Los sensores desempeñan un papel fundamental en una gran mayoría de aplicaciones de los microcontroladores. Nuestro primer contacto con el mundo de los sensores será a través de un sensor analógico de temperatura. Los conceptos que trataremos en este capítulo nos permitirán sentar las bases para trabajar con otros sensores analógicos, como pueden ser un potenciómetro, un sensor de sonido, una LDR (del inglés *Light Dependent Resistor*), un acelerómetro o incluso un joystick.

Este capítulo cierra el bloque de introducción a las señales analógicas: si en los dos capítulos anteriores escribimos señales analógicas, lo que haremos en este será programar un sensor para leerlas. Trabajaremos con el siguiente sensor:

* Sensor analógico de temperatura TMP36 (en encapsulado TO-92).
* Opcional: termistor NTC, con la resistencia externa integrada (módulo KY-013).­

9.1 Proyecto “Cambio climático”

«El cambio climático afecta, en la actualidad, a todos los países en todos los continentes. Tiene un impacto negativo en la economía y la vida de las personas, las comunidades y los países. En el futuro, las consecuencias serán todavía peores». Esta cita, extraída de los Objetivos de Desarrollo Sostenible de las Naciones Unidas –en concreto del número 13, “Tomar medidas urgentes para combatir el cambio climático”–, resume en pocas palabras lo que hace tiempo sabemos sobre el cambio climático y sus efectos.



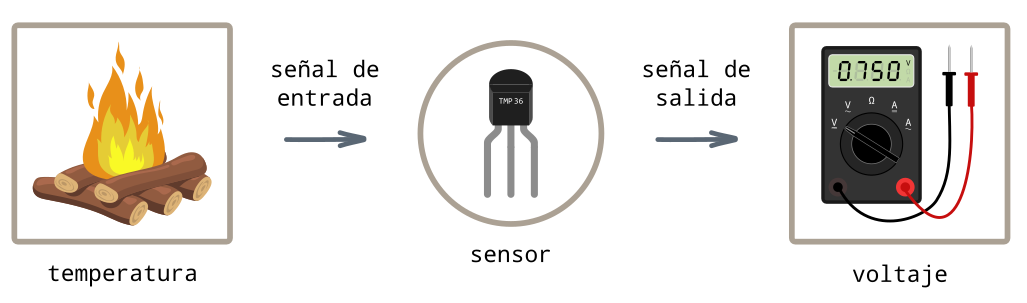
Para caracterizar el clima de la Tierra y poder obtener una imagen del cambio climático a escala mundial los científicos utilizan las llamadas variables climáticas esenciales. Estas variables, que proporcionan evidencia empírica sobre el cambio climático, son indicadores clave que se utilizan para evaluar riesgos climáticos, buscar las causas subyacentes de los fenómenos climáticos, planificar políticas a diferentes escalas, etc.

Existe una gran cantidad de sensores, económicos y fáciles de usar, que se pueden programar con un microcontrolador para estudiar algunas de las variables climáticas esenciales: concentración de CO2, precipitaciones, temperatura, humedad, radiación infrarroja y ultravioleta… En este capítulo, en concreto, programaremos un sensor de temperatura que podremos utilizar para elaborar proyectos más complejos relacionados con una de las variables que más importancia tiene en el estudio del cambio climático.

9.2 ¿Qué es un sensor?

Vivimos rodeados de sensores: puertas que se abren automáticamente ante nosotros, farolas que se encienden cuando hay poca luz, pantallas táctiles que responden a los movimientos de nuestros dedos… Todos estos ejemplos funcionan básicamente de la misma manera: un sensor detecta los valores de una propiedad del mundo físico y los transforma en señales eléctricas, que son enviadas a un microcontrolador. El microcontrolador procesa estas señales y manda a los dispositivos de salida –los actuadores– las órdenes pertinentes para que actúen de la manera programada. Este ciclo sensor-microcontrolador-actuador se repite una y otra vez, de manera que la respuesta obtenida se adapta siempre al estímulo recibido por el sensor.

Para que la magnitud detectada por el sensor pueda ser leída por el microcontrolador es imprescindible que se transforme en una magnitud eléctrica. Este es precisamente el papel de los sensores: recibir señales físicas, químicas o biológicas –temperatura, presión atmosférica, radiación ultravioleta, intensidad de campo magnético, aceleración, nivel de pH...– y transformarlas en señales eléctricas que puedan ser interpretadas por el microcontrolador.



La señal de entrada (la magnitud que queremos medir) es la temperatura. El sensor lee la temperatura y devuelve un voltaje como señal de salida.

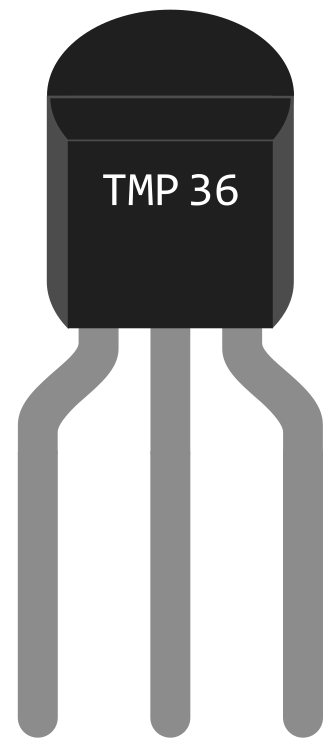
Cuando la salida de un sensor es una señal eléctrica como voltaje o resistencia –es decir, una señal analógica continua– se dice que es un sensor analógico; este es el tipo de sensores a los que dedicaremos este capítulo. Otros sensores, sin embargo, incorporan componentes electrónicos que transforman la medida en una secuencia de ceros y unos, es decir, proporcionan una salida digital; a estos se les llama sensores digitales.

Existen multitud de sensores analógicos que se pueden utilizar con MicroPython para medir las más variadas magnitudes; además, la mayoría de ellos son baratos y fáciles de conseguir. En este capítulo trabajaremos con el sensor de temperatura TMP36, pero los conceptos básicos que trataremos son aplicables a cualquier sensor analógico.

9.3 El sensor de temperatura TMP36

El TMP36 es un sensor de temperatura semiconductor, lo cual quiere decir que en su interior hay un diodo semiconductor en el que el voltaje cambia con la temperatura. Por tanto, la señal de entrada (la magnitud física que queremos medir) es la temperatura y la señal de salida (la magnitud eléctrica en la que se transforma la temperatura para poder ser leída por el microcontrolador) es el voltaje.

Este sensor es barato y fácil de conseguir (usaremos el que se presenta en el encapsulado TO-92 de tres pines, que es muy habitual) y, además, el montaje y la programación son muy sencillos.



El sensor de temperatura TMP36 en el encapsulado TO-92.

Lo primero que debemos hacer siempre para trabajar con un sensor es consultar su hoja de datos o ficha técnica –*data sheet*, en inglés–. Escribiendo en un buscador “tmp36 data sheet” aparecerán multitud de sitios web que alojan la hoja de datos del sensor. Este documento proporciona toda la información técnica del sensor, la mayoría de la cual no será de nuestro interés; habitualmente nos llegará con consultar la primera página, donde se suelen resumir las características generales de montaje y uso. A continuación vamos a analizar los datos que resultan pertinentes en la hoja de datos del TMP36 y los explicaremos brevemente.

Características del sensor

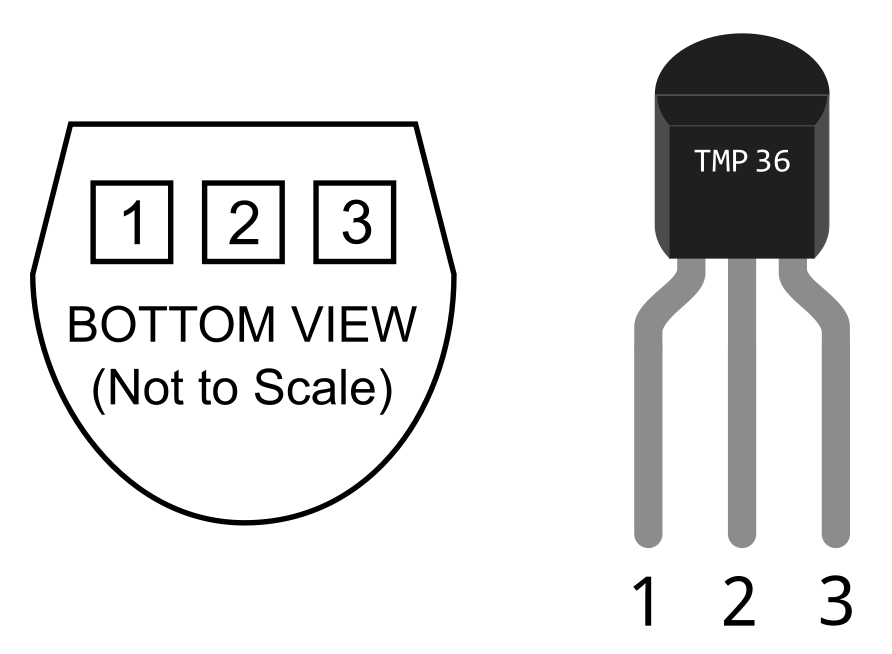
En las características generales del TMP36, que se resumen en la primera página de la hoja de datos, encontramos la siguiente información:

* «Mide temperaturas entre −40 ºC y +125 ºC.» Es importante saber en qué rango de medidas funciona un sensor; si lo sometemos a valores fuera de este intervalo no tenemos la garantía de obtener valores correctos y, además, podemos estropearlo.
* «Funciona con voltajes bajos de alimentación, entre 2.7 y 5.5 voltios.» Este es un dato que siempre debemos tener en cuenta cuando escojamos un sensor. Los microcontroladores con los que estamos trabajando suministran 3.3 V, así que estamos dentro del rango de utilización del sensor.
* «Proporciona un voltaje de salida proporcional a la temperatura en grados centígrados.» Esta característica hace que sea muy sencillo transformar el voltaje proporcionado por el sensor en temperatura, como veremos más adelante.
* «Se obtiene un voltaje de 750 mV a 25 °C y el factor de escala de salida es 10 mV/°C.» Estas especificaciones, que explicaremos más adelante, nos servirán para establecer la relación matemática entre el voltaje de salida y la temperatura.

No debemos confundir el voltaje de alimentación con el voltaje de salida. El voltaje de alimentación, indicado como *+VS* en la hoja de datos –la letra S indica *supply* o suministro– es el voltaje de la fuente de alimentación a la que está conectado el sensor (en nuestro caso, los 3.3 V que suministra el microcontrolador). A su vez, el voltaje de salida o *VOUT* es el voltaje variable que mide el sensor, que es proporcional a la temperatura.

9.4 Los pines de conexión

En la hoja de datos también se indica cuál es la configuración de los pines del sensor. Mirando de frente la parte plana del sensor, los terminales están numerados como 1, 2 y 3 de izquierda a derecha.



Izquierda: esquema proporcionado en la hoja de datos. Derecha: se muestra la numeración de los pines si el sensor se coloca mirando la parte plana.

Un poco más adelante en la hoja de datos se indica la función que cada uno de los pines tiene asignada, tal y como se recoge en la siguiente tabla:

|  |  |  |  |
| --- | --- | --- | --- |
| Pin | 1 | 2 | 3 |
| Función | *+VS* | *VOUT* | GND |

*+VS* es el voltaje de alimentación y *VOUT* es el voltaje de salida medido en el sensor; GND, como habitualmente, es la conexión a tierra.

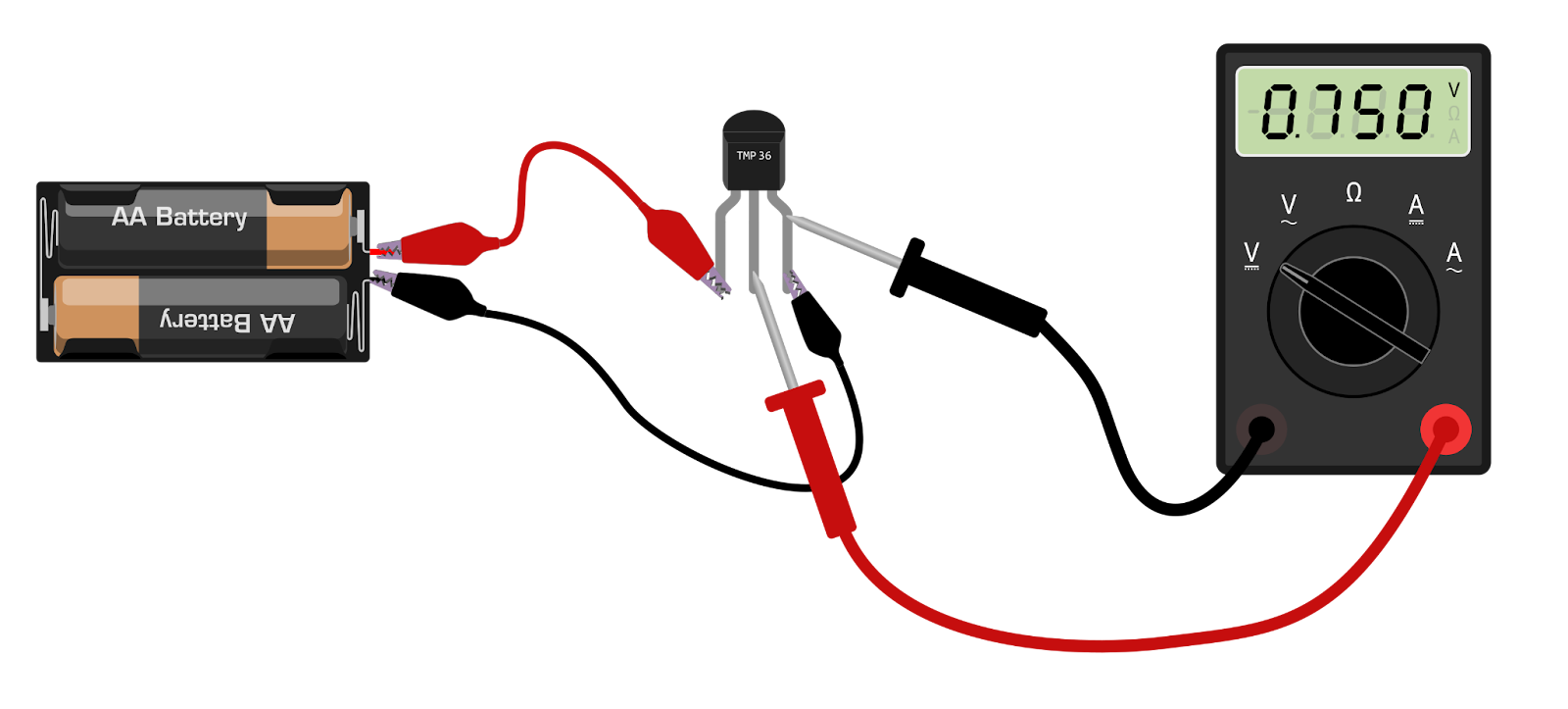
9.5 Probando el sensor

Antes de seguir adelante proponemos un pequeño experimento para ver cómo funciona este sensor. Para llevarlo a cabo usaremos el siguiente material (en esta prueba no vamos a usar la placa):

* El sensor de temperatura TMP36.
* Dos o tres pilas AA de 1.5 voltios en un portapilas, o una pila de petaca de 4.5 V.
* Un polímetro.
* Cables de cocodrilo.

Para funcionar, el sensor necesita estar conectado a una fuente de alimentación (nada sorprendente en un dispositivo electrónico). Como vimos en la hoja de datos, la tensión de alimentación debe estar comprendida entre 2.7 V y 5.5 V; podemos usar, por tanto, dos o tres pilas AA de 1.5 voltios asociadas en serie o una pila de petaca de 4.5 V (o cualquier otra fuente de alimentación continua que proporcione un voltaje en el rango indicado). Usamos los cables de cocodrilo para conectar el pin 1 del sensor (*+VS* ) al polo positivo de la pila y el pin 3 (GND) al polo negativo.

Vamos ahora a medir el voltaje de salida del sensor. Seleccionamos en el polímetro la medida de voltaje en corriente continua. Conectamos la terminal positiva (roja) del polímetro al pin central del sensor, y la negativa (negra) al pin GND, como en el dibujo.



El polo positivo de la fuente de alimentación está conectado al pin *+VS* del sensor y el negativo al GND. Con el polímetro se mide la diferencia de potencial (el voltaje) entre los pines *VOUT* y GND.

Si las conexiones están bien hechas, a una temperatura de 25 ºC se debería leer en el polímetro un valor de 750 mV. Tocando con los dedos el sensor, el voltaje debería acercarse a los 800 mV. Acercando algo frío, el voltaje debería disminuir a menos de 600 mV. De esta sencilla manera hemos comprobado cómo, efectivamente, el voltaje de salida del sensor depende de la temperatura.

9.6 Conversión analógico-digital

En este apartado analizaremos cómo funciona la conversión de señal analógica en digital. Vaya por delante que los conceptos pueden resultar un tanto áridos, pero es necesario que sepamos qué es lo que sucede para poder entender algunas de las instrucciones con las que trabajaremos más adelante.

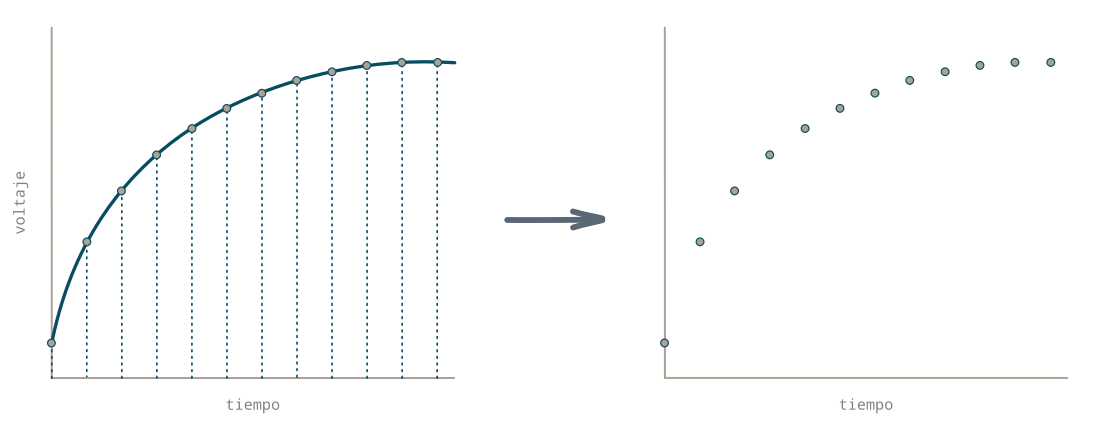
El conversor analógico-digital o ADC

Como hemos comentado anteriormente, el sensor TMP36 es un sensor analógico. Esto supone un gran problema, ya que los microcontroladores no pueden trabajar directamente son señales analógicas. Por tanto, para que puedan leer el voltaje hay que convertir la señal analógica en un formato legible por el microcontrolador; este proceso se conoce como conversión analógico-digital.

Para realizar la conversión de analógico a digital los microcontroladores disponen de un elemento de hardware llamado conversor analógico-digital o, por sus siglas en inglés, ADC (de *Analog to Digital Converter*).

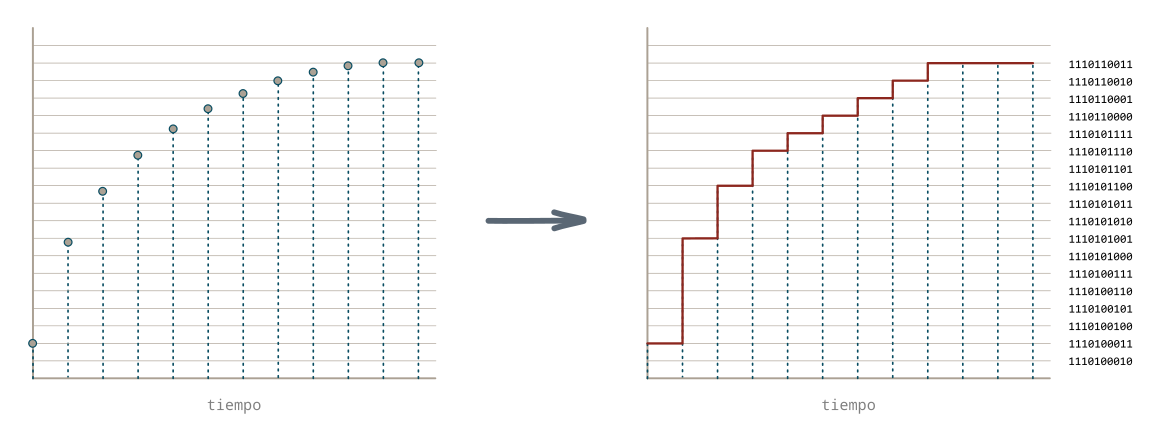
Muestreo, cuantificación y codificación

El proceso de conversión de una señal analógica en digital tiene tres pasos: muestreo, cuantificación y codificación. El primer paso es el muestreo de la señal original. El muestreo consiste en tomar valores de la señal a intervalos regulares de tiempo; de esta manera, la señal continua es sustituida por un conjunto finito de valores. Es decir, el muestreo convierte la señal analógica continua en una señal que aún es analógica pero ahora es discreta.



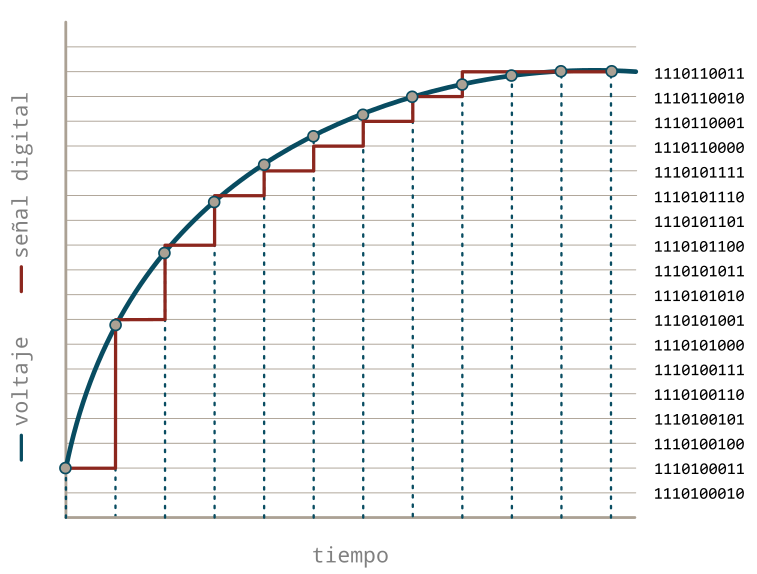
En la gráfica de la derecha se ha sustituido la señal analógica continua por los valores discretos que toma en los instantes señalados con la línea de puntos en la gráfica de la izquierda.

A continuación se procede a la cuantificación de la señal. En este proceso cada uno de los valores obtenidos del muestreo se hace corresponder con el valor que más se le aproxima de entre los valores predeterminados por la configuración del ADC. De esta manera la señal, que hasta ahora podía tomar infinitos valores, se restringe a un conjunto finito de valores, cada uno de los cuales está codificado, es decir, tiene asociado un código digital (un conjunto de ceros y unos). Por tanto, ya tenemos la señal analógica original convertida en una señal digital.



Cada valor obtenido del muestreo se hace corresponder con el valor más cercano dentro de los predefinidos en el ADC.

En conclusión, mediante la conversión analógico-digital la señal original se ha transformado en una secuencia de ceros y unos (representación binaria) que, ahora sí, pueden ser manejados por el microcontrolador. Sin embargo, dado que este proceso transforma un rango continuo de valores en un conjunto discreto, la señal digitalizada no es exactamente igual que la original, lo cual implica un error en la aproximación. El error de cuantificación es inevitable, pero podemos estar tranquilos porque no será significativo cuando utilicemos la conversión ADC en nuestro microcontrolador.

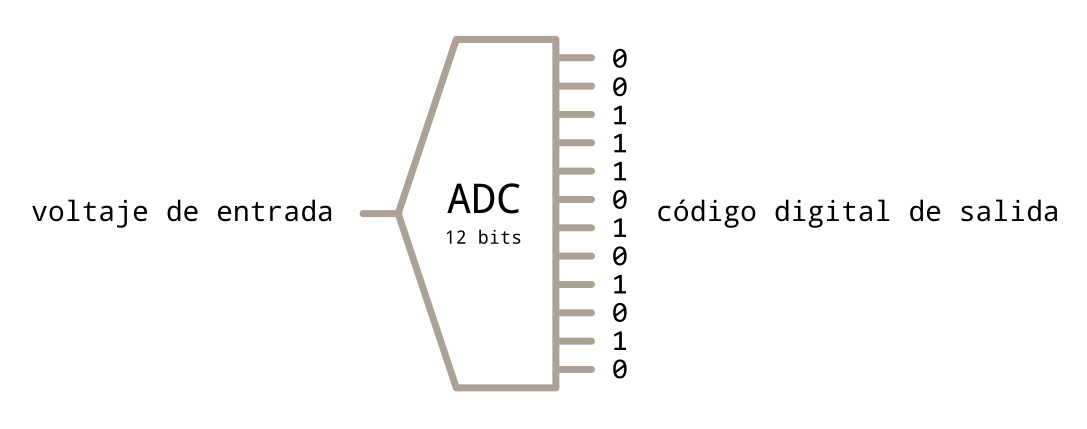


Señal analógica original y valores discretos en los que se ha transformado. Se aprecia el error introducido por la cuantificación ya que la señal digitalizada no es exactamente igual que la original.

Resolución del ADC

Cuando hablamos del proceso de cuantificación de la señal dijimos que el ADC tenía una serie de niveles predeterminados, cada uno de ellos asociado a un código digital, que eran los únicos valores que podía tomar la señal. La pregunta que surge inmediatamente es: ¿cuáles son esos niveles? La respuesta la proporciona una característica fundamental del conversor analógico-digital: su resolución.

El número de niveles de cuantificación de un ADC, es decir, el número de valores diferentes que puede tomar la señal, queda determinado por su resolución en bits. Imaginemos que tenemos un conversor cuya resolución es de 12 bits. Eso quiere decir que la salida del ADC es un número de 12 bits, es decir, un número binario compuesto por 12 cifras de las cuales cada una puede ser cero o uno.



Un ADC de 12 bits proporciona a su salida un número binario de 12 bits.

Haciendo unas cuentas rápidas, un conversor de 12 bits da 212 = 4096 combinaciones posibles de ceros y unos. Por tanto, hay 4096 niveles posibles, cada uno de los cuales está codificado, en binario, como un valor desde 0 (000000000000) hasta 4095 (111111111111). En general, un conversor de n bits da 2n niveles de salida con valores entre 0 y 2n − 1.

|  |  |  |
| --- | --- | --- |
| Resolución del ADC | Valores de salida | Rango |
| 10 bits | 1024 | 0 - 1023 |
| 12 bits | 4096 | 0 - 4095 |
| 16 bits | 65536 | 0 - 65535 |

Dado que el ADC es un elemento de hardware, su implementación varía entre los distintos microcontroladores. Para saber cuál es la resolución del conversor ADC de nuestra placa debemos consultar la documentación.

9.7 Los pines analógicos

Para escoger el pin del microcontrolador donde leeremos los datos medidos por el sensor debemos tener en cuenta que estamos trabajando con un sensor analógico. Acabamos de ver que la lectura de las señales analógicas las lleva a cabo el conversor analógico-digital, pero no todos los pines de la placa están conectados a este elemento de hardware. A los pines de la placa que podemos usar como entrada de una señal analógica se les llama entradas analógicas, pines analógicos o pines ADC.

Los pines analógicos dependen de la placa. El ESP32 puede tener hasta veinte pines ADC, dependiendo del modelo, mientras que la Raspberry Pi Pico solo pone a nuestra disposición tres entradas analógicas. Además, algunos de los pines ADC tienen adjudicadas otras funciones, así que en ciertos casos pueden no estar disponibles.

Mostramos aquí una tabla con las posibles entradas analógicas de las diferentes placas (estos valores se deben confirmar en la documentación de la placa). Es muy importante tener en cuenta esta información porque, por ejemplo, el GPIO14 puede ser utilizado para Arduino y ESP32, pero no para la Raspberry Pi Pico.

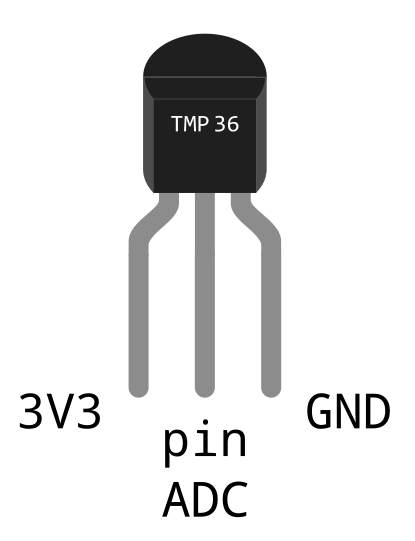
|  |  |  |
| --- | --- | --- |
| Arduino | ESP32 | Raspberry Pi Pico |
| A0: GPIO1  A1: GPIO2  A2: GPIO3  A3: GPIO4  A4: GPIO11  A5: GPIO12  A6: GPIO13  A7: GPIO14 | La mayoría de los GPIO (depende del modelo). | A0: GPIO26 (pin 31)  A1: GPIO27 (pin 32)  A2: GPIO28 (pin 34) |

Las entradas analógicas nunca deben ser sometidas a un voltaje superior a los 3.3 V, ya que se pueden dañar los pines o incluso la placa.

9.8 El circuito

Ahora que ya tenemos identificados los pines analógicos en la placa procedamos a montar el circuito.

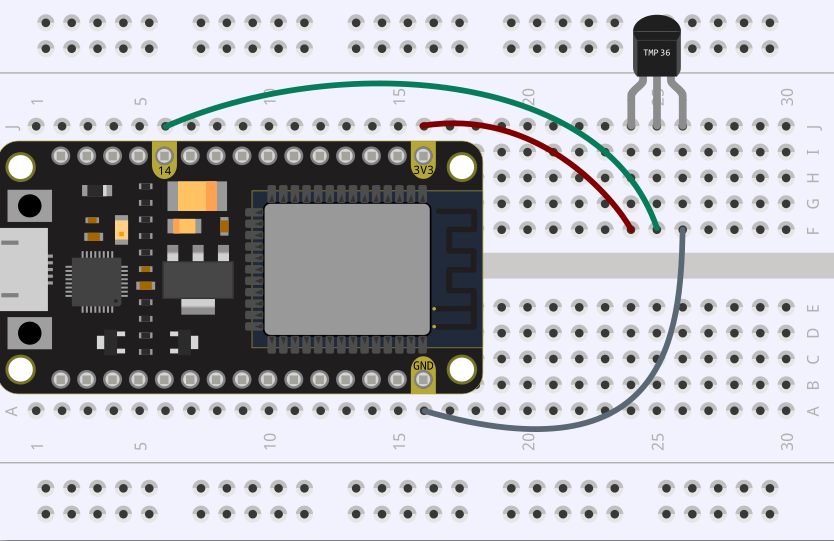
Como leímos en la ficha técnica del sensor, para que funcione hay que conectarlo a una fuente de alimentación de entre 2.7 y 5.5 voltios, por lo que podemos usar el microcontrolador como si fuese una pila de 3.3 V. El pin de datos lo conectaremos a una entrada analógica.



El pin de datos del sensor (el central) se conecta a una entrada analógica de la placa. Los de los extremos van a 3.3 V y a GND, respectivamente.

Por tanto, los tres pines del microcontrolador que usaremos para conectar el sensor son los siguientes:

* *+VS*: voltaje de alimentación (pin 3.3V, 3V3 o 3V3 Out).
* GND: tierra (pin GND).
* *VOUT*: cualquier pin analógico (ver apartado anterior).



|  |  |  |  |
| --- | --- | --- | --- |
|  | Arduino | ESP32 | Raspberry Pi Pico |
| Pin ADC | GPIO14 (pin A7) | GPIO14 | GPIO28 (pin 34) |

9.9 La clase **ADC** del módulo **machine**

La clase ADC del módulo machine nos proporciona el acceso al conversor analógico-digital de la placa; es, por tanto, lo que necesitamos para leer señales analógicas. Es importante tener en cuenta que, dado que el conversor ADC es un elemento de hardware, existen diferencias en su comportamiento en los distintos microcontroladores.

El constructor

La sintaxis del constructor es muy sencilla. Aunque acepta más argumentos, los vamos a pasar por alto ya que no son necesarios en un capítulo de introducción como este.

|  |
| --- |
| ADC(id) |
| Crea un objeto ADC que representa la implementación en software del pin indicado en el argumento.  El argumento id es un objeto Pin que identifica el pin de la placa donde se leerá la señal analógica. |

Los métodos

Para leer la señal en un pin analógico usaremos uno de los métodos siguientes, dependiendo de la placa con la que estemos trabajando. Estos métodos se aplican a un objeto ADC previamente creado.

|  |  |
| --- | --- |
| read\_uv() | Lee la señal medida en el pin. Devuelve el voltaje, expresado en microvoltios. Es el método que usaremos si estamos trabajando con una placa ESP32 o Arduino (no está disponible en la Raspberry Pi Pico). |
| read\_u16() | Lee la señal medida en el pin. Devuelve el valor bruto con una resolución de 16 bits, es decir, un entero entre 0 y 65535. Es el método que usaremos si estamos trabajando con la Raspberry Pi Pico. |

9.10 Comprobación del pin

Es posible que a estas alturas nos estemos preguntando lo siguiente: ¿cómo sabemos si, efectivamente, el pin al que hemos conectado el sensor está asociado a un conversor analógico-digital? La respuesta es fácil: en el REPL, creamos un objeto ADC asociado a dicho pin, y si no recibimos ningún mensaje de error es que el pin es adecuado.

>>> from machine import Pin, ADC

>>> TMP36 = ADC(Pin(14))

Si todo va bien, con estas dos instrucciones se ha creado un objeto ADC, llamado TMP36 (se puede usar cualquier otro nombre), asociado al GPIO14 de la placa –en la Raspberry Pi Pico es necesario cambiar este número, ya que los únicos que tienen funcionalidad ADC son los GPIO 26, 27 y 28–. Este objeto representa la implementación en software de la salida analógica asociada al sensor; dicho de otra manera, cuando necesitemos referirnos al sensor en el programa lo haremos a través de este objeto.

Por otro lado, si al crear el objeto anterior hemos utilizado un pin que no tiene la funcionalidad ADC el constructor devuelve el mensaje de error ValueError: invalid pin o ValueError: Pin doesn't have ADC capabilities. Esta es otra manera de comprobar cuáles son los pines ADC de la placa.

9.11 Lectura del voltaje con **read\_uv()**

Si estamos usando un microcontrolador basado en el ESP32, el método read\_uv(), que actúa sobre un objeto ADC, devuelve el voltaje medido en el pin correspondiente, expresado en microvoltios. Esta función no está disponible para la Raspberry Pi Pico; en este caso, para leer el voltaje hay que seguir los pasos que se indican en el siguiente apartado.

Apliquemos este método al objeto previamente creado para obtener el voltaje de salida del sensor, en microvoltios. Como un microvoltio –1 μV– es la millonésima parte de un voltio, transformamos el voltaje en voltios dividiendo el valor anterior entre un millón.

>>> TMP36.read\_uv()/1000000

0.758

El sensor ha devuelto un voltaje de 0.758 V. Cuando presentamos el sensor leímos en la hoja de datos que el voltaje a una temperatura de 25 ºC era 750 mV o, lo que es lo mismo, 0.750 V. El valor obtenido, por tanto, parece razonable; enseguida veremos cómo transformar este voltaje en temperatura.

9.12 Lectura del voltaje con **read\_u16()**

Si estamos usando la Raspberry Pi Pico no se puede obtener directamente el voltaje como hicimos en el apartado anterior, sino que hay que usar el método read\_u16(). Aunque este método también se puede usar con el ESP32, el voltaje de referencia (más abajo se explica su significado) es algo problemático, por lo que es preferible usar el método indicado en el apartado anterior.

Cuando hablamos de la conversión analógico-digital vimos que el conversor ADC devuelve el valor digital correspondiente al valor analógico de entrada. A este número le vamos a llamar valor bruto o valor sin procesar –en inglés, *raw value*–. La clase ADC pone a nuestra disposición el método read\_u16() para leer este valor bruto. Este método devuelve un valor entre 0 y 65535 que se corresponde con la transcripción digital, con una resolución de 16 bits, del voltaje de salida del sensor.

Medida del valor bruto

Utilizando el método read\_u16() sobre el objeto previamente creado podemos leer el valor bruto de salida del sensor.

>>> TMP36.read\_u16()

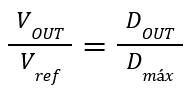
15075

Pero, ¿qué significa una lectura de 15075? Lo que necesitamos obtener es el voltaje proporcionado por el sensor (igual que hicimos con el polímetro); por tanto, el siguiente paso es transformar el valor bruto en voltaje. Y, para ello, necesitamos hacer unas pocas cuentas.

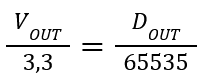
Relación entre el valor bruto y el voltaje

Para proporcionar el valor bruto de salida, el conversor ADC utiliza un voltaje interno, llamado voltaje de referencia, con el que compara la señal de entrada. En la Raspberry Pi Pico este voltaje es 3.3 V.

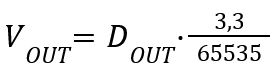
Un voltaje de referencia de 3.3 V implica que el ADC transforma 3.3 V en el valor 65535. Dado que 0 V se corresponde con el valor 0, cuando read\_u16() devuelve un valor intermedio entre 0 y 65535 el voltaje será la proporción correspondiente entre 0 V y 3.3 V. Matemáticamente, por tanto, es muy sencillo calcular el voltaje de salida que se corresponde con una lectura determinada. Si llamamos *Vref* al voltaje de referencia y *Dmáx* al valor máximo devuelto por el ADC, la relación que hay entre la medida proporcionada por el ADC, *DOUT*, y el voltaje correspondiente, *VOUT*, es:



En nuestro caso *Vref* es 3.3 V y *Dmáx* es 65535; por tanto, la proporción anterior equivale a:



Si, finalmente, despejamos el voltaje obtenemos:



Esta expresión proporciona el voltaje *VOUT*, en voltios, que equivale al valor bruto medido *DOUT*.

Medida del voltaje

Ya estamos en condiciones de obtener el voltaje medido por el sensor.

>>> lectura = TMP36.read\_u16()

>>> voltaje = lectura\*3.3/65535

>>> print(voltaje)

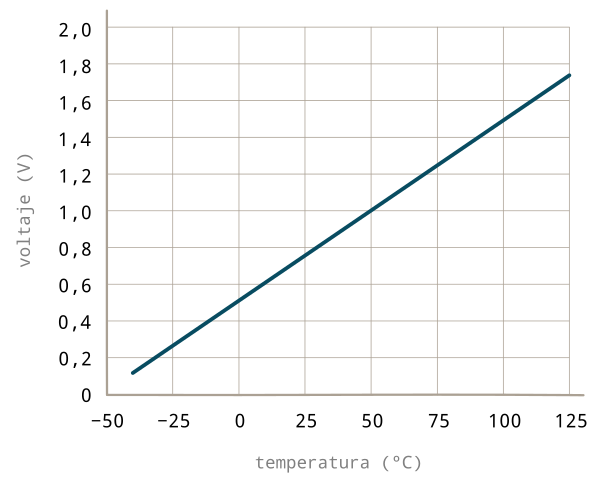
0.7590982

Es decir, el voltaje es 0.759 V. De esta manera podemos transformar en voltaje la lectura en bruto de cualquier señal analógica usando la Raspberry Pi Pico.

9.13 Voltaje vs. temperatura

Ya sabemos cómo leer el voltaje de salida del sensor TMP36; el último paso es convertir este voltaje en una temperatura. Para conseguirlo, debemos recurrir de nuevo a la hoja de datos del sensor.

En la ficha técnica se indican las siguientes características del sensor: el voltaje de salida es proporcional a la temperatura en grados centígrados, el voltaje a 25 ºC es de 750 mV y el factor de escala de salida es 10 mV/ºC. Estos datos, además, se incluyen en forma de gráfica: es la llamada curva característica del sensor. Veamos qué quiere decir esta información y cómo la podemos utilizar.



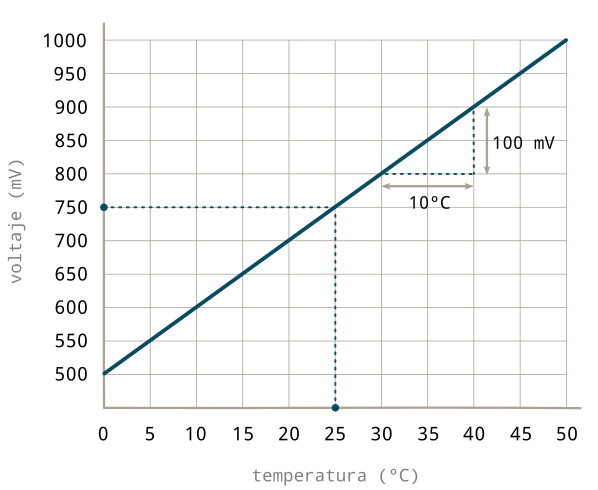
Curva característica del sensor TMP36, extraída de la hoja de datos.

***i***

Dado que la relación entre el voltaje y la temperatura es lineal, la curva característica del sensor es una recta y, como tal, queda determinada por dos atributos: el factor de escala y el *offset*.

El factor de escala es una medida de lo inclinada que está la recta: indica cuánto “se sube” en el eje vertical por cada unidad que “se avanza” en el eje horizontal. De acuerdo con la hoja de datos el factor de escala es 10 mV/ºC y, por tanto, el voltaje aumenta 10 mV por cada grado centígrado que aumenta la temperatura (o 100 mV por cada diez grados, etc.).

El *offset* (o, más apropiadamente, *zero offset*), indica el punto en que la recta corta al eje vertical. Es, por tanto, el valor del voltaje para una temperatura de 0 ºC. Según la hoja de datos, 750 mV corresponden a una temperatura de 25 ºC; por tanto, 0 ºC, que son 25 grados menos, serán –a razón de 10 mV por cada grado– 250 mV menos. Es decir, el *offset* es 500 mV.



Representación de la curva característica entre 0 y 50 grados centígrados.

En resumen, la curva característica nos da el voltaje proporcionado por el sensor en función de la temperatura. Lo que deseamos obtener, sin embargo, es la relación inversa a la mostrada: queremos saber cuánto vale la temperatura para un voltaje suministrado por el sensor. Veremos cómo conseguirlo en el siguiente apartado.

9.14 Cómo calcular la temperatura

En el apartado anterior hemos analizado la curva característica del sensor, que proporciona la relación lineal que existe entre el voltaje y la temperatura. Pues bien, a la vista de la gráfica muy sencillo deducir la expresión para hallar la temperatura *T* (en grados centígrados) que corresponde a cada voltaje medido *V* (en voltios):



La deducción de esta expresión no es complicada; basta, por ejemplo, con analizar varios pares de valores temperatura-voltaje para inferirla (cuidado con las unidades, porque en la gráfica el voltaje está en milivoltios pero en la relación anterior lo hemos puesto en voltios). Sin embargo, dedicaremos unas líneas a continuación a explicar la manera matemática de hallar la relación entre voltaje y temperatura; aunque esta demostración no es necesaria para el desarrollo del capítulo, es interesante saber cómo resolver el problema de una manera sistemática.

Temperatura en función del voltaje

La hoja de datos del sensor nos dice que la relación entre el voltaje y la temperatura es lineal. Afortunadamente, obtener la expresión matemática de una recta es muy sencillo (si la relación entre las variables no fuese lineal el asunto sería más complicado); solo necesitamos conocer dos parámetros: la pendiente y la ordenada en el origen. Vamos a obtener estos valores a partir de los datos suministrados en la hoja de datos del sensor para poder escribir la relación matemática entre el voltaje y la temperatura.

*Pendiente*

La pendiente de la recta es, simplemente, el factor de escala del sensor; en consecuencia, la pendiente es 10 mV/ºC o, en voltios, 0.01 V/ºC.

*Ordenada en el origen*

La ordenada en el origen indica el punto en que la recta corta al eje vertical, es decir, es el valor que hemos llamado *zero offset*. Este valor lo calculamos anteriormente y vale 500 mV o 0.5 V.

*Ecuación de la recta*

Si llamamos *m* a la pendiente (el factor de escala) y *n* a la ordenada en el origen (el *offset*), la recta se puede expresar como *y=mx+n*. Dado que la variable *x* es la temperatura en grados centígrados, la variable *y* es el voltaje en voltios, la pendiente es *m*=0.01 V/ºC y la ordenada en el origen es *n*=0.5 V, la relación entre voltaje y temperatura es:



Esta relación es la ecuación de la curva característica del sensor.

*Temperatura en función del voltaje*

Como lo que lo que buscamos es el valor de la temperatura, despejando *T* en la expresión anterior se obtiene la temperatura (en grados centígrados) en función del voltaje *V* (en voltios):



9.15 Por fin, la temperatura

Ha llegado finalmente el momento de hallar la temperatura medida por el sensor. Lo único que hay que hacer es añadir la relación recién obtenida entre la temperatura y el voltaje a lo visto anteriormente. Proponemos al lector que junte en un programa todas las instrucciones que hemos trabajado en secciones anteriores para que se muestre la temperatura actualizada cada segundo.

Como solución, una posible manera de hacerlo es la que mostramos a continuación.

# ESP32, Arduino

from machine import Pin, ADC

from time import sleep

TMP36 = ADC(Pin(14))

while True:

voltaje = TMP36.read\_uv()/1000000

temperatura = (voltaje - 0.5)\*100

print(round(temperatura, 1))

sleep(1)

El programa anterior es válido para la ESP32 y Arduino. Para la Raspberry Pi Pico se debe cambiar el número del GPIO y sustituir la instrucción de la medida del voltaje por la siguiente:

# Raspberry Pi Pico

voltaje = TMP36.read\_u16()\*3.3/65535

La salida del programa es, por fin, la temperatura en grados centígrados, que se actualiza cada segundo.

25.6

25.2

25.7

25.3

25.8

25.2

25.7

9.16 Suavizado de las medidas

Como se puede observar, los valores de la temperatura oscilan bastante. Esto es debido a que el conversor ADC es muy sensible a las interferencias, lo que da lugar a estas discrepancias. Para minimizar el ruido en la medida se puede añadir un condensador al circuito, aunque también se puede aplicar un método muy simple de suavizado de la medida: calcular la media.

Para proceder al cálculo de la media hemos creado una función, llamada voltaje\_medio(), que se encarga de hacer veinte medidas del voltaje y calcula su media.

# ESP32, Arduino

from machine import Pin, ADC

from time import sleep\_ms

TMP36 = ADC(Pin(14))

def voltaje\_medio():

i = 1

suma = 0

num = 20

while i <= num:

voltaje = TMP36.read\_uv()/1000000

suma += voltaje

i += 1

media = suma/num

return media

while True:

temperatura = (voltaje\_medio() - 0.5)\*100

print(round(temperatura, 1))

sleep\_ms(500)

Si en lugar de la ESP32 estamos utilizando la Raspberry Pi Pico, debemos cambiar el número del GPIO y sustituir la instrucción de lectura del voltaje por la siguiente:

# Raspberry Pi Pico

voltaje = TMP36.read\_u16()\*3.3/65535

Ahora las medidas son mucho más estables.

25.4

25.4

25.5

25.4

25.4

25.5

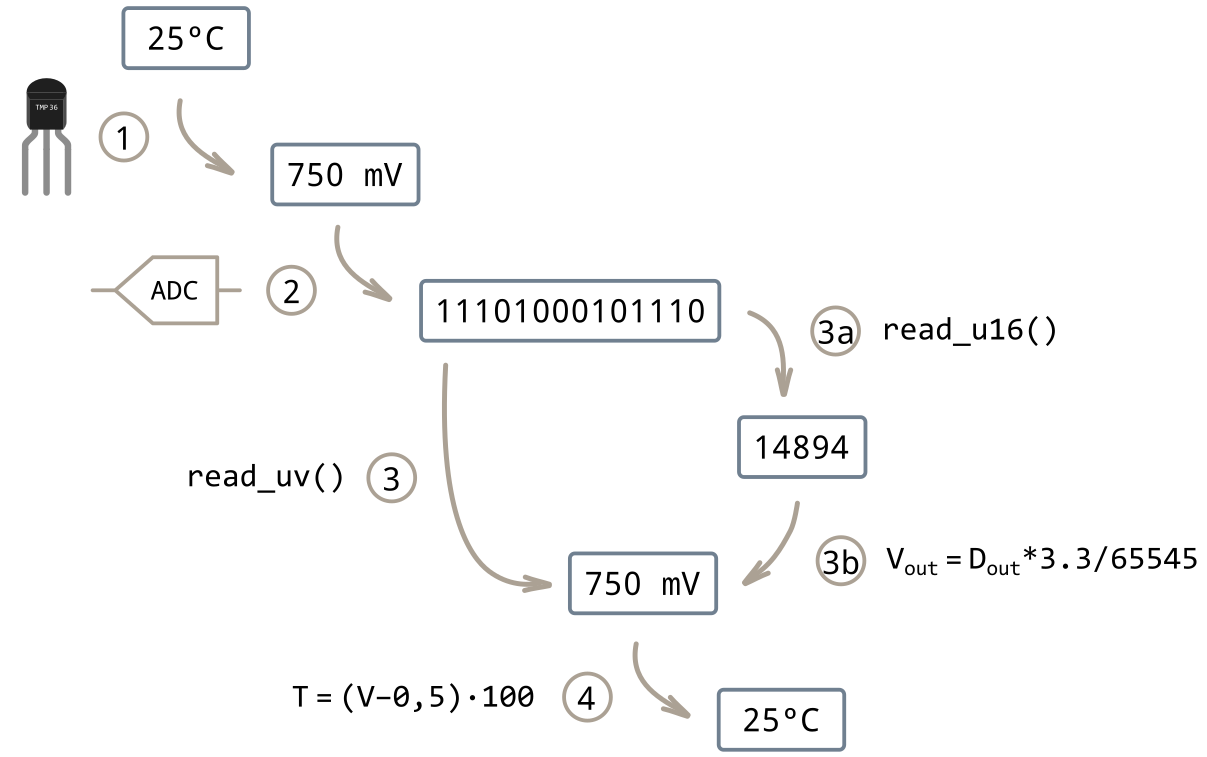
25.4

Existen otros métodos más sofisticados y efectivos para suavizar las medidas, pero por ahora este es suficiente.

9.17 Todo junto

A modo de resumen incluimos un esquema en el que se indican los pasos a seguir cuando se usa un sensor analógico. Si estamos usando la placa ESP32 o Arduino los pasos son los indicados en la secuencia 1-2-3-4; si usamos una Raspberry Pi Pico, los pasos son 1-2-3a-3b-4. Esta estructura será la misma para cualquier sensor analógico que usemos.

|  |  |
| --- | --- |
| **ESP32, Arduino** | **Raspberry Pi Pico** |
| (1) El sensor transforma la temperatura en un voltaje. | |
| (2) El conversor analógico-digital del microcontrolador traduce el valor analógico del voltaje en un código binario. | |
| (3) El valor digital proporcionado por el ADC se puede leer directamente como un voltaje usando el método read\_uv(). | (3a) El método read\_u16() nos da el valor bruto medido por el sensor y (3b) esta lectura se convierte en voltaje usando el voltaje de referencia (3,3 V) y la resolución de la medida (16 bits). |
| (4) El voltaje se transforma en temperatura usando la relación lineal que se extrae de la hoja de datos. | |



9.18 *Bonus track*: un sensor no lineal

A lo largo de este capítulo hemos usado el sensor TMP36 para ilustrar cómo se leen las señales analógicas con un microcontrolador. Los conocimientos que hemos adquirido se pueden aplicar a cualquier otro sensor analógico; lo único que cambia es la manera de traducir el voltaje a la magnitud de interés. Pues bien, existe otro tipo de sensor analógico de temperatura, ampliamente utilizado en aplicaciones reales, que proporciona medidas de gran precisión a un precio muy bajo: el termistor. El motivo por el que no nos hemos decantado por él desde el principio es que, mientras en el TMP36 la relación entre el voltaje y la temperatura es muy sencilla de obtener, en el termistor esta relación presenta una complicación matemática extra que hemos preferido evitar. Llegados a este punto, sin embargo, queremos cerrar el capítulo mostrando cómo proceder en un caso de que la relación entre el voltaje y la magnitud que queremos medir no sea lineal.

El termistor

Un termistor es simplemente una resistencia cuyo valor cambia con la temperatura. Usaremos un termistor NTC (del inglés *negative temperature coefficient*); en estos dispositivos, al aumentar la temperatura la resistencia disminuye. En concreto trabajaremos con el módulo KY-013, que es un sensor analógico de temperatura que consta de un termistor NTC y trae una resistencia externa integrada.

La primera complicación con que nos encontramos es que en los termistores la relación entre la resistencia y la temperatura no es lineal.

Relación entre la resistencia y la temperatura en un termistor de coeficiente negativo (NTC). Al aumentar la temperatura la resistencia disminuye. Además, la relación entre estas dos magnitudes no es lineal.

Por otro lado, para hallar la temperatura es necesario medir la resistencia del termistor. El problema es que los microcontroladores no pueden medir resistencias, solo voltajes. Por tanto, para medir la resistencia necesitamos un divisor de tensión.

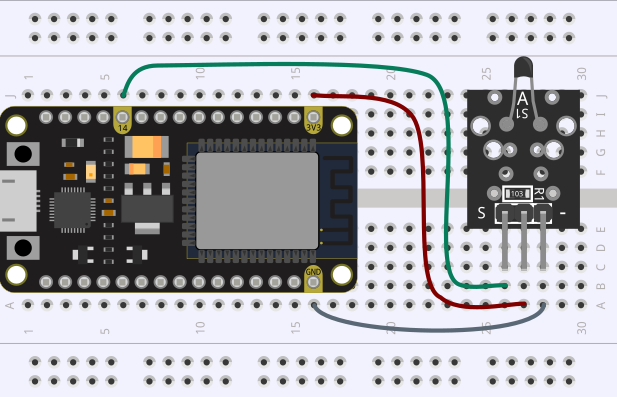
El divisor de tensión

Un divisor de tensión, o divisor de voltaje, consiste simplemente en dos resistencias asociadas en serie. Una de estas resistencias es fija y conocida; la otra es la que queremos medir. Para facilitarnos el trabajo, el módulo KY-013 trae una resistencia externa integrada formando el divisor de tensión que necesitamos (para entender el funcionamiento de un divisor de voltaje se pueden consultar numerosas referencias *online*).

El circuito

El módulo tiene tres pines:

* El pin de señal, S, se conecta a un pin del microcontrolador que tenga funcionalidad ADC (por ejemplo, el GPIO14 en el ESP32 o el Arduino Nano ESP32, o el GPIO28 en la Raspberry Pi Pico).
* El pin central se conecta a la fuente positiva de alimentación (el pin 3.3V o 3V3; en la Raspberry Pi Pico es el pin 3V3 Out).
* El pin marcado con un signo menos se conecta a tierra (GND).

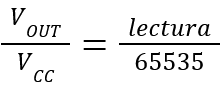


Medida de la temperatura

Los pasos que hay que seguir para medir la temperatura con el termistor son los siguientes: 1) Se mide la señal ADC en el pin S del módulo. 2) A partir de la lectura del ADC se calcula la resistencia del termistor. 3) La resistencia se transforma en temperatura.

1. Lectura de la señal

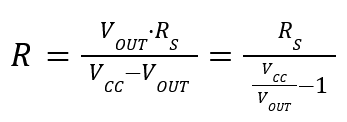
Para leer el valor medido por el ADC usaremos la función read\_u16(), que devuelve el valor de la señal con una resolución de 16 bits. Como vimos anteriormente, este valor es proporcional al voltaje. La relación entre el voltaje y la lectura del ADC viene dada por la expresión:



donde *lectura* es el valor bruto devuelto por read\_u16(), *VOUT* es el voltaje medido y *VCC* es el voltaje de alimentación del sensor.

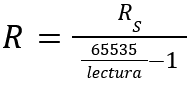
2. Cálculo de la resistencia

Teniendo en cuenta el funcionamiento de un divisor de tensión, la resistencia se puede calcular a partir del voltaje medido, *VOUT*, usando la siguiente expresión:



donde *R* es la resistencia del termistor (es la resistencia que queremos medir para obtener la temperatura) y *Rs* es la resistencia en serie del divisor de tensión, que en el módulo KY-013 tiene un valor de 10 kΩ.

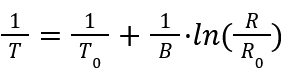
Dada la relación lineal entre el valor bruto y el voltaje, la expresión anterior es equivalente a:



Es decir, podemos calcular la resistencia del termistor a partir del valor bruto medido por el ADC.

3. Cálculo de la temperatura

Dado que el comportamiento del termistor no es lineal, la relación entre la resistencia y la temperatura es compleja. Se puede utilizar una aproximación a la llamada ecuación de Steinhart-Hart que establece que, para una resistencia *R*, la temperatura *T* (en kelvins) se puede calcular a partir de:



donde *R0* y *T0* indican la resistencia del termistor a temperatura ambiente y *B* es el valor beta del termistor. En el módulo KY-013 *R0* es 10 kΩ cuando *T0* es 25 ºC (o, lo que es lo mismo, 295.15 K) y *B* es 3950 K.

De la expresión anterior se obtiene la temperatura, en kelvins. Para calcularla en grados centígrados hay que restarle 273.15 K a ese valor.

El programa

Para finalizar, el siguiente programa recoge los pasos anteriores y devuelve la temperatura medida por el termistor actualizada cada segundo (cambiar el número del pin si el sensor no está conectado al GPIO14).

from machine import Pin, ADC

from time import sleep

from math import log

# Termistor conectado al GPIO14

SENSOR = ADC(Pin(14))

# Características del termistor

T0 = 298.15

R0 = 10000

B = 3950

# Resistencia del divisor de voltaje

Rs = 10000

while True:

# Lectura del ADC (valor bruto)

lectura = SENSOR.read\_u16()

# Resistencia

R = Rs/(65535/lectura - 1)

# Temperatura (en kelvins)

T = 1/(1/T0 + 1/B\*log(R/R0))

# Temperatura (en grados centígrados)

T = T - 273.15

print(round(T,1))

sleep(1)

En resumen

En el presente capítulo hemos aprendido a leer señales analógicas usando un sensor. Tras analizar qué son los sensores y en qué se basa su funcionamiento hemos examinado la hoja de datos del sensor de temperatura TMP36 para conocer sus características y conexiones. Hemos mostrado cómo funciona la conversión analógico-digital y hemos utilizado la clase ADC del módulo machine para leer la señal devuelta por el sensor, y la hemos convertido en temperatura. Aunque hemos basado el capítulo en un sensor de temperatura –la medida de esta magnitud es esencial para detectar el cambio climático y adaptar nuestras acciones para mitigar sus efectos–, los conocimientos adquiridos son aplicables a cualquier sensor analógico.