. Arriba a la izquierda, tenemos la representación en color del fotograma capturado por la cámara; arriba a la derecha, la transformación a blanco y negro de la misma imagen, y abajo a la derecha, la representación binaria de la imagen en blanco y negro. El umbral en esta binarización se ha fijado en un valor que permite distinguir fácilmente la mano (blanca) del fondo (negro).

3.2.3. Sustracción de fondo *o background substraction*

La sustracción de fondo o *background substraction* es una técnica de procesamiento de la imagen que permite "restar" una imagen del fondo de la escena con el fotograma actual y obtener, pues, el fotograma actual "menos" el fondo. De este modo, podemos aislar los objetos que han variado respecto de la imagen del fondo de una escena y determinar si ha habido movimiento.

Reflexión

Hay que tener en cuenta que esta técnica requiere poder "capturar" la imagen de "fondo". Su utilidad es en entornos relativamente controlados, en los que podamos conocer cómo será el fondo durante todo el procesamiento o bien en los que nos podamos ir adaptando.



Fuente: www.openframeworks.cc

En esta imagen vemos todo el proceso. La imagen de arriba a la izquierda es lo que "ve" la cámara, y la de abajo a la izquierda es la imagen del fondo o *background* que se ha de restar. Una vez hecha la resta y binarizada la imagen, vemos lo que se ve en la imagen de en medio a la derecha, en la que prácticamente solo aparece la mano en una imagen binaria (con píxeles 100% blancos y 100% negros). Esta última imagen binarizada y restada del fondo ya nos deja casi la silueta perfecta de la mano para aplicar el paso siguiente, que es identificar la "mancha" o *blob* de la mano para obtener cierta información, tal como veremos en el apartado siguiente.

3.2.4. Otras operaciones morfológicas

En muchas situaciones, una vez que hemos obtenido una imagen binaria (en blanco y negro), observamos que aparece **ruido** en la imagen, fruto de los cambios en la iluminación de ambiente y los pequeños reflejos de esta luz en los diferentes objetos y cuerpos en la escena. Este ruido suele consistir simplemente en píxeles que aparecen caóticamente en la imagen y que pueden estorbar seriamente la capacidad de los algoritmos de visión para reconocer *blobs* y patrones gráficos.

Para eliminar estos pequeños y molestos píxeles, hay una serie de algoritmos de "limpieza" de la imagen que nos ayudan a cumplir la tarea de obtención de una imagen binaria lo más neta posible.

1) Mediana (*median*)

El algoritmo de la mediana se aplica sobre las imágenes en escala de grises y provoca una cierta suavización de la imagen útil a la hora de simplificar contornos y áreas irregulares.

El algoritmo de la mediana divide la imagen en un conjunto de divisiones de radio definible y calcula el valor de luminosidad medio de cada una de las subdivisiones. Una vez calculado, sustituye los píxeles que hay dentro de cada división por una única mancha gris con el valor de luminosidad medio de los píxeles originales.

1. Imagen original

2. Aplicación de la media

2) Erosionar/dilatar (*Erode/dilate*)

Los algoritmos de erosión y dilatación se suelen aplicar en serie. Sobre la imagen "ruidosa" se aplica el algoritmo de erosión, que contrae los contornos de todas las áreas blancas un número determinado de píxeles y que elimina completamente las áreas de píxeles blancos más pequeñas e irrelevantes.

Una vez aplicado el algoritmo de erosión, se aplica el algoritmo de dilatación, que ayuda a recuperar la medida original de las áreas importantes y que expande los contornos de las áreas blancas tantos píxeles como sea necesario.

1. Imagen ruidosa

2. Imagen erosionada

3. Imagen dilatada

3.3. Extracción de datos

Una vez que tengamos la imagen preparada, aplicaremos una serie de algoritmos para extraer los datos de la misma. Podemos reconocer en la imagen, por ejemplo, las líneas, los círculos, las manchas, las esquinas, el movimiento, un determinado color..., o bien ciertos patrones de imagen previamente determinados.

Una vez reconocido en la imagen lo que queríamos extraer (sea por cada línea, círculo o mancha reconocidos) de cada elemento, podremos averiguar su posición, velocidad de movimiento, medida o color dominante.

A continuación, analizaremos una serie de algoritmos frecuentemente usados para la extracción de datos en el contexto de proyectos relacionados con visión artificial:

3.3.1. Reconocimiento de regiones o *blob detection*

Llamamos *blobs* a los puntos o las regiones de píxeles más luminosos (o más oscuros) que los de los alrededores y que, por lo tanto, forman una "mancha" aislada del resto de la imagen.

Las herramientas de **reconocimiento de regiones** o ***blob detection*** nos permiten analizar las "manchas" de una imagen binaria (blanca y negra) y extraer información de cada una, como la medida de la mancha, el centro de masas de la mancha, su contorno...

Este tipo de funciones son útiles en muchos casos, por ejemplo, para contar y localizar cuántos objetos tenemos en una escena. A partir de que los hayamos localizado, podremos seguir su trayectoria o analizar su movimiento. Por lo tanto, esta técnica se utiliza en muchos procesos interactivos.

Detección de *blobs*

Tened en cuenta que la detección de *blobs* requiere normalmente, de entrada, una imagen ya binarizada con las funciones de procesamiento de la imagen que acabamos de ver.

3.3.2. *Frame difference*: seguimiento del movimiento o *movement tracking*

Esta técnica nos permite obtener datos del movimiento que hay en la escena que estamos analizando, tanto de la cantidad de movimiento como de su dirección.

La técnica consiste básicamente en restar un fotograma capturado por la cámara con el anterior. La resta nos da una referencia de lo que ha cambiado entre las dos imágenes. Normalmente aplicaremos esta técnica después de haber hecho la sustracción de fondo, es decir, que habremos eliminado mucha información del fondo.

En caso de que no haya cambiado nada entre un fotograma y el anterior, la resta nos dará una imagen prácticamente negra.

Si, por ejemplo, en la cámara estamos viendo un objeto en movimiento, la resta entre dos fotogramas consecutivos nos dará las partes de la imagen que han cambiado. Si la imagen ha cambiado mucho (porque el objeto se desplazaba rápidamente), la resta será una mancha bastante grande; si la imagen ha variado muy poco, la operación de resta nos dará una mancha muy pequeña.

Para acabar de extraer toda la información, lo que haremos es una detección de *blobs*. Como resultado, tendremos, por ejemplo, la medida de la mancha del *blob,* que será una medida de la cantidad de movimiento. O bien, si comparamos los centros de masas de la mancha o *blob* en dos fotogramas consecutivos, podremos extraer una medida de la dirección del movimiento.

3.3.3. Seguimiento de color o *color tracking*

El seguimiento de color o *color tracking* nos permite hacer el seguimiento de un área de píxeles de un determinado color (es decir, de un valor específico RGB).

A pesar de que esta técnica no se incluye en la implementación clásica de las bibliotecas OpenCV, es bastante popular. De hecho, se trata más bien de la suma de un conjunto de técnicas, que implican la extracción selectiva de canales de color, para aislar el color seleccionado y obtener una imagen binaria en la que solo el color del que se ha hecho el seguimiento aparece como un *blob* o una mancha blanca de píxeles.

Generalmente, cuando hacemos un seguimiento de color, seleccionamos el color que queremos recorrer y definimos la medida del área alrededor de este píxel. El algoritmo tendrá que ir reescaneando permanentemente esta área para no perder el seguimiento en caso de que el píxel original cambie de color. Cuanto mayor sea esta área de seguridad, más dificultades tendrá el ordenador para hacer todos los cálculos, puesto que se trata de una operación con gran demanda de capacidad de cálculo por parte del procesador.

Gran popularidad del *color tracking*

El seguimiento de color es muy popular también en entornos de software más allá de las aplicaciones de interactividad en tiempo real. Se usa de manera intensiva en muchos programas de tratamiento profesional de vídeo, para recorrer un punto de una grabación de vídeo, y para añadir efectos de postproducción, estabilizar los movimientos involuntarios de la cámara, etc.

3.3.4. Buscador de caras o *face tracking*

Hay toda una serie de técnicas enfocadas al reconocimiento de patrones o formas concretas en una imagen. Estas técnicas nos permiten definir qué patrones o imágenes buscaremos dentro de la escena. Así es como funcionan la mayoría de técnicas de buscador de caras.

Esta función detecta si hay "caras humanas" dentro de la imagen y nos permite extraer cierta información de la cara (como su posición y su medida) y algunos aspectos morfológicos (que son la base para poder identificar las caras de diferentes usuarios).

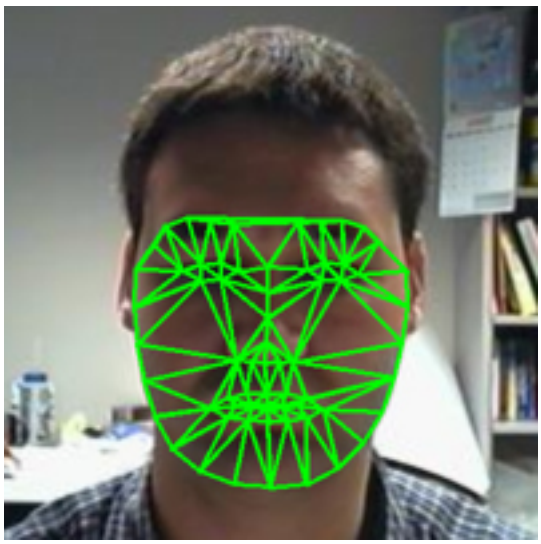
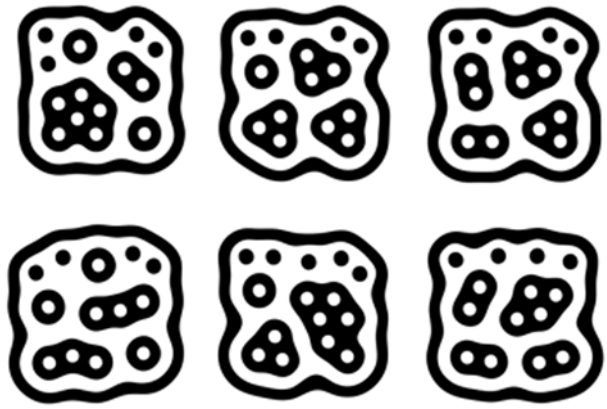


Imagen de ejemplo de buscador de caras en el proyecto AAM Fitting Algorithms
Fuente: Carnegie Mellon University. Robotics Institute

3.3.5. Buscador de características o *feature tracking*

A continuación, denominaremos bajo un mismo nombre toda una serie de técnicas basadas en la extracción de características o *features* de una imagen, que básicamente son puntos de la imagen fácilmente reconocibles por ciertos algoritmos (detección de esquinas, por ejemplo).

Otra aplicación de estas técnicas permite localizar los **fiduciales** de una escena. Los fiduciales serían patrones especialmente diseñados para ser reconocidos, como por ejemplo los fiduciales que utiliza el instrumento interactivo Reactable, desarrollado por el equipo de Sergi Jordà (MTG-UPF).



Marcadores fiduciales del sistema Reactivision, utilizados en Reactable
Fuente: reactivision.sourceforge.net

3.3.6. Realidad aumentada

Como habéis ido viendo, vamos acumulando unas técnicas tras otras y vamos llevando a cabo acciones cada vez más complejas. Del anterior concepto del buscador de características o *feature tracking* se desprende en cierto modo la técnica de la realidad aumentada o *augmented reality*.

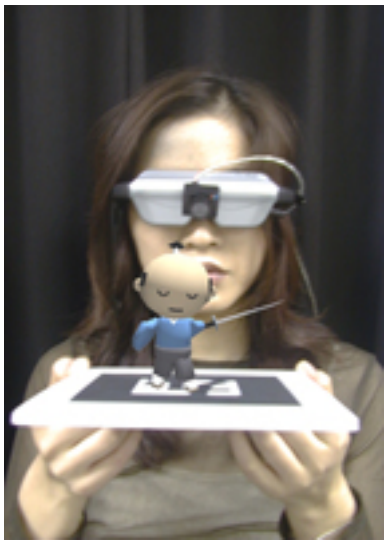
Por *realidad aumentada* entendemos todas las técnicas que de alguna manera integran una imagen virtual dentro de una imagen de un entorno real y, evidentemente, en tiempo real y de manera interactiva.

Una de las bibliotecas más utilizadas se denomina **ARToolKit**, que permite el trabajo desde muchos entornos de programación creativa.

Ejemplo de marcador de un sistema de realidad aumentada

Una de las aplicaciones más conocidas es la capacidad de introducir un elemento en 3D interactivo dentro de la imagen capturada por una cámara. Esta integración es coherente con el punto de vista de la cámara, y nos da la impresión de que los objetos en 3D están presentes en el entorno visto por esta. Evidentemente, esta técnica requiere una cámara para tener sentido, y siempre veremos el mundo en 3D integrado en la visión de la cámara en un monitor.

Esta imagen es un marcador o *marker* para un sistema de realidad aumentada. Este simple patrón es analizado por el sistema de visión artificial y de él podemos extraer la posición y la orientación relativas a la cámara. A partir de aquí, solo nos resta vincular un mundo virtual en 3D en relación con esta marca, copiando su perspectiva y su orientación.



Típica aplicación de la realidad aumentada. Se genera una escena en 3D en relación con la posición del marcador
Fuente: artoolkit.sourceforge.net

Aquí podemos ver una aplicación muy sencilla de realidad aumentada: el usuario tiene en sus manos un papel con un marcador impreso, el sistema de visión reconoce el marcador y su orientación, y genera una figura en 3D sobre este, coherente con su posición, su rotación y su inclinación.

3.4. Kinect

Cuando apareció en el mercado la cámara Kinect de Microsoft, se generalizó la posibilidad de trabajar en el campo de la visión volumétrica o tridimensional.

> Este tipo de cámaras volumétricas nos permiten adquirir una imagen plana en color y una imagen de profundidad, es decir, nos permiten saber a qué distancia (o profundidad) de la cámara se encuentran cada uno de los píxeles de la imagen.

Además, si dentro de la imagen aparece la figura de un cuerpo humano, la propia cámara es capaz de reconocer la postura de dicha figura y de enviarnos a nuestro software la posición y la orientación de tronco, cabeza, brazos y piernas.

Al cabo de pocos días de su aparición en el mercado, surgieron personas que habían conseguido desencriptar el protocolo USB de la Kinect usando métodos de ingeniería inversa y que lo publicaron en Internet. En pocas semanas,

una gran cantidad de entornos de programación interactiva permitían interactuar con la Kinect: Processing, openFrameworks, Quartz Composer, Max/MSP, EyesWeb, Cinder, vvvv, Flash... Así pues, un producto pensado para el mercado de las consolas XBox se empezó a utilizar extensamente en el mundo del diseño interactivo.

Fotografía infrarroja en la que se muestra la nube de puntos infrarrojos que emite la Kinect para obtener la profundidad de la imagen. Fuente: www.mattcutts.com

El hecho de que la cámara Kinect sea un producto comercial masivo permite que su precio sea muy asequible en relación con las prestaciones que nos da en un sistema de visión artificial. Poder disponer, por un lado, de una imagen con profundidad y, por otro, del reconocimiento de figuras humanas nos facilita muchísimo el trabajo de interacción basado en la visión.

Por ejemplo, con una imagen con profundidad de un cuerpo, podemos pedir a la aplicación que solo trabaje sobre los píxeles que hay entre 1,50 y 1,30 m de distancia de la cámara. Si, por ejemplo, el usuario está a un metro de la cámara y mueve las manos hacia delante hasta que entren dentro del rango de 1,50 a 1,30 m, el sistema podrá trabajar con una imagen en la que solo aparecen dibujadas las manos; por lo tanto, las tendremos fácilmente aisladas para poder detectarlas con un *blob-tracking* y extraer su posición.

Es decir, con unas técnicas muy sencillas podemos extraer la posición de las manos con mucha precisión. Si no tuviéramos la imagen de profundidad, nos sería mucho más difícil conseguir el mismo resultado, por no decir imposible.

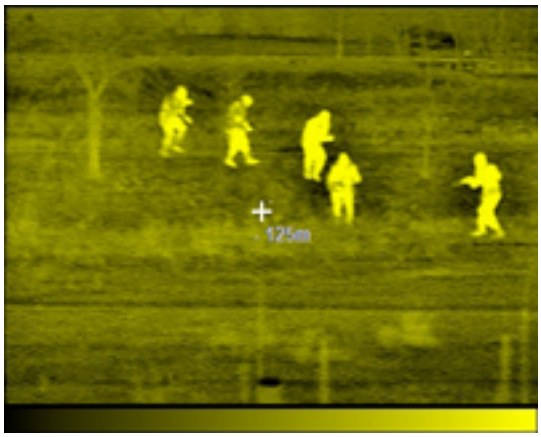
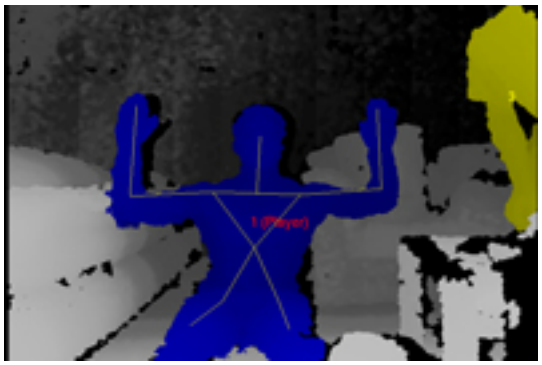


Imagen de profundidad obtenida con una Kinect. Las zonas más claras representan más proximidad a la cámara; las zonas más oscuras representan más alejamiento de la cámara. Fijaos que la parte del cuello está fuera del rango de profundidad y que, de alguna manera, no existe en esta imagen.
Fuente: www.lawriecape.co.uk

Otra característica impresionante de la Kinect es su capacidad para reconocer las posturas y la orientación de una figura humana ante la cámara. El propio hardware de la Kinect nos dice cuántas personas detecta en la imagen y cuál es la postura de cada una de ellas. Los datos que expresan una postura normalmente son los ángulos de las articulaciones. La Kinect divide el cuerpo en tres dimensiones (cabeza, antebrazo, brazo, tronco superior e inferior, cabeza, muslos y piernas). De esta capacidad han surgido multitud de técnicas para interactuar con los usuarios; por ejemplo, generando un modelo en 3D en tiempo real que se mueve tal como se mueve el usuario.



En esta imagen vemos cómo la Kinect pinta una estructura de esqueleto simplificada que copia la posición tridimensional del usuario ante la cámara.
Fuente: weblog.200ok.com.au

3.5. Diseño de interacciones

Una vez que ya hemos extraído los datos de la imagen, hemos de utilizar esta información para tomar decisiones dentro de nuestro sistema.

Esta es la etapa del proceso en la que, a partir de los datos extraídos, podemos tomar decisiones y acciones relacionadas.

Ejemplo

Por ejemplo, si el usuario ha movido los brazos, activaremos una secuencia determinada de operaciones, o si la pieza roja está muy cerca de la azul, haremos sonar una música.

En general, obtendremos unos datos numéricos (coordenadas del centro de masas, cantidades de píxeles de un determinado color, velocidad de movimiento...) que tendremos que transformar y que usar como control de todo tipo de acontecimientos. Obviamente, en esta parte hemos de ser creativos y analizar cuáles son los mapeos más eficientes para diseñar una interacción funcional y comprensible.

4. Más allá. Recursos y bibliografía específica

4.1. Referencias

Computer visions:
http://en.wikipedia.org/wiki/Computer_vision

Processing Tutorials: Getting Started with Video Processing via OpenCV:
http://createdigitalmotion.com/2009/02/
processing-tutorials-getting-started-with-video-processing-via-opencv/

OpenCV Motion Tracking, Face Recognition with Processing: I'm Forever Popping Bubbles:
http://createdigitalmotion.com/2009/02/opencv-
motion-tracking-face-recognition-with-processing-im-
forever-popping-bubbles/

Processing OpenCV Tutorial 1:
http://andybest.net/2009/02/processing-opencv-tutorial-1/

Introduction to programming with OpenCV:http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html

The Five Reactive Books:
http://www.youtube.com/watch?v=nA_UTUvC4h8

Project3 – New Interactions with Kinect and Computer Vision:
http://golancourses.net/2011spring/projects/project-3-interaction/

Microsoft explica cómo funciona Kinect:
http://materiageek.com/2011/03/microsoft-explica-como-funciona-kinect/

4.2. Bibliografía

Programming Interactivity. A Designer's Guide to Processing, Arduino, and openFramework:
http://www.amazon.com/Programming-Interactivity-Designers-Processing-Openframeworks/dp/0596154143

Comunicación y tratamiento de datos

Santiago Vilanova Ángeles

PID_00190488

Universitat Oberta de Catalunya

Índice

Introducción

Tal como veíamos al inicio de esta documentación, la secuencia clásica ENTRADA - PROCESO - SALIDA gobierna todos los procesos de diseño de interacción. Desde los sensores de captura (entrada), hasta el procesamiento de los datos provenientes de estos sensores (proceso) y la aplicación de estos datos como parámetro de control de algún sistema (salida), hasta ahora hemos ido reproduciendo esta secuencia a lo largo de toda la documentación.

Estos tres procesos (entrada, proceso y salida) necesitan algún tipo de canal de comunicación entre ellos, que se materializa físicamente, dependiendo de nuestro diseño, en forma de cableado, sistema de comunicación inalámbrica, etc.

Los datos que fluyen por estos canales físicos se codifican, según nuestras necesidades, siguiendo diferentes protocolos o lenguajes (audio, vídeo, TCP, MIDI, OSC, TTL, Serial, etc.); posteriormente, en cada parte del proceso deberán ser codificados y descodificados para que se puedan manipular y usar con el objetivo de crear sistemas interactivos.

Es importante conocer las particularidades de estos protocolos y sistemas de comunicación para adquirir agilidad en la práctica del diseño interactivo.

Objetivos

1. Analizar los sistemas de comunicación y protocolos habituales en el diseño de interacción.

2. Comprender los distintos métodos de transmisión de datos.

3. Aportar al alumno la capacidad de trabajo mediante distintos protocolos de comunicación.

4. Aportar métodos de tratamiento de datos útiles en el contexto del diseño interactivo.

1. Conceptos teóricos

1.1. Métodos de manipulación de datos: normalización numérica

A menudo encontramos que los datos provenientes de los periféricos de entrada están formateados en un rango numérico específico. Por ejemplo:

- en el caso del movimiento *XY* del ratón, estos valores estarán restringidos al número de píxeles de la pantalla (por ejemplo, 1.024 × 768);
- en el caso de un sistema de sensores en Arduino, el rango de entrada habitual es de 0 a 1.024, y
- en el caso de datos MIDI, hablaremos de un rango efectivo de valores entre 0 y 127.

Como vemos, se trata de rangos de valores numéricos muy diferentes entre sí y que en la mayoría de casos necesitaremos transformar para que se adapten a nuestras necesidades. Para facilitar la manipulación de estos rangos numéricos, usaremos un método muy extendido entre los programadores que se denomina ***normalización numérica***.

La normalización trata básicamente de uniformizar todos los rangos de números a un único rango de valores entre 0 y 1 (o entre –1 y 1) de números flotantes (es decir, con decimales: 0,23, 0,005...).

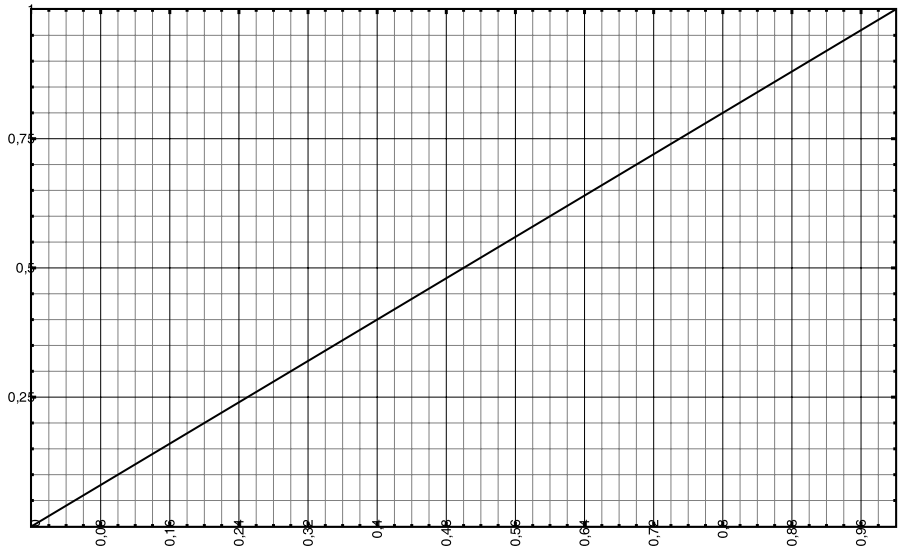
Las características particulares de este rango específico entre 0 y 1 permiten una gran maleabilidad numérica y facilitan procesos de transformación numérica, como la inversión, la exponenciación, la cuantización, etc.

Para normalizar un rango de números, dividiremos el rango original por el valor máximo del rango.

Pongamos por caso la serie de datos MIDI CC 115, 114, 112, 110, 107 normalizados (es decir, divididos por 127, que es el número máximo que puede adquirir un valor MIDI CC); la serie quedaría así: 0,905, 0,897, 0,881, 0,866, 0,842...

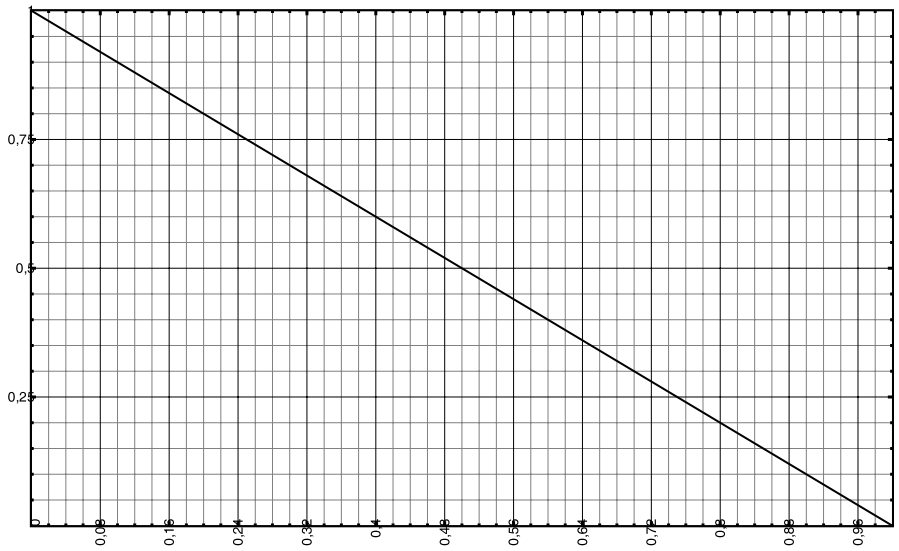
Una vez que disponemos de rangos numéricos normalizados, resulta muy sencillo manipularlos mediante operaciones matemáticas simples que nos permitan transformaciones radicales y útiles en muchos casos.

Si partimos del valor *x* normalizado, es decir, dentro del rango entre 0 y 1:

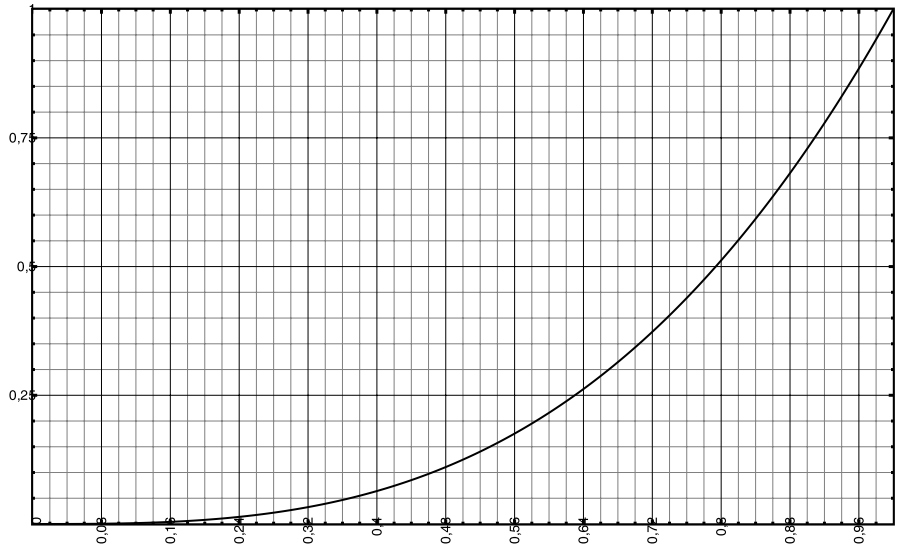


Rango original: $y = x$

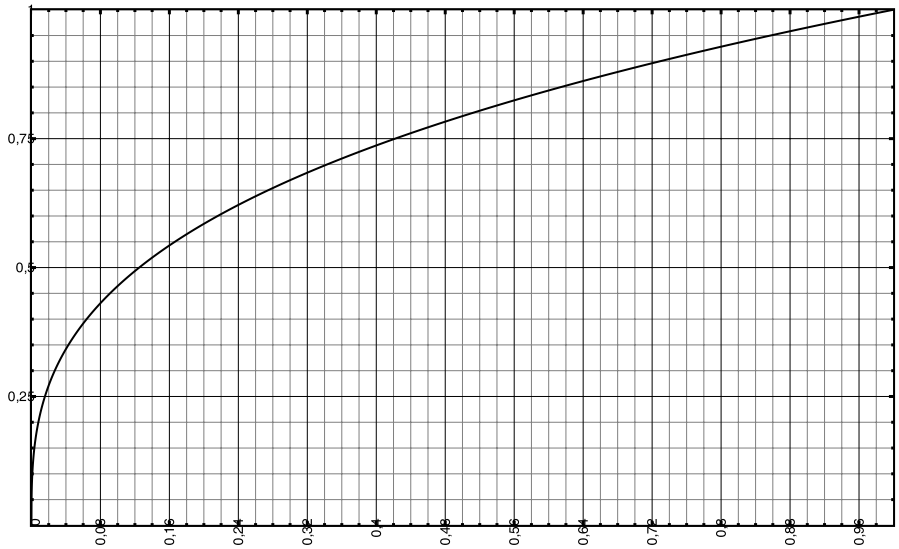
podemos obtener:



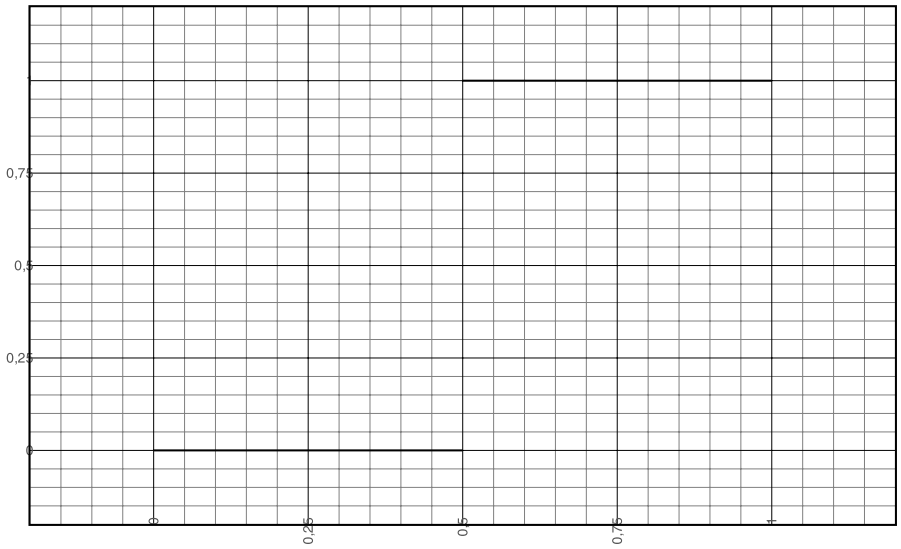
Rango invertido: $y = 1 - x$



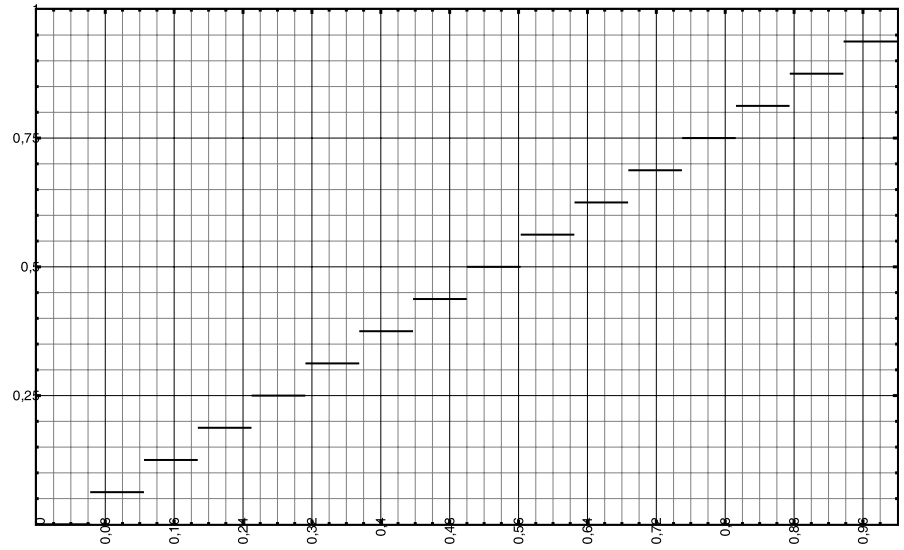
Rango exponencial: $y = x^3$



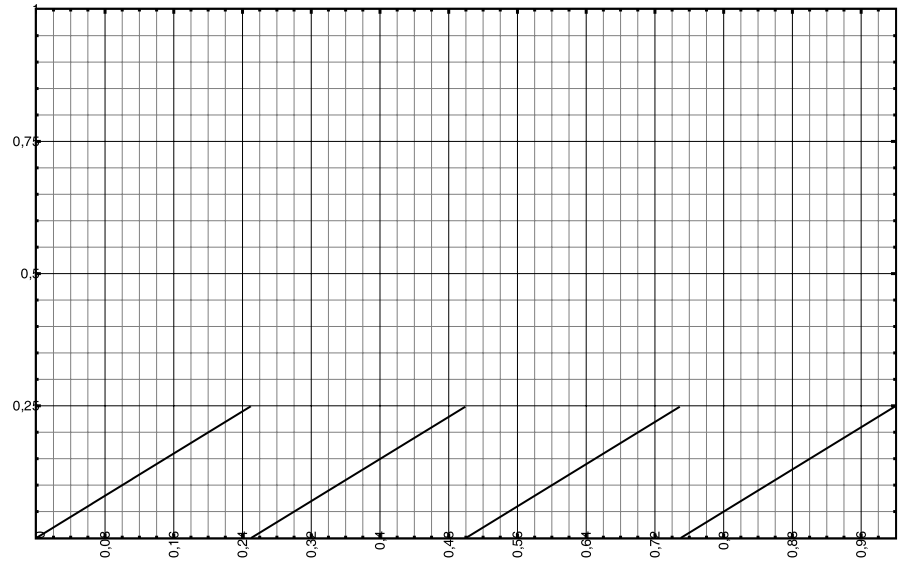
Rango logarítmico: $y = \sqrt[3]{x}$



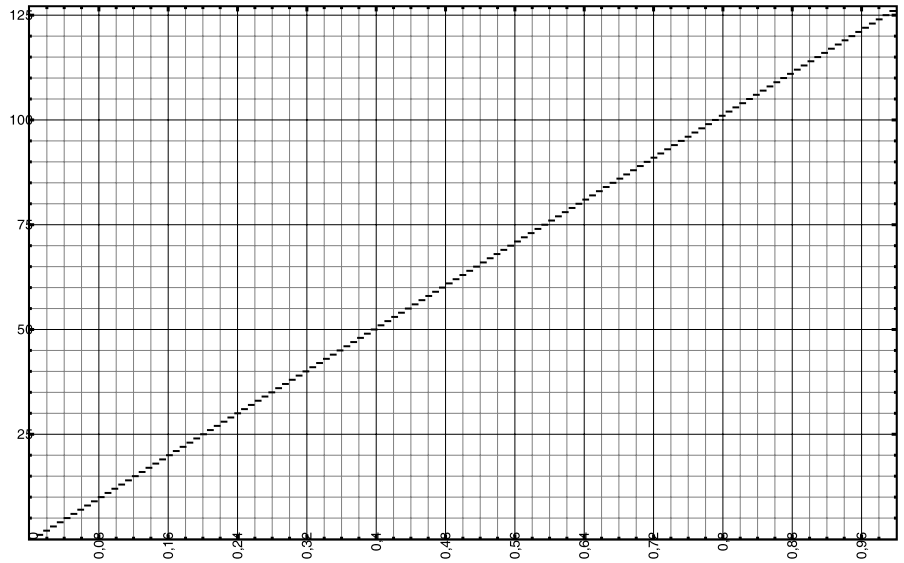
Rango binarizado: *y* = int(*x*)



Rango cuantizado en 16 pasos: $y = \text{int}(x \times 16) / 16$



Rango modulado: $y = x \,\%0{,}25$



Rango reescalado a 0-127: $y = \text{int}(x \times 127)$

Este tipo de transformaciones son muy útiles cuando queremos, por ejemplo, invertir los valores de lectura de una célula fotosensible, reescalar los rangos de valores, binarizar los valores provenientes de un potenciómetro o cuantizar en un número pequeño de pasos los datos obtenidos mediante un termómetro (por ejemplo, en décimas de grado más que en centésimas).

2. Las herramientas

2.1. Periféricos de entrada

A lo largo de este documento, hemos repasado diferentes sistemas de entrada de datos que permiten a los usuarios la interacción con un sistema digital.

Como hemos visto, los periféricos de entrada se encargan de recopilar datos provenientes del usuario o el entorno y de usarlos como desencadenantes de las interacciones diseñadas por el programador. Una lista resumida de estos dispositivos de entrada es la siguiente:

- Ratón
- Teclado
- Videocámara
- Micrófono
- Sensores electrónicos
- Palanca de control (*joystick*)
- Mando de juego (*gamepad*)
- Teclado MIDI

2.2. Unidades de procesamiento

Actualmente, disponemos de múltiples dispositivos que permiten el procesamiento de datos. Estos dispositivos tienen capacidad de entrada de datos por periféricos de entrada, y de salida de datos por periféricos de salida. Son, por lo tanto, los "cerebros" que podemos programar para ejecuten nuestros diseños de interacción:

- Teléfonos inteligentes (*smartphones*)
- Ordenadores
- Asistentes digitales personales (PDA)
- Microchips de sistemas electrónicos empotrados

2.3. Periféricos de salida

Cuando diseñamos una interacción, necesitamos algún periférico de salida que lleve a cabo las acciones consecuencia de la interacción del usuario. Entre estos periféricos de salida, podemos mencionar los siguientes:

- Monitor
- Videoproyector
- Impresora

- Sistema de audio
- Sistema mecánico
- Sistema electrónico

2.4. Contenidos de salida

Valdría la pena hacer un pequeño resumen de las posibilidades creativas de los contenidos que pueden alojar los periféricos de salida. Para no extendernos demasiado, podemos resumir algunas de las estrategias de creación de contenidos interactivos en la lista siguiente (sin considerarla una lista cerrada ni definitiva):

- **Sonido interactivo**. Sistemas musicales interactivos en los que los parámetros sonoros se modifican en tiempo real en función del comportamiento de los usuarios o de los datos contextuales recogidos por los periféricos de entrada.

- **Gráficos generativos**. Sistemas dinámicos de generación de gráficos o de visualización de datos que cambian la morfología en función del comportamiento de los usuarios o de los datos contextuales recogidos por los periféricos de entrada.

- **Vídeo**. Contenidos de vídeo digital "disparados" por las acciones diseñadas en el sistema interactivo en función del comportamiento de los usuarios o de los datos contextuales recogidos por los periféricos de entrada.

- **Texto**. Contenidos textuales que varían en función del comportamiento de los usuarios o de los datos contextuales recogidos por los periféricos de entrada.

- **Cinética y mecánica**. Creación de movimiento mecánico (electrónico o químico) en función del comportamiento de los usuarios o de los datos contextuales recogidos por los periféricos de entrada.

- **Iluminación**. Generación o disparo de efectos de iluminación en función del comportamiento de los usuarios o de los datos contextuales recogidos por los periféricos de entrada.

- **Domótica**. Regulación de las condiciones de espacios domóticos en función del comportamiento de los usuarios o de los datos contextuales recogidos por los periféricos de entrada.

2.5. Cableado

Para conectar los periféricos de entrada y de salida con las unidades de procesamiento, necesitamos un cableado específico. El cableado físico es imprescindible en la mayoría de aparatos relacionados con la conectividad o los sistemas multimedia, aunque cada vez más la popularidad de la tecnología inalámbrica o *wireless* (como Wi-Fi, Bluetooth o ZigBee) aumenta entre los fabricantes y los consumidores. Destaquemos los siguientes cableados estándar:

Tipos de datos transmitidos	Audio		Vídeo		
Nombre	Jack	Cánon	RCA	VGA	DVI
Imagen					

Tipos de datos transmitidos	Datos digitales				
Nombre	USB	FireWire	RS232	DIN	Ethernet
Imagen					
Protocolos	*Serial* Otros	Múltiples	*Serial* Otros	MIDI DMX	TCP UDP

2.6. Protocolos

Un protocolo es el conjunto de reglas de codificación de una señal electrónica.

Los hay de muchos tipos, según las especificaciones y las necesidades de cada sistema de comunicación.

Así, el protocolo de codificación de la señal proveniente de un sensor electrónico (una fotocélula, por ejemplo) no es el mismo que el de una señal de Internet.

Hemos de conocer algunos de los protocolos de codificación de datos más habituales para adquirir agilidad en el diseño de interacciones:

- **TTL**. Es uno de los protocolos de comunicación más sencillos. Se trata simplemente de una señal electrónica de entre 0 y 5 V usada para la transmisión de datos entre componentes en un circuito electrónico. Es el tipo de señal que utilizamos para enviar datos a Arduino provenientes de un sensor.

- **NTSC/PAL**. Son protocolos de codificación de imágenes de vídeo que consisten en un grupo de señales que permiten formar una imagen en movimiento. Hay diferentes convenciones según el país en el que nos encontremos. Así, en Estados Unidos el sistema estándar es el NTSC, mientras que en Europa el estándar es el PAL. El advenimiento de la televisión de alta

definición (HDTV) hace pensar que estos sistemas de codificación tienen los días contados, aunque todavía son muy populares y todos los sistemas de vídeo son compatibles con ellos.

- **TCP/UDP**. Son protocolos de codificación de datos pensados para la transmisión por redes de Internet (local o remoto).

- **OSC**. Es un protocolo de datos pensado para la transmisión en red muy popular entre los diseñadores de interacción, ya que permite una gran velocidad, resolución y flexibilidad. Más adelante, analizaremos en profundidad este protocolo.

- **Serial**. Es un protocolo de comunicación basado en la transmisión de datos digitales serializados. La transmisión de los paquetes de datos se codifica en grupos de bytes (8 bits ---> valores numéricos entre 0 y 255). Es el protocolo que usamos para comunicar Arduino con los ordenadores a través de cableado USB, pero también se utiliza en numerosos periféricos informáticos y en aparatos digitales de todo tipo.

- **MIDI**. Es un protocolo de codificación de información musical que permite la comunicación entre instrumentos digitales (sintetizadores, cajas de ritmos, ordenadores, etc.).

- **DMX**. Es un protocolo estándar para el control de sistemas de iluminación o sistemas mecánicos, habitualmente en el contexto de las artes escénicas y el mundo del espectáculo.

2.7. Redes

Actualmente podemos enviar todo tipo de datos por sistemas de red: tanto redes y redes remotas, como Internet. De hecho, los sistemas de conectividad actual permiten el diseño de interacciones con envío de datos a ordenadores remotos.

Pongamos por caso un sistema de sensores de temperatura en Barcelona que es enviado por Internet hasta una ubicación remota en Melbourne, donde se recogen esos datos y se usan como datos de control de un sistema mecánico.

Reflexión

Tratad de imaginar las posibilidades de un sistema deslocalizado de interacción en el que el usuario y los actuadores se encuentran en lugares del mundo diferentes.

Recientemente, se ha hablado mucho la **Internet de los objetos**, una nueva forma de Internet orientada a la conexión de todo tipo de objetos electrodomésticos a la red global para facilitar la operación remota. Es algo todavía muy reciente y está por ver cuál acabará siendo el impacto real de la aplicación de esta nueva Internet de las cosas.

Reflexión

Como ejercicio de prospectiva, intentemos pensar en las posibilidades funcionales de un diseño interactivo basado en la operación remota de electrodomésticos.

2.8. Sistemas y protocolos de comunicación inalámbrica

En los últimos años, se han abaratado y popularizado diferentes sistemas de comunicación inalámbrica que permiten la comunicación entre dispositivos sin cables. Entre estos **sistemas inalámbricos**, podemos destacar los siguientes:

1) Radio. El sistema clásico de transmisión de ondas de radio permite la comunicación a distancia entre emisores y receptores. A pesar de que es un sistema muy popular y extendido, presenta muchos inconvenientes para la transmisión de datos digitales, puesto que de él se derivan muchas pérdidas de información. Sin embargo, dada su generalización y un cierto romanticismo, no hay que descartar su uso en determinados proyectos. Habitualmente, las transmisiones de radio se codifican modulando la amplitud de las ondas (AM) o bien su frecuencia (FM). Las ondas de radiofonía clásicas se mueven en un rango entre los 100 kHz y los 100 mHz.

2) Wi-Fi. Este sistema permite grandes velocidades de transmisión de datos y está muy extendido en el contexto de la transmisión doméstica de señales de Internet. Es muy adecuado para la transmisión de datos digitales, a pesar de que no es tan fiable como la conexión por cable Ethernet, puesto que también presenta pérdidas y retrasos irregulares en la llegada de los paquetes de información. El rango frecuencial de transmisión de datos está alrededor de los 2,4 GHz.

3) ZigBee. El sistema ZigBee se ha hecho muy popular entre la comunidad de desarrolladores de sistemas electrónicos empotrados, puesto que permite una transmisión descentralizada de datos (no se ha de pasar por un servidor central) y es relativamente barato y sencillo de configurar. Arduino es compatible con él.

4) Bluetooth. Este sistema, presente en muchos computadores portátiles y teléfonos móviles, es muy popular y relativamente barato. Es un sistema comparable a ZigBee que se mueve dentro del rango de transmisión frecuencial entre 1 y 4 GHz.

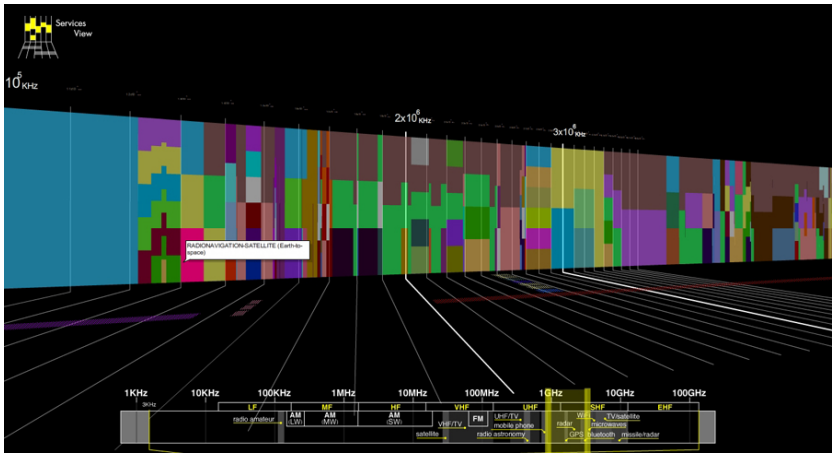
5) 3G. Las redes de telefonía móvil de tercera generación (3G) permiten la transmisión de datos por medio de satélites centralizados de telefonía. Aunque el acceso a la transmisión y la recepción de datos por este sistema continúa,

de momento, delimitado al uso de teléfonos móviles y asistentes personales digitales o PDA, hay una gran previsión de desarrollo de este tipo de tecnología en un futuro cercano.

6) GPS. El sistema GPS permite la geolocalización de dispositivos por satélite con errores aproximados de un metro.

7) RFID. El sistema de identificación por radiofrecuencia se usa como sistema de seguridad antirrobo en productos de consumo, como sistema de identificación y posicionamiento por triangulación de dispositivos o como dispositivo de identificación personal (en las tarjetas llave de los hoteles, por ejemplo). Se trata más bien de un sistema de identificación que de un sistema de transmisión de datos, pero puede ser útil en situaciones y en diseños específicos determinados.

Tipo de datos transmitidos



Visualización gráfica del espectro radioeléctrico.
Fuente: *Atles de l'espai electromagnètic*. Irma Vilà, J. Luis de Vicente, Bestiario.
www.spectrumatlas.org

3. Diseñando interacciones

En este módulo, proponemos la profundización en dos de los protocolos de datos más populares entre los diseñadores de interacciones: OSC y Serial.

3.1. OSC

Es un protocolo de comunicaciones que permite comunicar instrumentos de música, computadoras y otros dispositivos multimedia (por ejemplo, móviles o PDA equipados con Bluetooth o Wi-Fi), y que está pensado para compartir información en tiempo real sobre una red.

Aparece como sustituto de MIDI, respecto al cual es muy superior en características y capacidades.

La mayoría de software multimedia en tiempo real de última generación es compatible con OSC. Para hacer un pequeño repaso, a continuación tenéis una lista de softwares compatibles con OSC:

- **Software de audio**.
  Reaktor, IanniX, Ardour.
- **Software de vídeo**.
  Resolume, Modul8, VDMX.
- **Entornos de programación**.
  Quartz Composer, Processing, openFrameworks, Max/MSP, Pure Data, Arduino.
- **Otros**.
  TouchOSC, Lemur.

Para poder recibir y enviar datos OSC entre diferentes aplicaciones o entre diferentes computadoras, hemos de configurar una red local. Por regla general, asignaremos IP estáticas correlativas a cada una de las máquinas de la red. Una vez configurada la red, configuraremos los servidores y los clientes OSC para que envíen y reciban los datos a la IP o desde la IP, y el puerto que nos interesa. Además, tendremos que definir una serie de etiquetas (*tags*) identificadoras para cada uno de los mensajes que queramos enviar o recibir, para identificarlos y dirigirlos hacia el parámetro correspondiente.

Como ejemplo de etiquetaje y funcionamiento de los parámetros OSC, podemos proponer el siguiente:

/alumno1/sintetizador/volumen 0.15
/alumno1/ritmo/tempo 0.23
/alumno1/vídeo/opacidad 0.63

En este caso, estamos enviando tres parámetros OSC a las subrutas *sintetizador, ritmo* y *vídeo* de la ruta *alumno1* con tres valores diferentes para cada parámetro. Una vez mapados estos datos, podremos asignar estos valores de control a los parámetros *volumen*, *tempo* y *opacidad* de un sintetizador audiovisual.

3.2. Serial

A pesar de que ya hemos hecho una reseña de este protocolo en el módulo sobre dispositivos electrónicos y Arduino, vale la pena profundizar en su funcionamiento y mostrar un caso específico.

Como hemos visto, el protocolo Serial se basa en la transmisión serializada de paquetes de bytes (8 bits: valores entre 0 y 255).

Esta transmisión se hace a una velocidad específica, llamada *baud rate*, determinada en bits por segundo (bps). Hemos de seleccionar bien el valor de la *baud rate*, ya que si estamos diseñando un sistema con un gran volumen de transmisión de información en serie, necesitamos suficiente ancho de banda para todo este volumen de información.

> Se trata de encontrar un equilibrio entre velocidad de transmisión y seguridad de la comunicación, puesto que muchos sistemas a *baud rates* altas no son 100% fiables, sobre todo si el cableado es de baja calidad.

Una vez definida una velocidad de transmisión de datos en bits por segundo, habrá que analizar bien las necesidades de comunicación de nuestro sistema y diseñar un procedimiento de transmisión/recepción de datos serializados.

Pongamos por caso que queremos recibir dos grupos de datos de un rango numérico entre 0 y 2.048. Cada uno de estos dos grupos de datos moverá un motor conectado a Arduino.

Necesitaremos, pues, como mínimo, dos valores diferenciados:

1) Un valor identificador del motor que queremos mover (motor 1 o motor 2).
2) Un valor de posicionamiento de este motor.

Como los valores de posicionamiento de los motores exceden el byte (recordemos que un byte puede almacenar valores entre 0 y 255 y nosotros necesitamos poder almacenar valores entre 0 y 2.048), deberemos construir este valor con operaciones de 2 bytes.

Una posibilidad sería hacer algún constructo matemático del tipo:

El valor enviado está entre 0 y 255.
Byte 1: 0
Byte 2: 0 < *n* < 255

El valor enviado está entre 256 y 511.
Byte 1: 1
Byte 2: 0 < *n* < 255

El valor enviado está entre 512 y 767.
Byte 1: 2
Byte 2: 0 < *n* < 255

Siguiendo este sistema, podemos construir valores hasta 256^2 de la manera siguiente:

valor = (byte1 * 255) + byte2

Finalmente, podemos implementar este sistema en el código de Arduino tal como sigue:

```
if (Serial.available()>2){

 val1=Serial.read();              //id motor

 val2=Serial.read();              //constructe rotacio1

 val3=Serial.read();              //constructe rotacio2


rotación = (val2 * 255) + val3;

}
```

Esta no es la única manera de transmitir valores numéricos más grandes que un byte a Arduino: hay otros métodos (por ejemplo, concatenando tantos bytes como cifras tenga el número que queremos enviar, codificando cada una de las cifras con su valor ASCII), y cada caso específico puede presentar sus particularidades. Por eso es muy importante identificar *a priori* las necesidades de transmisión y recepción de datos que tendrá nuestro sistema.

4. Más allá. Recursos específicos

4.1. Recepción de datos de Internet. Minería de datos (*data mining*)

Dentro de esta área de comunicación por la red, se abren una serie de posibilidades muy interesantes en el campo del diseño interactivo y la experimentación creativa a partir de la adquisición de datos en tiempo real provenientes de bases de datos en línea (*on-line*).

Ejemplo

Pensad en datos provenientes de las cotizaciones bursátiles, datos de seguimiento por GPS de grupos de fauna en peligro de extinción, datos provenientes de estaciones de observación astronómica, etc.

La búsqueda en línea de este tipo de datos se denomina **minería de datos** (*data mining*), y es realmente sorprendente la cantidad de valores actualizados en tiempo real que podemos encontrar en Internet.

Hay dos estrategias básicas de extracción de datos:

- el uso de los agregadores (*feed agregators*) RSS, o
- el análisis (*parsing*) de webs.

Los agregadores RSS son un tipo de datos empaquetados con la intención de ser compartidos entre diferentes plataformas.

Agregadores RSS

Muchos diarios, publicaciones en línea e incluso blogs personales usan este sistema para compartir titulares u opiniones de manera distribuida en la red.

Entornos de programación como Processing son compatibles con bibliotecas que se encargan de recibir estos agregadores de la red.

Por otro lado, el análisis es un método de "fuerza bruta" que implica analizar todo un web para encontrar los campos que son actualizados en tiempo real y, por lo tanto, que contienen datos valiosos en el sentido del diseño de interacciones.

En este web podéis encontrar una descripción detallada de un proceso de análisis:

http://dataleech.wordpress.com/

En esta misma línea, también es interesante el proyecto www.pachube.com, que pretende convertirse en un punto de centralización de datos provenientes de sensores ubicados en todo el mundo y que ofrece los datos de captación ambiental en abierto para facilitar la creación de proyectos basados en el análisis de datos que provienen de contextos remotos.