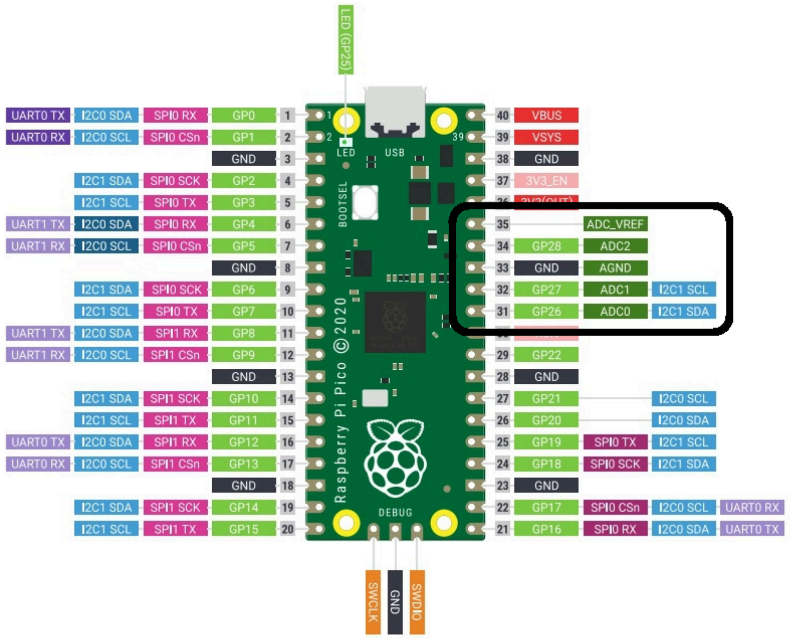
The ADC on RP2040 has a resolution of 12 bits, meaning that it can transform an analog signal into a number ranging from 0 to 4095 – though this is handled in MicroPython transformed to a 16-bit number ranging from 0 to 65535. RP2040 has five ADC channels total, four of which are brought out to chip GPIOs: GP26, GP27, GP28, and GP29. On Raspberry Pi Pico, 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. The ADC's fifth input channel is connected to a temperature sensor built into RP2040.



It is possible to read any ADC channel either by using the PIN or by channel:

```
adc = machine.ADC(26)     # pin number
adc = machine.ADC(0)      # channel
```

Task 1. Create a MicroPython script to read the internal temperature sensor and send this value to a computer.

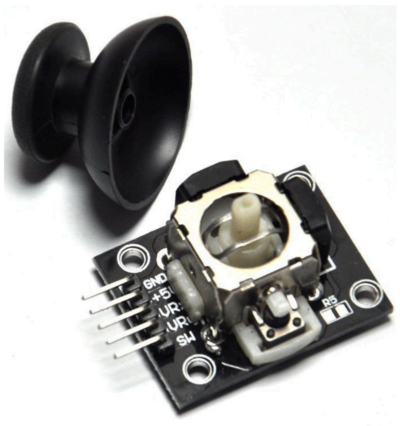
```
from machine import ADC
from utime import sleep
```

```
#use channel instead of a number of a pin
temp_sensor = ADC(4)

while True:
    result = temp_sensor.read_u16()
     temperature = 27 - (result*3.3/65535 -
                         0.706)/0.001721
    print("Temperature: {:.2f}°C".format(temperature))
    sleep(2)
```

Typically, 0.706V means 27 degrees Celsius, with a slope of -1.721mV (0.001721) per degree.

Task 2. Connect a joystick to the Raspberry Pi Pico board.



Pico board	Joystick board
Gnd	Gnd
+3V3	+5V
GP26	VRx
GP27	VRy
GP15	SW

Create a MicroPython script to send values proportional to the stick position to the computer.

```
from machine import ADC
from utime import sleep
```

```
joy_X = ADC(26)
joy_Y = ADC(27)

while True:
    t_string = "X position = " + str(joy_X.read_u16())
    print(t_string)
    t_string = "Y position = " + str(joy_Y.read_u16())
    print(t_string)
    sleep(0.5)
```

Task 3. Create a MicroPython script to send to the computer, values from 0 to 10, proportional to the current stick position.

```
from machine import ADC
from utime import sleep

def analog_map(x, in_min, in_max, out_min, out_max):
     return int((x-in_min)*(out_max-out_min)/
               (in_max-in_min)+ out_min)

joy_X = ADC(26)

while True:
     result = analog_map(joy_X.read_u16(),0,65535,0,10)
     t_string = "X position = " + str(result)
     print(t_string)
     sleep(1)
```

Task 4. Connect *Pico-LCD-0.96* module.

Open the *Demo code* from the *Resources* section on the website:

www.waveshare.com/wiki/Pico-LCD-0.96

Analyze the code.

Task 5. (*own work - 0.5 of a point*) Create a MicroPython script that displays ambient temperature in Celsius degrees, Kelvins, and Fahrenheit degrees values using *Pico-LCD-0.96* module.

Task 6. (*own work - 1 of a point*) Use a joystick to control the position of a bargraph LED segment. Use the joystick switch button to switch on and off the bargarph.

Task 7. (*own work - 1 of a point*) Use a joystick to control the position of the displayed character on the OLED display.

Task 8. (*own work - 1.5 of a point*) Display temperature vs time graph on the Node-RED-based interface.

To pass Exercise 1, you have to solve either Task 5, Task 6, Task 7, or Task 8. Presenting and sending the solution afterward is mandatory.

For those interested:

1. MicroPython web page:

   micropython.org/download/rp2-pico/

2. RandomNerd tutorial on analog inputs:

   randomnerdtutorials.com/raspberry-pi-pico-analog-inputs-micropython/