2. Onboard temperature sensor reading

I. Learning objectives

1. Learn the basic use of the ADC pins of the Raspberry Pi Pico 2 .
2. Learn how to read the temperature through the onboard temperature sensor.

II. Hardware construction

This course directly utilizes the temperature sensor on the Raspberry Pi Pico 2 motherboard without additional hardware equipment.

The Raspberry Pi Pico 2's RP2350 microcontroller is a digital device, just like all mainstream microcontrollers: it is composed of thousands of transistors, these tiny switch-like devices are either on or off. Therefore, your Pico 2 can't really understand an analog signal - it can be anything on the spectrum between fully off and fully on - without relying on additional hardware: an analog-to-digital converter (ADC).

An ADC has two key characteristics: its resolution, measured in digital bits, and its channels, or how many analog signals it can accept and convert at once. The ADC in your Pico 2 has a resolution of 12 bits, which means it can convert analog signals into digital signals with numbers ranging from 0 to 4095 - although this is processed in MicroPython and converted to 16-bit numbers ranging from 0 to 65535, so it behaves the same as the ADC on other MicroPython microcontrollers.

The RP2350 has a total of five ADC channels, four of which are brought out to the chip GPIO: GP26, GP27, GP28 and GP29. On the Pico 2, the first three of these are brought out to GPIO pins, and the fourth can be used to measure the VSYS voltage on the board.

3. Program Analysis

Thonny Programming

For the use of ThonnyIDE, please check the previous environment construction related courses.

```
import machine
import utime
sensor_temp = machine.ADC(29)
conversion_factor = 3.3 / (4096-1)
while True:
    reading = sensor_temp.read_u16()* conversion_factor
    temperature = reading#27 - (reading - 0.706)/0.001721
    print(temperature)
    utime.sleep(2)
```

import machine

The machine library contains all the instructions MicroPython needs to communicate with Pico and other MicroPython-compatible devices, extending the language of physical computing.

import utime

The "utime" library. This library handles everything to do with time, from measuring it to inserting delays into your program. The units are in seconds.

sensor_temp = machine.ADC(29)

Initialize an ADC (Analog to Digital Converter) object that will be used to read an analog signal connected to pin 29 of the microcontroller.

conversion_factor = 3.3 / (4096-1)

Calculate the conversion factor used to convert the ADC reading (range 0-4095) to a voltage value. The Raspberry Pi Pico is powered by 3.3V, so the conversion factor is 3.3V divided by the maximum value of the ADC minus 1 (i.e. 4095).

reading = sensor_temp.read_u16() * conversion_factor

Read the 16-bit unsigned integer from the ADC (read_u16()) and multiply the reading by the conversion factor defined above to get the voltage value.

temperature = reading

The voltage value is assigned to the temperature variable.

print(temperature)

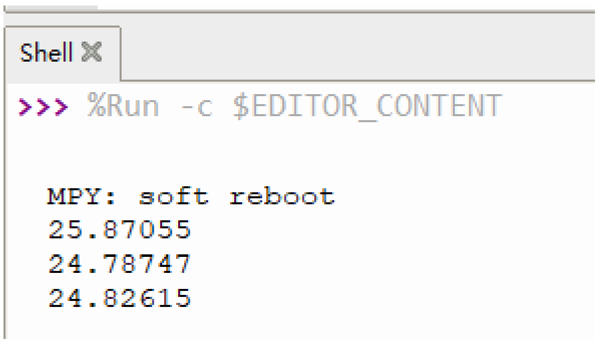
Print the value.

utime.sleep(2)

This calls the sleep function from the utime library and enters the loop again after 2s.

4. Experimental phenomenon

After the program is downloaded, we can see that the shell window prints the temperature value every 2 seconds.



```
>>> %Run -c $EDITOR_CONTENT

  MPY: soft reboot
  25.87055
  24.78747
  24.82615
```