Manual de formación
para soluciones generales en automatización
Totally Integrated Automation (T I A)

MÓDULO B2

Procesamiento de valores analógicos

Agradecemos al Ingeniero Michael Dziallas, a los tutores de las escuelas de formación profesional, así como a todas aquellas personas que nos han prestado su colaboración para la elaboración de este documento.

PÁGINA:

1.	**Introducción**	4
2.	**Señales Analógicas**	6
3.	**Tipos de Datos en STEP 7**	8
4.	**Operaciones Matemáticas**	9
4.1.	Cálculo con números enteros (INT y DINT)	9
4.2.	Cálculo con números en coma flotante (REAL)	10
4.3.	Operaciones de Conversión de Tipos de Datos	11
5.	**Valores de Entradas/Salidas analógicas**	12
5.1.	Entrada y valor analógico Normalizado	13
5.2.	Normalización y valor analógico de salida	14

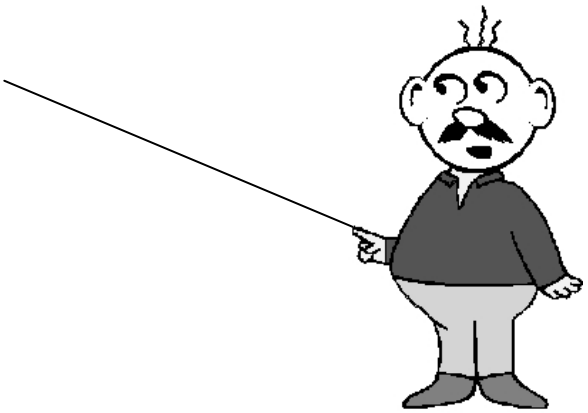
Los símbolos siguientes acceden a los módulos especificados:

Información

Programación

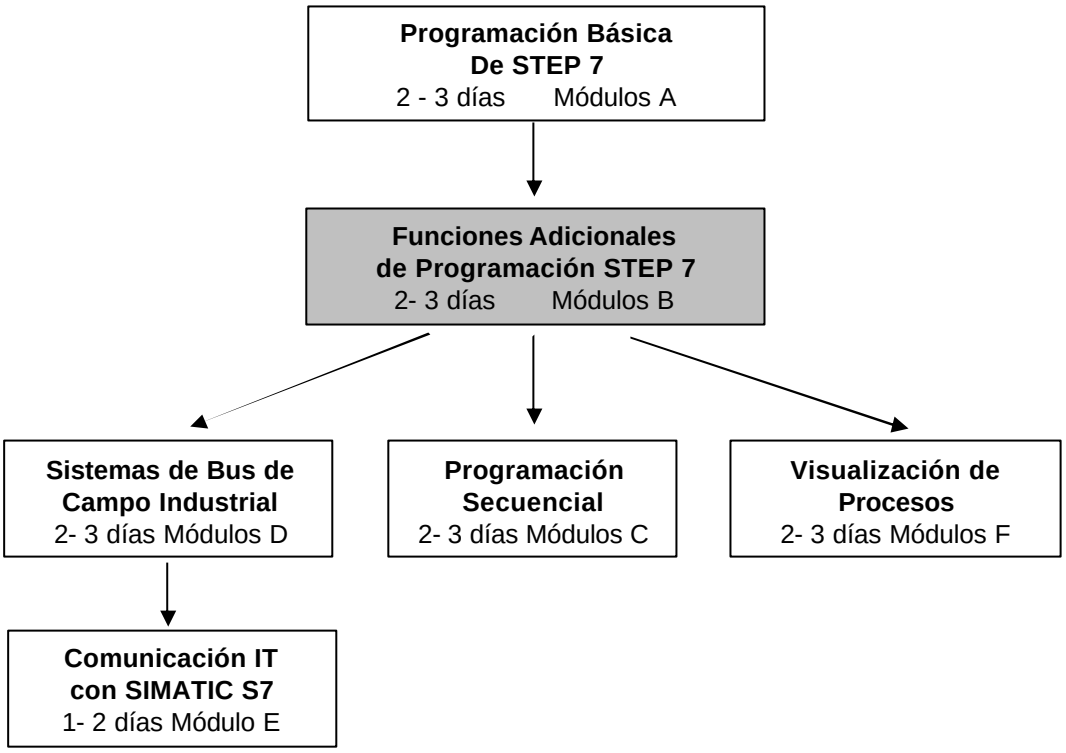
Ejercicio Ejemplo

Notas



1. INTRODUCCIÓN

El módulo B2 pertenece al contenido de las **Funciones Adicionales de Programación STEP 7**.



Finalidad del Aprendizaje:

En este módulo, el lector aprenderá sobre como los valores analógicos son introducidos, procesados y sacados en un PLC SIMATIC S7.

- Señales Analógicas
- Tipos de Datos en STEP 7
- Operaciones Matemáticas
- Conversión de Tipos de Datos en STEP 7
- Entradas y escalado de valores analógicos
- Desescalado y salida de valores analógicos

Requisitos:

Para el correcto aprovechamiento de este módulo, se requieren los siguientes conocimientos:

- Conocimientos de uso de Windows 95/98/2000/ME/NT4.0
- Programación Básica de PLC con STEP 7 (Módulo A3 - 'Puesta en Marcha' programando PLC con STEP 7)

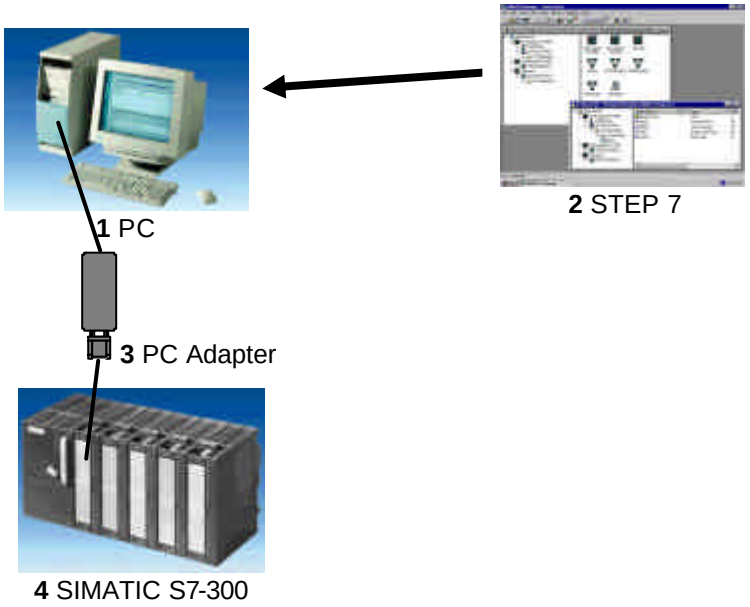
Hardware y software Necesarios

1 PC, Sistema Operativo Windows 95/98/2000/ME/NT4.0 con
- Mínimo: 133MHz y 64MB RAM, aprox. 65 MB de espacio libre en disco duro
- Óptimo: 500MHz y 128MB RAM, aprox. 65 MB de espacio libre en disco duro

2 Software STEP 7 V 5.x

3 Interfase MPI para PC (p.e. PC- Adapter)

4 PLC SIMATIC S7-300 con al menos un módulo de entradas/salidas analógicas, el cual debe tener un potenciómetro u otro transductor analógico conectado a una entrada analógica. También será necesario tener conectado a una salida analógica una visualizador de valores analógicos.
Ejemplo de configuración:
- Fuente de Alimentación: PS 307 2A
- CPU: CPU 314
- Entradas Digitales: DI 16x DC24V
- Salidas Digitales: DO 16x DC24V / 0.5 A
- Entradas/Salidas Analógicas: AI 4/ AO 2 x 8Bit



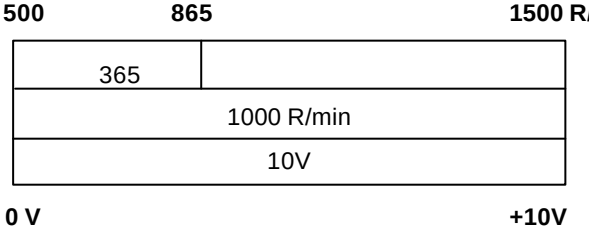
2. SEÑALES ANALÓGICAS

Al contrario que una señal binaria o digital, la cual puede aceptar solo dos valores 'Con Tensión +24V' y 'Sin Tensión 0V', las señales analógicas pueden aceptar tantos valores como se deseen, dentro de unos ciertos rangos. Un ejemplo típico de un transductor analógico es un potenciómetro. Dependiendo de la posición del mando, se proporciona un valor diferente de resistencia hasta un valor máximo.

Ejemplos de medidas analógicas en tecnologías de sistemas de control:

- Temperatura -50 ... +150°C
- Caudal 0 ... 200 l/min
- Revoluciones 500 ... 1500 R/min
- Etc…

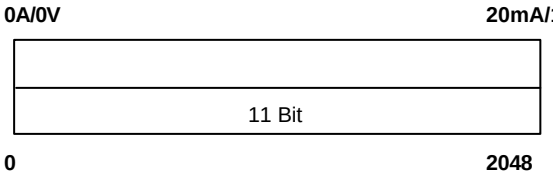
Estos valores son convertidos a valores de voltaje, intensidades de corriente o resistencias con la ayuda de transductores de medida. Por ejemplo, si se desea medir un determinado número de revoluciones, el cambio de velocidad puede convertirse en un rango de entre 500... 1500 R/min, a través de un captador de medida, en un voltaje que oscile entre 0... +10V. Cuando el número de vueltas sea de 865 R/min, el captador de medida emitirá un voltaje de + 3.65 V.



10V: 1000 R/min = 0,01 V/R/min

365 R/min x 0,01 V/R/min = 3,65

Si se procesan mediciones similares con un PLC, entonces el voltaje, intensidad o valor de resistencia introducido debe ser convertido a información digital. Esta conversión se denomina conversión Analógico--Digital (Conversión A/D). Esto significa que, por ejemplo, el valor de voltaje de 3.65V se deposita como información en un registro digital equivalente de 'unos' y 'ceros'. Cuanto mayor sea el número de dígitos binarios utilizados para la representación digital, mayor será la resolución. Si se hubiera utilizado, por ejemplo, un solo bit de resolución para el rango de voltaje 0... +10V, solo obtendríamos dos estados, uno en el rango de 0...+5V y otro en el de +5V...+10V. Con dos bits, el rango se puede dividir en 4 áreas individuales: 0... 2.5/2.5... 5/5... 7.5/7.5... 10V. Es muy usual que una conversión A/D en sistemas de control implique 8 u 11 bit de resolución. Se obtienen 256 áreas individuales con 8 bits y con 11 bits una resolución de 2048 áreas.



10V: 2048 = 0,0048828

® **Las diferencias de potencial <5mV son detectadas**

3. TIPOS DE DATOS EN STEP 7

En SIMATIC S7 existen diferentes tipos de datos, bajo los cuales pueden representarse diferentes formatos numéricos. A continuación, se muestra una lista completa de los tipos de datos

Tipo y descripción	Tamaño en Bits	Formato-Opciones	Rango y notación numérica (Valores máximo y mínimo)	Ejemplo
BOOL (Bit)	1	Texto Booleano	TRUE/FALSE	TRUE
BYTE (Byte)	8	Número Hexadecimal	B#16#0 a B#16#FF	B#16#10
WORD (Palabra)	16	Número Binario	2#0 a 2#1111_1111_1111_1111	2#0001_0000_0000_0000
		Número Hexadecimal	W#16#0 a W#16#FFFF	W#16#1000
		BCD	C#0 a C#999	C#998
		Número Decimal sin signo	B#(0,0) a B#(255,255)	B#(10,20)
DWORD (Doble Palabra)	32	Número Binario	2#0 a 2#1111_1111_1111_1111_1111_1111_1111_1111	2#1000_0001_0001_1000_1011_1011_0111_1111
		Número Hexadecimal	DW#16#0000_0000 a DW#16#FFFF_FFFF	DW#16#00A2_1234
		Número Decimal sin signo	B#(0,0,0,0) a B#(255,255,255,255)	B#(1,14,100,120)
INT (Entero)	16	Número Decimal con signo	-32768 a 32767	1
DINT (Int,32 bit)	32	Número Decimal con signo	L#-2147483648 a L#2147483647	L#1
REAL (Número en coma flotante)	32	Número en coma flotante IEEE	Máximo: +/-3.402823e+38 Mínimo: +/-1.175495e-38	1.234567e+13
S5TIME (Tiempo Simatic)	16	Tiempo S7 en pasos de 10 ms	S5T#0H_0M_0S_10MS a S5T#2H_46M_30S_0MS and S5T#0H_0M_0S_0MS	S5T#0H_1M_0S_0MS S5TIME#1H_1M_0S_0MS
TIME (Tiempo IEC)	32	Tiempo IEC en pasos desde 1ms, entero con signo	-T#24D_20H_31M_23S_648MS a T#24D_20H_31M_23S_647MS	T#0D_1H_1M_0S_0MS TIME#0D_1H_1M_0S_0MS
DATE (Fecha IEC)	16	Fecha IEC en pasos de 1 día	D#1990-1-1 a D#2168-12-31	DATE#1994-3-15
TIME_OF_DAY (Fecha y Hora)	32	Tiempo en pasos de 1ms	TOD#0:0:0.0 a TOD#23:59:59.999	TIME_OF_DAY#1:10:3.3
CHAR (Carácter)	8	Caracteres ASCII	´A´, ´B´ etc.	´B´

Nota: Para el procesamiento de valores analógicos, los tipos de datos **INT** y **REAL** juegan un papel fundamental, porque los valores analógicos introducidos existen como valores reales en el formato **INT**. Debido a errores de redondeo por el tipo **INT**, sólo los números reales **REAL** entran en juego para un posterior procesamiento preciso.

4. OPERACIONES MATEMÁTICAS

4.1 CALCULOS CON NÚMEROS ENTEROS (INT Y DINT)

Con números enteros, son posibles las operaciones unitarias matemáticas de suma, resta, multiplicación y división. No obstante, no se tienen en cuenta los lugares tras el punto decimal, lo cual genera errores de redondeo con la división.

Operación	Tamaño en Bits	Función
+I	16	Suma el contenido de la palabra baja de los ACCUs 1 y 2 y guarda el resultado el la palabra baja del ACCU 1.
-I	16	Resta el contenido de la palabra baja de los ACCUs 1 y 2 y guarda el resultado el la palabra baja del ACCU 1.
*I	16	Multiplica el contenido de la palabra baja de los ACCUs 1 y 2 y guarda el resultado (32 Bit) en ACCU 1.
/I	16	Divide el contenido de la palabra baja del ACCU 2 con la palabra baja del ACCU 1. El resultado es almacenado en la palabra baja del ACCU 1. El resto es almacenado en la palabra alta del ACCU 1.
+D	32	Suma los contenidos de los ACCUs 1 y 2 en el ACCU 1.
-D	32	Resta los contenidos de los ACCUs 1 y 2 en el ACCU 1.
*D	32	Multiplica los contenidos de los ACCUs 1 y 2 en el ACCU 1.
/D	32	Divide el contenido del ACCU 2 con el contenido del ACCU 1 y guarda el resultado en el ACCU 1.
MOD	32	Divide el contenido del ACCU 2 con el contenido del ACCU 1 y guarda el resto en el ACCU 1.

4.2 CÁLCULO CON NÚMEROS EN COMA FLOTANTE (REAL)

Con números en coma flotante, se pueden elaborar múltiples operaciones matemáticas. Aquí se consideran las posiciones a la derecha del punto decimal.

Operación	Función
+R	Suma de números en coma flotante (32 Bit, IEEE-FP) contenidos en los ACCUs 1 y 2 y guarda el resultado (32 bits) en el ACCU 1.
-R	Resta de números en coma flotante (32 Bit, IEEE-FP) contenidos en los ACCUs 1 y 2 y guarda el resultado (32 bits) en el ACCU 1.
*R	Multiplicación de números en coma flotante (32 Bit, IEEE-FP) contenidos en los ACCUs 1 y 2 y guarda el resultado (32 bits) en el ACCU 1.
/R	División de números en coma flotante (32 Bit, IEEE-FP). Se divide el contenido del ACCU 2 por el del ACCU 1. El resultado (32 bits) se guarda en el ACCU 1.
SQRT	Calcula la raíz cuadrada del número en coma flotante (32 Bit, IEEE-FP) contenido en el ACCU 1 y guarda el resultado (32 bits) en el ACCU 1.
SQR	Calcula el cuadrado del número en coma flotante (32 Bit, IEEE-FP) contenido en el ACCU 1 y guarda el resultado (32 bits) en el ACCU 1.
LN	Calcula el logaritmo neperiano del número en coma flotante (32 Bit, IEEE-FP) contenido en el ACCU 1 y guarda el resultado (32 bits) en el ACCU 1.
EXP	Calcula el número e del número en coma flotante (32 Bit, IEEE-FP) contenido en el ACCU 1 y guarda el resultado (32 bits) en el ACCU 1.
SIN	Calcula el seno del número en coma flotante (32 Bit, IEEE-FP) contenido en el ACCU 1 y guarda el resultado (32 bits) en el ACCU 1.
COS	Calcula el coseno del número en coma flotante (32 Bit, IEEE-FP) contenido en el ACCU 1 y guarda el resultado (32 bits) en el ACCU 1.
TAN	Calcula la tangente del número en coma flotante (32 Bit, IEEE-FP) contenido en el ACCU 1 y guarda el resultado (32 bits) en el ACCU 1.
ASIN	Calcula el arcoseno del número en coma flotante (32 Bit, IEEE-FP) contenido en el ACCU 1 y guarda el resultado (32 bits) en el ACCU 1.
ACOS	Calcula el arcocoseno del número en coma flotante (32 Bit, IEEE-FP) contenido en el ACCU 1 y guarda el resultado (32 bits) en el ACCU 1.
ATAN	Calcula el arcotangente del número en coma flotante (32 Bit, IEEE-FP) contenido en el ACCU 1 y guarda el resultado (32 bits) en el ACCU 1.

4.3 TIPOS DE DATOS- OPERACIONES DE CONVERSIÓN

Dado que frecuentemente los números no existen para posteriores procesamientos de formatos numéricos, estos números deben de ser ajustados con la ayuda de operaciones de conversión.

Operación	Función
BTI	Conversión BCD a entero (16 Bit). Esta operación convierte un número BCD contenido en el ACCU 1 en un entero (16 Bit). El resultado se deposita en el ACCU1
BTD	Conversión BCD a entero (32 Bit). Esta operación convierte un número BCD contenido en el ACCU 1 en un entero (32 Bit). El resultado se deposita en el ACCU1
ITB	Entero (16 Bit) convertido a BCD. Esta operación convierte un número entero (16 bits) contenido en el ACCU 1 en un número BCD. El resultado se deposita en el ACCU1
ITD	Entero (16 Bit) convertido a entero (32 bits). Esta operación convierte un número entero (16 bits) contenido en el ACCU 1 en un número entero (32 bits). El resultado se deposita en el ACCU1
DTB	Entero (32 Bit) convertido a BCD. Esta operación convierte un número entero (32 bits) contenido en el ACCU 1 en un número BCD. El resultado se deposita en el ACCU1
DTR	Entero (16 Bit) convertido a real (32 bits, IEEE-FP). Esta operación convierte un número entero (16 bits) contenido en el ACCU 1 en un número real (32 bits, IEEE-FP). El resultado se deposita en el ACCU1 (32 Bit, IEEE-FP).
RND	Redondeo a entero. Esta operación redondea el número convertido al entero superior. Cuando la fracción del número convertido sea de 5 o superior, se redondea al entero superior.
RND+	Redondeo al siguiente entero superior. Esta operación redondea el número convertido al siguiente entero superior.
RND-	Redondeo al entero inferior. Esta operación redondea el número convertido al valor de su parte entera.
TRUNC	Redondeo truncado. Esta operación toma sólo la parte entera del número.

Nota: En el caso de procesamiento del valor analógico, dicho valor analógico se muestra en formato **INT** y debería ser convertido a formato real para posteriores operaciones con precisión. Dado que dicha conversión no es directa, el valor se convertirá primero a DINT con **ITD** y después a **REAL** con **DTD**.

.

5. VALORES ANALÓGICOS DE ENTRADA/SALIDA

Los valores analógicos son introducidos en el PLC como información en tamaño palabra. El acceso a esta palabra se realiza con las instrucciones:

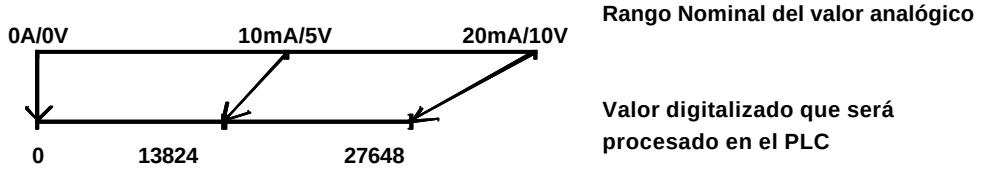
L PEW x para 'Cargar Palabra Analógica de Entrada'
T PAW x para 'Cargar Palabra Analógica de Salida'

Cada valor analógico ("Canal") corresponde a una palabra de entrada-salida. El formato es entero **INT**.

El direccionamiento de las palabras de entrada/salida analógicas dependen de la dirección de comienzo del módulo. Si el módulo analógico se coloca en el slot 4, su dirección de comienzo estándar es 256. La dirección de comienzo de cada módulo adicional se incrementa en 16 bytes. Esta dirección estándar se puede comprobar en la tabla de configuración hardware en la vista detallada.

Por ejemplo, la dirección de comienzo del primer canal del módulo de entradas analógicas colocado en el slot 6 del rack es la PEW 288. El segundo canal tiene la dirección PEW 290. Si el módulo fuera de entradas/salidas analógicas, el primer canal de salidas analógicas sería el PAW 288, etc. .

El proceso de transformación del valor analógico para el posterior procesamiento en el PLC (digitalización) es el mismo tanto para entradas como para salidas.
Para el módulo SM334, con 4 entradas y 2 salidas analógicas, con rango de tolerancias de 0 a 10V y de 20mA, respectivamente, el valor digitalizado se muestra de la siguiente forma:



Estos valores digitalizados deberán normalizarse en posteriores procesos en el PLC.

5.1. ENTRADA Y VALOR ANALÓGICO NORMALIZADO

Si un valor analógico es presentado como valor digitalizado, deberá normalizarse antes de ser procesado por el PLC.

De la misma forma, el valor de salida analógica del módulo de periferia es obtenido a través de un desescalado.

En un programa STEP 7, la normalización es exigida en la operación matemática.
Por esta razón, la operación matemática debe ser tan precisa como sea posible. Los valores que van a ser normalizados deben de ser convertidos a formato REAL para minimizar los errores de redondeo.

Ejercicio:

En el ejemplo siguiente, se introduce un valor de entrada analógica (rango de 0 a 10V) con un módulo analógico SM334 en el slot 6 (PEW288). Éste valor es representado como INT (16 Bits) y deberá normalizarse en un rango de 100 a 1000 en formato REAL, almacenándose después el resultado en la doble palabra de marcas MD10.

Solución en AWL:

```
L     PEW 288      //Valor analógico de entrada de 0 a 10 V: contiene valores enteros de 0 a 27648
                     (16 Bits)
ITD                //Conversión de entero (16 Bits) a entero (32 Bits)
DTR                //Conversión de entero (32 Bits) a valor real
L     2.7648e+4    //
/R                 //Division con el número real 27648
L     9.000e+2     //
*R                 // Multiplicación con el número real 900 (1000-100)
L     1.000e+2     //
+R                 // Suma con el número real 100 (Deriva)
T     MD10         // Valor normalizado 100 a 1000 en formato real
```

5.2. NORMALIZACIÓN Y VALOR ANALÓGICO DE SALIDA

Si se va a utilizar un valor estandarizado en un canal analógico de salida, éste deberá de procesarse. En un programa STEP 7, la normalización es exigida en la operación matemática.
Por esta razón, la operación matemática debe ser tan precisa como sea posible. Los valores que van a ser normalizados deben de ser convertidos de formato REAL a INT para minimizar los errores de redondeo. Los decimales que van después del punto se pierden.

Ejemplo:

En el ejemplo siguiente, se almacena un valor de 100 a 1000 en formato real en la doble palabra de marcas MD20 y se emitirá su valor normalizado de 0 a 10V en un módulo analógico de salidas SM334 (PAW288) .

Solución en AWL:

```
L     MD20          // Valor de 100 a 1000 en formato real
L     1.000e+2      //
-R                  // Resta con el valor real 100.0 (Deriva)
L     9.000e+2      //
/R                  // División con el valor real 900.0
L     2.7648e+4     //
*R                  // Multiplicación con el valor real 27648.0
RND                 // Redondeo a entero
T     PQW 288       // El número entero de 0 a 27648 (16 Bits) corresponde al valor analógico de
                        salida de 0 a 10 V
```