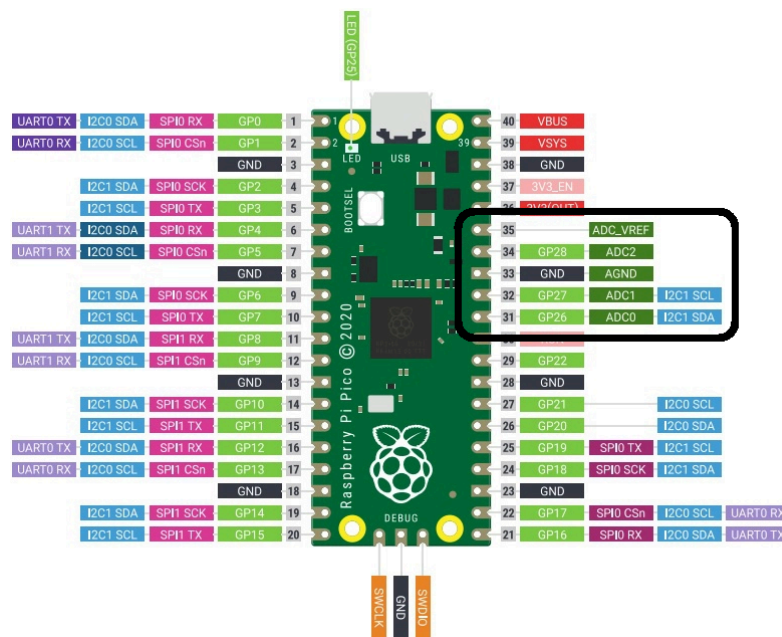


Exercise no 2: Analog inputs

Introduction. 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 number 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
```

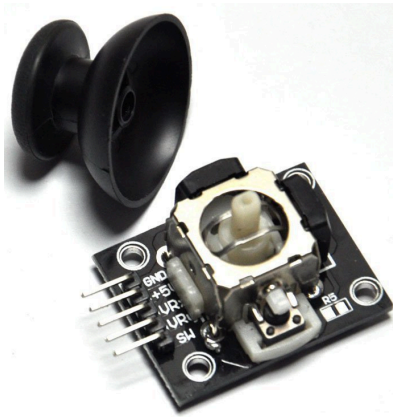
Exercise no 2: Analog inputs

```
temp_sensor = ADC(4) # channel

while True:
    result = temp_sensor.read_u16()
    temperature = 27 - (result*3.3/65535 - 0.706)/0.001721
    print(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 to the computer, values proportional to the current stick position.

```
from machine import ADC
from utime import sleep

joy_X = ADC(26)
```

Exercise no 2: Analog inputs

```
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 postion = " + str(result)
    print(t_string)
    sleep(1)
```

Task 4. Connect the SSD1306 OLED display to the Raspberry Pi Pico board.

| Pico board | OLED display |
|------------|--------------|
| Gnd | Gnd |
| +3V3 | Vcc |

Exercise no 2: Analog inputs

| | |
|------|-----|
| GP16 | SDA |
| GP17 | SCL |

```
from machine import Pin,I2C
from ssd1306 import SSD1306_I2C
from utime import sleep

WIDTH = 128
HEIGHT = 64

i2c = I2C(0, scl = Pin(17), sda = Pin(16), freq = 200000)
print("I2C adr :" + hex(i2c.scan()[0]).upper())
print("I2C config :" + str(i2c))
oled = SSD1306_I2C(WIDTH,HEIGHT,i2c)

oled.fill(0)
oled.text("RPI Pico",5,10)
oled.show()
```

Task.5. Create a MicroPython script that displays ambient temperature value using an OLED display.

Task.6. Use a joystick to control the position of a bargraph LED segment. Use the joystick switch button to switch on and off this segment.

Task 7. Use a joystick to control the position of the displayed character on the OLED display.

Task.8. Display temperature vs time graph.

For those interested:

1. MicroPython web page:

micropython.org/download/rp2-pico/

2. SSD1306 display programming tutorial:

microcontrollerslab.com/oled-display-raspberry-pi-pico-micropython-tutorial/