Captación y Digitalización

Félix de la Paz López

Departamento de Inteligencia Artificial. UNED.

Resumen

En este capítulo vamos a describir distintos dispositivos sensores necesarios para la captación de magnitudes físicas relevantes del medio, y cómo se traducen a formato digital para su manipulación por un programa. También describiremos algunos sistemas efectores, que son los que van a permitir que el sistema responda con alguna acción a la entrada sensorial.

1. Introducción: Captación y digitalización de datos sensoriales.

En numerosas aplicaciones tanto industriales como de investigación, es necesario conocer el valor de magnitudes físicas exteriores e interiores respecto al sistema en sí. El dispositivo encargado de realizar esta misión es el sensor. Las magnitudes físicas son detectadas por el sensor y traducidas a una señal eléctrica (captación). Posteriormente, esta señal ha de ser transformada en un dato con el que se pueda operar en una computadora (digitalización). Los sensores son muy utilizados en la industria para monitorización y supervisión de procesos, sistemas de alarma y vigilancia, teleoperación etc.. Si además, queremos que el sistema sea capaz de interaccionar con el medio son necesarios actuadores que puedan modificar el estado del conjunto sistema-medio. Un sistema genérico que integre percepción y acción consta de las siguientes partes.

- Un sistema sensorial para tomar datos del medio.
- Un sistema de proceso para tratar esos datos.
- Un sistema efector para interactuar con el medio.

En la figura 1 vemos como el sistema sensorial toma datos del medio. Estos datos son puestos en un formato adecuado por el sistema conversor para su posterior proceso. Después, el sistema de proceso ordena acciones al sistema efector que, evidentemente, afectan al estado del medio. Este nuevo estado del medio es de nuevo medido por el sensor, cerrando así el lazo de percepción-acción.

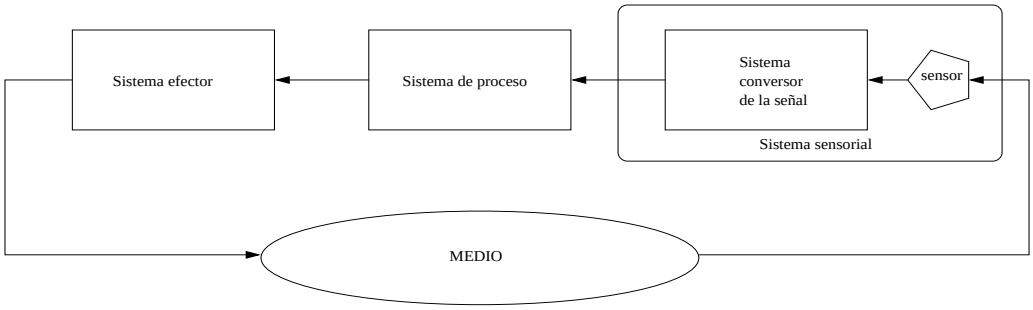


Figura 1: Sistema genérico de percepción acción.

2. Sistema sensorial genérico.

Los sistemas sensoriales son aquellos que permiten medir variables del mundo exterior al sistema y transformarlos a un formato que pueda ser fácil de manipular.

Un sistema sensorial genérico consta de los siguientes bloques:

- Un sensor o transductor que es capaz de traducir una variable física en una señal eléctrica.
- Un sistema de conversión de la señal, que sirve de interfaz con el sistema de proceso.

En general, los datos adquiridos por el sistema sensorial suelen ser magnitudes analógicas, esto es, voltajes o frecuencias variando en un rango determinado y que han de transformarse a magnitudes digitales, que son al final los datos que podemos almacenar en la memoria de una computadora para su posterior manipulación mediante un programa.

Por tanto, la conversión analógico-digital es una herramienta fundamental para traducir las magnitudes físicas que tomamos del medio a través de los sensores en datos digitales (bits) para su posterior interpretación y manipulación.

Pongamos por ejemplo un sensor que es capaz de medir temperaturas en un rango de 0 a 100 grados, variando el voltaje que atraviesa el sensor linealmente entre 0 y 5V. Evidentemente, podemos establecer una simple regla de tres y deducir que cada grado se corresponde a una variación de 0.05V. Pero, suponiendo que el sensor sea efectivamente tan sensible y que la variación es contínua, tenemos que tener en cuenta con qué resolución digital vamos a muestrear este rango. Si disponemos de un conversor A/D de 4 bits cuya entrada pueda variar entre 0 y 5V, tendremos 16 posibles estados del conversor con lo que sólamente podremos dividir los 100 grados (5V) en 16 partes (figura 2). Con esto, solo apreciaríamos variaciones de 5/16 V (0.3V) o lo que es lo mismo 100/16 grados (6,25). Si tuviéramos un conversor con una resolución de 8 bits, tendríamos 256 posibles estados con lo que se podrían representar hasta 100/256 (0.4) grados. Es, por tanto, el número de bits con los que puede trabajar el conversor A/D lo que nos da la resolución con la que vamos a trabajar y lo que nos indica si nuestro sistema es mejor o peor.

En realidad, ningún sensor se comporta de manera lineal en todo su rango. Casi todos suelen tener una respuesta de tipo logaritmo o exponencial. Como método general, deberemos de calibrar la respuesta del sensor con un patrón.

Aprovechando esta pequeña incursión en la electrónica, podemos comentar que la conversión D/A (Digital-Analógica), que es el camino contrario al que hemos comentado hasta ahora, se utiliza, por ejemplo, para transformar los resultados conseguidos tras

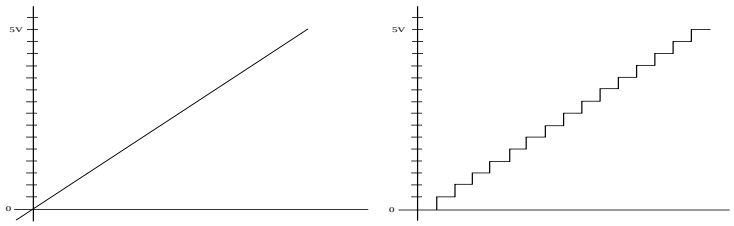


Figura 2: Conversión A/D. La señal analógica (izda) solo puede ser representada por 16 valores discretos (dcha), correspondientes a los tramos horizontales.

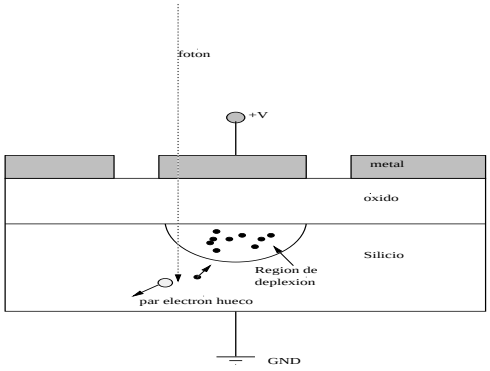


Figura 3: Estructura de un dispositivo CCD

procesar los datos sensoriales mediante un programa, en señales analógicas que puedan ser usadas para gobernar motores, que son los componentes esenciales de los sistemas efectores.

2.1. Tipos de Sensores.

Los tipos de sensores más utilizados son aquellos que tienen que ver con la visión (Cámaras de video), y con la medida de distancias (sonar, infrarrojos, contacto), aunque también se pueden utilizar, dependiendo de la aplicación que se vaya a desarrollar, sensores de temperatura, presión, humedad, detectores de metales, sensores de gases etc...

2.1.1. Camaras de Video.

El sensor

En la actualidad las cámaras que mas se utilizan son las que están basadas en la tecnología CCD. Estos dispositivos han abaratado mucho el coste de las cámaras, con unas prestaciones excelentes. CCD se corresposde a las siglas inglesas Coupled Charge Device (dispositivo acoplado por carga).

Una cámara CCD es una matriz bidimensional de transistores MOS que funcionan como condensadores. El tamaño aproximado de un pixel es de 10 x 10 μm. El substrato tipo P de Silicio se recubre con una capa de oxido de silicio que aisla el substrato del metal. Cuando un fotón, con una energía sufuciente incide en el substrato, crea un par electrón-hueco en la región de deplexión. Este electrón va hacia el electrodo positivo y se almacena una carga en el condensador MOS. Así se va almacenando en cada pixel la imagen, que luego puede ser recuperada (figura 3).

Figura 4: Disposición de los transistores MOS en una cámara CCD (izda). Dispositivos CCD (dcha).

Figura 5: Cámara comercial

La imagen se va formando mediante la carga de cada uno de estos condensadores. Cada MOS es un pixel. Las dimensiones de la matriz CCD nos dan, por tanto, la resolución de la cámara. Previamente al CCD existe una óptica que se encarga del enfoque, zoom etc...

En la figura 4, vemos que hay una zona de 4 pixels iluminada. La intensidad de la luz que incide está relacionada con la carga que almacena el transistor. En este caso, tendríamos una resolución de 12 x 14 pixels . Evidentemente, este dibujo sólo sirve como ejemplo, ya que las dimensiones normales para una cámara comercial suelen ser 768 x 494 pixels, indicando la primera magnitud la resolución horizontal, y la segunda la vertical.

En la figura 5 vemos un ejemplo del aspecto de una cámara comercial. Su tamaño no es mayor que un paquete de cigarrillos.

El sistema de conversión de la señal

Para poder trabajar con cámaras, necesitanos una tarjeta digitalizadora de video (frame grabber) que podemos ver en la figura 6. Las condiciones de este tipo de tarjetas, en cuanto a canales, velocidad de transferencia y preproceso, dependerán de los requerimientos de la aplicación. Con esto conseguimos adaptar la señal que nos da la cámara a un formato que podemos tratar con un programa.

2.1.2. Sensores de Sonar.

El sensor de sonar se utiliza cuando es necesario conocer distancias del orden de metros con precisión de unos pocos centímetros. El más utilizado es el sensor de membrana Polariod (c) 6500.

Figura 6: Tarjeta digitalizadora (frame grabber).

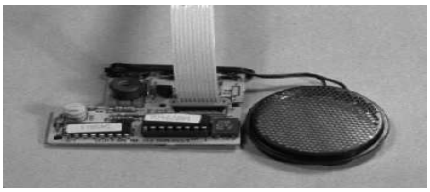


Figura 7: Sensor de Sonar y circuitería.

El sensor.

Se trata de una membrana plana que puede emitir y recibir pulsos de ultrasonidos (figura 7). La distancia se calcula midiendo el tiempo transcurrido entre la emisión y la recepción de un pulso. Puede medir distancias entre 15cm y 11m con una precision del 1 % sobre todo el rango.

En la configuración más usual, cada sensor emite un pulso de onda cuadrada de 56 ciclos a 49.4 kHz. A continuación existe un periodo muerto, durante el cual, el circuito de control se resetea y estabiliza. En ese momento, el sensor actua como receptor, alimentando con los ecos detectados, un amplificador de ganancia variable con el tiempo. El factor de ganancia de este dispositivo aumenta con el tiempo para compensar las pérdidas debidas a la amplitud del medio y la atenuación del sonido en el aire. La salida del amplificador pasa por un circuito de umbral. Cuando este umbral se excede, se mide el tiempo transcurrido desde el comienzo de la transmisión, y se convierte en distancia mediante un factor de calibración.

Fuentes de error debidas al dispositivo.

El sensor no emite energía de forma homogénea en todas direcciones, sino que lo hace en forma de lóbulos de intensidad decreciente (figura 8). Además, resultados experimentales muestran que el patrón de radiación de los sensores Polaroid (c) no es simétrico y que varía de un sensor a otro. Esto afecta sobre todo a los lóbulos laterales.

La atenuación de los ultrasonidos depende de la temperatura y de la humedad. Valores

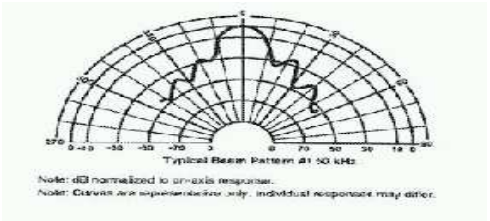


Figura 8: Patrón de emisión de un sensor de Sonar. Cortesía de Polaroid (c).

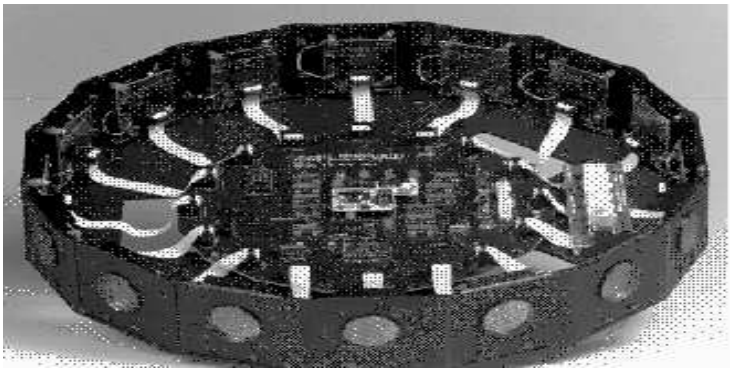


Figura 9: Sistema sensorial Sensus 200 (tm). Cortesía de Nomadic (c).

típicos para 50 kHz presenta una atenuación de 0.6 a 1.8 dB/m para variaciones de temperatura entre 17 y 28 C y variaciones de la humedad relativa entre 15 y 70 %. La velocidad del sonido en el aire se puede expresar como:

$$v = 331{,}4\sqrt{\frac{T}{273}} m/s$$

donde T se expresa en grados Kelvin. Vemos que si cambian las condiciones ambientales, cambia la velocidad con que se propaga el sonido y, por lo tanto, afectará a la calibración de nuestro sensor.

Existen además varias fuentes de error debidas a la circuirtería elecrtrónica. En primer lugar, la duración del pulso transmitido. Toda la medida del "tiempo de vuelo" está basada en que el inicio del pulso transmitido es la parte del eco de vuelta que en ese momento excede el umbral del detector. Si este no es el caso, el error es como mucho, 20cm.

El amplificador de ganancia variable con el tiempo utiliza una curva exponencial aproximada por una funcion constante a tramos de 12 pasos. Esta función puede ser diferente si cambian las condiciones de humedad y temperatura. Como la energia retornada es función del ángulo de incidencia, el ángulo de visibilidad cambia.

El circuito de umbral dispone de un dispositivo de carga capacitiva. Se almacenan varios pulsos hasta que la intensidad alcanza un cierto valor umbral. Normalmente, si la señal reflejada tiene una intensidad alta, son suficientes 3 ciclos para alcanzar el umbral. Este retraso se acomoda bien con la calibración. Sin embargo, si la señal es débil, el periodo de carga puede ser bastante largo, dando lugar a valores erróneos.

Fuentes de error debidas al medio.

También hay que tener en cuenta los objetos en donde se van a reflejar los ultrasonidos. Se pueden definir dos grupos: objetos reflectivos, de dimensiones mayores que una longitud de onda (6.95 mm a 20C) y objetos difractores de dimension menor que una longitud de onda. Objetos menores que la longitud de onda suelen ser más escasos que los otros, sin embargo, superficies rugosas como el cemento presentan asperezas que actuan como difractores. Superficies pulidas, pintadas o metalizadas, actúan como reflectores.

Cuando la superficie es suave (pulida) refleja todos los lobulos del sonar, lo que provoca un patron de reflexión tipo arco. Esto es debido a que los lóbulos secundarios se reflejan con menor energía y esto produce errores en la medida. Cuando la superficie es rugosa, sin embargo, al dispersar los lóbulos es menos probable que obtengamos la reflexión de los lóbulos secundarios y que sólo tengamos reflexión del principal. Esto que en principio

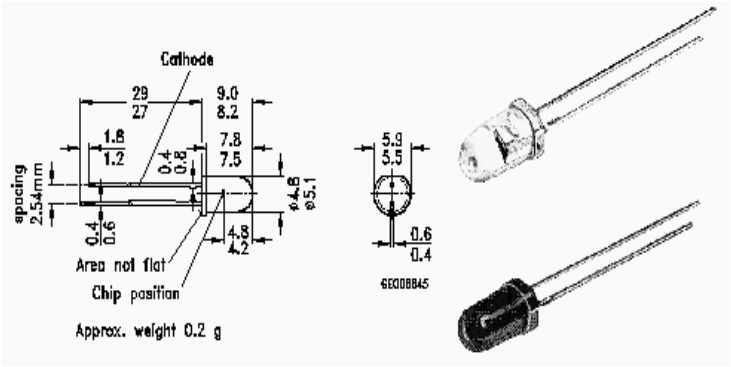


Figura 10: Aspecto de un sensor de infrarrojos. Cortesía de Siemens (c).

parece favorecer la medida no es así, ya que la reflexión del lóbulo principla al estar en parte dispersado, se produce con menor energía, lo que hace que se tarde más en cargar el dispositivo de umbral y éste retraso produce graves errores en la medida.

El sistema de conversión de la señal.

Este tipo de sensor se suele agrupar junto con otros del mismo tipo en disposiciones lineales o circulares. Estas agrupaciones suelen estar controladas por microprocesador, que toma los datos de los sensores cíclicamente y los prepara para pasarlos por un bus al procesador central. De esta manera se descarga a la CPU principal de un trabajo grande y monótono. En la figura 9, vemos una disposición circular de sensores de sonar "Sensus 200 (tm)" de la empresa Nomadic (c), utilizada frecuentemente en robótica. Se trata de 16 sensores de sonar controlados por un microprocesador Motorola (tm) HC11.

2.1.3. Sensores de infrarrojos.

Antes hemos comentado en los sensores de sonar que, al ser el mismo dispositivo el que emite y recibe los ecos, existe un tiempo muerto mientras el dispositivo pasa de ser emisor a receptor. Por lo tanto si un objeto está lo suficientemente cerca, el eco llega al sensor antes de que este funcionando como receptor, con lo que la distancia no se puede medir. Para salvar este inconveniente, se utilizan los sensores de infrarrojos que, si bien tienen un rango de medida muy pequeño, pueden medir precisamente en la zona muerta de los sensores de sonar.

El sensor.

Cada sensor está compuesto de dos diodos emisores SIEMENS SFH 434-3 (figura 10) y un diodo receptor PIN SIEMENS SFH 2030F encapsulados en un mismo set. Su rango de acción teórico es desde 0cm a 90cm. El receptor esta montado entre los dos emisores.

La distancia a un objeto se determina por la intensidad de la luz infrarroja reflejada. Esta intensidad se mide digitalmente, usando un oscilador controlado por voltaje (VCO), que es un dispositivo que proporciona como salida una señal de onda cuadrada con una frecuencia proporcional al voltaje que le entra. Cuanto mayor sea el voltaje de entrada, más rápida será la oscilación.

El orden de magnitud para unas condiciones estándar (iluminación normal de oficina, pared blanca a 40cm) es 0.0025 segundos por cada ciclo de medida.

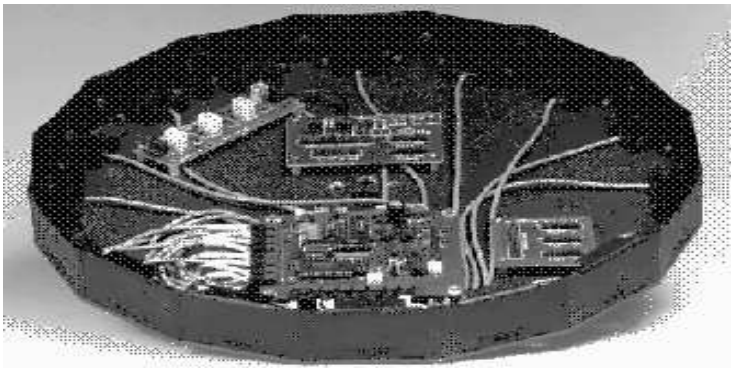


Figura 11: Sensus 300 (tm). Cortesía de Nomadic (c).

Fuentes de error debidas al dispositivo.

Este sistema, como hemos dicho, funciona dependiendo de la intensidad de luz infrarroja reflejada. Cuanto mas lejano sea el objeto, menor será la intensidad de la luz reflejada y, por lo tanto, consideraremos que está mas lejano. El angulo de reflexión del haz juega un importante papel. Asumimos que cuando el haz alcanza el objeto, parte se refleja y parte se difracta, pero, si encima la incidencia no es normal, solo recibiremos una pequeña parte de la energía emitida, proviniente sobre todo de la luz difractada, lo que dara errores de medida, debido a que la calibración del sistema esta hecha para incidencia normal.

Fuentes de error debidas al medio.

La iluminación del medio juega un papel crítico en este tipo de sensores. Lo ideal es funcionar a oscuras, ya que la cantidad de radiación infrarroja en el ambiente es mínima. Esto no es operativo pues siempre existen fuentes de luz, sobre todo en un ambiente tipo oficina. El problema es que existen fuentes de luz con mayor y menor contenido de luz infrarroja, lo que influye mucho en la calibración y rango de los sensores. La luz fluorescente no contiene apenas luz infrarroja, lo que prácticamente no perturba el sistema de infrarrojos. Sin embargo la luz incandescente, la solar y sobre todo la halógena, tienen gran cantidad de luz infrarroja en su espectro, lo que puede incluso dejar inoperativo el sistema sensorial por infrarrojos.

El color y la textura de la superficie reflectora también influye en la precisión de las lecturas. Superficies pulimentadas muestran un pico de intensidad para los ángulos de incidencia normal, pues actúan como espejos. Superficies como cemento u otras superficies de textura similar absorben más.

El sistema de conversión de la señal.

Los sensores de infrarrojos tienen múltiples usos. Se pueden utilizar para medir distancias por reflexión como hemos comentado, pero también sirven para transmitir información a distancia. Todos tenemos la experiencia del mando a distancia de la televisión o el video. En este caso, las distancias donde este sensor es operativo son mayores, debido a que no dependemos de la intensidad reflejada por una superficie, sino de la emisión directa entre emisor y receptor. Incluso nos podemos ayudar de lentes para estrechar el haz y conseguir mayores distancias. También se pueden utilizar para detectar radiación infrarroja producida por fuentes de calor.

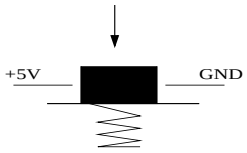


Figura 12: Esquema elemental de un pulsador.

En el caso de medida de distancias, hemos comentado que el fabricante proporciona dos emisores y un receptor en un mismo encapsulado. Este dispositivo, al igual que el sonar, se suele disponer en conjuntos de forma lineal o circular y también se utilizan microprocesadores dedicados para obtener los datos rápidamente y pasarlos por un bus al procesador central. Este es el caso del Sensus 300 (tm) de la empresa Nomadic(c) que vemos en la figura 11 con 16 sensores.

2.1.4. Sensores de impacto.

Los sensores de impacto, salvo en aplicaciones muy específicas en las que es necesario conocer la presión que se ejerce sobre una superficie, suelen ser sensores del tipo todo-nada, es decir, simples interruptores que cuando están desactivados indican un estado lógico y cuando se activan indican el complementario. El uso más extendido es como sensores de colisión.

El sensor.

Existen muchos modelos de microinterruptores que pueden servir como sensores de impacto. El más utilizado es el tipo pulsador (figura 12), esto es, no es necesario una nueva pulsación para cambiar de estado.

El sistema de conversión de la señal.

En este caso, no hay dispositivo intermedio entre el sensor y la CPU, pues la información ya esta como bit, y simplemente se introduce por un puerto serie o paralelo. Los sensores de impacto se suelen disponer en conjuntos dentro de bandas de caucho que protegen el sensor y proporcionan una elasticidad añadida al sistema. En inglés, este dispositivo se conoce como "bumper". Estas bandas llenas de interruptores se colocan alrededor de los sistemas y son la frontera última entre éstos y el medio. En la figura 13 podemos ver un sistema doble de bandas de sensores de impacto Sensus 100(tm).

Fuentes de error debidas al dispositivo.

Al ser un dispositivo todo-nada, es muy fiable y las únicas fuentes de error pueden ser debidas a averías mecánicas en el pulsador.

Fuentes de error debidas al medio.

Las únicas funtes de error en este tipo de dispositivos es que el objeto a golpear sea tan liviano que no active el pulsador y sea arrastrado por el robot.

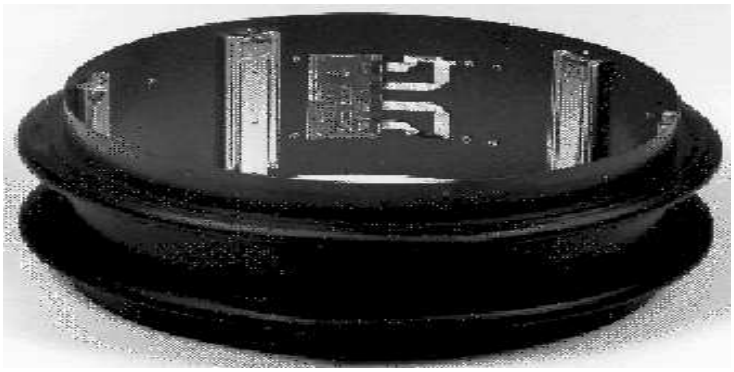


Figura 13: Doble anillo de sensores de impacto. Cortesía de Nomadic (c).

Figura 14: Sensor de Láser. Cortesía de Sick Electro-Optics (c).

2.1.5. Sensores de Láser.

Existen numerosos dispositivos que utilizan el Láser como instrumento de medida de distancias, incluso en combinación con cámaras de vídeo. Son dispositivos precisos pero bastante caros.

El sensor.

Se trata de transmitir un pulso de luz láser y reflejarla en un espejo rotatorio para transmitir ese haz en un patrón que cubra 180 grados. De esta manera se crea un campo sensorial alrededor del dispositivo. Cuando un objeto intersecta este campo, refleja parte de la luz láser al dispositivo. Usando el intervalo de tiempo en que la luz va y viene, y el ángulo del espejo, se determina la localización del objeto.

El sistema de conversión de la señal.

La circuitería asociada a este dispositivo es muy compleja. Básicamente nos entrega un vector de 361 componentes que cubre 180 grados (tiene una resolución de 0.5 grados). Este dispositivo puede medir distancias hasta de 50m con una precisión de 50mm (figura 14).

3. Sistema Efector.

Los efectores (figura 15) son los dispositivos que permiten al sistema interaccionar con el medio, desencadenando las acciones consecuentes al proceso que el sistema hace de los datos tomados por los sensores. Dentro de este apartado destacamos los sistemas de movimiento, y los brazos articulados.

Figura 15: Manipulador (izda) cortesía de Nomadic (c). Grupo motor (dcha) cortesía de Robosoft (c).

Figura 16: Robots de Nomadic (c).

Los sistemas de movimiento pueden ser también sensores del sistema si llevan incorporados dispositivos que puedan medir desplazamientos, velocidades y aceleraciones. Los brazos articulados pueden también incorporar sensores de sonar e infrarrojos para usarlos en su desplazamiento y sesores de presión en las pinzas para medir la fuerza utilizada en la captura de un objeto o en la manipulación de una herramienta.

Dentro de los sistemas motores, los más usados son los motores con ruedas, pero también se pueden utilizar cadenas (tipo tanque), con patas (tipo insecto), motores de aeromodelismo, globos aerostáticos etc...

Un buen ejemplo donde podemos encontrar juntos todos estos dispositivos de percepción, procesado y acción son los robots autónomos. En la figura 16 vemos el aspecto de algumos modelos de la empresa Nomadic (c).

4. Los datos.

4.1. Estructura.

Como hemos comentado a lo largo de este tema, los sistemas de acondicionamiento sirven para transformar la señal eléctrica proporcionada por el sensor, en datos que sean manipulables por el programa.

Estos datos suelen almacenarse en estructuras intermendias que faciliten su manipulación mediante librerías estándar. La estructura que más se utiliza es la matriz.

Pongamos por ejemplo la visión. La propia estructura física de los dispositivos CCD

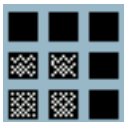
 $\Longleftrightarrow \begin{pmatrix} 0 & 0 & 0 \\ a & b & 0 \\ c & d & 0 \end{pmatrix}$

Figura 17: Representación matricial de los datos de un CCD.

$$\overrightarrow{x} = (\ x_1, ..., x_n)$$

Figura 18: Expresión genérica de un vector de datos.

nos indica que es muy útil representar la información que venga de la cámara en una matriz m x n, donde existe una clara correspondencia entre elemento de la matriz y pixel (figura 17).

En la mayoría de los casos, un elemento de matriz puede representar más que un simple número, ya que pueden ser punteros a listas con las propiedades del pixel (intensidad, brillo, contraste etc...).

En cuanto a los sensores tipo sonar e infrarrojo, también se almacenan en matrices 1 x n, donde n es el número de sensores. Las componentes de estas matrices son números enteros o en coma flotante cuya magnitud es la propia distancia medida en pulgadas o centímetros, o una magnitud proporcional a éstas.

Los sensores de impacto, sin embargo, se almacenan en una palabra de n bits de la siguiente manera. Se cuenta el número de sensores que hay, por ejemplo 20 y se asigna un bit de una palabra de, en este caso, al menos 20 bits codificando 1=impacto, 0 =nada. Normalmente este número se nos entrega en decimal para que tenga un formato parecido a los que dan los sensores de bumper e infrarrojos, pero al pasar a binario, obtenemos que sensor está activado y cuál no.

Muchos fabricantes de este tipo de sistemas, entregan librerías de desarrollo que permiten leer directamente las distancias sin tener que preocuparnos de formatos y calibraciones. Por ejemplo, pueden entregar una matriz 1 x n donde:

n= número de sonars+número de infrarrojos+bumper(solo una componente)+una serie de magnitudes internas.

En total , podemos leer un vector de unas 40 componentes donde las 16 primeras son el infrarrojo, del 17 al 32 son el sonar, el 33 son los bumpers (ya hemos dicho que se codifican en binario y que se pueden recodificar en su equivalente decimal), y de ahí hasta el 40 son datos sobre el voltaje de las baterías, velocidades, aceleraciones, posición etc...). La expresión general es la de la figura 18.Hay que tener en cuenta que estamos utilizando la notación "vector" como estructura para almacenar y tratar datos y no como elementos de un espacio vectorial de n dimensiones. En el caso matricial, para una imagen de resolución m x n sólo hay que generalizar el caso vectorial (figura 19).

$$I = \begin{pmatrix} x_{11} & . & . & . & x_{1n} \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ x_{m1} & . & . & . & x_{mn} \end{pmatrix}$$

Figura 19: Expresión genérica de una matriz de datos.

Figura 20: Ejemplo de un robot en un entorno simple (izda) y datos muestreados por sonar (dcha).

4.2. Manipulación.

Una vez que tenemos los datos en un formato fácil de manipular, podemos aplicar técnicas de filtrado espacio-temporal, memoria, umbralización, segmentación, extracción de características, identificación de patrones etc... que serán materia del capítulo siguiente, pero antes, vamos a ver un caso real.

Como ejemplo, vamos a utilizar un simulador de un robot representado por un círculo que dispone de 16 sonar, 16 infrarrojos y 20 bumpers situados de manera equidistante en su periferia. Conectamos el robot en una habitación cuadrada y sin objetos y vamos a ver de qué datos disponemos para averigüar que la habitación es, en efecto, cuadrada.

Evidentemente, deducir de esos puntos que estamos en una habitación cuadrada es una extrapolación exagerada, máxime si no hemos contemplado antes la parte izquierda de la figura 20. Los datos se almacenan en una estructura como la siguiente:

$\overrightarrow{x}$=($x_{i1}, ...x_{i16}, x_{s1}.....x_{s16}, x_b$)

donde el subíndice "i" indica infrarrojo, el "s" sonar, y el "b" bumper. En estas componentes se almacena directamente la distancia en centímetros o pulgadas salvo para el bumper, en el que se almacena en decimal un número que en binario tiene tantos bits como bumpers hay, y en el que cada bit vale 1 ó 0 dependiendo si el bumper está o no activado. Como el robot está muy alejado de las paredes, los sensores de infrarrojo no indicarán nada. Su lectura será la de la máxima distancia alcanzable ya que no recibirán señal alguna. Si nos fiaramos ahora de esta lectura, creeríamos que el robot se encuentra en una habitación circular de radio el máximo alcance de infrarrojo. Los sensores de sonar nos dan 16 lecturas de la distancia a las paredes, en este caso, las lecturas si son fiables. Por último, si no hay colisión, el bumper devuelve el número decimal 0. Esta claro que la asignación de semántica a los datos sensoriales que se almacenan en éste vector ha de ser llevada a cabo por el programador.

Con sólo 16 puntos de sonar poco se puede hacer. Parece necesario, por tanto, que aparezcan mecanismos de memoria que almacenen los datos muestreados en varios puntos del espacio y en distintos instantes de tiempo para que juntos contribuyan a tener una información más fiable. Si movemos el robot y tomamos datos en varios puntos del medio obtenemos algo como lo indicado en la figura 21.

En el ejemplo, hemos desplazado arbitrariamente el robot hasta que colisione con una pared. En este caso, mientras el robot se encuentra alejado de las paredes, solo recibe datos del sonar. Cuando se aproxima a la esquina, empiezan a aparecer datos de infrarrojo. Finalmente, en la colisión, aparece un dato puntual en el bumper correspondiente.

Para almacenar los datos sensoriales en un mapa que tenga cierta relación de isomorfismo con el medio real, es necesario medir en ese medio. Para esto necesitamos conocer las coordenadas del robot en todo momento y con gran precisión. Esto se consigue a través de sensores que codifican el movimiento de las ruedas y que están situados en los propios

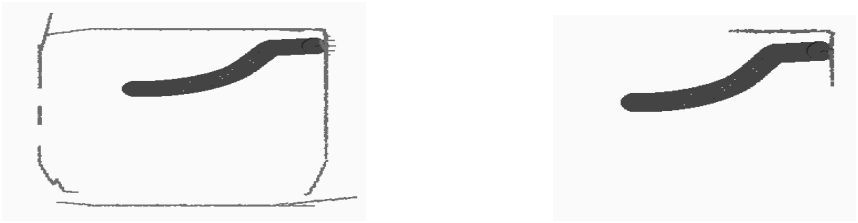


Figura 21: Datos de sonar (izda) e infrarrojo (dcha) almacenados durante la navegación. El trazo grueso indica la trayectoria del robot.

motores. Almacenamos así datos sensoriales en varias posiciones en el espacio que nos van revelando más detalles sobre el medio. Sin embargo también es necesario acumular datos temporales en la memoria. Esto nos va a permitir desechar datos inexactos (reflexiones espúreas) que evidentemente no se mantienen en el tiempo y también nos permite realizar la detección de objetos móviles.

Evidentemente, los entornos reales no están compuestos de habitaciones vacías con paredes rectas e imperturbables en el tiempo. En cuanto queremos añadir la más mínima complejidad, necesitamos acudir a la visión artificial como herramienta fundamental. No es desdeñable, sin embargo, el esfuerzo realizado en conocer otros dispositivos sensoriales en principio más pobres. La visión artificial tiene cierto éxito en el reconocimiento de formas y colores, sin embargo es más fácil y preciso medir una distancia con un sonar que con un sistema de visión estereoscópica. La estrategia fundamental es la fusión multisensorial, esto es, utilizar datos de distintas fuentes sensoriales que, juntos, dan una visión completa y complementaria del entorno del sistema.

Las estrategias, técnicas y algoritmos necesarios para procesar toda esta información aquí obtenida, y la interpretación de los resultados serán materia del siguiente capítulo.

5. Para conseguir más información

La mejor manera de conocer a fondo un sensor y lo que puede dar de sí, es recurriendo a las especificaciones del fabricante. Al final de este documento puedes ver unos ejemplos de este tipo de información. Internet también es una buena fuente de informanción actual y puntera en el campo de la robótica en general y de los sensores en particular. Vamos a citar a continuación algunas direcciones de interés.

5.1. Enlaces bibliográficos genéricos.

http://www.dia.uned.es/~delapaz/html/agentes.html. *Enlaces sobre robótica.*

http://www.neurop2.ruhr-uni-bochum.de/MIT_lit.html *Publicaciones del MIT AI Lab.*

http://www.cs.washington.edu/research/jair/ *Journal of Artificial Intelligence Research.*

http://www.wkap.nl/ *Kluwer Academic Publishers.*

http://www.interaccess.org/arg/arg-list/threads.html *Art & Robotics Group Mailing List.*

http://soma.npa.uiuc.edu/courses/neuroethol/model_systems.html *Estudio de modelos biológicos.*

5.2. Enlaces sobre sensores.

http://www.matrox.com *Cámaras y frame-grabbers.*
http://www.polaroid.com *Cámaras y fotografía en general.*
http://www.bpgprod.sel.sony.com/allcategories *Cámaras.*
http://www.icv.ac.il/DataBases/biblio/bibliography/stereo234.html *Shape Computations from Multiple Sensors.*
http://www.cs.utah.edu/projects/robot/navigation.html *Vision-Based Navigation ,Robotics and Vision, University of Utah.*
http://www.interaccess.org/forums/arg/messages/43.html *Some Sensor Sites.*
http://www.wirz.com/sonar/index.html *Wirz Electroncis - Polaroid Sonar Modules.*

5.3. Empresas que hacen robots.

http://www.robots.com *Nomadic Technologies.*
http://www.rwii.com *Real World Interface.*
http://www.robosoft.fr *Robosoft.*

5.4. Kits de montaje.

Para construir tus propios robots.
http://www.edacom.com/owi/robots.html *Kits de Robótica.*
http://www.robotstore.com *Kits de Robótica.*
http://www.legomindstorms.com *Para construir robots con LEGO.*

5.5. Para buscar.

Este es un buscador para temas de robótica.
http://www.robohoo.com *Robohoo!*

5.6. Ejemplos de robots.

En esta dirección puedes encontrar un robot con todos sus sistemas sensoriales, encargarle tareas para que se mueva en su laboratorio y ver como las hace en tiempo real.
http://www.cs.cmu.edu/~Xavier/ *Xavier the Robot.*

Aquí puedes programar tu propio robot en un simulador. Es software libre. puedes usarlo, copiarlo y modificarlo. Requiere LINUX.
http://www.gnu.org/software/robots/index.html *GNURobots.*

En esta otra página puedes encontrar el simulador del robot Khepera (tm). Se trata de un robot miniatura (figura 22) con 8 sensores de infrarrojo.
Requiere LINUX.
http://lamiwww.epfl.ch/lami/robots/K-family/Kfamily.html#environment. *Simulador del robot Khepera (tm).*

Figura 22: Robot Khepera (tm) (dcha) y modelo usado en el simulador (izda).

Con este simulador se puede construir un mundo de bloques para probar algoritmos de procesado de datos sensoriales y su relación con el movimiento del robot en ese mundo. El simulador es gratis y solo necesita las librerías de X11.

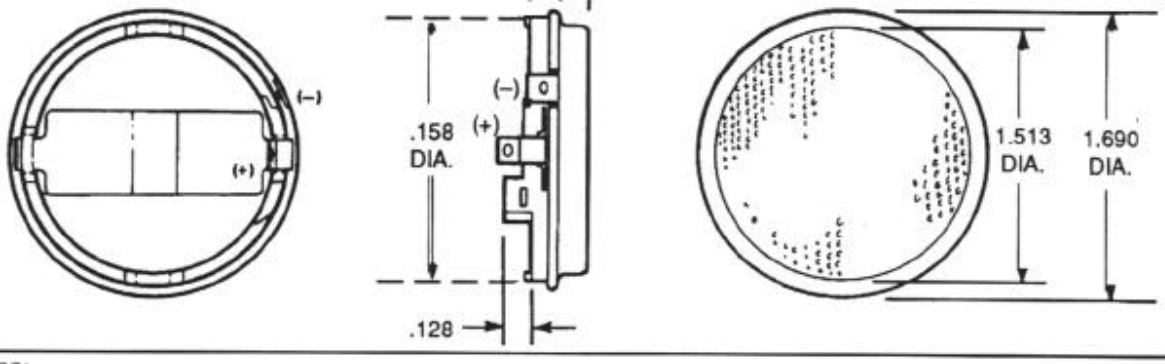
6. Conclusión.

La integración de la percepción y la acción es un estupendo banco de pruebas para cualquiera de las perspectivas de la IA, tanto simbólica como conexionista y, por supuesto, para la vía híbrida. En este capítulo hemos visto los diferentes sistemas de captación y digitalización de datos sensoriales, y en qué formato se suelen entregar. No es necesario, aunque sí conveniente, conocer desde el punto de vista electrónico los dispositivos sensoriales, pero con el enfoque que en este capítulo se ha dado, será más facil la elección de los sistemas sensoriales de un hipotético sistema que, como Ingeniero, tengas que diseñar. Evidentemente es necesario conocer a fondo las especificaciones tanto técnicas como económicas de la aplicación y elegir en consecuencia. También creemos que es conveniente saber el formato las estructuras de datos más utilizadas y más fácilmente manipulables mediante una computadora. En los siguientes capítulos se explicará qué hacer con los datos aquí obtenidos.

Technical Specifications for

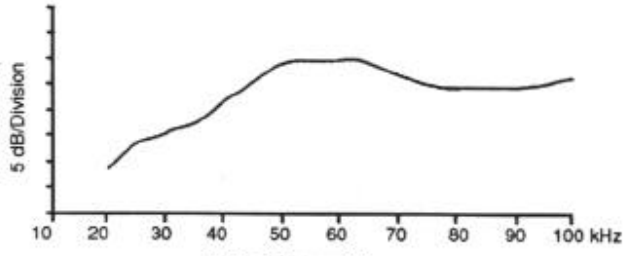
600 Series
Instrument Grade Electrostatic Transducer

This Instrument Grade electrostatic transducer is specifically intended for operation in air at ultrasonic frequencies. The assembly comes complete with a perforated protective cover.

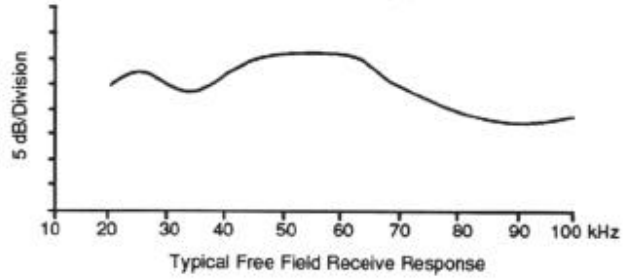


Specifications:

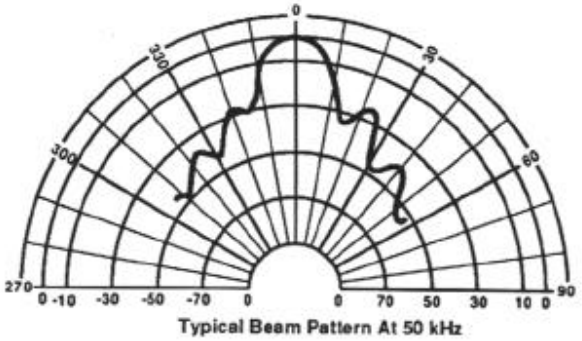
Usable Transmitting Frequency Range	See Graph	Maximum Combined Voltage	400V
Usable Receiving Frequency Range	See Graph	Capacitance at 1 kHz (Typical)	400-500 pf
Beam Pattern	See Graph	150 vdc bias	
Minimum Transmitting Sensitivity at 50 kHz	110 dB	Operating Conditions	
300 vac pk-pk, 150 vdc bias		Temperature	-20°-160°F
(dB re 20 µPa at 1 meter)		Relative Humidity	5%-95%
Minimum Receiving Sensitivity at 50 kHz	–42dB	Standard Finish	
150 vdc bias (dB re lv/Pa)		Foil	Gold
		Housing	Flat Black
Suggested DC Bias Voltage	150V		Cold Roll Steel
Suggested AC Driving Voltage (peak)	150V	Specifications subject to change without notice.	



Typical Transmit Response



Typical Free Field Receive Response



Typical Beam Pattern At 50 kHz

Note: dB normalized to on-axis response.

Note: Curves are representative only. Individual responses may differ.

Technical Specifications for

Polaroid

6500 Series Sonar Ranging Module

Features:

- Accurate Sonar Ranging from 6" to 35 ft.
- Drives 50 kHz Electrostatic Transducer with No Additional Interface
- Operates from Single Supply
- Accurate Clock Output Provided for External Use
- Selective Echo Exclusion
- TTL-Compatible
- Multiple Measurement Capability
- Uses TI TL851 and Polaroid 614906 Sonar Chips
- Socketed Digital Chip
- Convenient Terminal Connector
- Variable Gain Control Potentiometer

Description:

The 6500 Series is an economical sonar ranging module that can drive all Polaroid electrostatic transducers with no additional interface. This module, with a simple interface, is able to measure distances from 6" to 35 ft. The typical absolute accuracy is ± 1% of the reading over the entire range.

This module has an external blanking input that allows selective echo exclusion for operation on a multiple echo mode. The module is able to differentiate echos from objects that are only 3" apart. The digitally controlled-gain, variable-bandwidth amplifier minimizes noise and side-lobe detection in sonar applications.

The module has an accurate ceramic resonator-controlled 420 kHz time-base generator. An output based on the 420 kHz time base is provided for external use. The sonar transmit output is 16 cycles at a frequency of 49.4 kHz.

The 6500 Series module operates over a supply voltage range from 4.5 volts to 6.8 volts and is characterized for operation from 0° C to 40° C.

Absolute Maximum Ratings:

Voltage from any pin to ground (see Note 1) ..... 7 V
Voltage from any pin except XDCR to Vcc (see Note 1) ..... -7 to 0.5 V
Operating free-air temperature range ..... 0° C to 40° C
Storage temperature range ..... -40° C to 85° C

NOTE 1: The XCDR pin may be driven from -1 volt to 400 volts typical with respect to ground

Specifications subject to change without notice.

Recommended Operating Conditions

		MIN.	MAX.	UNIT
Supply Voltage, Vcc		4.5	6.8	V
High-level Input Voltage, V_{IH}	BLNK, BINH, INIT	2.1		V
Low-level Input Voltage, V_{IL}	BLNK, BINH, INIT		0.6	V
ECHO and OSC Output Voltage			6.8	V
Delay Time, Power Up to INIT High		5		ms
Recycle Period		80	ms	
Operating Free-air Temperature, T_A		0	40	° C

6500 Series Sonar Ranging Module (Cont.)

Electrical Characteristics Over Recommended Ranges of Supply Voltage and Operating Free-Air Temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
Input Current	BLNK, BINH, INIT	V_I = 2.1 V			1	mA
High-level Output Current, I_{OH}	ECHO, OSC	V_{OH} = 5.5 V			100	µA
Low-level Output Voltage, V_{OL}	ECHO, OSC	I_{OL} = 1.6 mA			0.4	V
Transducer Bias Voltage		T_A = 25° C		200		V
Transducer Output Voltage (peak-to-peak)		T_A = 25° C		400		V
Number of Cycles for XDCR Output to Reach 400 V		C = 500 pF			7	
Internal Blanking Interval				2.38†		ms
Frequency During 16-pulse Transmit Period	OSC output			49.4†		kHz
	XMIT output			49.4†		
Frequency After 16-pulse Transmit Period	OSC output			93.3†		kHz
	XMIT output			0		
Supply Current, I_{CC}	During transmit period				2000	mA
	After transmit period				100	

†These typical values apply for a 420 kHz ceramic resonator.

Operation With Polaroid Electrostatic Transducer:

There are two basic modes of operation for the 6500 Series sonar ranging module: single-echo mode and multiple-echo mode. The application of power (Vcc), the activation of the Initiate (INIT) input, and the resulting transmit output, and the use of the Blanking Inhibit (BINH) input are basically the same for either mode of operation. After applying power (Vcc) a minimum of 5 milliseconds must elapse before the INIT input can be taken high. During this time, all internal circuitry is reset and the internal oscillator stabilizes. When INIT is taken high, drive to the Transducer XDCR output occurs. Sixteen pulses at 49.4 kHz with 400-volt amplitude will excite the transducer as transmission occurs. At the end of the 16 transmit pulses, a DC bias of 200 volts will remain on the transducer as recommended for optimum operation.

In order to eliminate ringing of the transducer from being detected as a return signal, the Receive (REC) input of the ranging control IC is inhibited by internal blanking for 2.38 milliseconds after the initiate signal. If a reduced blanking time is desired, then the BINH input can be taken high to end the blanking of the Receive input anytime prior to internal blanking. This may be desirable to detect objects closer than 1.33 ft. corresponding to 2.38 milliseconds and may be done if transducer damping is sufficient so that ringing is not detected as a return signal.

In the single-echo mode of operation (Figure 1), all that must be done next is to wait for the return of the transmitted signal, traveling at approximately 0.9 milliseconds per foot out and back. The returning signal is amplified and appears as a high-logic-level echo output. The time between INIT going high and the Echo (ECHO) output going high is proportional to the distance of the target from the transducer. If desired, the cycle can now be repeated by returning INIT to a low-logic level and then taking it high when the next transmission is desired.

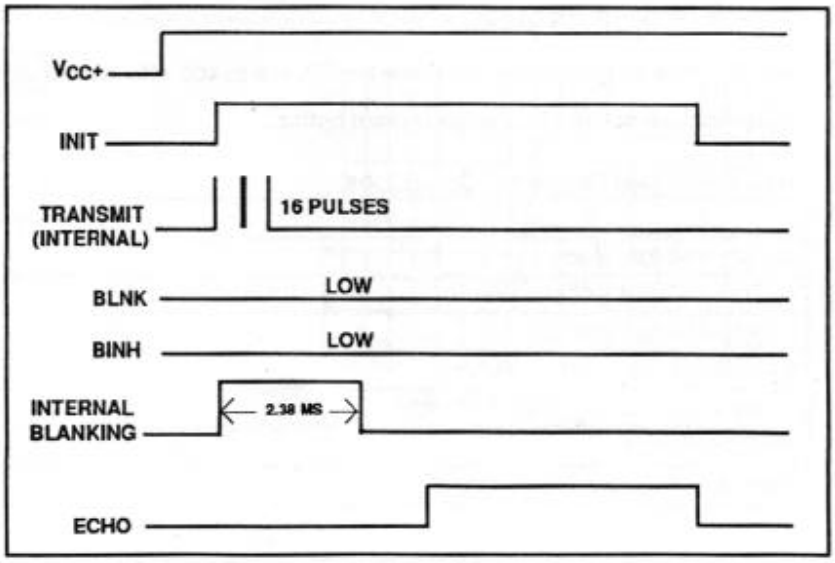
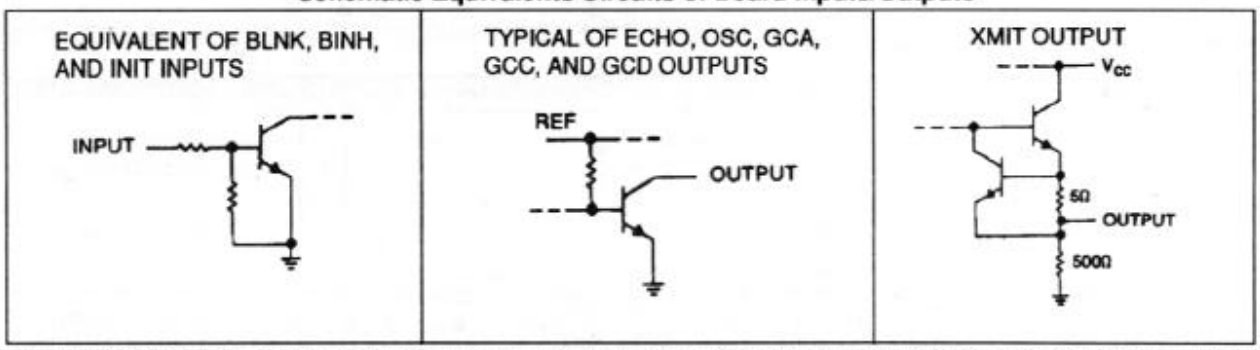


FIGURE 1: EXAMPLE OF A SINGLE-ECHO-MODE CYCLE WITHOUT BLANKING INPUT

6500 Series Sonar Ranging Module (Cont.)

Schematic Equivalents Circuits of Board Inputs/Outputs



INPUT/OUTPUT SCHEMATICS ***NOTE: The echo output is an open collector transistor output and requires a 4.7 k ohm pull up resistor between Vcc and the output.***

If there is more than one target and multiple echos will be detected from a single transmission, then the cycle is slightly different (Figure 2). After receiving the first return signal which causes the ECHO output to go high, the Blanking (BLNK) input must be taken high then back low to reset the ECHO output for the next return signal. The blanking signal must be at least 0.44 milliseconds in duration to account for all 16 returning pulses from the first target and allow for internal delay times. This corresponds to the two targets being 3" apart.

During a cycle starting with INIT going high, the receiver amplifier gain is incremented higher at discrete times (Figure 3) since the transmitted signal is attenuated with distance. At approximately 38 milliseconds, the maximum gain is attained. For this reason, sufficient gain may not be available for objects greater than 35 ft. away. Although gain can be increased by varying R1 (Figure 4), there is a limit to which the gain can be increased for reliable module operation. This will vary from application to application. The modules are "kitted" prior to their final test during manufacture. This is necessary because the desired gain distribution is much narrower than the module gain distribution if all were kitted with one value resistor. As kitted, these modules will perform satisfactorily in most applications. As a rule of thumb, the gain can be increased up to a factor of 4, if required, by increasing R1 correspondingly. Gain is directly proportional to R1.

Potentiometer VR1 (Figure 4) provides an interstage gain adjustment for the module. It can be used to trim the overall range of gain set by fix resistor R1.

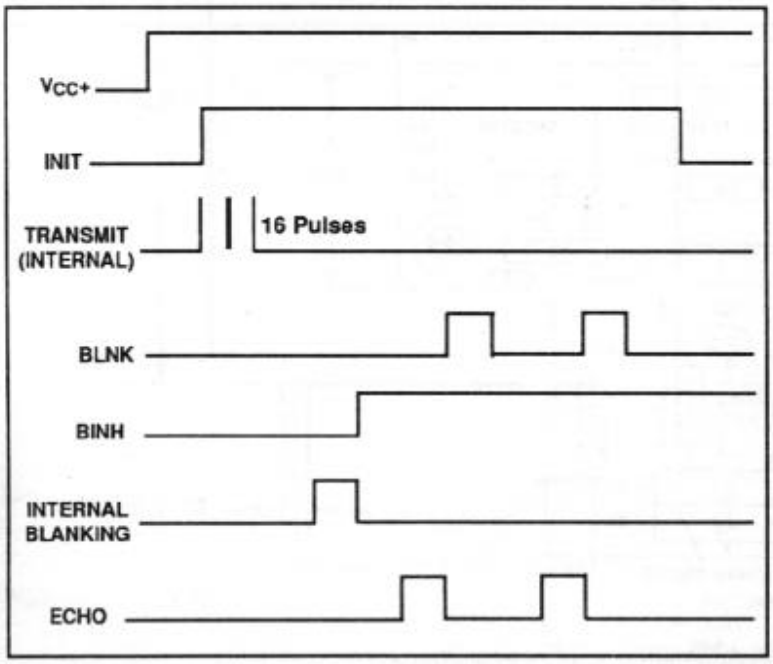


FIGURE 2: EXAMPLE OF A MULTIPLE-ECHO-MODE CYCLE WITH BLANKING INPUT

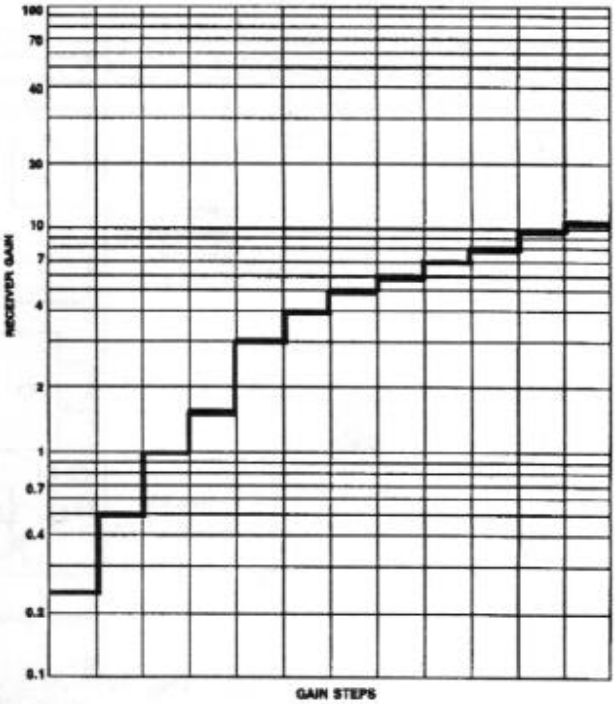


FIGURE 3: RECEIVER GAIN VS GAINSTEP NUMBERS

6500 Series Sonar Ranging Module (Cont.)

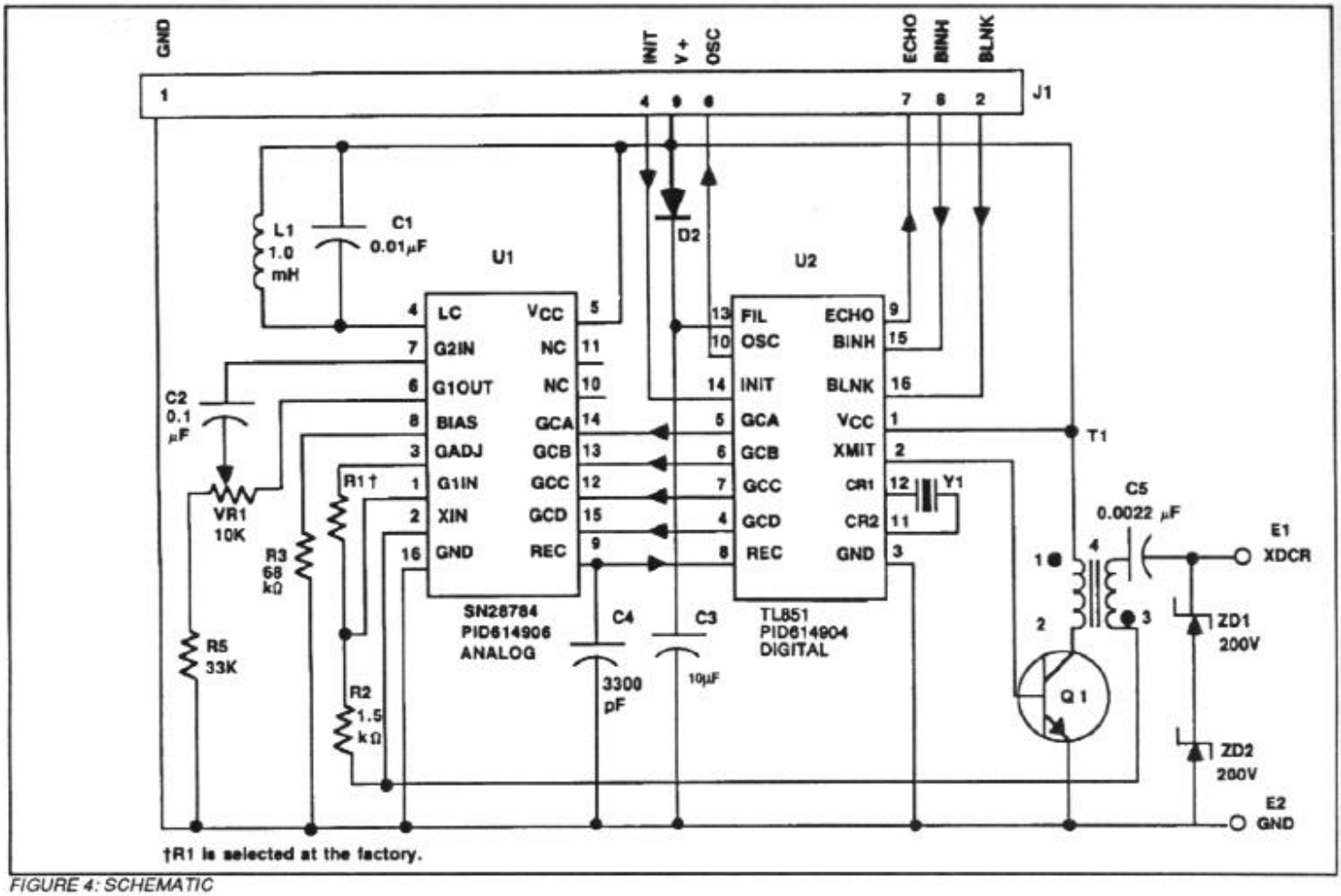


FIGURE 4: SCHEMATIC

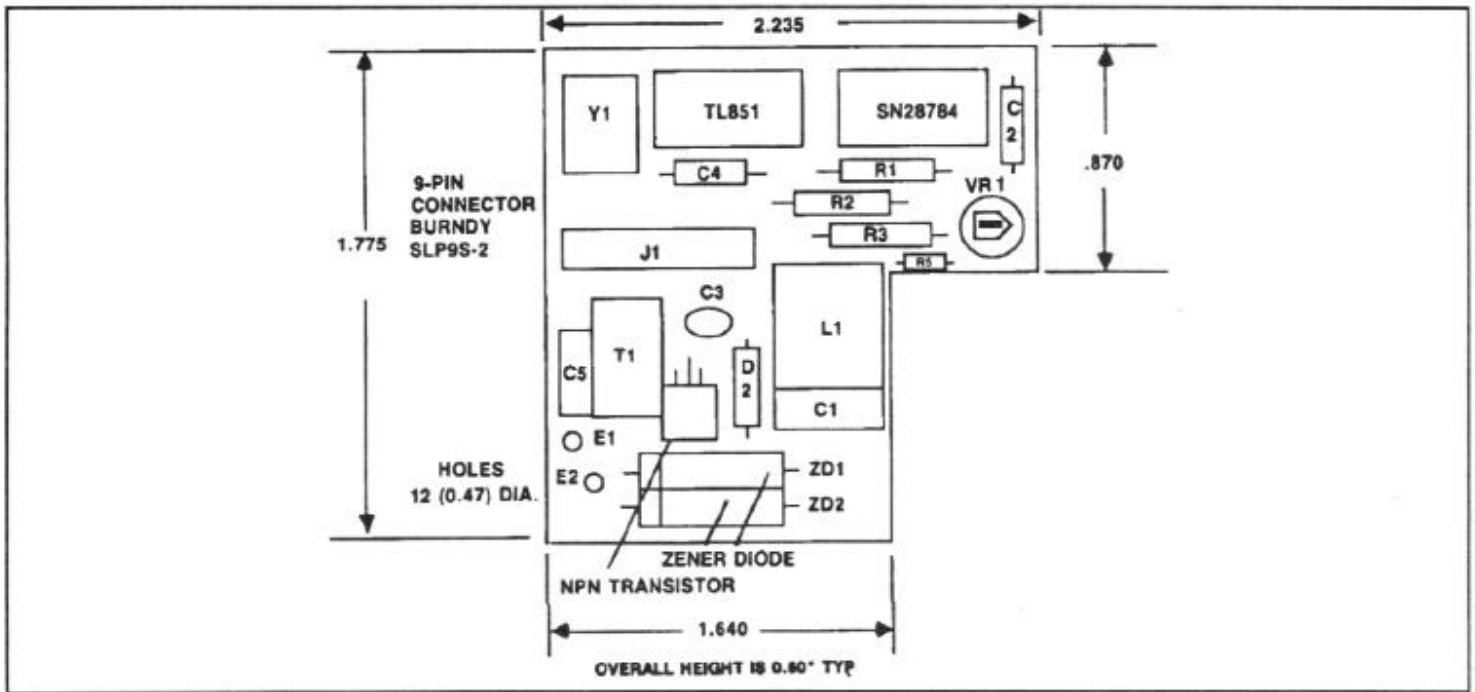
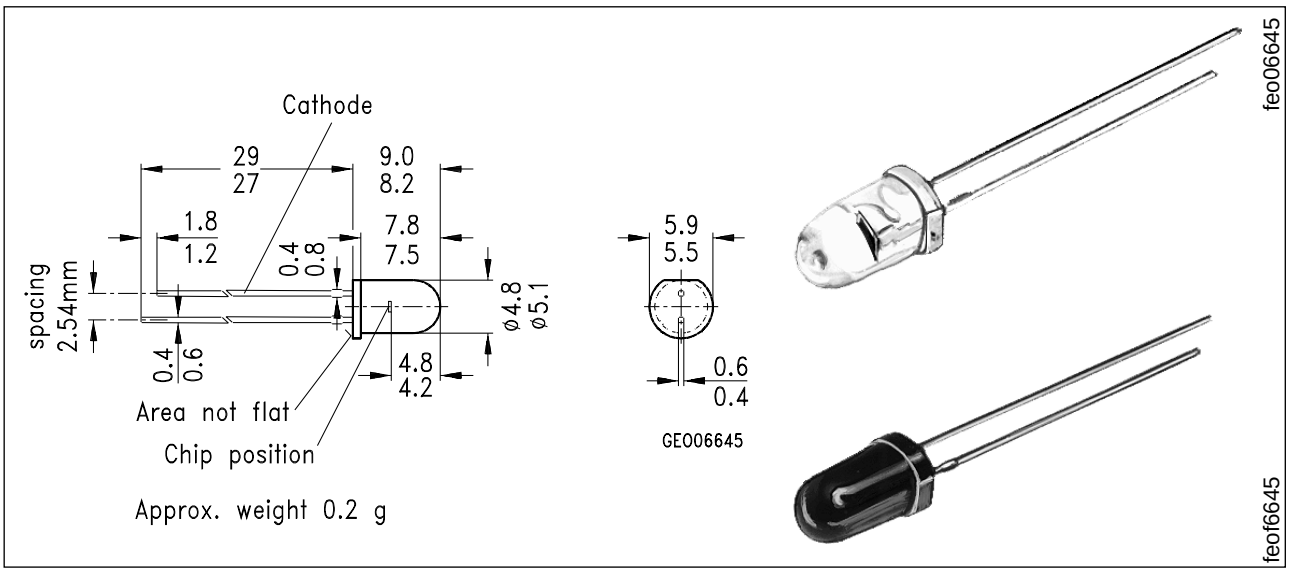


FIGURE 5: COMPONENT LAYOUT AND DIMENSIONS OF MODULE

SIEMENS

Silizium-PIN-Fotodiode mit sehr kurzer Schaltzeit — SFH 203
Silicon PIN Photodiode with Very Short Switching Time — SFH 203 FA



Maße in mm, wenn nicht anders angegeben/Dimensions in mm, unless otherwise specified.

Wesentliche Merkmale

- Speziell geeignet für Anwendungen im Bereich von 400 nm bis 1100 nm (SFH 203) und bei 880 nm (SFH 203 FA)
- Kurze Schaltzeit (typ. 5 ns)
- 5 mm-Plastikbauform im LED-Gehäuse
- Auch gegurtet lieferbar

Features

- Especially suitable for applications from 400 nm to 1100 nm (SFH 203) and of 880 nm (SFH 203 FA)
- Short switching time (typ. 5 ns)
- 5 mm LED plastic package
- Also available on tape

Anwendungen

- Industrieelektronik
- “Messen/Steuern/Regeln”
- Schnelle Lichtschranken für Gleich- und Wechsellichtbetrieb
- LWL

Applications

- Industrial electronics
- For control and drive circuits
- Photointerrupters
- Fiber optic transmission systems

Typ (*vorher) Type (*formerly)	Bestellnummer Ordering Code
SFH 203 (*SFH 2030)	Q62702-P955
SFH 203 FA (*SFH 2030 F)	Q62702-P956

Grenzwerte
Maximum Ratings

Bezeichnung **Description**	**Symbol** **Symbol**	**Wert** **Value**	**Einheit** **Unit**
Betriebs- und Lagertemperatur Operating and storage temperature range	T_{op}; T_{stg}	– 55 ... + 100	°C
Löttemperatur (Lötstelle 2 mm vom Gehäuse entfernt bei Lötzeit $t \leq 3$ s) Soldering temperature in 2 mm distance from case bottom ($t \leq 3$ s)	T_S	230	°C
Sperrspannung Reverse voltage	V_R	50	V
Verlustleistung Total power dissipation	P_{tot}	100	mW

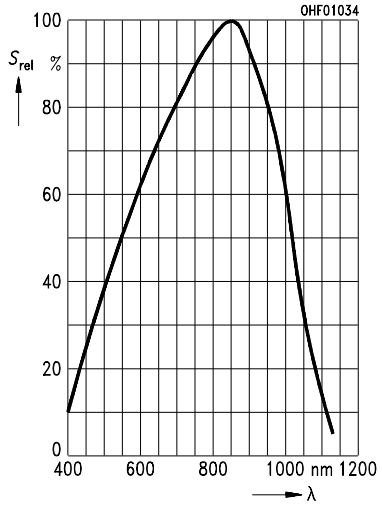
Kennwerte (T_A = 25 °C)
Characteristics

Bezeichnung **Description**	**Symbol** **Symbol**	**Wert** **Value**		**Einheit** **Unit**
		SFH 203	**SFH 203 FA**	
Fotoempfindlichkeit Spectral sensitivity V_R = 5 V, Normlicht/standard light A, T = 2856 K, V_R = 5 V, λ = 950 nm, E_e = 1 mW/cm^2	S S	80 (≥ 50) –	– 50 (≥ 30)	nA/lx µA
Wellenlänge der max. Fotoempfindlichkeit Wavelength of max. sensitivity	$\lambda_{S\ max}$	850	900	nm
Spektraler Bereich der Fotoempfindlichkeit S = 10 % von S_{max} Spectral range of sensitivity S = 10 % of S_{max}	λ	400 ... 1100	800 ... 1100	nm
Bestrahlungsempfindliche Fläche Radiant sensitive area	A	1	1	mm^2
Abmessung der bestrahlungsempfindlichen Fläche Dimensions of radiant sensitive area	$L \times B$ $L \times W$	1 × 1	1 × 1	mm × mm
Abstand Chipoberfläche zu Gehäuseoberfläche Distance chip front to case surface	H	4.0 ... 4.6	4.0 ... 4.6	mm

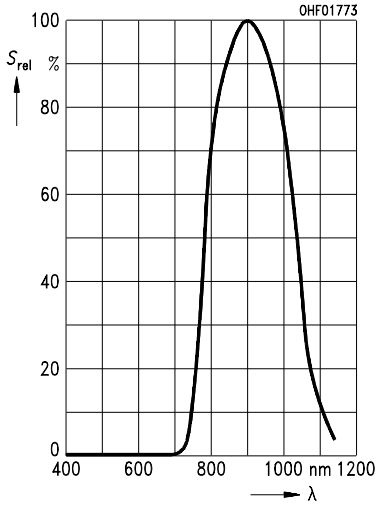
Kennwerte (T_A = 25 °C)
Characteristics (cont'd)

Bezeichnung **Description**	**Symbol** **Symbol**	**Wert** **Value**		**Einheit** **Unit**
		SFH 203	**SFH 203 FA**	
Halbwinkel Half angle	φ	± 20	± 20	Grad deg.
Dunkelstrom, V_R = 20 V Dark current	I_R	1 (≤ 5)	1 (≤ 5)	nA
Spektrale Fotoempfindlichkeit, λ = 850 nm Spectral sensitivity	S_λ	0.62	0.59	A/W
Quantenausbeute, λ = 850 nm Quantum yield	η	0.89	0.86	Electrons Photon
Leerlaufspannung Open-circuit voltage E_v = 1000 lx, Normlicht/standard light A, T = 2856 K E_e = 0.5 mW/cm^2, λ = 950 nm	V_O V_O	420 (≥ 350) –	– 370 (≥ 300)	mV mV
Kurzschlußstrom Short-circuit current E_v = 1000 lx, Normlicht/standard light A, T = 2856 K E_e = 0.5 mW/cm^2, λ = 950 nm	I_{SC} I_{SC}	80 –	– 25	µA µA
Anstiegs- und Abfallzeit des Fotostromes Rise and fall time of the photocurrent R_L = 50 Ω; V_R = 20 V; λ = 850 nm; I_p = 800 µA	t_r, t_f	5	5	ns
Durchlaßspannung, I_F = 80 mA, E = 0 Forward voltage	V_F	1.3	1.3	V
Kapazität, V_R = 0 V, f = 1 MHz, E = 0 Capacitance	C_0	11	11	pF
Temperaturkoeffizient von V_O Temperature coefficient of V_O	TC_V	– 2.6	– 2.6	mV/K
Temperaturkoeffizient von I_{SC} Temperature coefficient of I_{SC} Normlicht/standard light A λ = 950 nm	TC_I	0.18 –	– 0.2	%/K
Rauschäquivalente Strahlungsleistung Noise equivalent power V_R = 20 V, λ = 850 nm	NEP	2.9×10^{-14}	2.9×10^{-14}	$\frac{W}{\sqrt{Hz}}$
Nachweisgrenze, V_R = 20 V, λ = 850 nm Detection limit	D*	3.5×10^{12}	3.5×10^{12}	$\frac{cm \cdot \sqrt{Hz}}{W}$

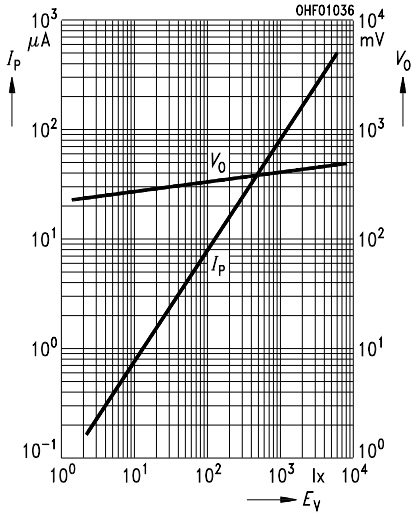
Relative spectral sensitivity SFH 203
$S_{rel} = f\ (\lambda)$



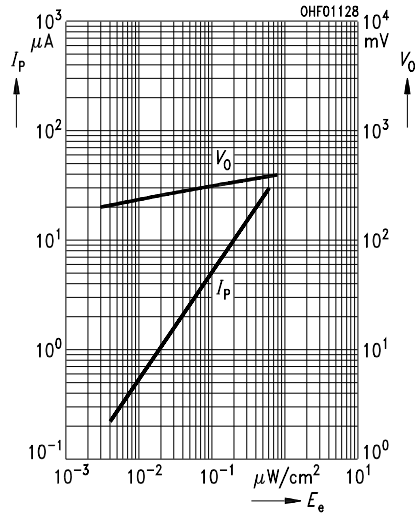
Relative spectral sensitivity SFH 203 FA
$S_{rel} = f\ (\lambda)$



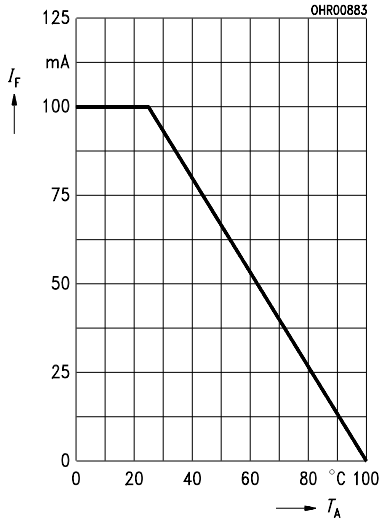
Photocurrent $I_P = f\ (E_V)$, V_R = 5 V
Open-circuit voltage $V_O = f\ (E_V)$
SFH 203



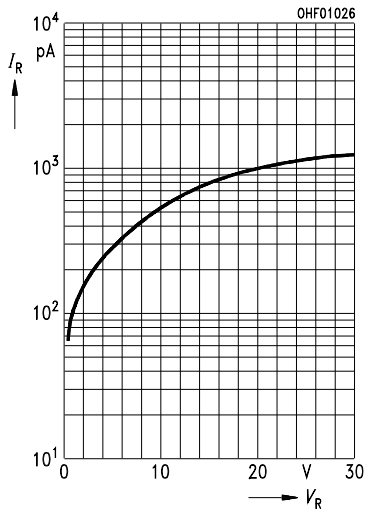
Photocurrent $I_P = f\ (E_e)$, V_R = 5 V
Open-circuit-voltage $V_O = f\ (E_e)$
SFH 203 FA



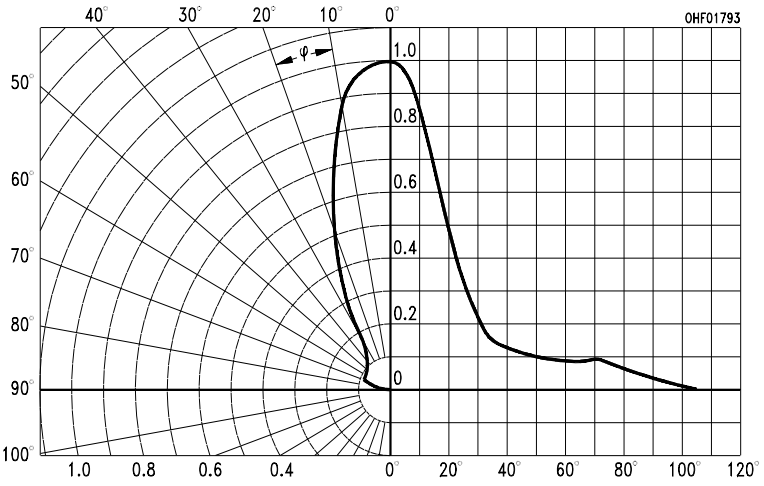
Total power dissipation
$P_{tot} = f\ (T_A)$



Dark current
$I_R = f\ (V_R)$, E = 0

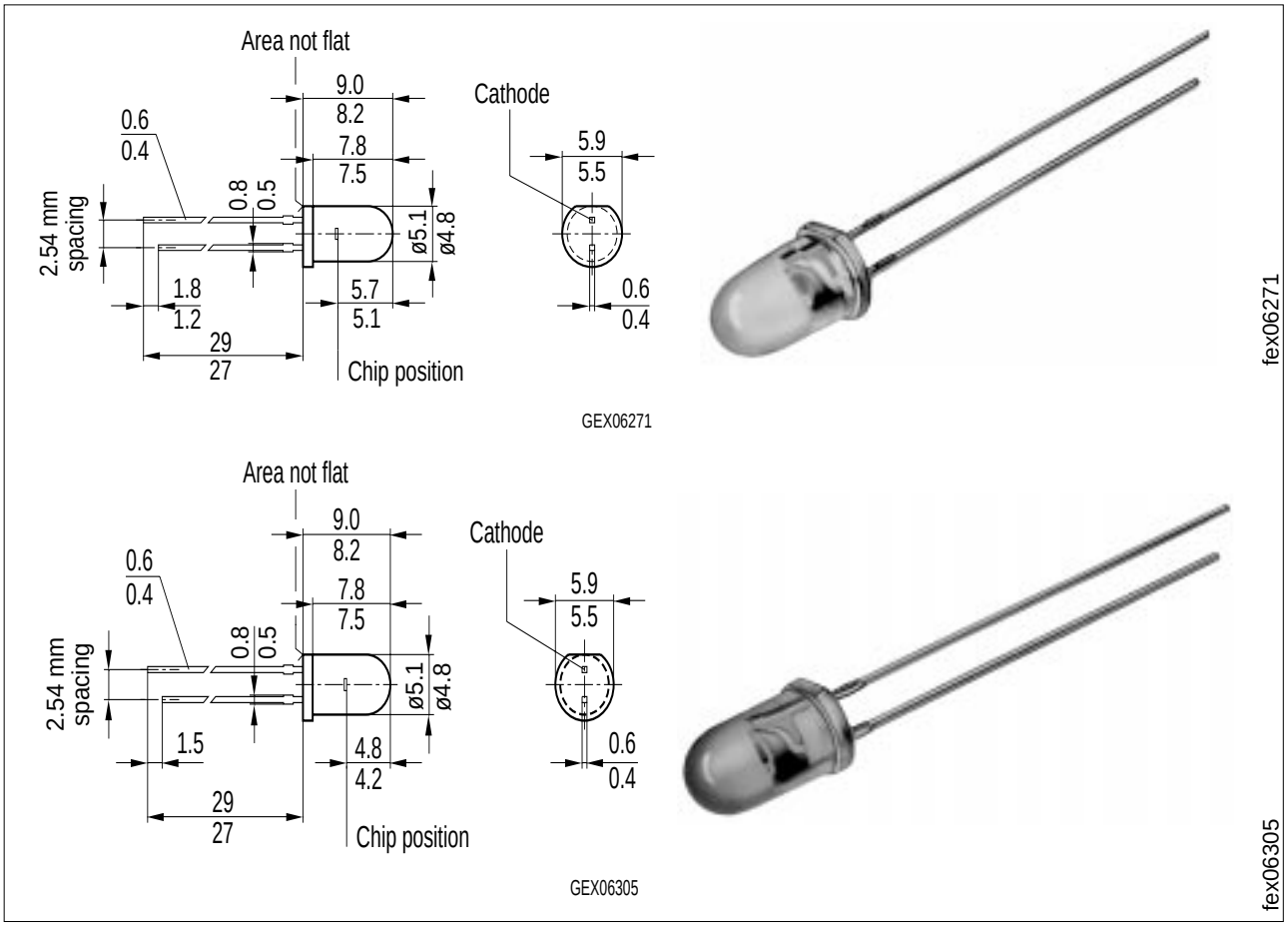


Directional characteristics $S_{rel} = f\ (\varphi)$



SIEMENS

GaAlAs-IR-Lumineszenzdioden (880 nm) SFH 484
GaAlAs Infrared Emitters (880 nm) SFH 485



Maße in mm, wenn nicht anders angegeben/Dimensions in mm, unless otherwise specified.

Wesentliche Merkmale

- Hergestellt im Schmelzepitaxieverfahren
- Hohe Zuverlässigkeit
- Gute spektrale Anpassung an Si-Fotoempfänger
- SFH 484: Gehäusegleich mit LD 274
- SFH 485: Gehäusegleich mit SFH 300, SFH 203

Features

- Fabricated in a liquid phase epitaxy process
- High reliability
- Spectral match with silicon photodetectors
- SFH 484: Same package as LD 274
- SFH 485: Same package as SFH 300, SFH 203

Anwendungen

- IR-Fernsteuerung von Fernseh- und Rundfunkgeräten, Videorecordern, Lichtdimmern
- Gerätefernsteuerungen für Gleich- und Wechsellichtbetrieb

Applications

- IR remote control of hi-fi and TV-sets, video tape recorders, dimmers
- Remote control for steady and varying intensity

Typ Type	Bestellnummer Ordering Code	Gehäuse Package
SFH 484	Q62703-Q1092	5-mm-LED-Gehäuse (T 1 $^3/_4$), klares violettes Epoxy-Gießharz, Anschlüsse im 2.54-mm-Raster ($^1/_{10}$''), Anodenkennzeichung: kürzerer Anschluß 5 mm LED package (T 1 $^3/_4$), violet-colored epoxy resin, solder tabs lead spacing 2.54 mm ($^1/_{10}$''), anode marking: short lead
SFH 484-1	Q62703-Q1755	
SFH 484-2	Q62703-Q1756	
SFH 485	Q62703-Q1093	
SFH 485-2	Q62703-Q1547	

Grenzwerte (T_A = 25 °C)
Maximum Ratings

Bezeichnung Description	Symbol Symbol	Wert Value	Einheit Unit
Betriebs- und Lagertemperatur Operating and storage temperature range	T_{op}; T_{stg}	– 40 ... + 100	°C
Sperrschichttemperatur Junction temperature	T_j	100	°C
Sperrspannung Reverse voltage	V_R	5	V
Durchlaßstrom Forward current	I_F	100	mA
Stoßstrom, t_p = 10 µs, D = 0 Surge current	I_{FSM}	2.5	A
Verlustleistung Power dissipation	P_{tot}	200	mW
Wärmewiderstand, freie Beinchenlänge max. 10 mm Thermal resistance, lead length between package bottom and PC-board max. 10 mm	R_{thJA}	375	K/W

Kennwerte (T_A = 25 °C)
Characteristics

Bezeichnung **Description**	**Symbol** **Symbol**	**Wert** **Value**	**Einheit** **Unit**
Wellenlänge der Strahlung Wavelength at peak emission I_F = 100 mA	λ_{peak}	880	nm
Spektrale Bandbreite bei 50 % von I_{rel} Spectral bandwidth at 50 % of I_{rel} I_F = 100 m A	$\Delta\lambda$	80	nm
Abstrahlwinkel Half angle SFH 484 SFH 485	 φ φ	 ± 8 ± 20	 Grad deg.
Aktive Chipfläche Active chip area	A	0.16	mm^2
Abmessungen der aktive Chipfläche Dimension of the active chip area	$L \times B$ $L \times W$	0.4 × 0.4	mm
Abstand Chipoberfläche bis Linsenscheitel Distance chip front to lens top SFH 484 SFH 485	 H H	 5.1 ... 5.7 4.2 ... 4.8	 mm mm
Schaltzeiten, I_e von 10 % auf 90 % und von 90 % auf 10 %, bei I_F = 100 mA, R_L = 50 Ω Switching times, I_e from 10 % to 90 % and from 90 % to10 %, I_F = 100 mA, R_L = 50 Ω	t_r, t_f	0.6/0.5	µs
Kapazität Capacitance V_R = 0 V, f = 1 MHz	C_o	25	pF
Durchlaßspannung Forward voltage I_F = 100 mA, t_p = 20 ms I_F = 1 A, t_p = 100 µs	 V_F V_F	 1.50 (≤ 1.8) 3.00 (≤ 3.8)	 V V
Sperrstrom Reverse current V_R = 5 V	I_R	0.01 (≤ 1)	µA
Gesamtstrahlungsfluß Total radiant flux I_F = 100 mA, t_p = 20 ms	Φ_e	25	mW

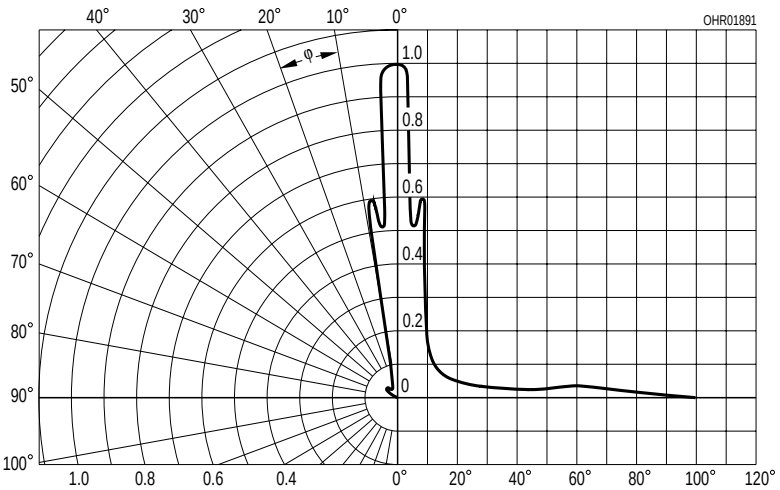
Kennwerte (T_A = 25 °C)
Characteristics (cont'd)

Bezeichnung Description	Symbol Symbol	Wert Value	Einheit Unit
Temperaturkoeffizient von I_e bzw. Φ_e, I_F = 100 mA Temperature coefficient of I_e or Φ_e, I_F = 100 mA	TC_I	– 0.5	%/K
Temperaturkoeffizient von V_F, I_F = 100 mA Temperature coefficient of V_F, I_F = 100 mA	TC_V	– 2	mV/K
Temperaturkoeffizient von λ, I_F = 100 mA Temperature coefficient of λ, I_F = 100 mA	TC_λ	0.25	nm/K

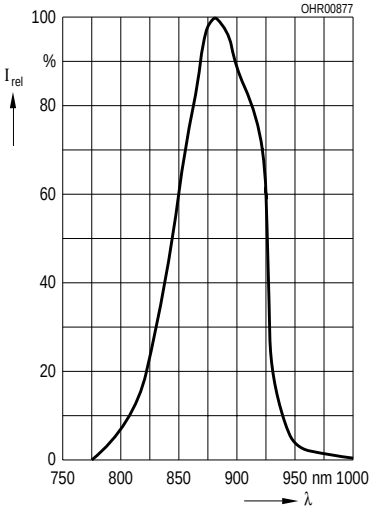
Strahlstärke I_e in Achsrichtung
gemessen bei einem Raumwinkel Ω = 0.001 sr bei SFH 484 bzw. Ω = 0.01 sr bei SFH 485
Grouping of radiant intensity I_e in axial direction
at a solid angle of Ω = 0.001 sr at SFH 484 or Ω = 0.01 sr at SFH 485

Bezeichnung Description	Symbol	Wert Value					Einheit Unit
		SFH 484	SFH 484-1	SFH 484-2	SFH 485	SFH 485-2	
Strahlstärke Radiant intensity I_F = 100 mA, t_p = 20 ms	$I_{e\ min}$ $I_{e\ max}$	50 160	50 100	> 80 –	16 80	> 25 –	mW/sr mW/sr
Strahlstärke Radiant intensity I_F = 1 A, t_p = 100 µs	$I_{e\ typ.}$	800	700	900	300	340	mW/sr

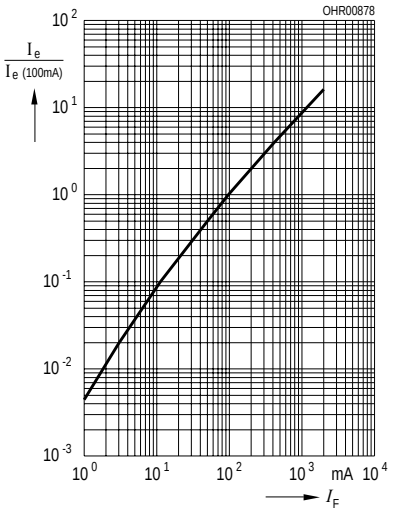
Radiation characteristics, SFH 484 $I_{rel} = f(\varphi)$



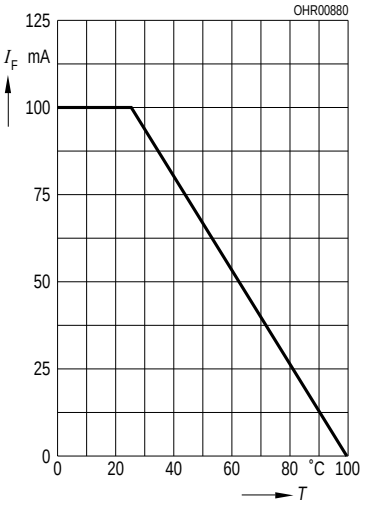
Relative spectral emission
$I_{rel} = f(\lambda)$



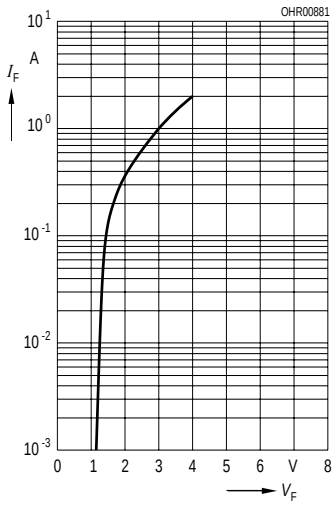
Radiant intensity $\frac{I_e}{I_e\ 100\ mA} = f\ (I_F)$
Single pulse, $t_p = 20\ \mu s$



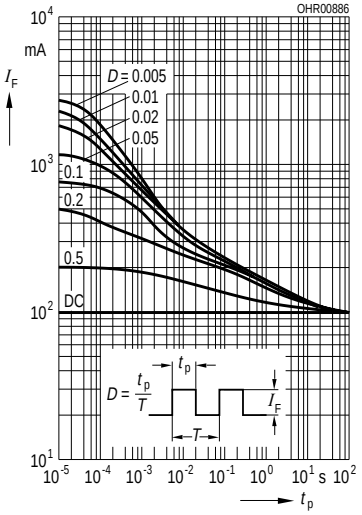
Max. permissible forward current
$I_F = f(T_A)$



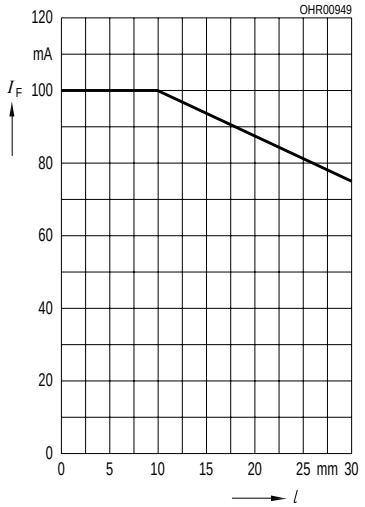
Forward current
$I_F = f(V_F)$, single pulse, $t_p = 20\ \mu s$



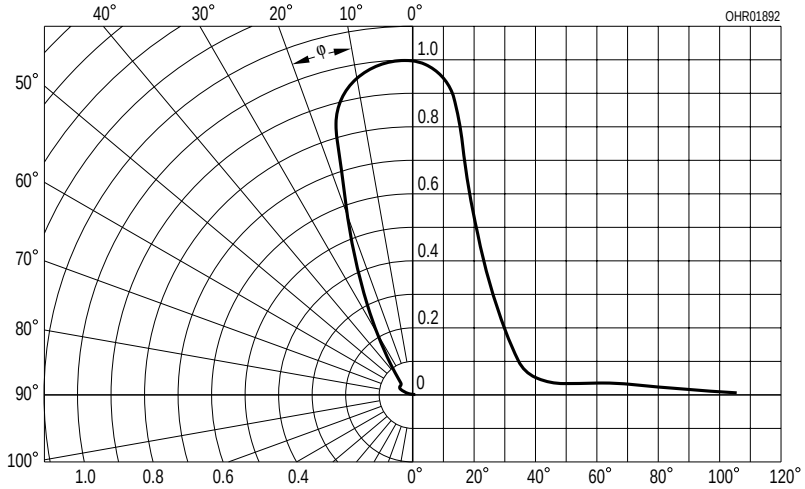
Permissible pulse handling capability
$I_F = f(\tau)$, $T_A = 25$ °C,
duty cycle D = parameter



Forward current versus lead length between the package bottom and the PC-board $I_F = f(l)$, $T_A = 25$ °C



Radiation characteristics, SFH 485 $I_{rel} = f(\varphi)$



Ejemplos sencillos de control en Robótica.

Félix de la Paz López.

17 de enero de 2007

1. Introducción.

Hasta ahora hemos visto los conceptos básicos sobre el control. Ahora vamos a realizar una serie de ejemplos prácticos utilizando la robótica como campo de acción. Para ello, no es necesario que dispongamos de un robot físico, ni si quiera de un simulador. De hecho, vamos a describir algoritmos de control que se puedan "dibujar" en un papel y que, posteriormente se puedan traducir a un lenguaje de programación para su ejecución en un simulador o en un robot real. En el apartado de captación y digitalización se os habló de como conseguir un simulador del robot Khepera(tm) de libre distribución. En este mismo capítulo ya vimos una descripción de los distintos dispositivos sensoriales, de sus funcionalidades y sus campos de aplicación. Estos dispositivos son los que permiten cerrar en el sistema el lazo de percepción-acción, donde toma sentido el concepto de realimentación explicado en este capítulo.

Esta realimentación se lleva a cabo, por ejemplo cuando los codificadores de posición situados en los ejes de los motores nos permiten comparar la distancia que ha girado la rueda, con la cantidad que hemos ordenado girar, o cuando los sensores de posición nos permiten comparar la distancia medida entre dos intervalos de tiempo, con el despalzamiento que se debía haber producido.

También se pueden establecer controles de tipo proporcional, del tipo que se estudia en teoría de control clásica y que hemos mencionado capítulo anterior, asignando a un motor una velocidad proporcional a la lectura de un sensor y, en general, combinaciones de esto con realimentación.

Sin embargo, en los ejemplos que aquí se describen no nos vamos a parar tanto en el detalle de los sitemas lineales en sí, sino en estrategias de control de un nivel más alto, que tienen más que ver con la supervisión que con el control en sí.

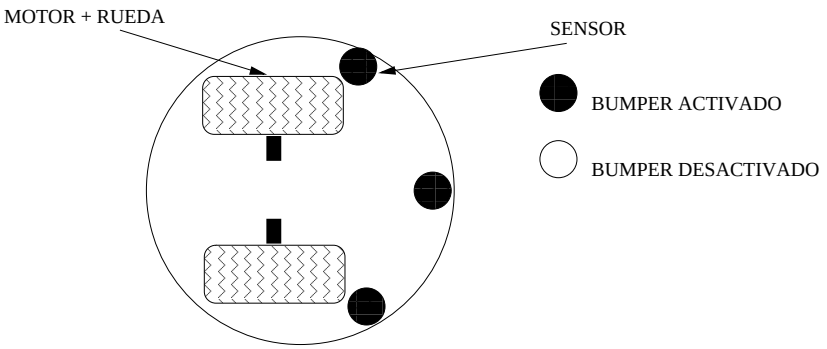


Figura 1: Vista superior esquemática del robot para ejemplos

2. El Robot.

Primero vamos a ver los elementos constructivos de nuestro robot imaginario (figura1) . Disponemos de motores ideales que transmiten a una rueda una velocidad constante v, hacia adelante o hacia atrás según el signo. También disponemos de dos tipos de sensores, de impacto y de proximidad (tipo sonar o infrarrojos) representados por un mismo círculo. El sensor de proximidad mide 0 cuando prácticamente hay colisión y 255 cuando es el máximo rango. El sensor de impacto mide 1 cuando hay colisión (circulo negro) y cero cuando no la hay (círculo blanco). Evidentemente estamos simplificando el problema e idealizando los sensores y motores para que no compliquen los algoritmos.

3. Ejemplos.

La estructura genérica de todos los ejemplos que vamos a desarrollar aquí es la que se puede observar en la figura 2. La planta esta formada por el conjunto de sensores y motores del sistema. El control se efectua por medio de un programa y es del tipo discreto en el tiempo. El programa controla una serie de parámetros que pasa a la planta y se realimenta con la salida de esta.

3.1. Ejemplo 0.

Para empezar vamos a simplificar al máximo el problema. Supongamos un robot con sólo un motor ideal que mueve solidariamente las dos ruedas y que, por lo tanto, sólo se puede desplazar en los dos sentidos de una dirección. Como sistema sensorial, dispone únicamente de un sensor de impacto y uno

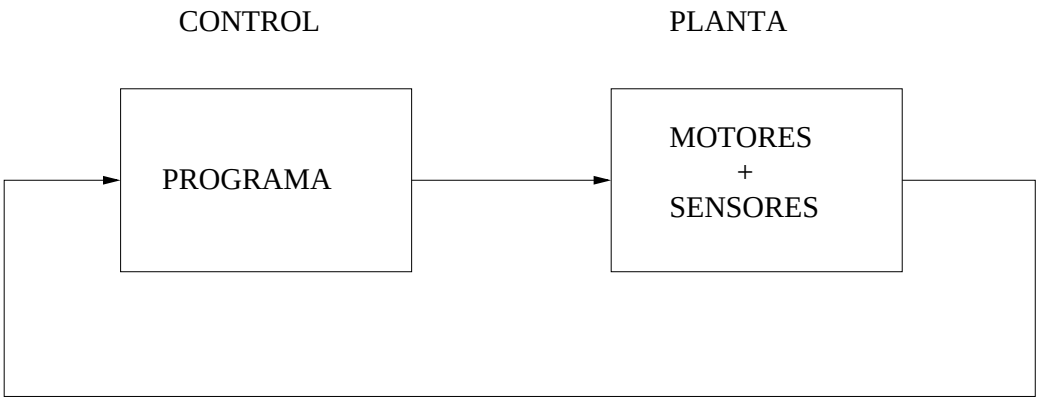


Figura 2: Estructura genérica del lazo de control en robótica.

de distancia situados en lo que consideraremos parte frontal. El algoritmo que vamos a diseñar tiene por objeto detener el robot cuando se encuentre a una distancia determinada de un objeto (figura 3).

La planta aquí es un sencillo sistema lineal de primer orden, con una entrada, una salida y un estado..

$$\begin{cases} \dot{x} = Bu \\ y = Cx \end{cases}$$

Donde $\boldsymbol{x}$ es la posición del robot (estado), su derivada, $\dot{x}$, es la velocidad del motor, B es la constante que relaciona la entrada u con la velocidad que se va a dar al motor, y es la salida igual al estado y C es la constante que incluye las dependencias del sensor en la salida codificada. Supondremos que el origen de referencia coincide con el valor cero del sensor.

Nuestro programa es bien sencillo.

```
Fijamos un parámetro de umbral para la entrada del sensor umbral_sensor
Fijamos una velocidad normal de avance v_normal que actúa un tiempo fijo
t_v
Si y >umbral_sensor , entonces parar (u=0)
En el otro caso, u=v_normal
Repetir.
```

Con esto conseguiremos que el robot quede a una distancia del obstáculo que viene dada por el umbral del sensor.

3.2. Ejemplo 1.

Siguiendo con la idea de ejemplos muy sencillos, vamos a añadir al sensor de distancias anterior, uno de impacto ("bumper"). El algoritmo que vamos a diseñar se va encargar de ajustar el umbral del sensor de distancia, para

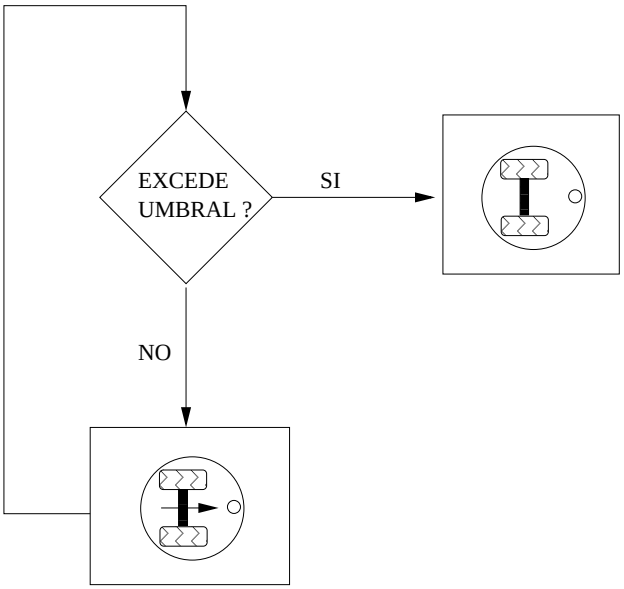


Figura 3: Diagrama conceptual y esquemático para visualizar el algoritmo del ejemplo 0.

que el robot se aproxime a un obstáculo y se detenga a una distancia de seguridad elegida.

Para ello, el algoritmo deberá de comprobar si hay o no impacto en el bumper y la lectura del sensor de distancia. Inicialmente, haremos que el robot se detenga si la lectura del sensor de distancia es cero (condiciones iniciales), pero que si, además hay impacto, retroceda una distancia y sume un valor fijo al valor del sensor de distancia para el cual el robot ha de detenerse. Con esto se consigue que el robot "aprenda" a no chocarse con la pared. Un esquema del algoritmo se puede ver en la figura 4.

Vamos a traducir este algoritmo a terminología clásica de control:

Entradas: La entrada del sistema es la velocidad del motor que mueve las ruedas.

Salidas: El sistema tiene dos variables de salida, el valor del sensor de proximidad y el valor del sensor de impacto.

Estado: Posición del sistema.

La planta vuelve a ser el conjunto de motores y sensores y el control lo hace el programa que ahora se complica algo pues hay que hacer dos comprobaciones (dos bifurcaciones condicionales). Tendremos ahora dos variables de salida,

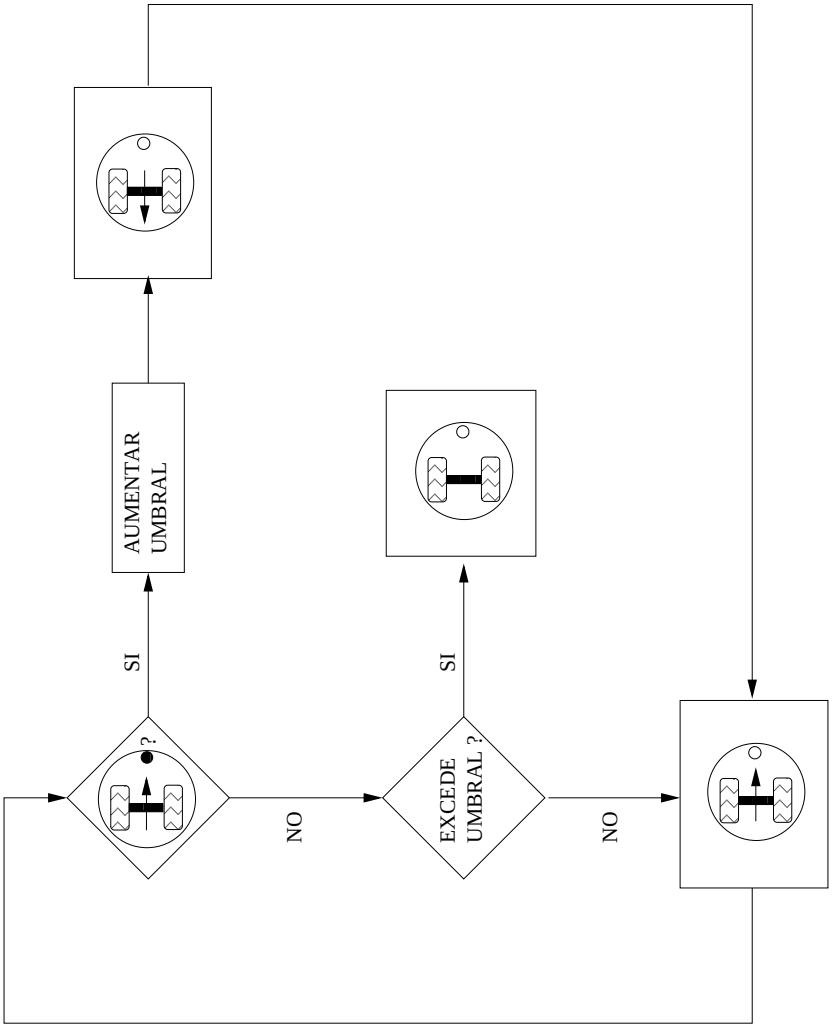


Figura 4: Diagrama conceptual y esquemático para visualizar el algoritmo del ejemplo 1.

la correspondiente al sensor de rango *y,* y la correspondiente al sensor de impacto que llamaremos *z.* Esta variable sólo puede tomar los valores 0 o 1 y su ecuación es del tipo *z=if(x=0)then, else....* , que no es lineal.

```
   Inicialmente, umbral_sensor = 0
   Definimos la funcion retrocede_seguridad que retrocede una distancia fija.
   Fijamos una velocidad normal de avance v_normal que actúa un tiempo fijo
t_v.
   Definimos una cantidad fija para sumar al umbral mas_umbral.
   Si z=1 entonces:
          umbral_sensor = umbral_sensor + mas_umbral
          retrocede_seguridad
   Si y >umbral_sensor , entonces parar (u=0)
   En el otro caso, u = v_normal
   Repetir.
```

3.3. *Ejemplo* 2.

Ahora vamos a complicar un poco el sistema. Supongamos un robot con tres sensores de distancia y con motores independientes para cada rueda. Vamos a describir un algoritmo que permita al robot navegar sin chocar.

En este caso las salidas son las lecturas de distancias de cada uno de los sensores *y_ i, y_ c, y_ d.* Las entradas son las velocidades de los motores. En cada caso, la entrada sensorial es comparada con una cierta magnitud umbral que el programador determina a priori. Si esta magnitud es superada se produce el giro durante un cierto tiempo t también puesto por el programador. Despues, el robot recupera la trayectoria rectilínea y se efectua otra vez el ciclo. En este caso no se utilizan los sensores de impacto.

En este ejemplo la descripción en forma de ecuaciones diferenciales se complica mucho, pero no son necesarias para entender el programa de control propuesto.

```
   Fijamos un parámetro de umbral para la entrada del sensor izquierdo
   umbral_sensor_i
   Fijamos un parámetro de umbral para la entrada del sensor central
   umbral_sensor_c
   Fijamos un parámetro de umbral para la entrada del sensor derecho
   umbral_sensor_d
   Fijamos una velocidad para el movimiento de los motores
   v_normal
   Definimos la función motor(v_izquierdo,v_derecho, t_motor),
   donde v_izquierdo y v_derecho son las velocidades de los motores y t_motor
el tiempo que actúa la función motor.
   Si y_i >umbral_sensor_izquierdo entonces
      motor(v_normal , 0 , t_motor)
         Si no, Si y_c >umbral_sensor_central entonces
             motor(-v_normal , v_normal, t_motor )
```

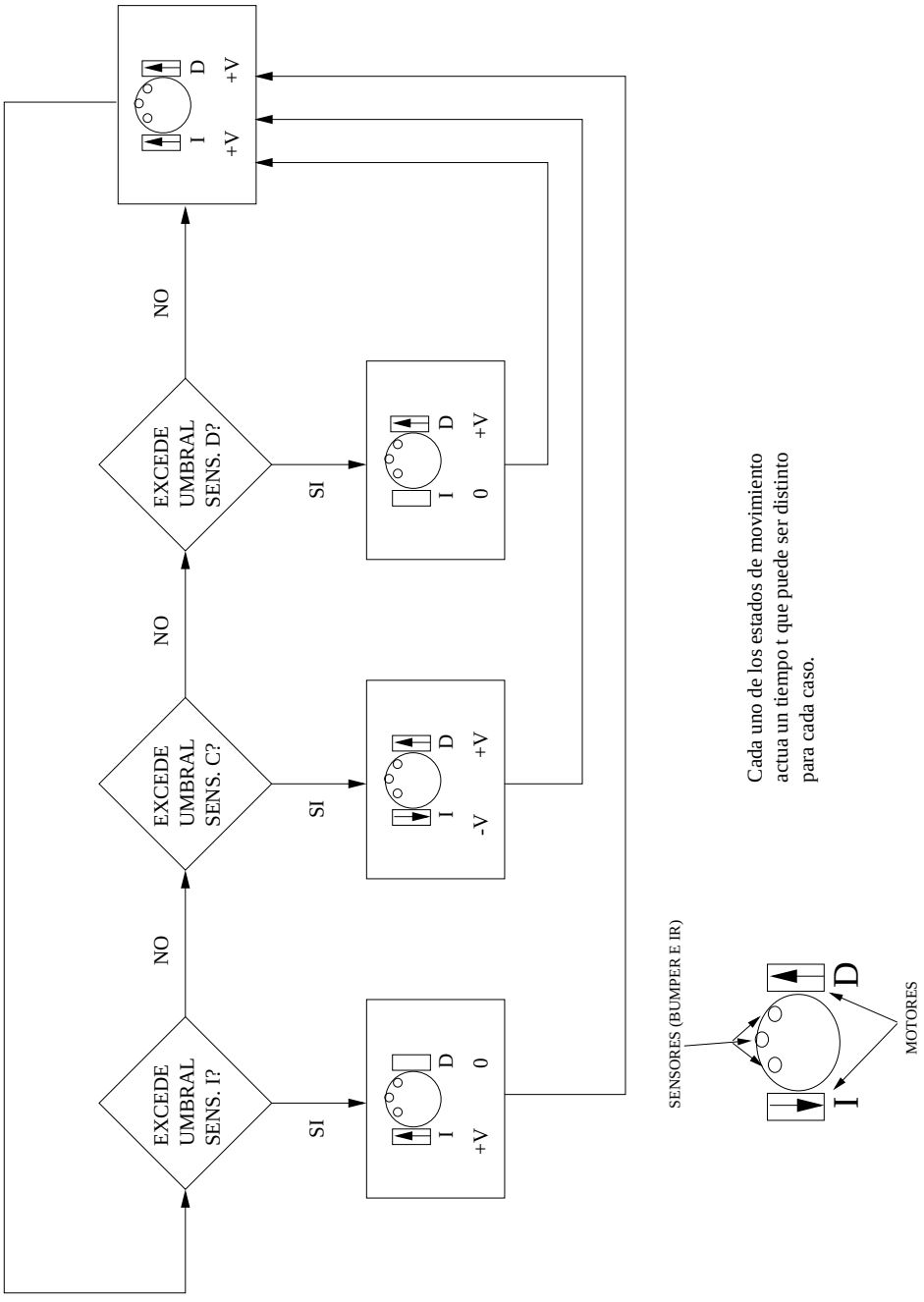


Figura 5: Diagrama conceptual y esquemático para visualizar el algoritmo del ejemplo 2.

```
            Si no, Si y_d >umbral_sensor_derecho entonces
               motor( 0, v_normal , t_motor )
motor( v_normal , v_normal, t_motor)
Repetir
```

La función *motor(v_izquierdo,v_derecho, t_motor)* es la u de nuestro sistema. Con este sencillo programa se controla el comportamiento del robot para evitar obstáculos (figura 5).

3.4. Ejemplo 3.

Todavía se puede complicar algo más el algoritmo anterior si intentamos que esa magnitud umbral no sea puesta por el programador sino ajustada por el propio robot durante la ejecución de programa (figura 6). Para ello es necesario usar sensores de impacto en combinación con los de distancia, de manera similar al ejemplo 1. El siguiente algoritmo utiliza el ajuste de esos tres parámetros de umbral para dar una mayor robustez al sistema ante casos en los que con el algoritmo anterior, el robot pudiera quedar trabado. La técnica aquí utilizada es conocida en el campo de la IA com "aprendizaje por refuerzo" aunque el nombre es un poco pretencioso y no deja de ser un método de ajuste paramétrico apoyado en la realimentación que produce la lectura continuada de los datos sensoriales.

La estrategia a seguir es ajustar todos los umbrales a cero. Evidentemente la primera vez el robot va a chocar, pero cada vez que lo haga, retrocederá una distancia fija y se sumará un cierto valor al umbral. Esta estrategia se repite para todos los sensores.

Un resultado interesante es que no todos los ángulos con los que el robot incide en un obstáculo producen el mismo umbral y, de hecho, si el umbral es fijo como en el caso anterior, o bien giramos mucho antes de llegar al obstáculo o bien el robot no es capaz de evitar todos los obstáculos para todos los ángulos. Con este método aseguramos que siempre se va a evitar el obstáculo.

Aquí las salidas del sistema son los valores de los sensores (tres bumpers y tres de distancia) y las entradas las velocidades de los motores. El sistema se realimenta mediante los sensores de bumper para poder modificar los valores de umbral que producen el giro.

3.5. Más ejemplos.

El robot se puede complicar tanto como uno quiera si es que queremos trabajar con este tipo de robot "imaginarios". Si vamos a trabajar con el robot del simulador Khepera(tm) tenemos que restringirnos a lo máximo que nos

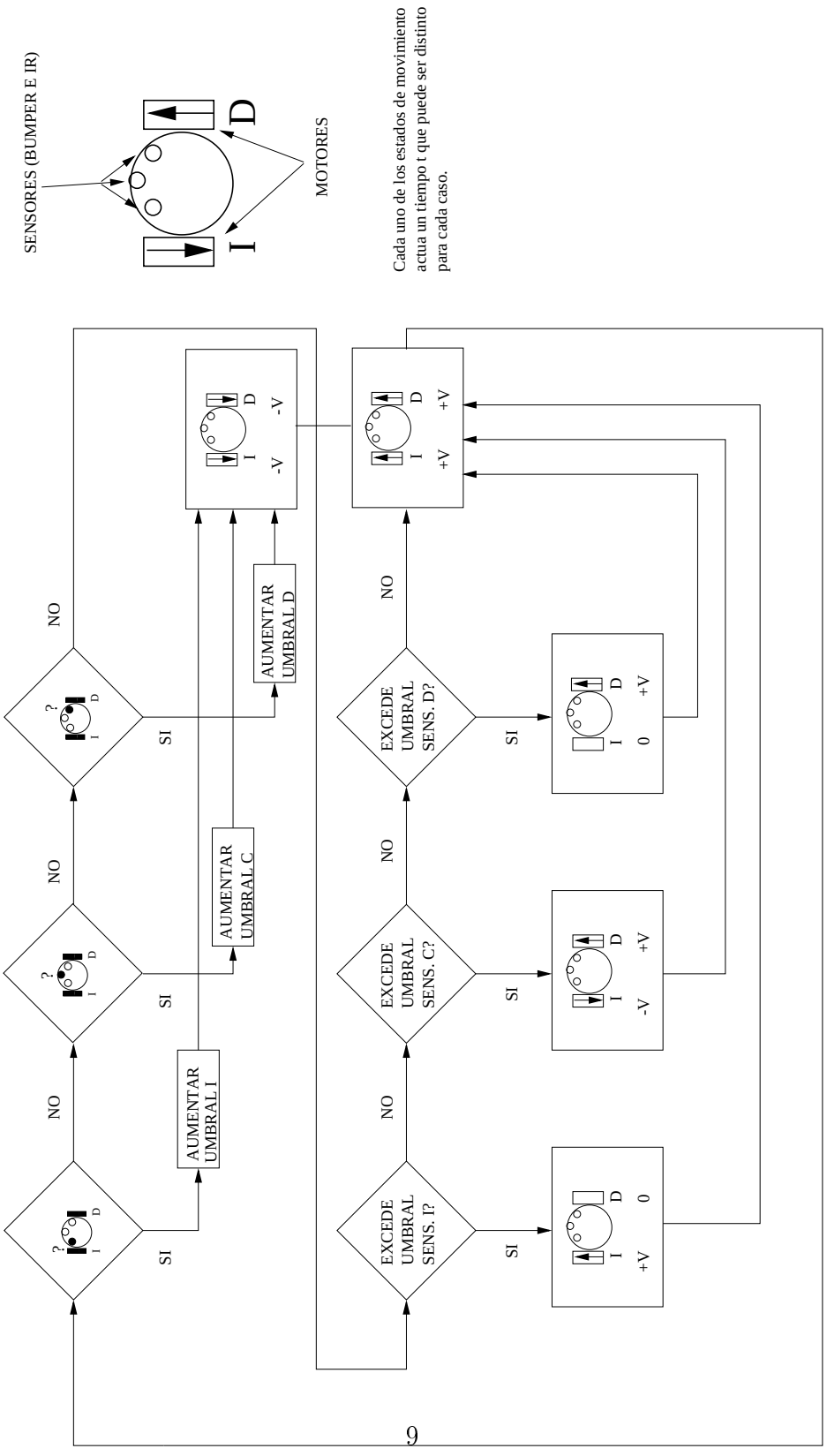


Figura 6: Diagrama conceptual y esquemático para visualizar el algoritmo del ejemplo 3.

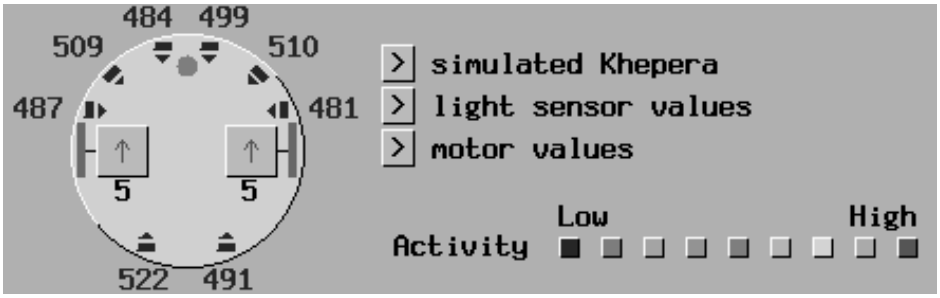


Figura 7: Modelo de Robot Khepera(tm) usado en el simulador.

pueda dar. En este caso (figura 7) el robot dispone de 8 sensores de distancia de infrarrojo y un bumper frontal . El ejemplo 2 sería directamente realizable en el simulador, simplemente obviando la lectura de varios sensores.

Los algoritmos se van complicando cuando aumenta el número de sensores y los diagramas que resultan pierden el carácter didáctico que aquí se pretende. Sin embargo es un buen ejercicio hacer un ejemplo en el que se usen todos los sensores, incidiendo en la posible interacción entre sensores vecinos y ver como estos mecanismos pueden mejorar los algoritmos probados para 2 o 3 sensores.

Dentro de los ejemplos que vienen con el simulador del robot Khepera(tm), hemos extraído el "EXAMPLE1" para comentarlo. El algoritmo esta programado en C usando las librerías que proporciona el propio simulador.

```
#define FORWARD_SPEED 5 /*define la velocidad de avance normal*/
#define TURN_SPEED 4 /* define la velocidad de giro normal*/
#define COLLISION_TH 900 /*umbral de colisión (valor de los sensores para los
que se considera que puede haber colisión*/

boolean StepRobot(struct Robot *robot)
{
   /*sensores de la parte frontal izquierda*/
  if ((robot->IRSensor[0].DistanceValue >COLLISION_TH) ||
      (robot->IRSensor[1].DistanceValue >COLLISION_TH) ||
      (robot->IRSensor[2].DistanceValue >COLLISION_TH))
/*si hay cloisión en la parte frontal izquierda*/
  {
    robot->Motor[LEFT].Value  =  TURN_SPEED;
    robot->Motor[RIGHT].Value = -TURN_SPEED; /* girar derecha*/
  }
/*idem pero en la otra parte*/
  else if ((robot->IRSensor[3].DistanceValue >COLLISION_TH) ||
           (robot->IRSensor[4].DistanceValue >COLLISION_TH) ||
```

```
            (robot->IRSensor[5].DistanceValue >COLLISION_TH))
   {
       robot->Motor[LEFT].Value  = -TURN_SPEED;
       robot->Motor[RIGHT].Value =  TURN_SPEED; /* girar izquierda */
     }
     else
     {
       robot->Motor[LEFT].Value  = FORWARD_SPEED;
       robot->Motor[RIGHT].Value = FORWARD_SPEED;  /*sigue recto*/
     }

     if ((robot->IRSensor[6].DistanceValue >COLLISION_TH)|| /*para la parte
trasera*/
          (robot->IRSensor[7].DistanceValue >COLLISION_TH))
      return(FALSE);  /* parar*/
     else
      return(TRUE);   /* continuar */
   }
```

En el programa se pueden observar tres bloques diferenciados. Cada bloque se corresponde con la comprobación de colisión por la parte izquierda, derecha y posterior respectivamente. La parte izquierda comprueba tres sensores y si hay posible colisión se hace girar el robot a la derecha. Lo mismo ocurre con la parte derecha, girando hacia la izquierda. Si la colisión es por la parte trasera, el robot se para.

En este ejemplo se han usado los ocho sensores que tiene el robot para realizar un algoritmo clásico de navegación evitando obstáculos. Es muy interesante ver sus virtudes y sus deficiencias en el simulador (figura 8), intentando plantear situaciones complicadas y viendo si es capaz de salir de ellas. La experiencia nos hara ver que es necesario complicar bastante el código (aumentando el número de "if's") para salvar ciertas situaciones.

3.6. Ejemplo de control con redes neuronales artificiales.

Al igual que hemos ilustrado de forma sencilla los programas de control que se usan en robots, podemos también usar con el mismo propósito didáctico, ejemplos sencillos de control de robots en los que intervienen redes de neuronas artificiales. La red Neuronal es un proceso paramétrico, ajustable que, partiendo de un conjunto de entrenamiento, permite clasificar este conjunto en una serie de clases linealmente separables.

El ejemplo aquí usado va a ser el "ejemplo 0" mencinado anteriormente. Un robot que se ha de aproximar a una pared y detenerse a una distancia. La

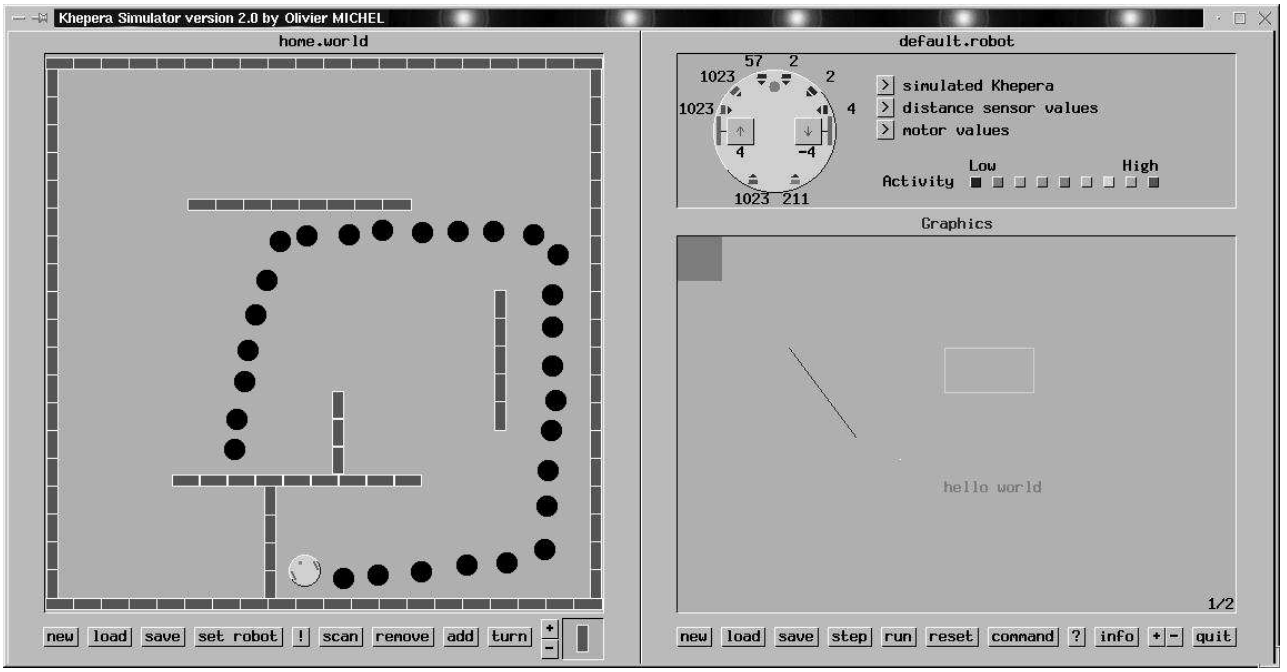


Figura 8: Trayectoria seguida por el robot en el simulador.

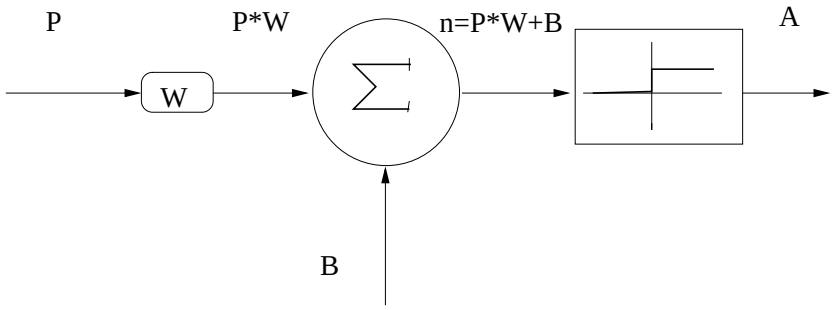


Figura 9: Neurona Artificial utilizada en el ejemplo.

red, va a ajustarse para "aprender" esa distancia. Supondremos que la red se entrena fuera del robot con un conjunto de entrenamiento y unos parámetros de peso y "bias"(desplazamiento de origen) elegidos adecuadamente y que, con posterioridad al aprendizaje, se pone a funcionar en el robot.

Supondremos que la variable de entrada es la posición del robot, de manera que éste se situa en el origen de una recta coordenada imaginaria y se tiene que parar a una distancia de 6 unidades. La pared se encuentra a 8 unidades. El control de la red lo hace a traves de 2 valores que conectan o desconectan el motor (1 conecta, 0 desconecta).

El vector de entrada será :$P = (1, 2, 3, 4, 5, 6, 7, 8)$

La salida esperada será: $T = (1, 1, 1, 1, 1, 0, 0, 0)$

El significado de esta dupla es que para $P = 1$,esto es, el robot esta en x=1 por lo tanto a 7 unidades de la pared, el robot debe de seguir avanzando, por lo tanto $T = 1$. Para $P = 6$, el robot se encuentra a 2 unidades de la pared y debe de detenerse, por lo tanto $T = 0$.

Esta dupla constituye el conjunto de entrenamiento.

La "neurona" que vamos a utilizar tiene una única entrada, un peso y un "bias"(despazamiento del origen).

La función de umbralización es del tipo abrupto, de tal manera que valores negativos o cero son convertidos en cero, y valores positivos son convertidos en uno.

Esta neurona computa su valor de la siguiente manera:

$n = P*W+B*I$, donde I es la matriz identidad, necesaria para mantener la coherencia de la ecuación matricial.

Los valores del peso y del bias se modifican con las ecuaciones matriciales:

$W_{nuevo} = W_{antigüo} + \alpha(T - A) * P$

$B_{nuevo} = B_{antigüo} + \alpha(T - A) * I$

En esta ecuación, A es la matriz resultante de evaluar la activación mediante la función umbral. Esta funcion, y para este caso, se ha elegido que

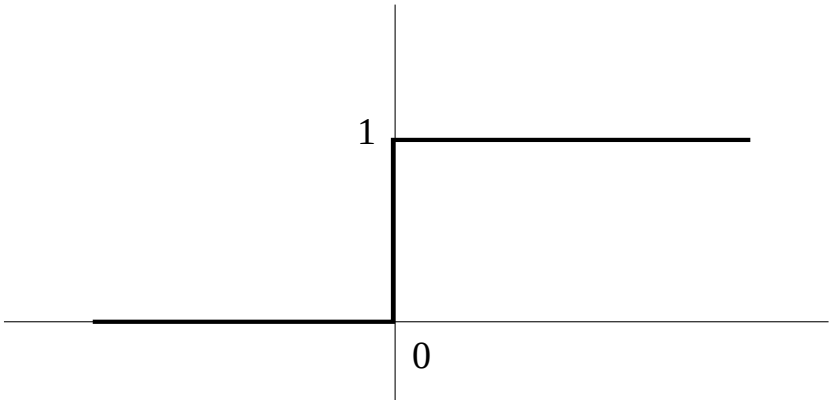


Figura 10: Función umbral para el ejemplo.

sea 1 si el valor es positivo y cero si es negatvo o cero (figura 10). El factor αse denomina, factor de aprendizaje, y regula cuanto afecta el error al nuevo valor del peso y el bias.

Evidentemente, este ejemplo es tan trivial que por simple inspección se puede inferir que con $B = 6$, $W = -1$ se consigue que la neurona funcione como queremos, pero, por motivos didácticos, vamos a usar valores próximos para ver como funciona el algoritmo de aprendizaje.

Si introducimos el conjunto de entrenamiento obtenemos, con $B = 5$, $W = -1$, $\alpha = 0{,}1$:

$P * W = (-1, -2, -3, -4, -5, -6, -7, -8)$

$n = (4, 3, 2, 1, 0, -1, -2, -3)$

$A = (1, 1, 1, 1, 0, 0, 0, 0)$

$[T - A] = (0, 0, 0, 0, 1, 0, 0, 0)$

Si efectuamos las correcciones:

$W = -1 + 0{,}5 = -0{,}5$

$B = 5 + 0{,}1 = 5{,}1$

Con estas nuevas By W, se vuelva a calcular $P * W$, n, A y $[T - A]$.

Este proceso se repite hasta conseguir un valor que cumpla con el conjunto de entrenemiento. Despues de hacerlo 5 veces más se obtiene:

$W = -1{,}1$

$B = 5{,}6$

Con estos valores se consigue el resultado esperado. Estos valores se fijan ahora y se utilizan como control para el robot.

La elección del conjunto de entrenamiento, los valores iniciales de W, B, α y la función de umbralización son cruciales en el diseño de la red. Hemos visto en este ejemplo tan simple que esta elección condiciona y mucho la convergencia y el número de pasos necesarios para llegar a un valor aceptable.

Las redes que se utilizan en control nunca son tan simples como éstas y elegir estos parámetros previos lleva gran parte del esfuerzo necesario. La fase de diseño es, por consiguiente, fundamental y es necesario el conocimiento del dominio de la aplicación para obtener resultados coherentes.

4. Conclusión

Hemos intentado poner de manifiesto una serie de ejemplos muy sencillos utilizando la robótica como campo de aplicación. Hemos hecho más énfasis en el programa controlador y hemos propuesto un diseño para cada caso. Evidentemente este diseño no es único y cada programador puede darle una orientación distinta. Sería muy conveniente que el alumno utilizara el simulador del robot Khepera (tm) que puede encontrarse en la dirección web:

`http://diwww.epfl.ch/lami/team/michel/khep-sim/index.html`

También se puede ampliar el conocimiento sobre el robot Khepera descargando el artículo:

"Mobile Robot Miniaturization: a Tool for Investigation in Control Algorithms" de F. Mondada, E. Franzi y P. Ienne, creadores del robot Khepera de la dirección web:

`ftp://lamiftp.epfl.ch/khepera/papers/mondada.ISER93.ps.Z`

El simulador funciona bajo GNU-Linux en el entorno gráfico X11. Tanto el sistema operativo Linux como el simulador son gratuitos y de libre distribución por lo que no supone coste alguno para el alumno trabajar en este area tan puntera de la Inteligencia Artificial Aplicada.

Para ampliar conocimientos es recomendable la lectura del libro *"Vehicles: Experiments in Synthetic Psychology"* de Valentino Braitenberg, publicado en MIT Press en 1986, que nos ha servido de inspiración para algunos de los ejemplos aquí descritos.

Control de sistemas

José Ramón Álvarez Sánchez

La teoría de control de sistemas es un área muy extensa que engloba varias disciplinas. En este material pretendemos dar al alumno de la asignatura de "Percepción y Control Basados en el Conocimiento" una base mínima, que le permita entender el control de sistemas, sin entrar en los detalles analíticos y matemáticos que caen fuera del alcance del contexto docente de la titulación a la que pertenece esta asignatura.

1. Conceptos básicos de control

Cuando hablamos de control, nos referimos al sistema de nuestro interés como *planta*, esta nomenclatura se utiliza en los textos de teoría de control en referencia a los procesos industriales o "plantas" industriales.

En todos los procesos industriales se intenta conseguir el máximo rendimiento posible sin superar los límites físicos de funcionamiento o de seguridad del propio sistema. Este objetivo se consigue *actuando* sobre elementos, que modifican la evolución y el comportamiento del sistema, llamados *actuadores*. Para calcular qué acciones hay que llevar a cabo a través de los actuadores necesitamos *observar* el sistema mediante elementos *sensores* para evaluar si la evolución del sistema corresponde a las condiciones deseadas de funcionamiento.

Las condiciones óptimas o deseadas de funcionamiento para el sistema pueden ir variando a lo largo del tiempo y por tanto se deben proporcionar al controlador las *consignas* de funcionamiento. Estas consignas son criterios del funcionamiento deseado y pueden tan simples como el valor deseado en alguna de las observaciones (valor de los sensores) directamente.

En la figura 1 hemos representado el sistema (planta o proceso industrial) como un medio o entorno que rodea al controlador. Los sensores y actuadores son la interfaz entre ambos. Al mismo tiempo el controlador puede recibir de otra fuente externa las consignas de funcionamiento deseado en la planta.

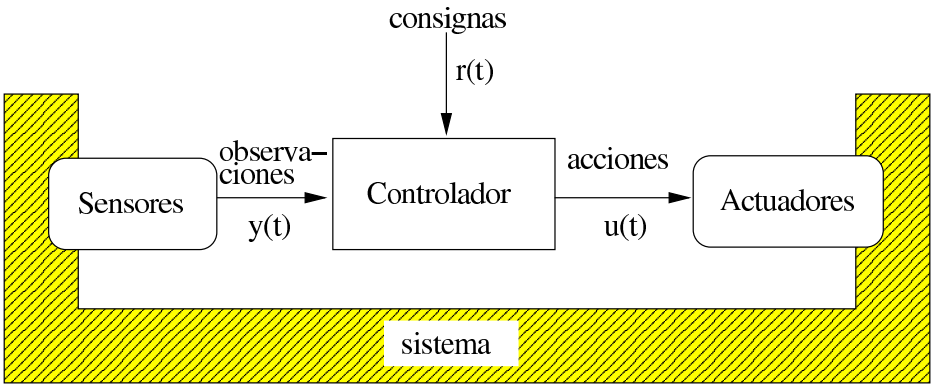


Figura 1: Situación de un controlador entre los sensores y los actuadores que lo conectan con el sistema.

1.1. Lazo de realimentación

La secuencia causal formada por el Controlador, los Actuadores, el Sistema y los Sensores se cierra en forma de lazo de realimentación[1]. La realimentación significa que las acciones solicitadas por el controlador afectan la evolución del sistema y que a su vez esto dará lugar a la modificación de las observaciones que afectan a los cálculos y decisiones del controlador.

En teoría de control este lazo de realimentación se suele dibujar de la forma esquemática que mostramos en la figura 2. Donde el controlador recibe las señales de consigna, y los valores de los sensores, y calcula los valores enviados a los actuadores. Habitualmente, en el tratamiento y diseño del controlador, se consideran los actuadores y los sensores formado parte de la planta, sustituyendo el conjunto por un modelo adecuado.

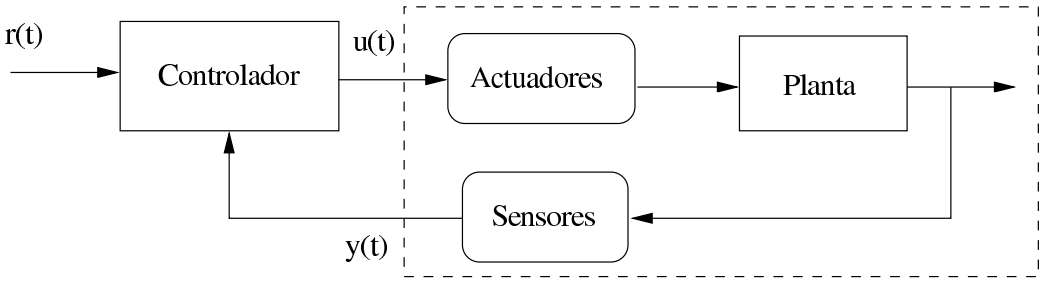


Figura 2: Forma habitual de representar esquemáticamente un controlador en un lazo de realimentación.

[1]En algunos textos también se puede encontrar con el nombre de *retroalimentación*, con el mismo significado.

Amplificador con realimentación negativa

Vamos a estudiar el caso más sencillo de controlador[2]. Supongamos que solamente observamos una variable de la planta y que sólo podemos actuar sobre otra variable en la planta, es decir tenemos un sistema con una entrada (acción) y una salida (observación). Además supondremos que la relación entre la variable de entrada y la de salida en el sistema es *positiva*, es decir que cuando crece la entrada también aumenta la salida (esto es lo más habitual) que cuando la entrada es nula, la salida se mantiene constante. Al mismo tiempo, como ya dijimos antes, el tipo de consigna más sencillo es el valor deseado de la variable observada, que implícitamente lleva asociada una función de optimización de mínimo error. Con estas condiciones, el método de control más sencillo es utilizar un amplificador con realimentación negativa, que consiste en calcular la diferencia (error) entre el valor deseado (consigna) y el valor observado, aplicando como control un valor proporcional (amplificador) a esa diferencia. La etiqueta de "negativa" hace referencia a que la acción siempre es de signo opuesto al exceso de la salida respecto a la consigna.

En la figura 3 hemos representado el esquema correspondiente a un controlador simple por amplificador con realimentación negativa. Se puede observar que la diferencia (error) $e(t) = r(t) - y(t)$ se ha representado, como es habitual en teoría de control, por un círculo con los signos correspondientes a cada entrada y que el amplificador se ha representado por un triángulo. El amplificador calcula la salida $u(t) = K_c \cdot e(t)$ proporcional al error con un parámetro K_c, llamado ganancia del controlador.

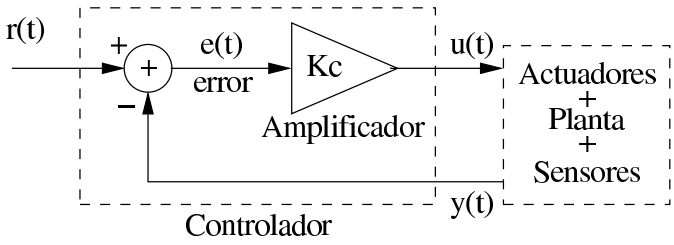


Figura 3: Esquema de un controlador por amplificador con realimentación negativa.

Al efectuar algún cambio en la consigna $r(t)$ se producirá una diferencia instantánea con el valor de salida que generará un valor $u(t)$ distinto de cero (de signo opuesto a la variación de la salida), haciendo de esta forma que la

[2]Como curiosidad, la utilización de la realimentación negativa fue propuesta por primera vez en 1929 (hace 70 años) por H. S. Black para reducir la distorsión en los amplificadores de válvulas.

entrada de la planta obligue a variar la salida de la planta hasta que vuelva a coincidir con la consigna, de forma que el error vuelva ser nulo.

1.2. Modelos de planta o de control

En el control simple mostrado antes, mediante amplificador con realimentación negativa, hemos impuesto algunas condiciones sobre el tipo de planta (una sola entrada y una salida con relación positiva y salida constante para entrada nula). Estas condiciones restringen los posibles tipos de planta que podríamos controlar con el tipo de controlador tan sencillo propuesto.

El comportamiento o evolución temporal de las variables de cualquier planta puede ser bastante más complejo. El aspecto más importante es que los cambios en la entrada de la planta pueden tardar un cierto *tiempo* en reflejarse en los cambios correspondientes en la salida. Además de los retardos, las variaciones inducidas por los cambios en la entrada no tienen porqué ser tan sencillos como un cambio proporcional. El sistema físico formado por los actuadores, la planta y los sensores podría recibir perturbaciones (desde el punto de vista del controlador) que no se estén observando (no hay sensores sobre todas las variables físicas). Estas perturbaciones pueden modificar la variable observada aunque el control $u(t)$ estuviera ya con un valor que compensara todo el error ($e(t) = 0$).

Si se observan los diagramas de las figuras 2 y 3 anteriores, podemos interpretar que, debido al cierre del lazo entre el control y la planta, el control realiza algo "similar" a la *función inversa de la planta*. Es decir, lo que para la planta es entrada, $u(t)$, para el control es salida y viceversa ($y(t)$), si consideramos que las consignas no varían (son constantes en el tiempo). Por lo tanto podríamos utilizar los modelos de la planta para hallar la función inversa correspondiente y así obtener el control adecuado. Desgraciadamente, las restricciones físicas o de coste habitualmente invalidan esa función inversa (cuando existe o se puede calcular) como función de control. Sí podemos deducir, en cambio, que los métodos que utilicemos para modelar y representar matemáticamente una planta se pueden usar para representar también los modelos del control.

Habitualmente, los modelos de la planta se realizan para *analizar* el comportamiento de la planta. En cambio, los modelos del controlador se realizan para *diseñar* un controlador adecuado a la planta analizada.

En la teoría de sistemas clásica se estudian las propiedades de los sistemas (planta y controlador) desde el punto de vista de las respuestas en frecuencia (dominio de frecuencia). Este método proviene de las interesantes propiedades de los sistemas lineales (variación temporal como combinación lineal de variables) que permiten convertir una ecuación diferencial en una ecuación

algebraica mediante una transformación integral (transformada de Laplace) al dominio de frecuencias. Para el estudio de estos métodos el alumno necesitaría conocer un conjunto de herramientas matemáticas que no entran en los planes de estudios a los que pertenece esta asignatura, por lo tanto, nosotros realizaremos el estudio principalmente desde el punto de vista del dominio del tiempo[3] haciendo referencia a ecuaciones diferenciales y ecuaciones en diferencias.

1.3. Modelos genéricos por variables de estado

Una de las formas más habituales y directas de describir matemáticamente los sistemas físicos que forman las plantas y los controladores, es decir sus modelos, en las cuales también se pueden incluir los sensores y los actuadores correspondientes (para la planta), es mediante *variables de estado*. Se debe suponer que el estado interno de la planta o el controlador, en cada instante, está determinado por un conjunto de variables o estados y que ese estado va cambiando según una serie de funciones correspondientes. La función de cambio de cada estado puede depender de todos los estados y de todas las "entradas" (que provienen de los actuadores en la planta). Además se necesita un conjunto de funciones que proporcionan el valor de las "salidas" (que irán a los sensores en la planta) dependiendo de los valores de los estados. El concepto de estado está relacionado con el concepto de *memoria* o almacenamiento de información, de tal forma que en la perspectiva de sistemas discretos o digitales podemos hablar de registros de memoria que almacenan el estado.

En el resto de este apartado haremos referencia a los modelos de planta con la notación habitual para estos, pero hay que recordar que el método de modelos para las plantas también sirven para los controladores, haciendo el cambio adecuado de entradas (por las salidas de la planta), de salidas (por el control de entrada a la planta) y de estados (el controlador tendrá sus propios estados diferentes de los de la planta).

Los modelos mediante variables de estado se pueden hacer especificando los cambios en los estados mediante ecuaciones diferenciales (derivadas del estado respecto al tiempo), en cuyo caso estaríamos modelando el sistema con ecuaciones continuas en el tiempo. Los modelos también se pueden hacer considerando solamente los valores en instantes puntuales equiespaciados en el tiempo (modelos discretos en el tiempo), para lo cual es necesario utilizar ecuaciones en diferencias (cambios de un instante al siguiente). En este material expresaremos las ecuaciones en diferencias en forma de relación de

[3] Al final se incluirán algunas pistas para poder enlazar con el dominio de frecuencia.

recurrencia, ya que ambas formas son equivalentes[4]. En la notación utilizada el índice entre paréntesis que sigue a la variable (habitualmente k) indica el número de instante discreto es decir $x(k) = x(t(k))$ y $t(k) = t_o + k \cdot T$, donde t_0 es el instante inicial (habitualmente cero) y T es el intervalo entre instantes (muestreo equiespaciado), aunque en algunos textos específicos de ecuaciones en diferencias se utiliza, en cambio, un subíndice para indicar el instante (u orden en la secuencia).

Éste último método, representación mediante ecuaciones en diferencias o modelos discretos en el tiempo, es más conveniente para los cálculos por ordenador puesto que se pueden realizar con variables que guardan el estado y funciones de cambio puntual y secuencial sobre esas variables. Actualmente, en la mayoría de los procesos industriales el control se realiza mediante ordenadores en forma digital. Además, considerando el contexto docente del alumno de esta asignatura, es más adecuado utilizar el enfoque *discreto del tiempo,* y éste es el enfoque en el que haremos más énfasis a lo largo de este material, pero incluimos a continuación un resumen de las formas de descripción de ecuaciones que incluye ambas perspectivas. No se incluye aquí la explicación sobre el método de obtener los datos digitalmente (muestreo en los sensores), puesto que ese tema ya se ha tratado en otra parte de esta asignatura.

1.3.1. Plantas de una entrada, una salida y un estado

El caso más sencillo es el de una planta con una sola variable de entrada, una variable de salida (en inglés SISO = *Single Input Single Output*) y también una variable de estado interno.

La representación matemática de este modelo continuo en el tiempo se hace mediante una ecuación diferencial ordinaria de primer orden que expresa la variación en el tiempo (derivada) del estado en función del propio estado y de la entrada. También es necesario especificar el valor inicial del estado en el instante $t = 0$ (ó en el instante de tiempo que sea el inicial), para que la ecuación diferencial tenga una solución única. La salida es una función del estado y de la entrada. La ecuaciones correspondientes a este modelo SISO son

$$\begin{cases} \dot{x} = \frac{d}{dt}x(t) = f\left(x(t); u(t)\right); \quad x(0) = X \\ y(t) = g\left(x(t); u(t)\right) \end{cases} \tag{1}$$

[4]Las ecuaciones en diferencias usan el operador Δ para indicar diferencias $\Delta x(k) = x(k+1) - x(k)$ y las ecuaciones son de la forma $\Delta x(k) = F\left(x(k)\right)$, pero reordenando los términos se pueden escribir como relaciones de recurrencia $x(k+1) = x(k) + F\left(x(k)\right) \equiv f\left(x(k)\right)$.

donde las funciones f y g pueden ser no-lineales[5]. Habitualmente la función g solamente depende del estado, e incluso en ocasiones la salida es el propio estado ($y(t) = g(x(t)) = x(t)$).

Las ecuaciones diferenciales proporcionan o constituyen una herramienta matemática para el análisis de modelos físicos, especialmente en el caso de que las funciones f y g sean lineales o se aproximen por funciones lineales ($f(x(t); u(t)) = \alpha x(t) + \beta u(t)$ y $g(x(t); u(t)) = \delta x(t) + \gamma u(t)$ donde α, β, δ y γ son constantes), puesto que entonces existen métodos muy sistemáticos de análisis y diseño de controladores (teoría clásica de sistemas dinámicos lineales).

La representación equivalente del modelo discreto de planta será una ecuación en diferencias finitas para el estado en el instante siguiente dependiendo del propio estado en el instante actual (memoria) y de la entrada. Igual que en el caso continuo la salida correspondiente es una función del estado actual y de la entrada actual

$$\begin{cases} x(k+1) = f\left(x(k); u(k)\right); \quad x(0) = X \\ y(k) = g\left(x(k); u(k)\right) \end{cases} \tag{2}$$

donde $x(k)$ representa el estado en el instante número k (entero positivo), $x(0)$ es el valor inicial dado para el estado y las funciones f y g pueden ser no-lineales en general.

1.3.2. Plantas de una entrada, una salida y múltiples estados

Cuando la dinámica de una planta es más compleja que la presentada en el apartado anterior es necesario utilizar más cantidad de variables de estado para definir el modelo.

Múltiples estados en modelos continuos Añadiendo las ecuaciones diferenciales correspondientes, en el caso de variables continuas en el tiempo, obtenemos el equivalente a las ecuaciones (1) para n estados

$$\begin{cases} \dot{x}_1 = \frac{d}{dt}x_1(t) = f_1\left(x_1(t), \cdots, x_n(t); u(t)\right); \quad x_1(0) = X_1 \\ \vdots \\ \dot{x}_n = \frac{d}{dt}x_n(t) = f_n\left(x_1(t), \cdots, x_n(t); u(t)\right); \quad x_n(0) = X_n \\ y(t) = g\left(x_1(t), \cdots, x_n(t); u(t)\right) \end{cases} \tag{3}$$

[5]En este material consideraremos que estas funciones son constantes en el tiempo, es decir que no dependen *explícitamente* de t (es decir que no contienen t en su expresión).

que podemos abreviar utilizando notación vectorial y suponiendo implícita la dependencia en el tiempo de todas las variables,

$$\begin{cases} \dot{\overrightarrow{x}} = \frac{d}{dt}\overrightarrow{x} = \overrightarrow{f}\left(\overrightarrow{x}; u\right); \quad \overrightarrow{x(0)} = \overrightarrow{X} \\ y = g\left(\overrightarrow{x}; u\right) \end{cases} \tag{4}$$

Si el modelo que tenemos de la planta necesita alguna ecuación diferencial de *orden superior*, como por ejemplo (obsérvese la dependencia extra de f_i^* en $\dot{x}_i(t)$) para una ecuación diferencial de segundo orden

$$\ddot{x}_i = \left(\frac{d}{dt}\right)^2 x_i(t) = f_i^*\left(x_1(t), \cdots, x_n(t), \dot{x}_i(t); u(t)\right)$$

simplemente se transforma el conjunto de ecuaciones, añadiendo una variable de estado nueva, $x_{n+1}(t) = \dot{x}_i(t)$ que es también la nueva función f_i, esto es

$$\dot{x}_i = \frac{d}{dt}x_i(t) = f_i\left(x_1(t), \cdots, x_i(t), \cdots, x_{n+1}(t); u(t)\right) = x_{n+1}(t)$$

y trasladando la anterior función f_i^* a ser la nueva función $f_{n+1} = f_i^*$ correspondiente a $\dot{x}_{n+1}(t)$, sustituyendo en ella $\dot{x}_i(t)$ por $x_{n+1}(t)$ y cambiando las condiciones iniciales a las nuevas variables.

Múltiples estados en modelos discretos De la misma forma que para el sistema continuo en el tiempo, también podemos utilizar más cantidad de estados en el modelo discreto en el tiempo, añadiendo las ecuaciones en diferencias correspondientes para los nuevos estados a las ecuaciones (2), por tanto nos queda para n estados

$$\begin{cases} x_1(k+1) = f_1\left(x_1(k), \cdots, x_n(k); u(k)\right); \quad x_1(0) = X_1 \\ \vdots \\ x_n(k+1) = f_n\left(x_1(k), \cdots, x_n(k); u(k)\right); \quad x_n(0) = X_n \\ y(k) = g\left(x_1(k), \cdots, x_n(k); u(k)\right) \end{cases} \tag{5}$$

que también podemos abreviar con notación vectorial de la forma

$$\begin{cases} \overrightarrow{x}(k+1) = \overrightarrow{f}\left(\overrightarrow{x}(k); u(k)\right); \quad \overrightarrow{x(0)} = \overrightarrow{X} \\ y(k) = g\left(\overrightarrow{x}(k); u(k)\right) \end{cases} \tag{6}$$

Igual que en el caso continuo en el tiempo, también aquí podría darse el caso de un modelo que necesitase ecuaciones en diferencias de orden superior,

esto es que apareciesen en las funciones f dependencias en valores de algún estado en instantes anteriores, por ejemplo

$$x_i(k+1) = f_i\left(x_1(t), \cdots, x_i(k), x_i(k-1) \cdots, x_n(k); u(k)\right) \tag{7}$$

en cuyo caso simplemente se añade un nuevo estado $x_{n+1}(k) = x_i(k-1)$, sustituyendo su valor en la ecuación (7) y añadiendo la nueva ecuación en diferencias correspondiente

$$x_{n+1}(k+1) = f_{n+1}\left(x_1(t), \cdots, x_n(k), x_{n+1}(k); u(k)\right) = x_i(k) \tag{8}$$

(más el cambio de variables en las condiciones iniciales) de forma que el sistema volvería a quedar constituido por ecuaciones en diferencias de primer orden.

1.3.3. Plantas de múltiple entrada y múltiple salida

Finalmente llegamos a los modelos más generales en los cuales podemos tener varias entradas y varias salidas (en inglés MIMO = *Multiple Input Multiple Output*). La extensión de las ecuaciones se hace simplemente añadiendo las dependencias de las entradas en todas las ecuaciones y añadiendo más funciones para todas las salidas.

Si estamos representando un modelo continuo en el tiempo, utilizaremos ecuaciones diferenciales para la variación de los estados y en notación vectorial abreviada tendremos un conjunto de ecuaciones equivalentes a las ecuaciones (4) de la forma

$$\begin{cases} \dot{\overrightarrow{x}} = \frac{d}{dt}\overrightarrow{x} = \overrightarrow{f}\left(\overrightarrow{x}; \overrightarrow{u}\right); \quad \overrightarrow{x(0)} = \overrightarrow{X} \\ \overrightarrow{y} = g\left(\overrightarrow{x}; \overrightarrow{u}\right) \end{cases} \tag{9}$$

Si, en cambio, estamos representando un modelo discreto en el tiempo, utilizaremos de forma equivalente ecuaciones en diferencias para los estados, añadiendo las dependencias respecto a las entradas en todas las ecuaciones (6) y también ecuaciones para las salidas, que en notación vectorial abreviada nos quedaría

$$\begin{cases} \overrightarrow{x}(k+1) = \overrightarrow{f}\left(\overrightarrow{x}(k); \overrightarrow{u}(k)\right); \quad \overrightarrow{x(0)} = \overrightarrow{X} \\ \overrightarrow{y}(k) = g\left(\overrightarrow{x}(k); \overrightarrow{u}(k)\right) \end{cases} \tag{10}$$

Como ya hemos dicho, en este material utilizaremos con mayor frecuencia la representación de modelos discretos en el tiempo (ecuaciones en diferencias), por ser más sencillos para el alumno de esta asignatura. En particular,

los modelos más sencillos son aquellos en los que las funciones $\overrightarrow{f}$ y $\overrightarrow{g}$ son lineales y se pueden sustituir por productos y sumas matriciales[6] de la forma

$$\begin{cases} x(k+1) = A \cdot x(k) + B \cdot u(k); \quad x(0) = X \\ y(k) = C \cdot x(k) + D \cdot u(k) \end{cases} \tag{11}$$

donde A, B, C y D son *matrices* de las dimensiones adecuadas (dependiendo del número de estados, n, y de la cantidad de entradas y de salidas) y donde hemos eliminado las flechas sobre las variables suponiendo implícitamente que son vectores. Los modelos lineales son especialmente interesantes puesto que podemos aproximar prácticamente cualquier otro modelo no-lineal a uno lineal dentro de unos rangos de operación. Esta operación de aproximación se llama *linealizar* un modelo.

1.4. Modelos digitales

En el apartado anterior hemos presentado el modelo más genérico de planta con múltiples entradas, salidas y estados, tanto en la representación continua en el tiempo (ecuaciones diferenciales), como en la representación discreta en el tiempo (ecuaciones en diferencias). Tal como ya hemos mencionado al alumno de esta asignatura y carrera, le será más provechosa (y sencilla de entender) la representación discreta en el tiempo, sin más que sustituir las ecuaciones en un programa de ordenador de la forma adecuada. Si se sustituye el modelo de la planta hablaremos de *programa simulador* y si sustituimos el control hablaremos de *programa controlador*.

1.4.1. Transformación en programa de ordenador

Los pasos generales para hacer la sustitución de las ecuaciones en diferencias 10 en forma de programas, o mejor dicho, de función dentro de un programa, es la siguiente (en lenguaje informal):

1. Transformar los vectores matemáticos (x, u, y) en las *variables* de tipo vector ("array") correspondientes dentro del programa, cuidando de mantener dos copias (variables distintas) para el vector de estados (la "actual" y la "siguiente").

2. Asignar a las variables del estado los *valores iniciales* correspondientes.

3. Recibir los valores del vector de *entradas* (esto puede estar implícito en la forma de organizar los bucles, dependiendo del lenguaje de programación).

[6] La misma simplificación puede hacerse en modelos contínuos en el tiempo.

4. Calcular los valores del vector de *salida* (bucle para cada salida) ejecutando una llamada a las funciones correspondientes de las funciones matemáticas $g(\,)$ con los argumentos adecuados (estados actuales y entradas).

5. Calcular los valores del vector de *estados* "siguiente" (bucle para cada estado) ejecutando una llamada a las funciones correspondientes de las funciones matemáticas $f(\,)$ con los argumentos adecuados (estados actuales y entradas).

6. Sustituir el vector de estados "actuales" por los valores del vector de estados "siguientes".

7. Devolver como resultado el vector de salidas (la forma de "enviar" estos valores también depende de la organización de los bucles y del lenguaje de programación).

8. Cambiar de instante de tiempo (sumando T si se necesita t explícitamente) y *repetir* desde el punto [3], hasta que se llegue al instante final de simulación.

Con estos pasos tan inmediatos, si las ecuaciones matemáticas se expresan en un lenguaje de programación adecuado (Fortran, C, Scheme, etc.), podemos transformar los modelos de una planta o un controlador directamente en un programa. A continuación se muestra el listado en lenguaje C de un ejemplo de transformación para un modelo SISO discreto con un sólo estado (ecuaciones 2 de la pág. 7). En los comentarios, al final de cada línea, se han indicado entre corchetes los números de paso, según el esquema anterior, correspondientes en el programa.

```
double siso1(A, B, C, D, X, n)
{
  double A, B, C, D, X;   /* parámetros del modelo */
  int n;          /* número de instantes a simular */
  double x, xsig, u, y;   /* variables corresp. [1]*/
  int k;                  /* indice de instantes */
  x = X;                  /* estado inicial     [2]*/
  for (k = 0; k < n; k++) /* bucle exter. tiempo[8]*/
  {
     u = leer_entrada(k);                    /*[3]*/
     y = C*x+D*u;                            /*[4]*/
     xsig = A*x+B*u;                         /*[5]*/
```

```
      x = xsig;                                    /*[6]*/
      enviar_salida(y,k);                          /*[7]*/
  };
}
```

La estructura tan sistemática y la posibilidad de recurrencia en las funciones de los lenguajes de programación nos sugieren la posibilidad de descomponer estos modelos en elementos más pequeños y más simples, agrupándolos en pequeños programas (o funciones) genéricos reutilizables. En efecto, este tipo de programas genéricos existen (simuladores dinámicos) y se han usado durante mucho tiempo para comprobar y los cálculos y el diseño de controladores antes de probarlo en el sistema físico real.

1.4.2. Descomposición modular

Hasta ahora hemos hablado de los modelos de planta o de controlador como un único bloque descrito por funciones (ecuaciones diferenciales o en diferencias), pero otra forma de realizar la descripción (modelo) es la descomposición del bloque en otros bloques o módulos más pequeños. Esto ya lo hemos hecho implícitamente al descomponer el modelo del sistema (lazo de realimentación) completo en dos partes: planta y controlador. También lo hemos hecho al englobar los elementos sensores y actuadores dentro de la planta en la figura 2, o al dibujar los elementos que componen el controlador en la figura 3, tal y como se hace habitualmente en ingeniería.

Los subbloques o módulos en los que dividimos un sistema representan "partes" del modelo del sistema, que se pueden considerar por separado y que se deben conectar entre sí para obtener el modelo total. Los módulos también se describen enumerando sus entradas y sus salidas, y definiendo los estados internos propios. De esta forma vemos que un modelo de planta o de controlador se puede descomponer en otros elementos más sencillos que están *conectados* entre sí (las salidas de unos módulos pueden ser las entradas de otros) para formar en su conjunto el modelo del sistema (planta + controlador).

El concepto de descomposición modular en el diseño de modelos dinámicos ha propiciado que algunos entornos de simulación utilicen esta idea para permitir la "programación gráfica", es decir el diseño de un modelo, a través de interacción en un entorno gráfico (como dibujar el modelo), que después se transformará en un programa de forma automática (tal como hemos explicado en el apartado anterior) para simular el sistema modelado.

En la figura 4hemos representado la descomposición en bloques más simples del mismo modelo de planta que se ha mostrado en el apartado anterior

(ecuaciones 2 de la pág. 7). La descomposición da lugar a tres bloques, dos de ellos son las funciones de las ecuaciones que se pueden seguir descomponiendo y el otro bloque representa un registro de memoria que guarda el valor del estado anterior. El registro de memoria "retarda" el valor de su entrada un instante de tiempo a la vez que sirve de almacén. Este tipo de bloque se puede encontrar habitualmente representado con la etiqueta $\boxed{z^{-1}}$ en entornos de control clásico[7].

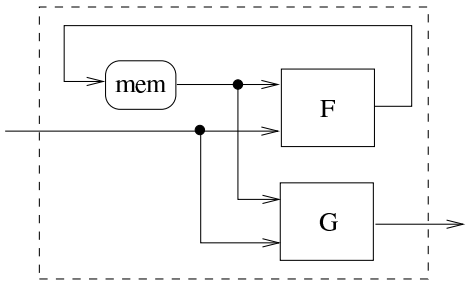


Figura 4: Descomposición modular de un modelo de planta SISO discreto.

Los dos bloques de función en la primera descomposición modular se han representado, en la figura 5, descompuestos a su vez en otros bloques más pequeños. En esta descomposición se utilizan bloques amplificadores (triangulares) con su correspondiente ganancia (parámetro) y bloques sumadores de dos entradas.

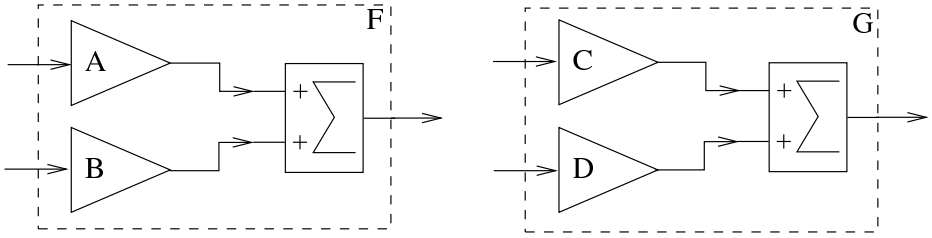


Figura 5: Descomposición modular de las funciones f y g del modelo de planta SISO discreto de la figura 4.

En la figura 6 se han recopilado los tres tipos de bloque básicos y muy sencillos que se han utilizado en esta descomposición: el registro de memoria, el amplificador y el sumador de dos entradas.

[7]Esa nomenclatura con la variable z para el retardo, proviene de la "transformada Z" que es la versión digital (discreta), mediante sumas, de la transformada integral de Laplace para la representación en el dominio de frecuencias.

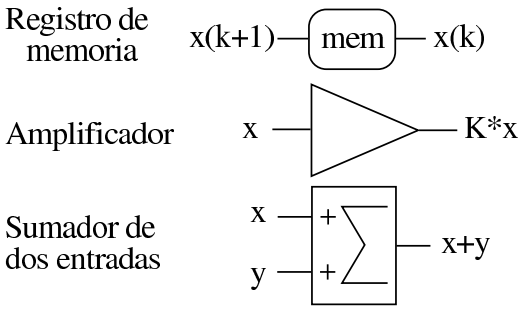


Figura 6: Módulos básicos utilizados en la descomposición de un modelo SISO en las figuras 4 y 5

1.4.3. Ejemplos de simulación con diseño modular

Uno de los programas que permiten hacer el modelado de forma gráfica, para luego simular el sistema es SCICOS que forma parte del programa SCILAB[8] (del INRIA). A continuación mostramos dos ejemplos sencillos de planta y controlador realizados con este programa. Ambos tienen un controlador por amplificador por realimentación negativa como el que se ha explicado en el apartado 1.1 de la página 3. La diferencia es que el primero tiene un modelo continuo de la planta y el segundo tiene un modelo discreto.

Ejemplo de modelo continuo La planta tiene una entrada, una salida y un estado. El modelo utiliza las ecuaciones diferenciales lineales siguientes:

$$\begin{cases} \dot{x} = \frac{d}{dt}x(t) = Ax(t) + Bu(t); \quad x(0) = X \\ y(t) = Cx(t) + Du(t) \end{cases} \tag{12}$$

donde hemos tomado para el ejemplo $A = 0$, $B = 1$, $C = 1$ y $D = 0$, junto con la condición inicial para el estado $X = 0$.

El controlador es un amplificador aplicado sobre el error (diferencia) entre la salida de la planta y la señal de referencia (valor deseado) que corresponde al control simple que se explica en el apartado (ver pág. 3) sobre la realimentación negativa. En la figura 7 se puede observar este sistema tal como se ha introducido en SCICOS, donde además de los bloques de la planta y del controlador se han incluido otros bloques para visualizar la simulación y generar las entradas.

La simulación de este sistema se ha realizado utilizando una señal de referencia $r(t)$, que cambia de cero a 1 (bruscamente) en el instante $t =$

[8]Este programa es de libre distribución, hay versiones para varias plataformas y se puede encontrar en la página WEB: `http://www-rocq.inria.fr/scilab/`

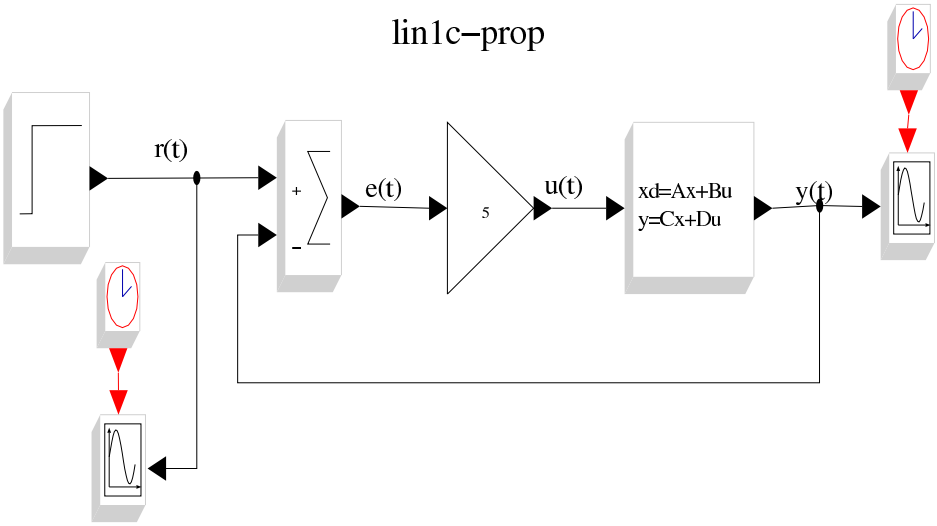


Figura 7: Control por amplificador de realimentación negativa sobre una planta lineal de primer orden (un estado) con modelo continuo (en SCICOS).

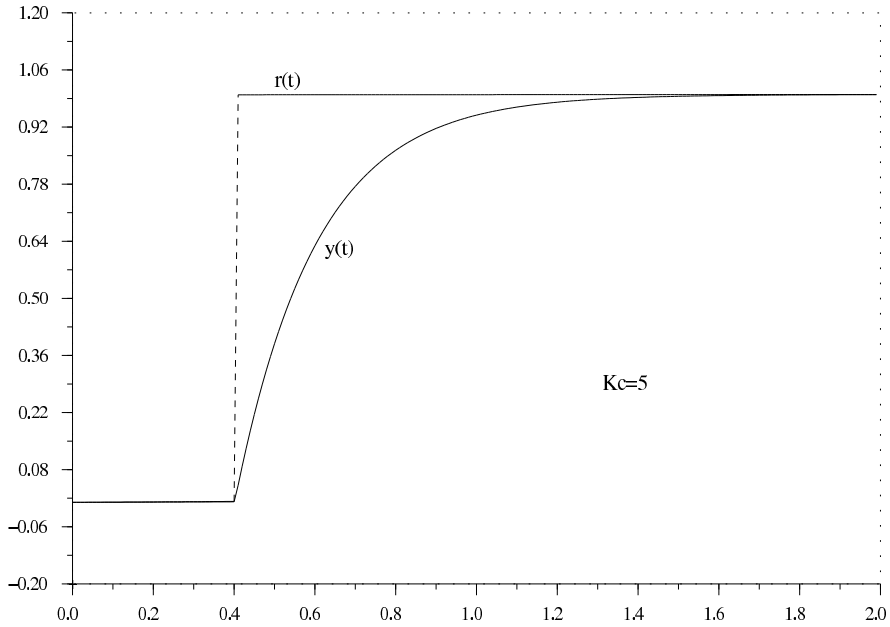
0,4., para ver la respuesta de la planta con el controlador. En la figura 8 se pueden ver los resultados para dos casos, utilizando $K_c = 5$ en el controlador (constante del amplificador) y utilizando $K_c = 20$. Se puede apreciar que para este tipo de planta (lineal de primer orden) la respuesta "mejora", es decir se parece más a la señal de referencia, cuando se aumenta la constante, por lo tanto se debería elegir para K_c el valor más alto permitido por las restricciones físicas del sistema.

Ejemplo de modelo discreto La planta tiene una entrada, una salida y un estado digital. El modelo utiliza las ecuaciones de recurrencia lineales siguientes:

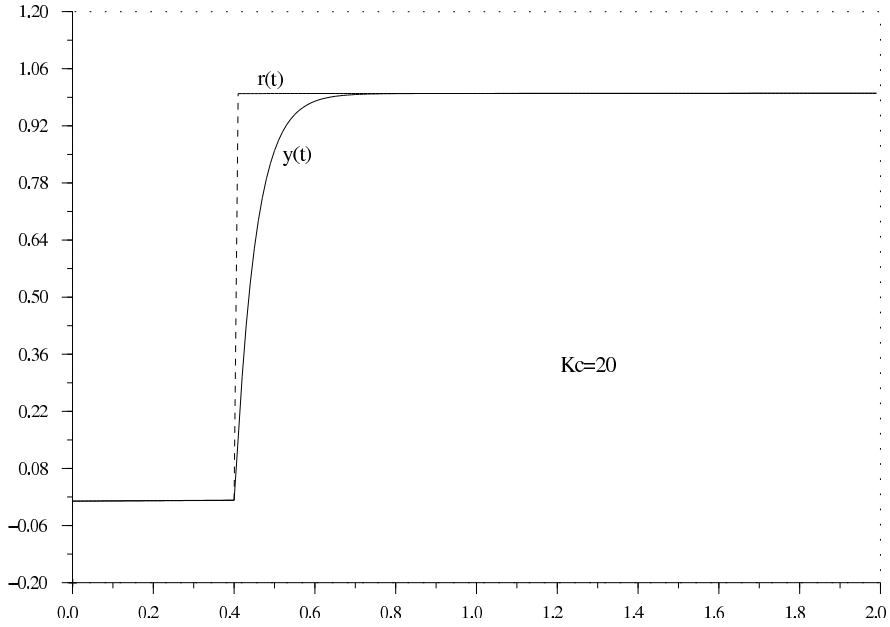
$$\begin{cases} x(k+1) = Ax(k) + Bu(k); \quad x(0) = X \\ y(k) = Cx(k) + Du(k) \end{cases} \tag{13}$$

donde hemos tomado para el ejemplo $A = 1$, $B = 1$, $C = 1$ y $D = 0$, junto con la condición inicial para el estado $X = 0$ y el intervalo de muestreo o periodo de actualización es $T = 0{,}05$.

El controlador es, como en el caso continuo (ver también pág. 3), un amplificador aplicado sobre el error (diferencia) entre la salida de la planta y la señal de referencia (valor deseado). En la figura 9 se puede observar este sistema tal como se ha introducido en SCICOS, donde además de los bloques de la planta y del controlador se han incluido otros bloques para visualizar la simulación y generar las entradas. En especial, en este caso el bloque de la



(a) $K_c = 5$



(b) $K_c = 20$

Figura 8: Ejemplos de simulación (en SCICOS) del control por amplificador de realimentación negativa sobre una planta lineal de primer orden (un estado) con modelo continuo.

planta tiene conectado un bloque que genera los eventos (actualización) con el periodo necesario T.

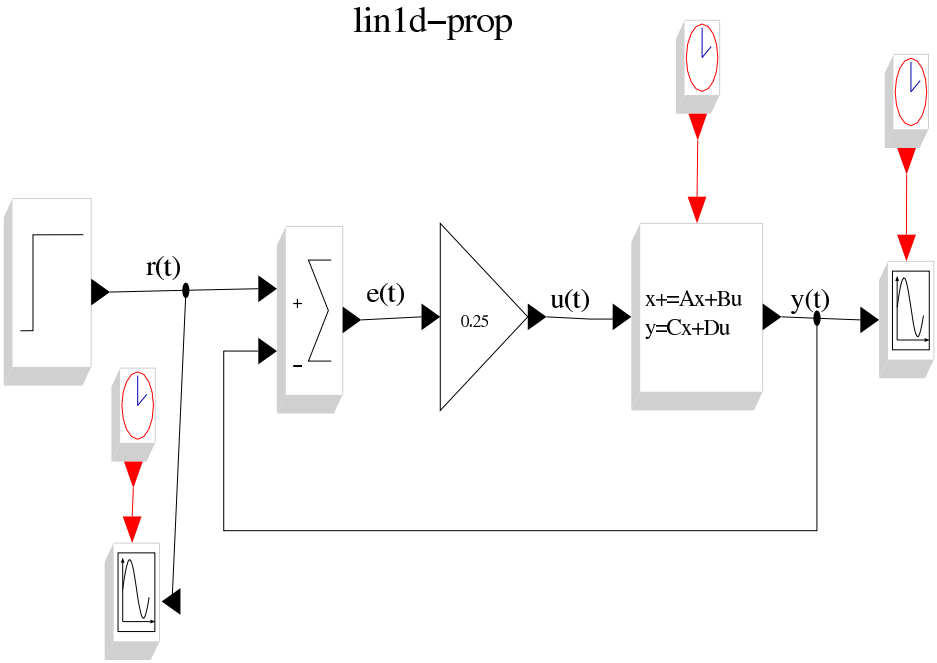
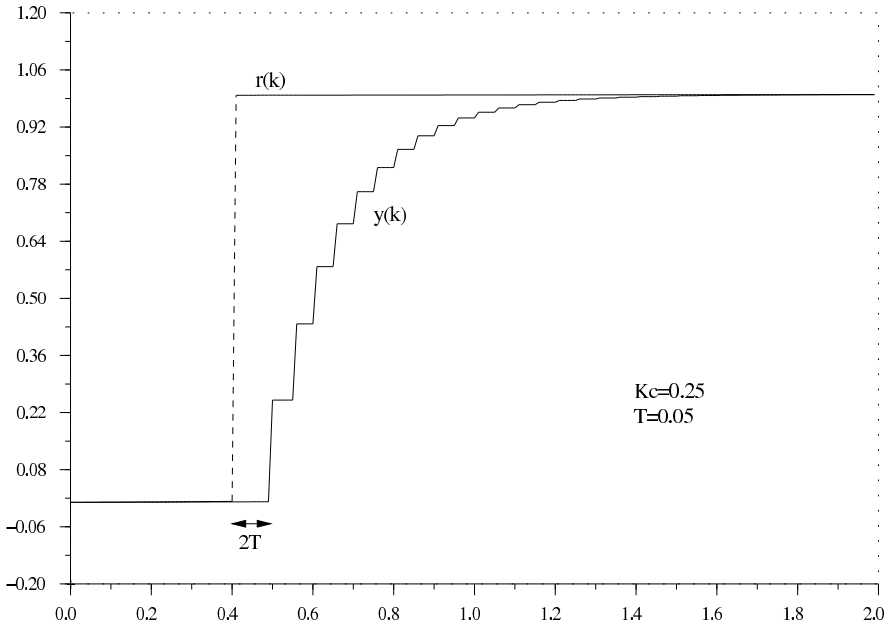


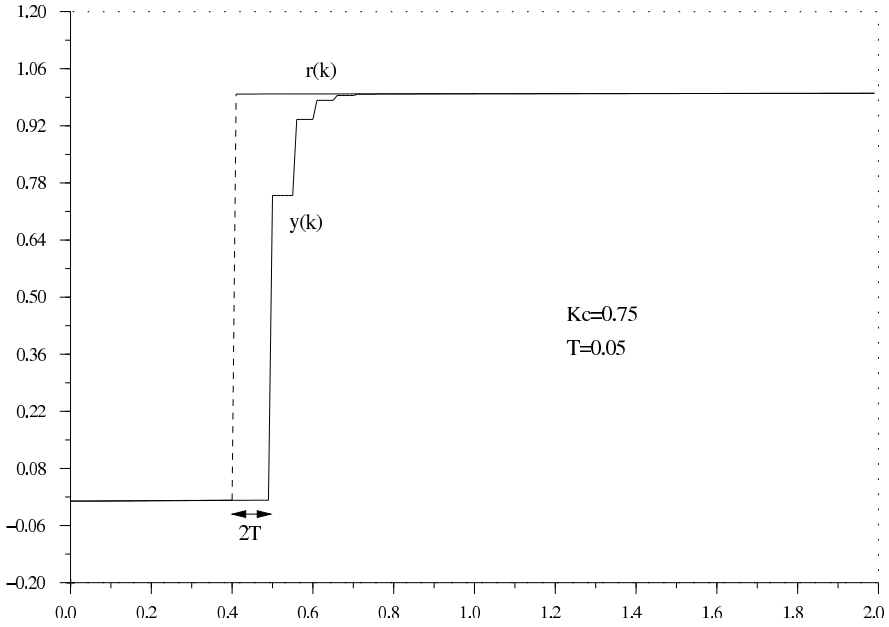
Figura 9: Control por amplificador de realimentación negativa sobre una planta lineal de primer orden (un estado) con modelo discreto (en SCICOS).

La simulación de este sistema también se ha realizado utilizando una señal de referencia $r(t)$, que cambia de cero a 1 (bruscamente) en el instante $t = 0{,}4$., para ver la respuesta de la planta con el controlador. En la figura 10 se pueden ver los resultados para dos casos, utilizando $K_c = 0{,}25$ en el controlador (constante del amplificador) y utilizando $K_c = 0{,}75$. Igual que en el ejemplo de planta continua, se puede apreciar que para este tipo de planta (lineal de primer orden) la respuesta "mejora", es decir se parece más a la señal de referencia, cuando se aumenta la constante. Pero si se sobrepasa el valor $K_c > 1$ el comportamiento del sistema es distinto, sobrepasando la planta el valor de la referencia para después acercarse al valor final estable oscilando, si utilizamos valores aún más altos ($K_c \geq 2$) se puede llegar a la inestabilidad oscilatoria (el valor de salida oscila entre límites cada vez más grandes). Estos dos últimos casos se pueden apreciar en los dos ejemplos de la figura 11.

En el caso discreto, hay que tener en cuenta que la respuesta de la planta también depende del periodo de actualización T, mejorando la respuesta cuando el periodo es más pequeño. En la figura 12 se muestran dos simula-

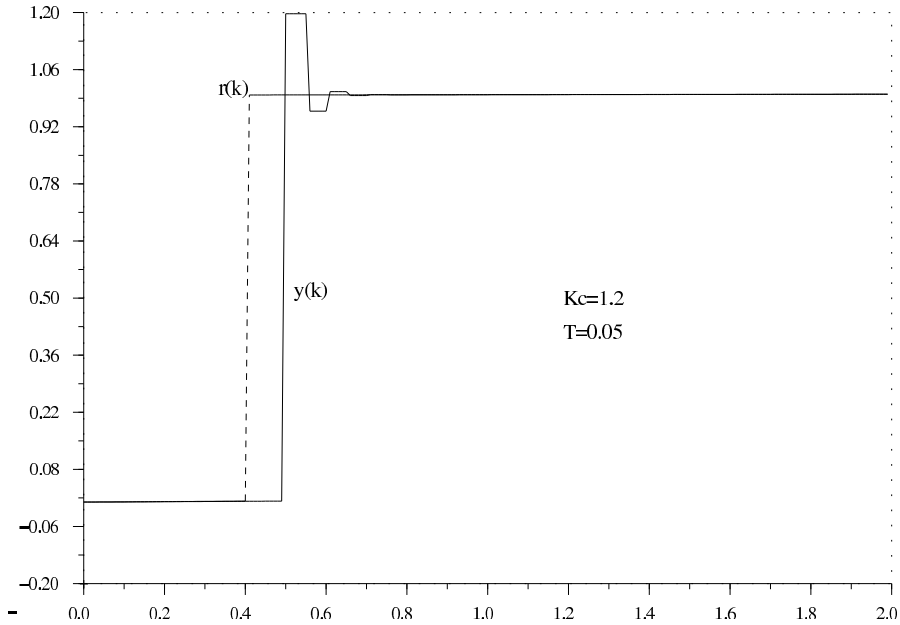


(a) $K_c = 0{,}25$

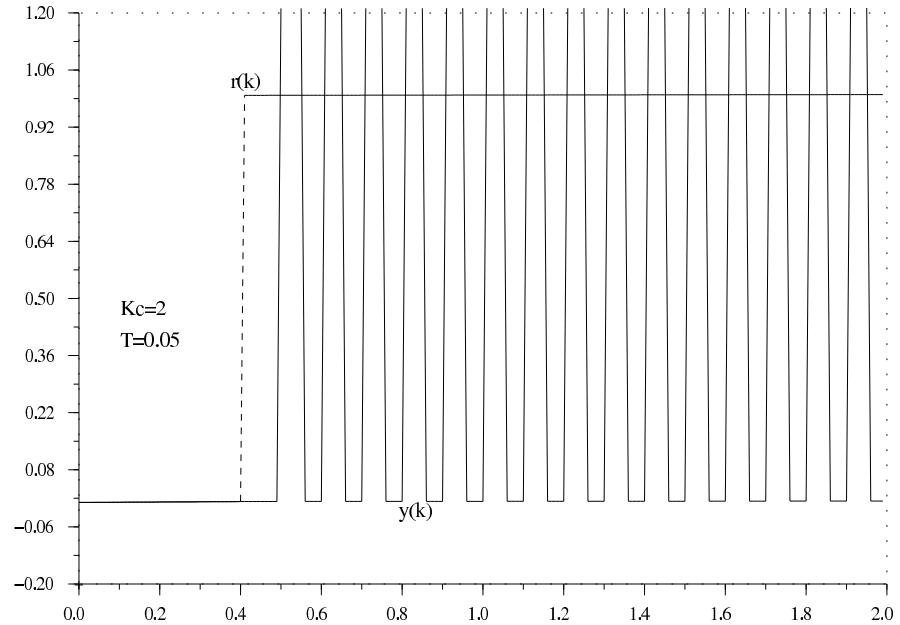


(b) $K_c = 0{,}75$

Figura 10: Ejemplos de simulación (en SCICOS) del control por amplificador de realimentación negativa sobre una planta lineal de primer orden (un estado) con modelo discreto.



(a) $K_c = 1{,}2$



(b) $K_c = 2$ (la imagen se ha truncado por arriba para mantener la escala comparativa).

Figura 11: Ejemplos de simulación (en SCICOS) del control por amplificador de realimentación negativa sobre una planta lineal de primer orden (un estado) con modelo discreto.

ciones más realizadas con la misma constante en el amplificador del control $K_c = 0{,}25$ pero con dos valores del periodo distintos, en el primer caso con el doble ($T = 0{,}1$) y en el segundo caso con la mitad ($T = 0{,}025$). Es decir, que en el caso de modelos discretos (control o planta) para un controlador por amplificador con realimentación negativa la *ganancia* depende también del periodo de muestreo.

Además, en la figura 12, hay que notar que el retraso hasta que se inicia la corrección es mayor cuando el periodo es mayor y viceversa. Esto es lógico si pensamos que la planta no se "entera" de los cambios de la entrada salvo en los instantes de muestreo. En este caso, el retraso aparente es doble debido a que el salto brusco de la referencia, $r(t)$, se produce justo después del instante de muestreo (inmediatamente después de $t = 0{,}4$), lo normal (si fuera justo antes) sería un retraso igual al periodo (recordar que la salida se calcula con estado actual, ver la ecuación 13).

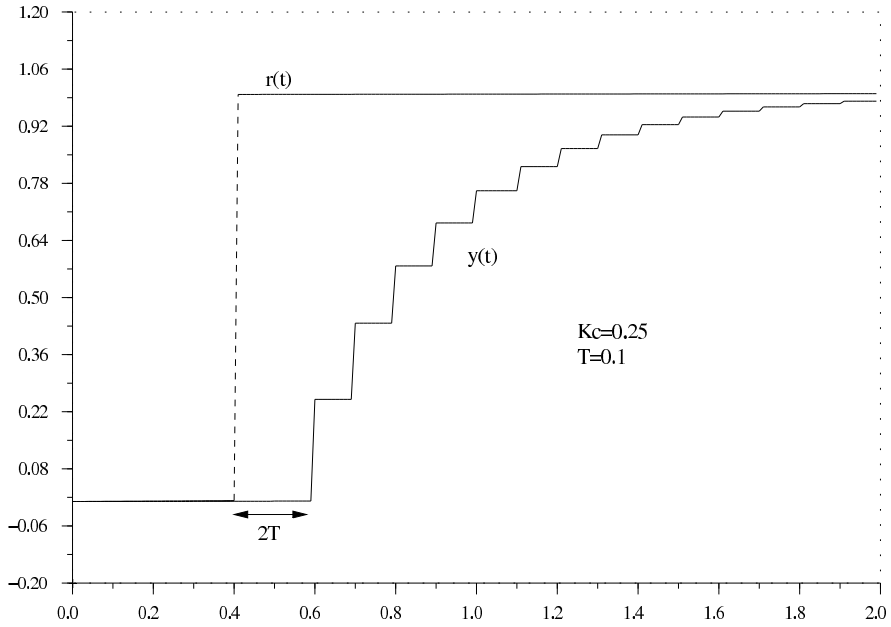
1.5. Especificación de control

Los sistemas de control industriales se diseñan teniendo en cuenta que la forma de las señales de referencia y de las posibles perturbaciones externas desconocidas puede ser muy complicada (no tan simple como un escalón mostrado en estos ejemplos). Para poder realizar esos diseños teóricos, se utilizan habitualmente especificaciones sobre las condiciones de respuesta de un sistema a los cambios en las señales de control y referencia en distintas frecuencias.

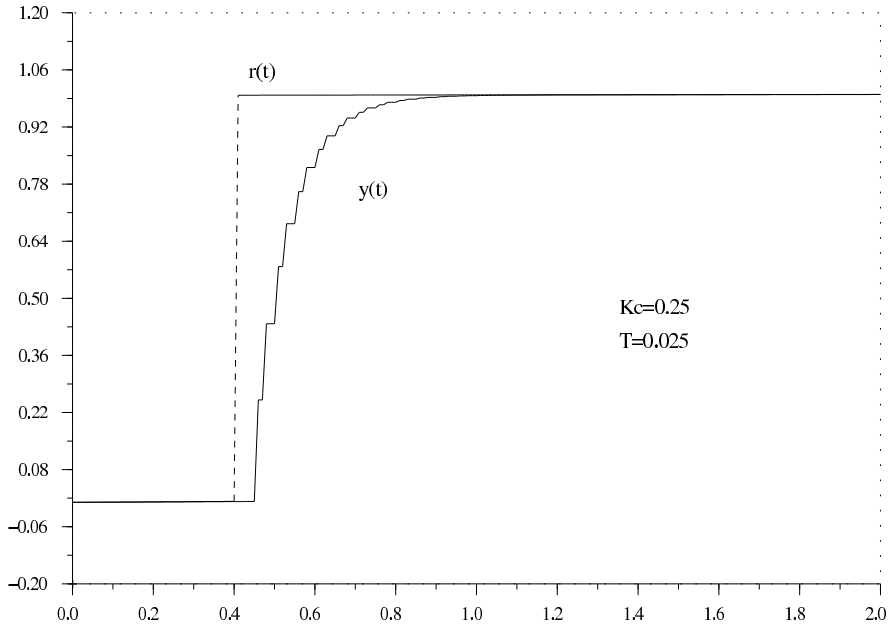
Representación en el dominio de frecuencia Cualquier señal se puede descomponer en un conjunto de señales de una frecuencia pura sumadas en distinta proporción y con distintos desfases (adelantos o retrasos). Una señal de una frecuencia pura es de forma sinusoidal (función seno o coseno), es decir que se puede representar por una función del tiempo de la forma $A\sin(\omega t+\phi)$ donde A es la *amplitud* (proporción para la suma), ω es la frecuencia angular ($\omega = 2\pi f$, donde f es aquí el número de veces que se repite el ciclo cada segundo[9]) y ϕ es el desfase (en radianes). Por lo tanto, cualquier señal se podrá escribir como una suma de señales sinusoidales puras

$$r(t) = \sum_{n=-\infty}^{\infty} A_n \sin\left(\omega_n t + \phi_n\right)$$

[9]Por tanto w se mide en radianes por segundo (rad/s), y f se mide en Hercios (1 Hz = 1 segundo^{-1}).



(a) $T = 0{,}1$



(b) $T = 0{,}025$

Figura 12: Ejemplos de simulación (en SCICOS) del control por amplificador de realimentación negativa sobre una planta lineal de primer orden (un estado) con modelo discreto, con diferentes pasos de tiempo.

En el análisis de plantas, se utiliza como prueba una señal de una sola frecuencia y se observa la variación de la salida (la respuesta) en amplitud y en desfase. Se supone que la frecuencia en la respuesta es la misma cuando el sistema es lineal (en el sentido de las ecuaciones diferenciales). Cuando se realiza esa prueba utilizando sucesivamente señales de entrada de varias frecuencias, se pueden obtener diagramas representando la variación en amplitud y en desfase dependiendo de la frecuencia. Existen operadores de transformación que nos permiten obtener de forma analítica esta información como son la "Transformada Integral de Laplace" y la "Transformada Z" (para el caso discontinuo).

Como ya hemos mencionado anteriormente, el tratamiento en el dominio de frecuencia de los sistemas dinámicos cae fuera del alcance de esta asignatura, sin embargo, en el apartado siguiente vamos a dar una breve referencia mínima sobre las especificaciones de respuesta de un sistema que aparece habitualmente en control clásico.

Especificaciones en espacio de frecuencia Los requisitos de un controlador se especifican en base a características y parámetros respecto a la respuesta en frecuencias. Estos parámetros o especificaciones son, por ejemplo, la máxima o mínima ganancia (cambio en amplitud), el máximo desfase, o también el conjunto de frecuencias (banda) con ganancia máxima (filtros), etc.

Si aplicamos una señal de referencia sinusoidal al sistema lineal de primer orden con control por amplificador de realimentación negativa usado anteriormente (ver apartados 1.3.1 y 1.4.3), obtenemos una respuesta del sistema que también es sinusoidal (seguidor) de la misma frecuencia (por ser un sistema lineal) pero con amplitud y desfase distintos. En la figura 13 se puede ver un ejemplo con una señal de referencia de 0.5 Hz y amplitud 1, donde se han marcado el *desfase,* (diferencia de tiempo entre picos de ambas señales) de $-0{,}17$ s. aprox. (negativos por ser un retraso) que equivalen a $-0{,}17 \cdot 2\pi \cdot 0{,}5$ rad. $= -30{,}6$ grados, y el cambio de amplitud (aprox. $-0{,}16$) que, por tanto, indica una *ganancia* (relación entre ambas amplitudes) que es $\frac{1-0{,}16}{1} = 0{,}84$ (ó $20 \log 0{,}84 \simeq -1{,}5$ dB).

Si realizásemos esta experiencia para muchas frecuencias y dibujáramos los resultados en forma gráfica obtendríamos algo similar a la figura 14, realizada mediante SCILAB, que corresponde al *diagrama de Bode* del sistema lineal con control proporcional, donde se representa la ganancia ("magnitud") en decibelios (unidades logarítmicas muy habituales en teoría de sistemas) y el desfase ("fase") en grados, ambos respecto a la frecuencia en escala semilogarítmica, para varios valores de la constante del controlador ($K_c = 5$, 10,

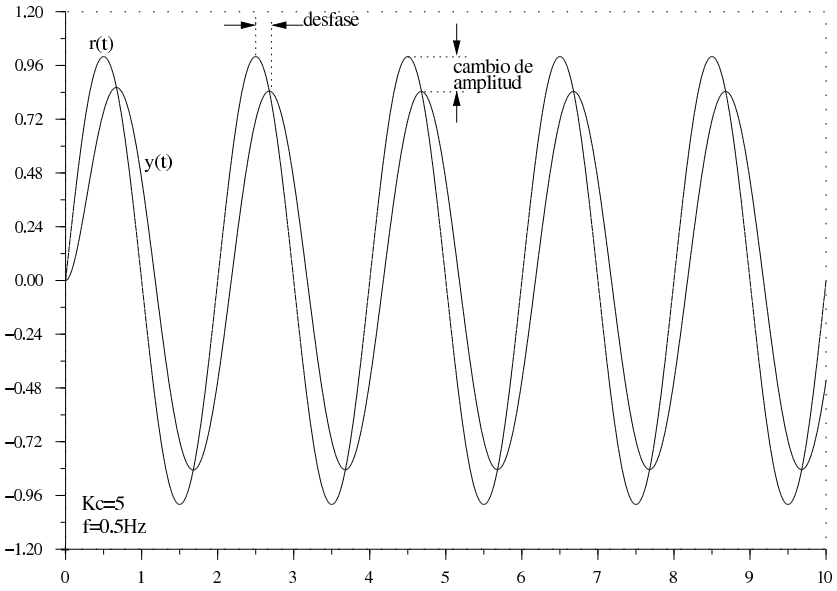


Figura 13: Respuesta de un sistema lineal con controlador proporcional a una entrada de referencia sinusoidal de frecuencia 0.5 Hz y amplitud 1 (obtenido con SCILAB).

20 y 40), también se han marcado los valores para la frecuencia 0,5 Hz. utilizada en la figura 13. De esos diagramas podemos comprobar que el conjunto formado por la planta y el controlador responden "bien" para frecuencias bajas ($< 10^{-1}$ Hz.). Las zonas con más calidad de la respuesta son aquellas en las que la salida no pierde (ni gana) amplitud, esto es, la respuesta se mantiene cerca de los 0 decibelios (que corresponden a una relación entre amplitudes igual a 1), y la salida no se retrasa (ni se adelanta) mucho, esto es, el desfase se mantiene cerca de cero grados. Como ya habíamos comentado anteriormente (ver apartado 1.4.3), la respuesta del sistema mejora cuando se aumenta la constante del controlador, esto es, el rango de frecuencias, para el cual la respuesta se aproxima a la deseada, es mayor (aunque siempre dentro de las bajas frecuencias).

Como ejemplo, la forma de describir un posible controlador deseable para un sistema, podría ser especificar que el controlador debe proporcionar en la salida una ganancia mayor de -5 dB y con un desfase en valor absoluto menor que 10 grados para frecuencias menores de 1Hz. Con estas especificaciones y observando la figura 14, podríamos deducir que el controlador con amplificador por realimentación negativa con una constante de $K_c > 40$ sería válido.

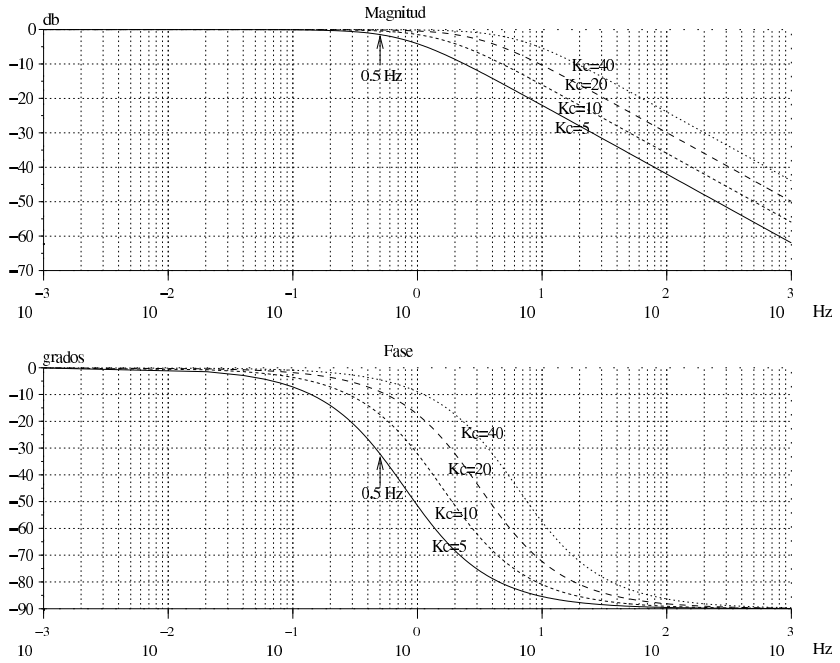


Figura 14: Diagrama de Bode (obtenido con SCILAB) para el conjunto de planta lineal de primer orden y controlador proporcional para valores de $K_c = 5$, 10, 20 y 40.

1.6. Utilización de modelos en control

Cuando se conoce un modelo adecuado de una planta (o una aproximación válida) se pueden diseñar controladores específicos para obtener un comportamiento dado. Habitualmente los métodos de diseño de controladores tradicionales basados en modelos los pueden utilizar de varias formas, que exponemos de forma resumida a continuación.

1.6.1. Control por planta inversa

Ya hemos comentado anteriormente que el método de control más intuitivo, e inmediato, sería calcular el modelo inverso del modelo de la planta. Esto es posible realizarlo cuando se cumplen las condiciones siguientes:

1. Se puede calcular el modelo inverso de la planta, es decir que se conoce un modelo y además existe la inversa del modelo, esto es, que existen las ecuaciones que permiten calcular las entradas en función de las salidas de la planta.

2. El modelo inverso de planta es realizable físicamente y cumple las restricciones de funcionamiento de la planta.

3. Las consignas de funcionamiento para el control consisten en proporcionar señales de referencia para las salidas deseadas en la planta.

La primera condición es fácil de cumplir cuando los modelos de planta que utilizamos son *lineales* (o se aproximan a lineales en un rango de funcionamiento), y entonces se puede calcular directamente la planta inversa, ya que esencialmente, se trata de invertir matrices de números (salvo que la matriz sea singular). Este caso es bastante habitual en el control tradicional.

La segunda condición significa que aunque exista y conozcamos el modelo inverso matemático de la planta, no siempre este modelo es realizable físicamente o no siempre es adecuado para el control debido a limitaciones físicas en los dispositivos que efectúan el control.

La última condición se cumple en la mayoría de los casos habituales, puesto que la forma más sencilla de especificar las consignas de control (criterios del funcionamiento) consiste precisamente en proporcionar señales de referencia para las salidas de la planta.

De esta forma, si se cumplen las condiciones mencionadas, el diagrama para el control aplicado a la planta *no tiene realimentación* (no se cierra el lazo), tal como se muestra en la figura 15. Este tipo de control puede tener problemas debido a las imperfecciones y aproximaciones del modelo de planta utilizado (los modelos no suelen ser perfectos) que pueden conducir a un comportamiento deficiente del control y que no se puede corregir ya que no hay realimentación. Este método de control se denomina en inglés "direct inverse control" que se debería traducir por "control inverso directamente", haciendo referencia a que se aplica directamente (sin otra transformación) como control el modelo inverso de la planta.

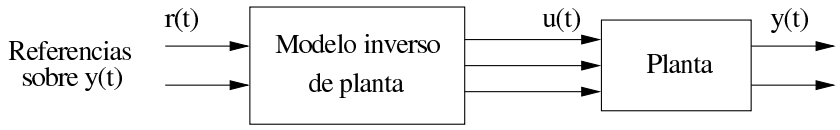


Figura 15: Esquema de utilización del modelo inverso de planta como controlador.

1.6.2. Control por predicción según el modelo

Uno de los problemas al controlar una planta son los retrasos en la reacción al control. Una forma de resolverlo es que el controlador use un modelo de planta para predecir la respuesta de esta. En este método se trata de que el controlador puede anticipar los valores que la planta va a generar en su salida para cada valor de las entradas (valor de control) simulando el cálculo con el

modelo. Incluso se puede utilizar el modelo para calcular de antemano cuáles de los valores de control propuestos en un conjunto dará mejor respuesta (respecto a las consignas) en la planta. En la figura 16 se ha representado el esquema estructural de este tipo de controlador.

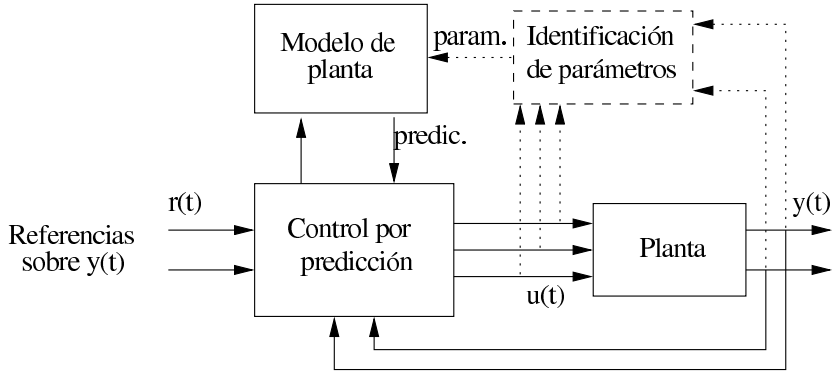


Figura 16: Esquema de un controlador por predicción según el modelo.

En algunos casos, solamente se conoce la estructura de la planta (el tipo de modelo) pero no se conocen exactamente los valores de los parámetros que la definen, o bien esos parámetros van cambiando lentamente a lo largo del tiempo (desgastes, transformaciones, etc.). Entonces, es necesario que formando parte del controlador exista un elemento de identificación de parámetros de la planta (indicado con trazos en la figura 16). Este elemento recibe las entradas y salidas de la planta y las compara con los valores dados por el modelo calculando los parámetros del modelo que permitan una predicción más fiel a la planta real.

1.6.3. Control por estimación de parámetros

Cuando no se conoce perfectamente la planta, en algunas ocasiones es útil suponer que la planta pertenece a una familia de modelos, y que nos falta conocer algunos parámetros que definen ese modelo. De esta forma se diseña un control en función de los parámetros o características del modelo. El modelo no tiene porqué ser muy fiel a la planta real, sólo es necesario que las características extraídas del modelo (tiempos de retardo, frecuencias naturales, etc.) permitan hacer mejor el control.

En la figura 17 se ha representado el esquema de este tipo de control. Como se puede observar también contiene un elemento identificador, que en este caso llamamos estimador (y que puede incluir internamente un modelo de planta), pero, a diferencia del caso anterior, los parámetros o características

que se extraen no tienen porqué coincidir con parámetros estructurales del modelo que, además, se usan directamente en el control (no hay predicción).

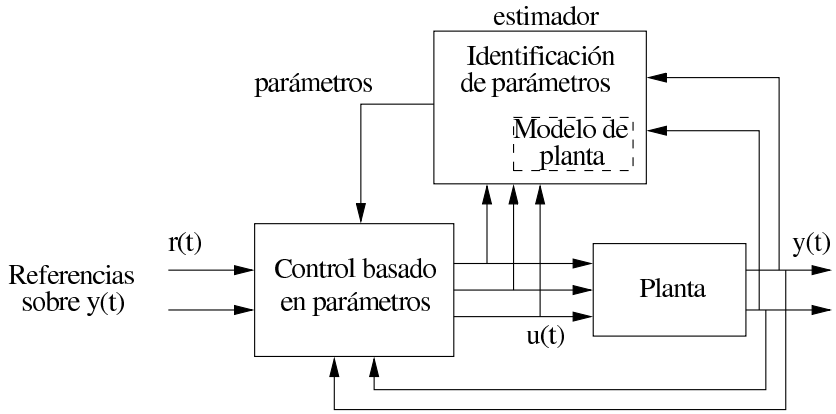


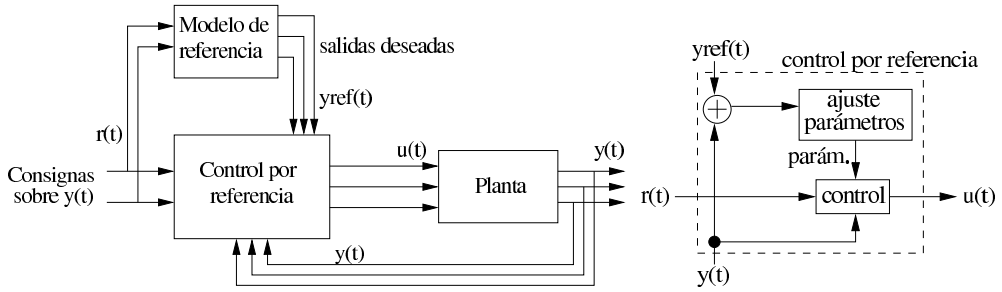
Figura 17: Esquema estructural del control por estimación de parámetros.

1.6.4. Control adaptativo por referencia a un modelo

Otra forma de realizar un control es por referencia a un modelo, aunque esto en realidad se podría considerar como un método de especificación de objetivos de control. Este método consiste en utilizar un modelo de referencia que recibe las señales de consigna (referencias) y genera el valor deseado para la salida en cada instante, estos valores se irán aproximando a las referencias gradualmente en el tiempo, pero siguiendo pautas de cambio distintas (más ajustadas a las posibilidades físicas).

El objetivo del control es generar las señales de control necesarias para que la planta tenga en su salida los mismos valores (o lo más próximos posible) que los del modelo de referencia (y no directamente las consignas). En la figura 18 se muestra el esquema de este tipo de control donde se puede ver que el control por referencia recibe la salida de la planta $y(t)$ y la salida deseada $y_{ref}(t)$ calculada por el modelo de referencia sobre los valores de las consignas.

Lo más habitual es que el bloque de control, que recibe las salidas de la planta y del modelo de referencia, haga una comparación (diferencia) entre ambas $(y(t) - y_{ref}(t))$ para usar ese valor en el ajuste de los parámetros de control (ver figura 18.b).



(a) Esquema general (b) Detalle de controlador

Figura 18: Esquema de control adaptativo por referencia a un modelo.

2. Redes Neuronales en Control

Las redes neuronales artificiales[10] son sistemas computacionales que en sus orígenes se inspiraron en las capacidades de aprendizaje y paralelismos observados en los sistemas nerviosos biológicos. Estos sistemas se utilizan ampliamente en la actualidad en muchos campos de la tecnología y la ingeniería. Aquí haremos un resumen descriptivo de las redes neuronales que permita al alumno entender el uso que se puede hacer de estos sistemas en el control, principalmente, pero también para percepción.

2.1. Introducción a Redes Neuronales

Una red neuronal está formada por un conjunto de elementos de computación paramétricos de "grano fino" (funciones sencillas), conectados[11] entre sí. Los elementos calculan una función a partir de los valores de sus entradas y colocan el resultado en su propia salida. Estos elementos tienen también una función de *aprendizaje* que modifica los valores de los parámetros en la función de computación. Las características más destacables de las redes neuronales son el paralelismo, la redundancia, el aprendizaje, la tolerancia a fallos.

2.1.1. Red neuronal básica

En el contexto de esta asignatura, nos vamos a restringir a un tipo concreto de red neuronal, que es la más ampliamente utilizada y por tanto paradigmática de hecho. Aunque existen infinidad de arquitecturas de redes

[10]En este contexto es normal reducir el nombre a "redes neuronales", sobreentendiendose "artificiales".

[11]De ahí que en algunos ámbitos se llame "conexionismo" a este campo de conocimiento.

neuronales, de elementos, de métodos de aprendizaje y de variantes, en este material describiremos únicamente la arquitectura de "perceptrón multicapa con aprendizaje por retropropagación del error por descenso del gradiente".

El nombre tan largo que hemos dado hace referencia a los diferentes elementos que describen la red neuronal. A continuación vamos a hacer una breve descripción cada una de las partes de ese nombre:

perceptrón: Es el nombre genérico que reciben el tipo de elementos que forman la red. La función de cálculo está formada por la suma ponderada (con coeficientes o pesos que son los parámetros en el aprendizaje) de las entradas, usada como argumento en una función discriminante (o de dicotomía) no lineal que habitualmente es de forma "sigmoidea" que corresponde con alguna variante de la tangente hiperbólica.

multicapa: Hace referencia a la arquitectura de conexión de los elementos de la red, que se disponen en capas (conjuntos de elementos no conectados entre sí) que se reciben como entradas los valores de los elementos de una capa "anterior" y entregan sus salidas a los elementos de una capa "posterior". La primera capa capa (que recibe sus entradas del exterior) se llama capa de entrada[12] (o primera capa) y la última capa (que envía sus salidas al exterior) se llama capa de salida (o última capa). Las capas intermedias se llaman capas ocultas.

retropropagación del error: El algoritmo de aprendizaje (ajuste de parámetros o pesos) es de tipo supervisado (significa que hay un agente externo que proporciona información sobre el error) donde se hace uso del error en las salidas (respecto a los valores deseados) para corregir los pesos. Además la contribución del error sobre las salidas se transmite (propaga) hacia atrás, a las capas ocultas, de forma proporcional a su contribución en las salidas.

por descenso del gradiente: Significa que el reparto de la parte proporcional de error, hacia capas oculta en la retropropagación, se hace siguiendo el mínimo gradiente (derivada) de una función externa de error, que habitualmente es la suma del error cuadrático medio en las salidas.

Si continuamos con el punto de vista modular aplicado anteriormente a los modelos de planta, podemos describir las redes neuronales en forma de módulos. En primer lugar la capa es una estructura que se repite y por tanto

[12]Aviso: En algunos libros y artículos se llama capa de entrada, simplemente al conjunto de entradas del exterior, contabilizandose como primera capa. En este material *no* consideraremos esta interpretación, pero se debe tener en cuenta al consultar otros materiales.

formará el primer nivel de la descripción. Cada capa está formada por un conjunto de elementos (que se llaman neuronas, históricamente por afinidad con la biología) que forman el segundo nivel modular, y así sucesivamente podemos descomponer la red en módulos más pequeños, hasta llegar a ecuaciones simples.

2.1.2. Interpretación de tarea de clasificación

La tarea representativa a la que se aplica el tipo básico de redes neuronales descritas aquí, es la de *clasificación*. Entendiéndose por este nombre la interpretación de las salidas como la identificación de una clase a la cual pertenece la descripción dada en las entradas, consideradas como características que definen un "objeto". Evidentemente, la representación de las clases, los objetos y las características, están supeditadas a una interpretación en el dominio del observador. El entrenamiento consiste en la presentación repetida de un conjunto de "ejemplos" (pares entradas, salidas) de valores correctos, que permiten ajustar los parámetros de la red neuronal para responder de la misma forma ante las mismas entradas.

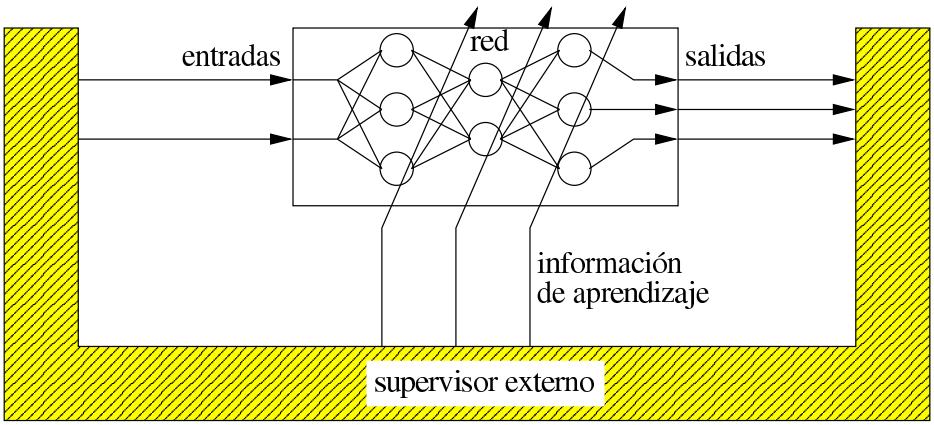


Figura 19: Esquema de aprendizaje con un supervisor externo.

Desde el punto de vista computacional esta tarea se traduce en la construcción de un sistema que puede almacenar una *tabla* para interpolar valores desconocidos. Si consideramos las entradas como los valores de las coordenadas en un espacio de representación, entonces las salidas son los valores que representan las clases.

2.1.3. Descripción resumida del algoritmo completo

A continuación damos un resumen de las etapas y pasos que se realizan para el entrenamiento de una red neuronal básica de perceptrón multicapa con aprendizaje por retropropagación del error por descenso del gradiente [*Mira et al., 1995,* "Introd. a la I.A."]. Este resumen incluye los cálculos asociados con el supervisor externo, el cálculo normal y las modificaciones de parámetros para el aprendizaje en una red de L capas, con N entradas y M salidas:

1. Asignar valores iniciales aleatorios a los pesos de cada capa:

$$w_{ij}^{l} = \text{rand}(-1, 1)$$

2. Elegir un ejemplo del conjunto de entrenamiento consistente en los valores de las entradas y los valores de las salidas deseadas: $\{(x_1^*, \ldots x_N^*), (y_1^*, \ldots y_M^*)\}$

3. Calcular para cada capa, empezando desde la entrada, y capas ocultas hasta la salida, el valor de la respuesta de cada neurona de la capa l, mediante las expresiones generales :

$$z_j^l = \sum w_{ij}^l x_i^l + w_{0j}^l$$
$$y_j^l = \text{sigmoid}\left(z_j^l\right)$$

   recordando que: $x_i^l = y_i^{l-1}$, y que para la capa de entrada: $x_i^1 = x_i^*$.

4. Calcular para cada capa en orden inverso, empezando por la salida y capas ocultas hasta la entrada, los valores de corrección de pesos, usando los valores de salida deseados para la red en capa de salida y_i^*y con las expresiones generales para la capa l:

$$\delta_j^l = \text{dersigm}\left(z_j^l\right) \cdot \sum \delta_k^{l+1} w_{jk}^{l+1}$$
$$\Delta w_{ij}^l = \alpha \cdot \delta_j^l \cdot x_i^l$$

   recordando que para la última capa L (primera calculada): $\delta_j^L = \text{dersigm}\left(z_j^L\right) \cdot \left(y_j^* - y_j^L\right)$.

5. Actualizar los pesos utilizando las correcciones calculadas en el paso anterior:

$$w_{ij}^l \rightarrow w_{ij}^l + \Delta w_{ij}^l$$

6. Volver al punto 2 hasta que el error cuadrático medio en las salidas de la red sea menor que un cierto valor para todos los ejemplos del conjunto de entrenamiento.

En este algoritmo las variables que se usan son las siguientes[13]:

x_i^* es la entrada externa i-ésima de un ejemplo del conjunto de entrenamiento.

y_i^* es la salida externa i-ésima de un ejemplo del conjunto de entrenamiento.

w_{ij}^l es el peso de la entrada i-ésima de la neurona j en la capa l.

w_{0j}^l es el desplazamiento ("bias" o umbral) de la neurona j en la capa l.

x_i^l es la entrada i-ésima en la capa l.

y_j^l es la salida j-ésima en la capa l.

z_j^l es la suma ponderada (por w_{ij}^l) de las entradas (más el desplazamiento) de la neurona j en la capa l.

α es el coeficiente de aprendizaje que, en general, es una constante positiva pequeña ($< 0{,}01$), aunque se podría elegir variable con el tiempo (normalmente decreciente), que es igual para todas las neuronas, pero también se podría elegir distinta para cada peso (α_{ij}^l), para cada neurona (α_j^l) o para cada capa (α^l).

δ_j^l es la corrección (delta) de la neurona j-ésima en la capa l.

$\text{sigmoid}(z) = \frac{a-b}{2} \cdot \tanh\left(\frac{2p}{a-b} \cdot z\right) + \frac{a+b}{2}$, es la función sigmoide que varía entre a y b con una pendiente p en el origen. La sigmoide típica con valores $a = 1$, $b = 0$, $p = \frac{1}{4}$, también se puede expresar como $\frac{1}{1+e^{-z}}$.

$\text{dersigm}(z) = \frac{d}{dz}(\text{sigmoid}(z)) = p \cdot \left(1 - \tanh^2\left(\frac{2p}{a-b} \cdot z\right)\right)$, es la derivada de la función sigmoide. En el caso de la sigmoide típica (que llamamos $G(z)$ con $a = 1$, $b = 0$, $p = \frac{1}{4}$), la derivada cumple la propiedad $\frac{dG(z)}{dz} = G(z) \cdot (1 - G(z))$, muy útil para implementar la función derivada con productos y sumas únicamente.

[13]Atención para *no confundir* las variables x, utilizadas aquí como entradas en las neuronas, con las variables del mismo nombre utilizadas en los apartados de control para referirnos a los estados internos.

2.2. Uso de Redes Neuronales en Control

Por redes neuronales para control se entiende el uso de redes neuronales que van más allá de la simple clasificación de sus entradas, sino que además pueden influenciarlas. Es decir explícitamente diseñadas para aprender de la interacción en lazo cerrado con su entorno. El control en lazo cerrado implica un conjunto muy diferente de requerimientos para los métodos de aprendizaje que los usualmente considerados en otras aplicaciones de las redes neuronales. Por ejemplo, en control es mucho mas importante el aprendizaje conectado directamente ("on-line"), incremental y, en muchas ocasiones, sin un supervisor externo que especifica el comportamiento deseado.

Teniendo en cuenta estos criterios, en este material vamos a centrarnos en los diferentes tipos de uso que se puede hacer de una red neuronal en control, principalmente, desde el punto de vista de los modelos y del lazo de control. Más concretamente, la mayor parte de los modos de utilizar redes neuronales en control están relacionados con sustituir, por una red neuronal, algunas de las partes del control en los métodos de control tradicional que se apoyan en modelos (resumidos en el apartado 1.6). En algunos casos la sustitución permite el entrenamiento o ajuste de la red simultáneamente con el funcionamiento para control, aunque en otros casos es necesario un periodo previo de entrenamiento y después la red se utiliza con la red ya entrenada con sus pesos fijados.

Esta utilización de las redes neuronales en control sustituyendo a algunas partes del control, es posible debido a la características como bloque computacional ajustable (por aprendizaje) que tienen las redes neuronales. En todos estos métodos que se exponen a continuación lo común es el uso de las redes neuronales como clasificador y en todos ellos se puede utilizar el tipo básico de red neuronal explicado en el apartado 2.1 con aprendizaje supervisado. El origen de los ejemplos de entrenamiento es diverso, pero en general proviene de las señales de la planta y de las consignas.

El material que se presenta en este apartado está extensamente basado en algunos de los contenidos del libro:

> "*Neural Networks for Control*" de W. T. Miller, R. S. Sutton y P. J. Werbos (editores) MIT Press, 1990.

que figura en la bibliografía básica de la asignatura.

2.2.1. Planta inversa

Cuando no se dispone de información completa del modelo de una planta, se puede calcular el modelo inverso mediante una red neuronal. Se utilizan

las salidas de la planta como entradas de los pares de entrenamiento, y las entradas de la planta como salidas deseadas en los pares de entrenamiento. Se generan valores en la planta introduciendo varios tipos de señales de prueba que cubran el rango de operación deseado. Esto se esquematiza en la figura 20, que presenta la conexión para el entrenamiento. Después del entrenamiento, se usa la red neuronal entrenada como controlador, en la forma de la figura 15 de la pág. 25.

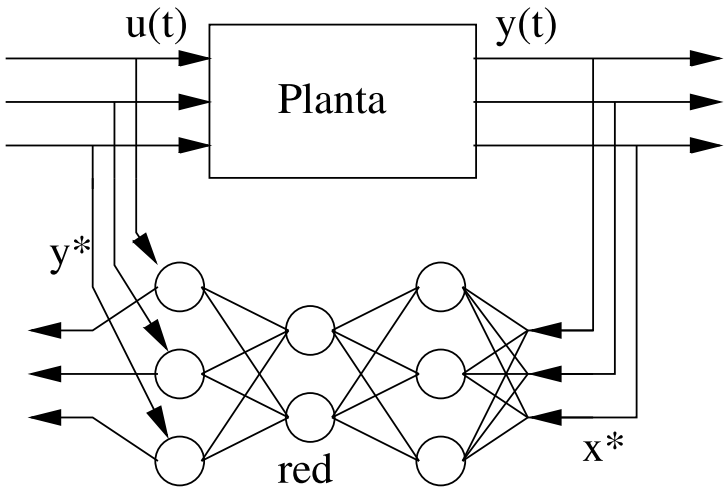


Figura 20: Entrenamiento de una red para identificar la planta inversa.

2.2.2. Predicción de la planta directa

Al igual que en el caso anterior, si no disponemos de información sobre el modelo de la planta podemos utilizar una red neuronal "copiar" la planta y utilizar esa red para sustituir el modelo de planta en un sistema de control por predicción como el explicado en el apartado 1.6.2. En la figura 21 se ha representado el modo de colocar la red neuronal para ajustar sus parámetros para responder igual que la planta y al mismo tiempo ir corrigiéndose para el entrenamiento.

En los métodos que utilizan las redes neuronales para hallar un modelo de planta, directa (como en este caso) o inversa (como en el apartado anterior), es muy útil recordar que la estructura de las redes neuronales que hemos comentado tiene muchas similitudes con la descripción modular que se ha explicado en el apartado 1.4.2.

Si recordamos que la manera de transformar las ecuaciones en diferencias de orden superior, (dependencias en valores de varios instantes anteriores) en

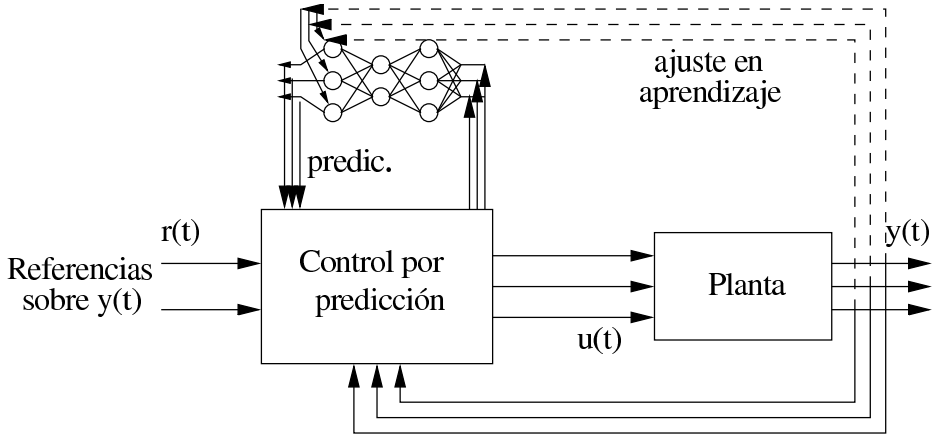


Figura 21: Uso de red neuronal como modelo de planta en control por predicción.

ecuaciones de primer orden, es generando nuevos estados sobre cada instante anterior, vemos que la estructura modular de una red neuronal puede representar un modelo de planta genérico simplemente incluyendo en la estructura de red neuronal básica los bloques de registro de memoria (o de retardo $\boxed{z^{-1}}$) adecuados.

2.2.3. Identificación de parámetros de la planta

En este caso se utiliza una red neuronal para realizar la estimación de parámetros de un modelo de planta para usarlos en un control basado en estimación de parámetros. La sustitución consiste en cambiar el estimador de parámetros por una red neuronal pero se mantiene el modelo de planta que aparecía representado internamente al estimador en la figura 17 de la pág. 27. Adicionalmente las diferencias entre las salidas del modelo y las de la planta real se comparan (diferencia) para utilizarlas como correcciones en el ajuste de la red (aprendizaje). Las salidas de la red son los parámetros calculados para la planta que se envían tanto al control basado en parámetros como al modelo ajustable. Todos estos elementos y conexiones se han representado en la figura 22.

2.2.4. Ajuste de parámetros de control tradicional

En algunas ocasiones se dispone de un control diseñado de forma paramétrica pero el ajuste de esos parámetros del controlador se le asigna como tarea a una red neuronal. Se entrena la red con ejemplos conocidos de ajustes para determinadas condiciones en la planta y en las consignas. Después se

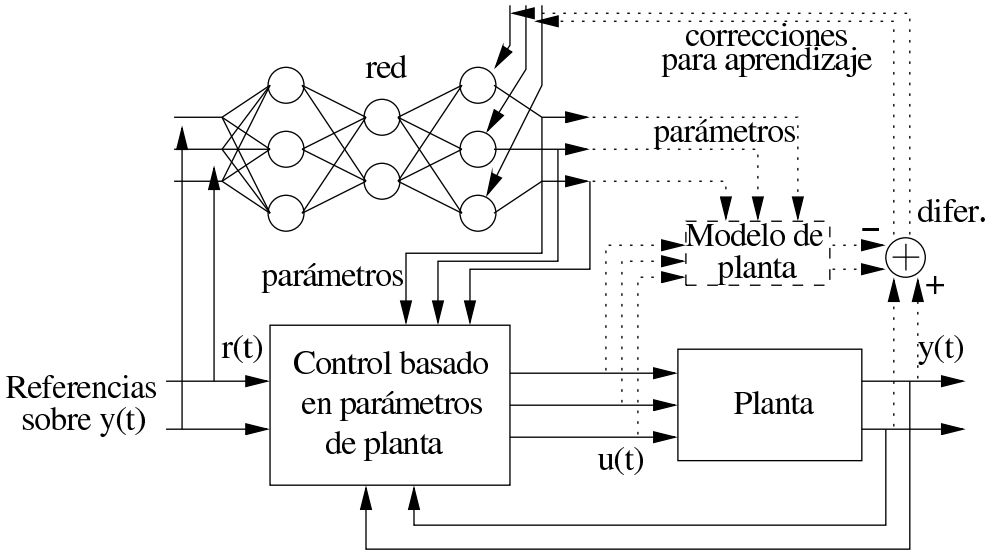


Figura 22: Uso de una red neuronal para calcular los parámetros de un modelo de planta con un controlador por estimación de parámetros.

conecta la red neuronal tal como se indica en la figura 23 para que genere los valores de los parámetros del controlador. No se debe confundir este modo con el anterior, aquí las salidas de la red neuronal son los parámetros *del controlador*, en el caso anterior eran los parámetros de la planta (que el controlador usaba).

2.2.5. Copiar un controlador tradicional

La capacidad de identificación de modelos basándose en ejemplos, que tienen las redes neuronales, también se puede aplicar en la identificación de un modelo de controlador, aparte de la identificación de la planta. En esta forma de utilización, la red neurona se entrena previamente, "aprendiendo" a imitar a un humano, o a un programa, o a un controlador tradicional que ya "sabe" controlar la planta. La forma de conexión para el entrenamiento se ha representado en la figura 24.

Después del entrenamiento, se puede sustituir la red neuronal directamente en el lugar del controlador. Evidentemente, esta forma de uso de las redes neuronales tiene más sentido y más utilidad cuando se trata de copiar a un control humano (p. ej. pilotos, conductores, etc.) para conseguir automatizar un sistema de control.

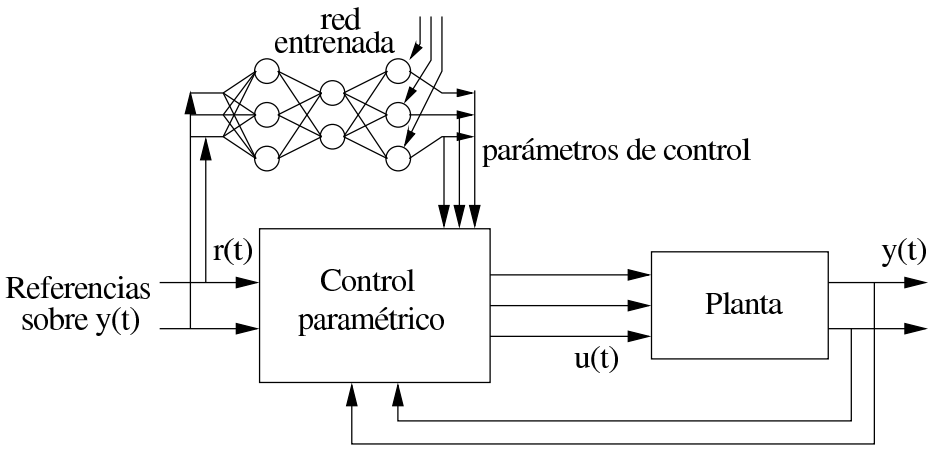


Figura 23: Ajuste de parámetros de control mediante una red neuronal ya entrenada previamente con ejemplos conocidos.

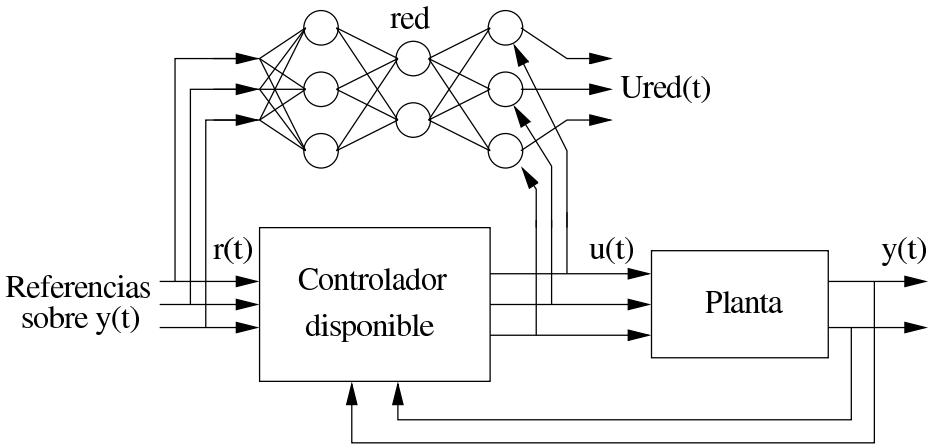


Figura 24: Conexión de una red neuronal para el entrenamiento en identificación de un controlador existente.

2.2.6. Control propio por redes

Finalmente, las redes neuronales se pueden utilizar directamente para realizar un controlador, utilizando criterios diferentes para el ajuste o entrenamiento. En las fórmulas de modificación de pesos, que aparecen en el resumen del algoritmo básico de retropropagación (pág. 31), se decía que el factor de corrección para la última capa es $\delta_j^L = \text{dersigm}\left(z_j^L\right) \cdot \left(y_j^* - y_j^L\right)$, el término de diferencia $\left(y_j^* - y_j^L\right)$ proviene de utilizar en la deducción una función de error a minimizar, que corresponde a la mitad de la suma de errores cuadráticos en las salidas. Cambiando la función a minimizar (o función de error) se pueden obtener otros valores para la corrección de pesos. De esta forma la red se puede ir ajustando al mismo tiempo que está controlando la planta. En la figura 25 se representa la forma de utilizar una red directamente para control.

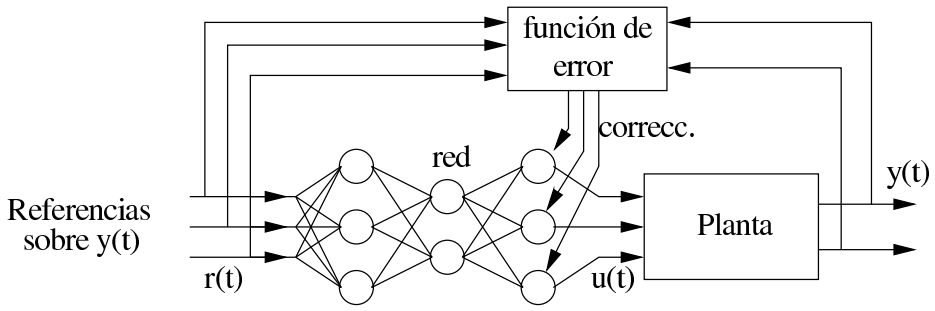


Figura 25: Red neuronal utilizada directamente como controlador.

Este método no se puede usar cuando una señal de control inadecuada pueda causar daños en la planta o en el entorno, ya que al principio la red generará valores de control poco eficientes e incluso erróneos para ir mejorando con el ajuste por el uso. Esto es si los ejemplos de entrenamiento se van produciendo directamente con el uso, pero también se podría entrenar a la red previamente con valores extraídos y guardados (o bien de un simulador) para después colocar la red ya entrenada a controlar.

3. Supervisión y protección en control

3.1. Sistemas de supervisión

Hasta ahora hemos visto que el objetivo de los sistemas de control es conseguir que una planta, o proceso industrial, siga un comportamiento marcado por señales de consigna. Estas señales de consigna son generadas por otros

sistemas, de un nivel superior, encargados de obtener otros objetivos generales (que pueden incluir a otros procesos y sus controles). Estos sistemas los denominamos de supervisión y los objetivos que deben cumplir se suelen especificar como minimización de funciones de coste (eficiencia, ahorro, producción, etc.). Las funciones de coste dependen de los valores de las salidas generadas por la planta que está siendo controlada, y también por las señales de control utilizadas para conseguir esa salida, coste del control (p. ej. combustible de propulsión, energía de motor, etc.).

En la figura 26 se ha representado el esquema de un bloque de supervisión acoplado a un circuito de control. La función de coste que se utiliza con los criterios de optimización se ha representado internamente en la supervisión.

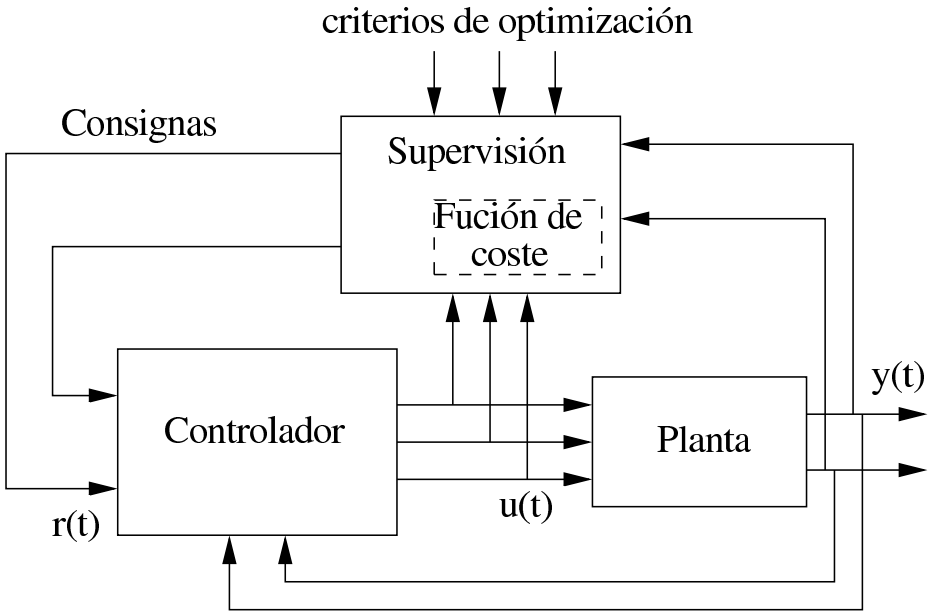


Figura 26: Esquema de supervisión sobre un lazo de control en una planta o proceso industrial.

3.2. Sistemas de protección

Los procesos industriales pueden involucrar acciones o tener consecuencias no deseables, en caso de que un control inadecuado sobrepase los límites de seguridad o en el caso de un accidente (que es una "modificación" instantánea de la planta, con la que el control no puede seguir actuando correctamente). Para evitar esta eventualidad, los procesos industriales con algún tipo de riesgo (rotura, explosión, contaminación, etc.) o que puedan resultar peligrosos por mal funcionamiento, incluyen, junto con los sistemas de control,

otros sistemas que permiten llevar *inmediatamente* el proceso o la planta a un estado seguro tomando acciones específicas de seguridad.

Normalmente las acciones de seguridad, llevadas a cabo por un sistema de protección, suelen ser del tipo "todo o nada" (apertura o cierre total de una válvula, desconexión de corriente, etc.) y tienen preferencia de actuación frente a las acciones del sistema de control (precisamente porque deben prevenir posibles fallos del control). Este tipo de sistemas deben estar incluidos en el diseño del mismo proceso industrial e incluso forman parte de él.

Para entrar en acción los sistemas de protección verifican constantemente que se cumplan una serie de condiciones de seguridad, en el caso de que alguna falle se acciona inmediatamente la protección adecuada. En la figura 27 se ha representado el esquema de conexión de un sistema de protección en un sistema controlado, el sistema de protección recibe los valores de la planta (incluso otros no recibidos por el control), los valores del control y las condiciones de seguridad.

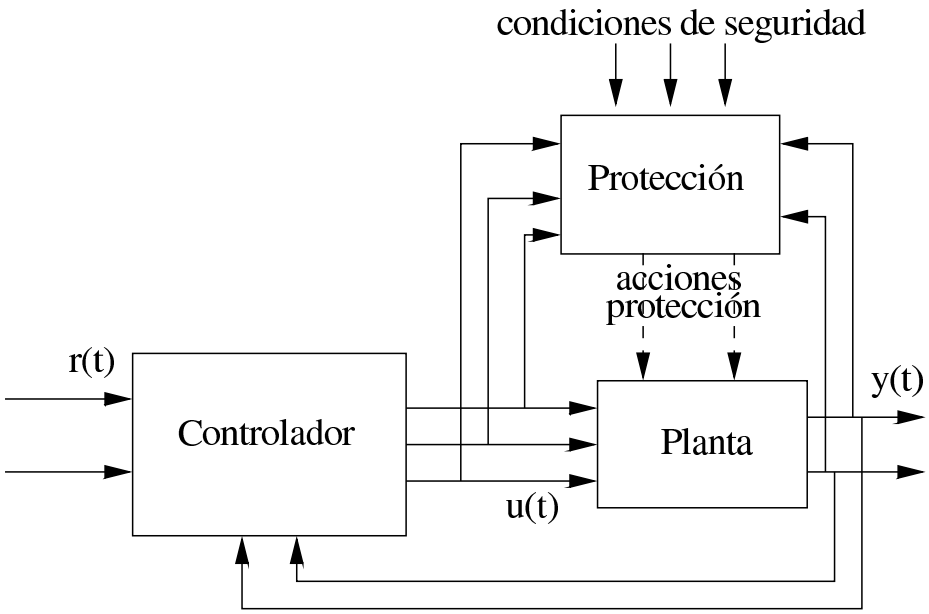


Figura 27: Esquema de conexión de los sistemas de protección sobre un circuito de control.

3.3. Lazo de realimentación en supervisión o en protección

Tanto los sistemas de supervisión como los sistemas de protección tienen la misma estructura de lazo de realimentación (ver figuras 26 y 27) que vimos

al principio (ver apartado 1.1 en la pág. 2), por lo tanto podemos considerar estos tipos de sistemas como un tipo especial de sistemas de control. Los sistemas de supervisión pueden ser considerarlos como controles de un nivel superior. Los sistemas de protección se pueden considerar, en cambio, como sistemas de control separados.

La diferencia, o característica, específica de los sistemas de supervisión frente a los sistemas de control normales es la constante de tiempo sobre la que actúan. Los sistemas de supervisión actúan más lentamente, cambiando poco a poco las consignas y evaluando el funcionamiento de la planta a lo largo de periodos de tiempo mayores que en el caso de los controles normales (que deben responder a la velocidad de la dinámica de la planta).

Los sistemas de protección se diferencian de los sistemas de control normal en que sus actuaciones de protección son del tipo "todo o nada", como ya hemos dicho, y que la función que realizan para calcular esas actuaciones suele ser de tipo lógico, de la forma:

```
REPETIR siempre
   PARA TODOS los criterios de seguridad i
      SI no se cumple criterio de seguridad i,
      ENTONCES activar protecciones de i
```

donde los criterios de seguridad son listas de condiciones sobre valores en variables que se deben cumplir simultáneamente para mantener la seguridad.