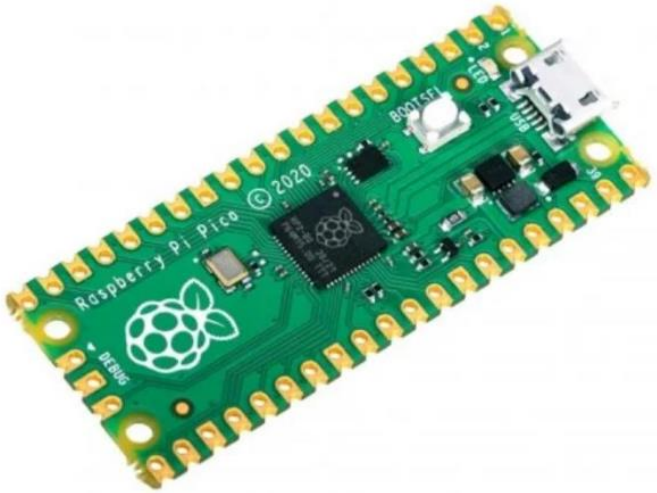
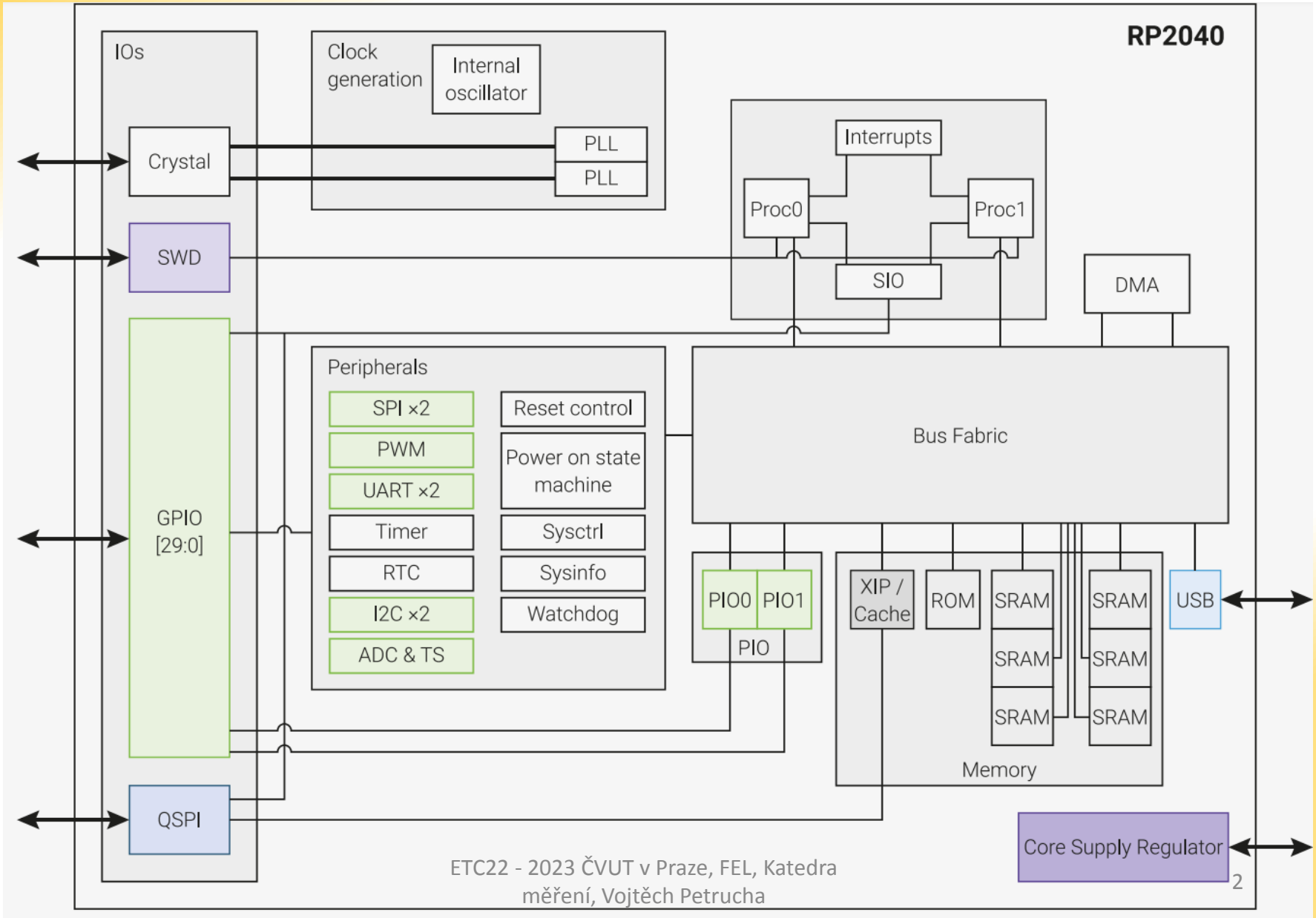
Raspberry Pi Pico

Programování v MicroPythonu



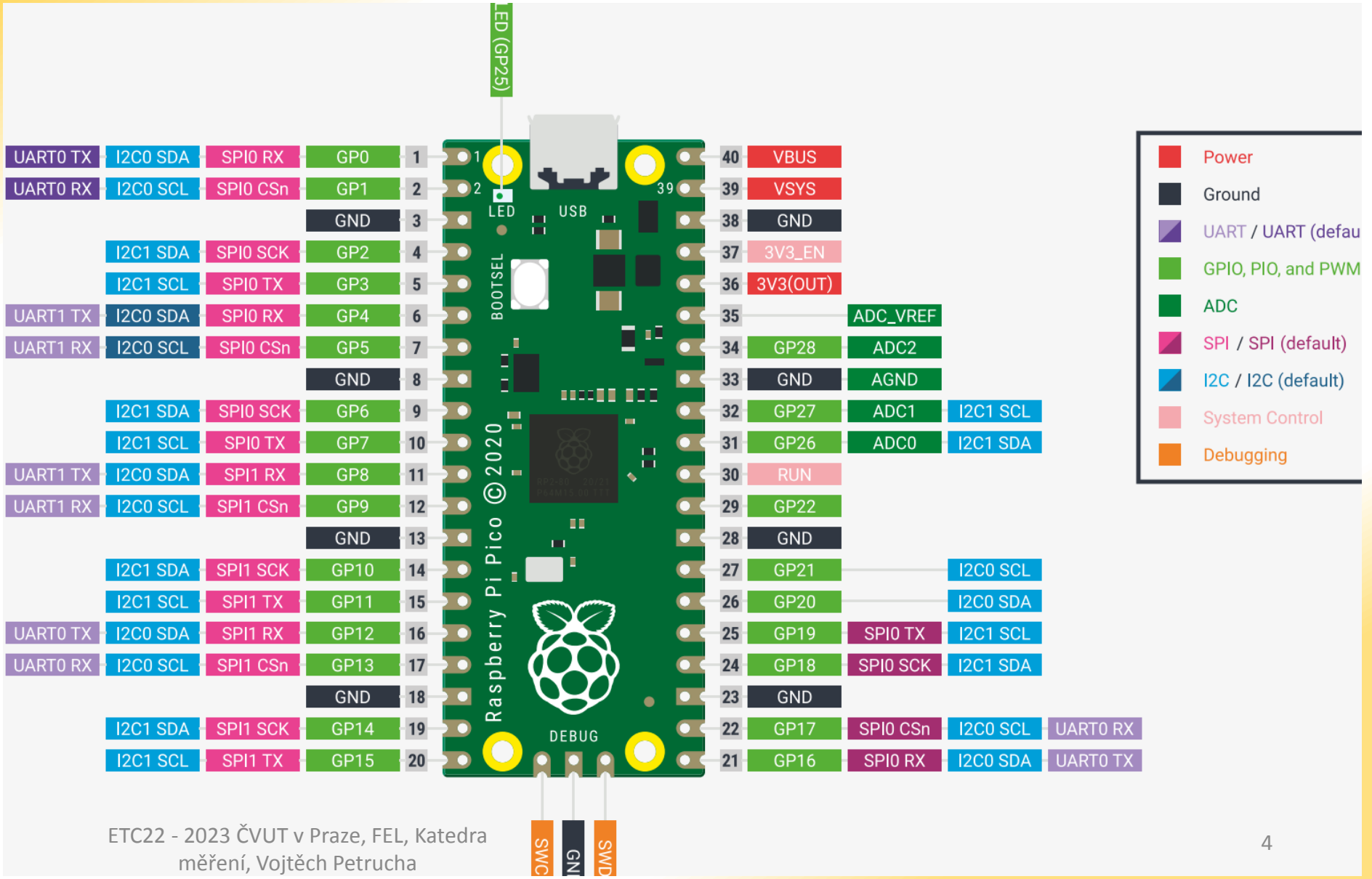
RP2040 – srdce Rasp. Pi PICO



Pro nás je důležité

- **GPIO** – můžeme nastavit jako výstup a například ovládat LED, nebo jako vstup a můžeme číst stav tlačítka
- **ADC** – můžeme měřit napětí, pouze na několika pinech..
- **PWM** – můžeme snadno generovat periodický obdélníkový signál, pípat s PIEZO bzučákem... nebo rozsvěcet LED různým jasem, dle střídy PWM
- **USB** – tím to bude připojené do PC a můžeme si vypisovat zprávy z programu, naměřené hodnoty
- **QSPI** – rozhraní pro externí paměť FLASH

Piny na Raspberry Pi PICO



Jak lze PICO programovat?

- **C** (např. pomocí Visual Studio Code)

https://github.com/raspberrypi/pico-setup-windows/releases/latest/download/pico-setup-windows-x64-standalone.exe

s využitím „SDK“ knihoven a příkladů

- **MicroPython** <- to použijeme

je třeba nejdřív do modulu nahrát firmware (.uf2)

https://micropython.org/download/RPI_PICO/

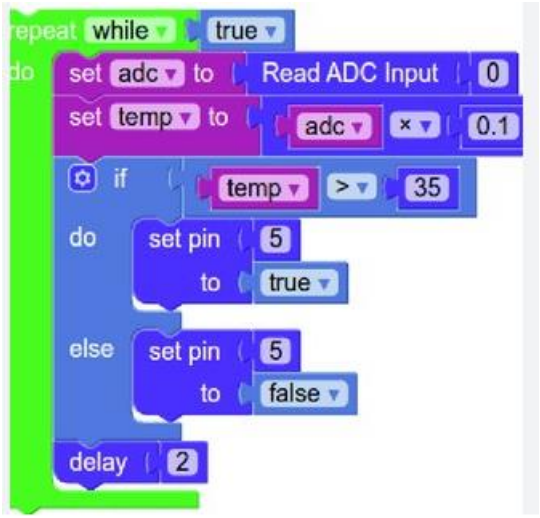
potom programovat pomocí software Thonny

https://thonny.org/

Jak lze PICO programovat?

- **MicroPython**

ale s grafickým rozhraním https://bipes.net.br/ide/



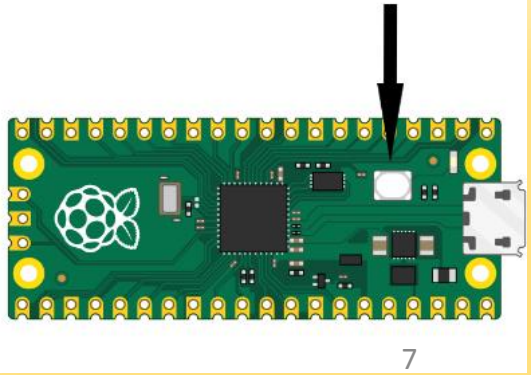
viz návod zde:

https://embedded.fel.cvut.cz/sites/default/files/stredni_skoly/Detska_univerzita/Detska_Universita_2022_Bipes_Raspberry_PI_Pico_2022_7_15_1.pdf

Jak nahrát soubor *.uf2

- před připojením modulu do USB zmáčknout bílé tlačítko na modulu, držet ho a připojit k PC, poté tlačítko pustit (aktivuje se bootloader, ten se aktivuje i pokud se tlačítko nestiskne a není v něm žádný program...)
- modul se objeví jako externí disk (flashka) kam lze normálně přetažením nebo Ctrl+C, Ctrl+V vložit daný soubor **RPI_PICO-20231005-v1.21.0.uf2**

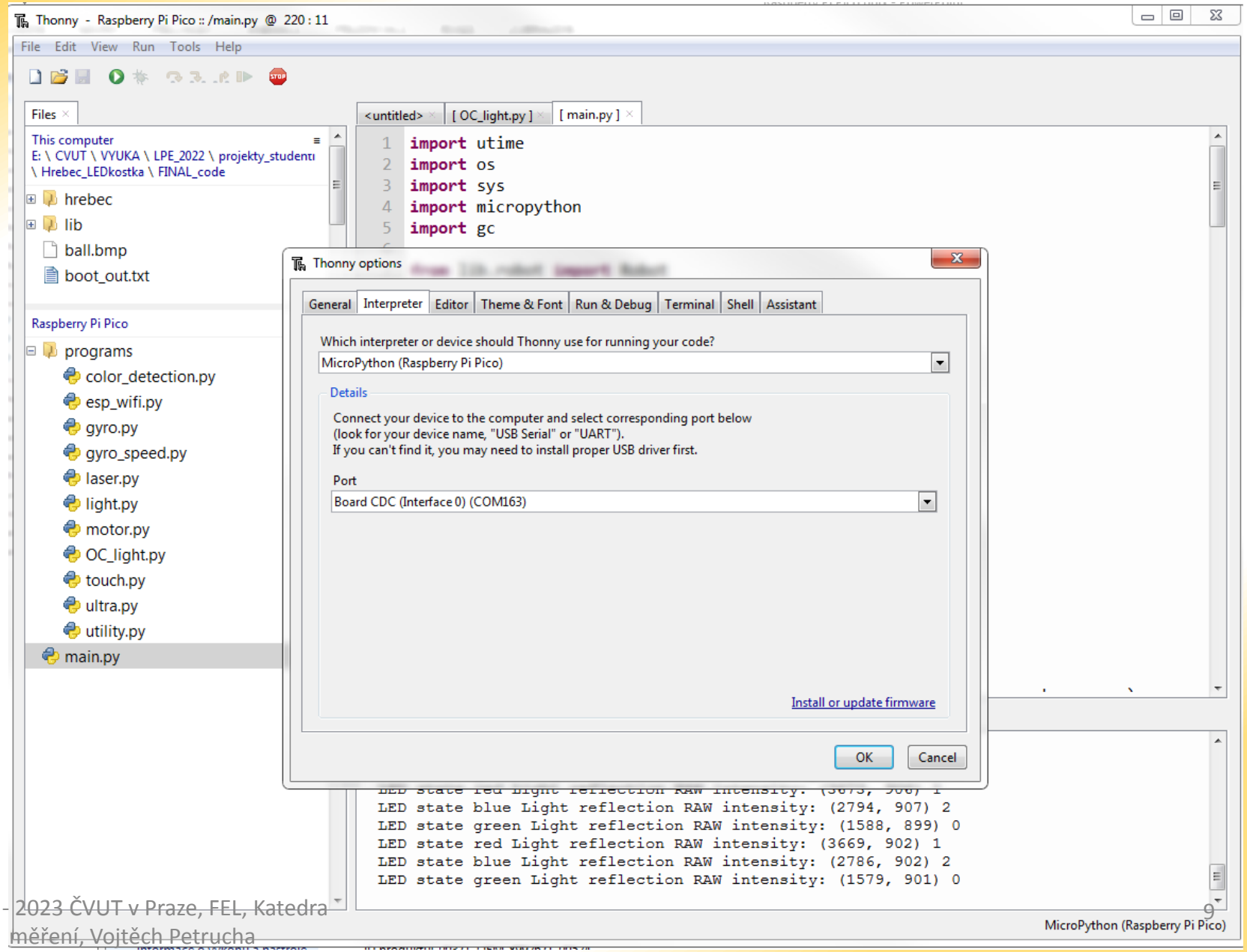
https://micropython.org/download/RPI_PICO/



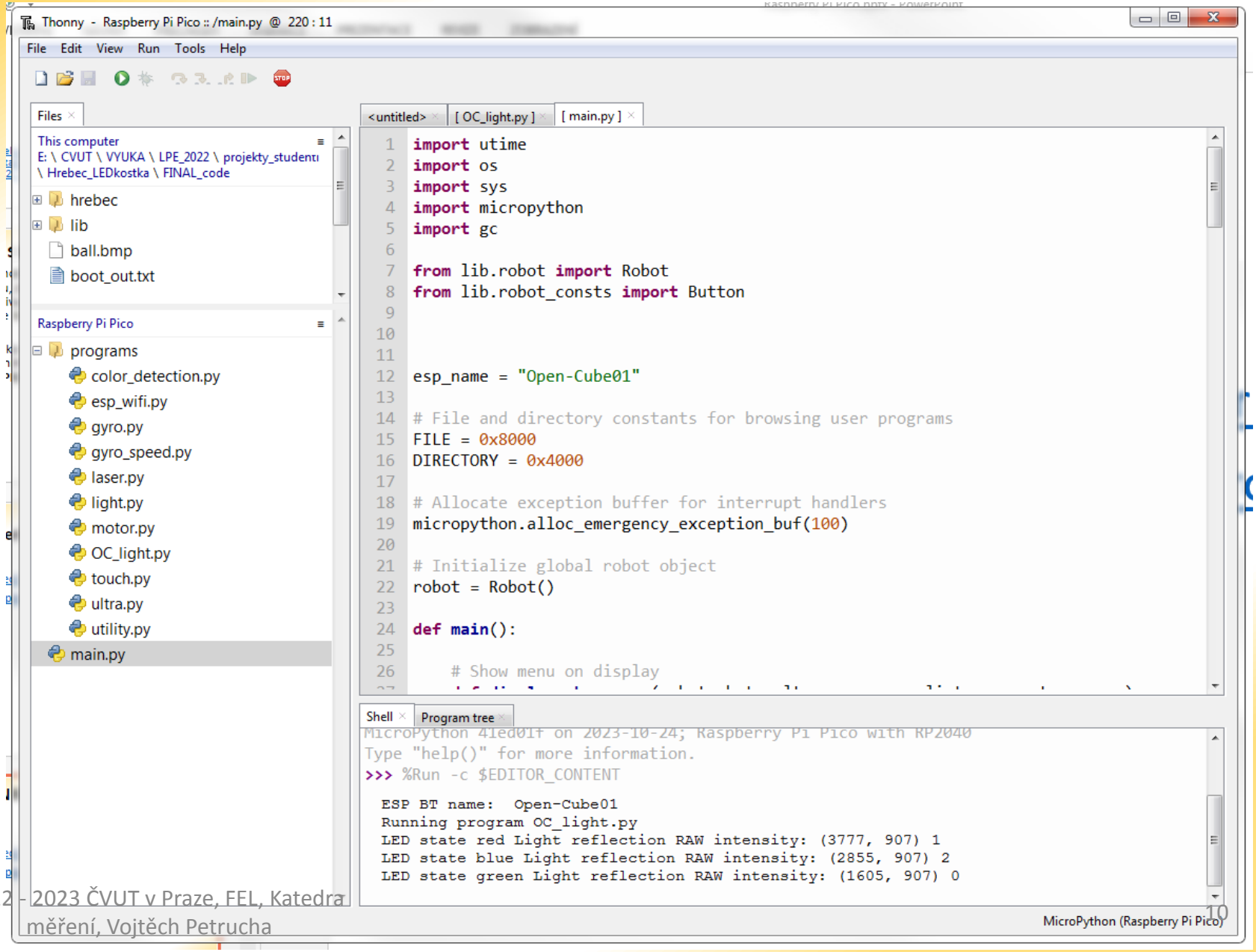
Kde najít materiály k programování

- https://embedded.fel.cvut.cz/kurzy/etc22/RP_PICO
- https://www.raspberrypi.com/products/raspberry-pi-pico/
- https://thonny.org/

Prostředí THONNY



Prostředí THONNY



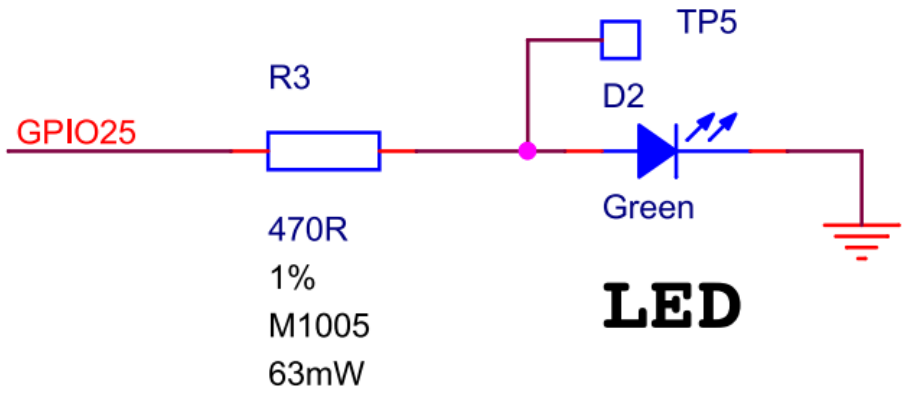
Prodlevy v programu - čekání

```
from utime import sleep, sleep_ms
sleep(1)
sleep_ms(100)
```

GPIO – výstup (LED)

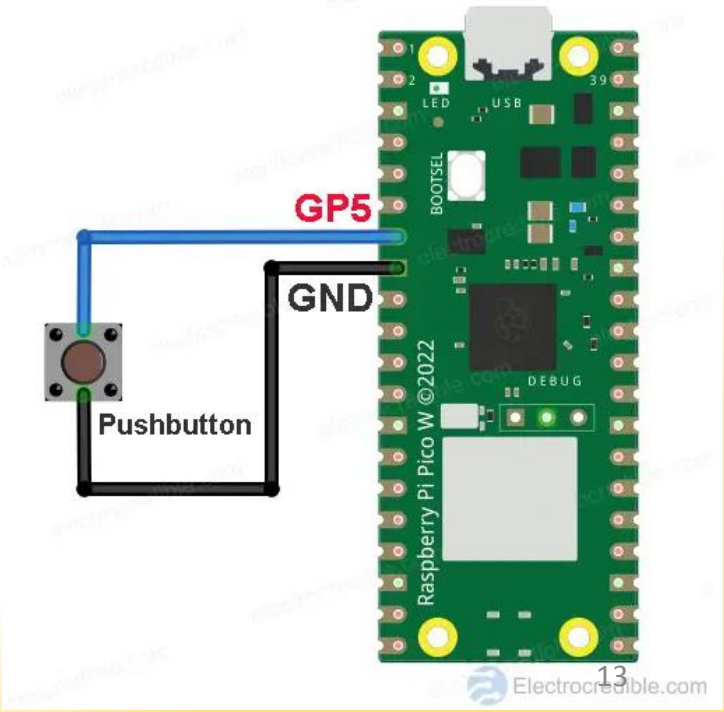
```
from machine import Pin
inbuiltLed = 25   #číslo GPIO pinu
led = Pin(inbuiltLed, Pin.OUT)

led.toggle()     #změní stav pinu
led.value(1)    # 3V3
led.value(0)    # 0V
```



GPIO – vstup (tlačítko)

```
from machine import Pin
pin5 = Pin(5, Pin.IN, Pin.PULL_UP)
#(zapneme „rezistor“ mezi daným vstupem a 3V3 )
print(pin5.value())      #přečte stav pinu 5
```



PWM výstup

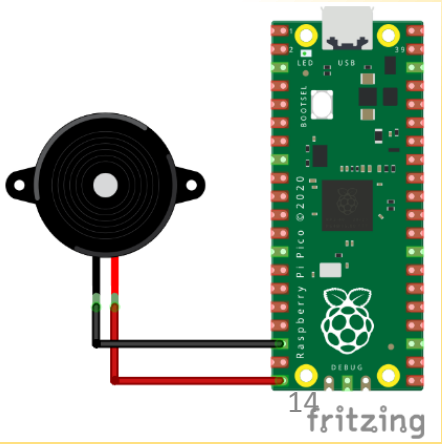
```
from machine import Pin, Timer, PWM
buzzer = PWM(Pin(15))   # kde budeme PWM generovat
buzzer.freq(500)              # frekvence 500 Hz
buzzer.duty_u16(32768)  # střída PWM 0...65535, 50%
buzzer.duty_u16(0)          # 0 – výstup bude na 0V
```

https://tarikgit.github.io/coding/using-raspberry-pico

(Playing a tune on your Pico)

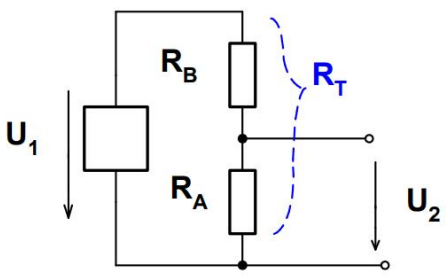
PIEZO připojit přes rezistor 470 ohm

nebo více

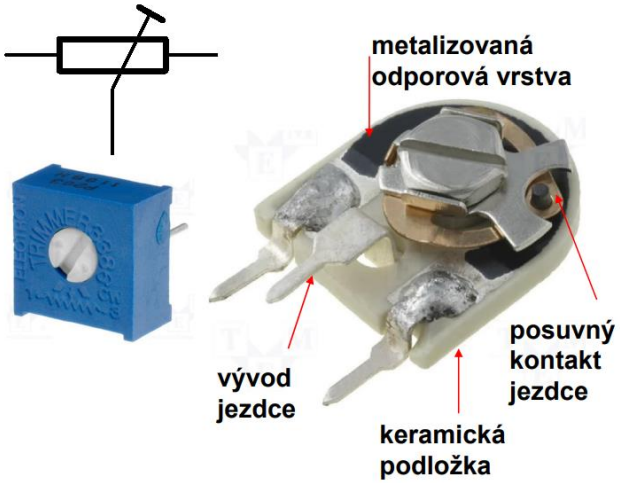


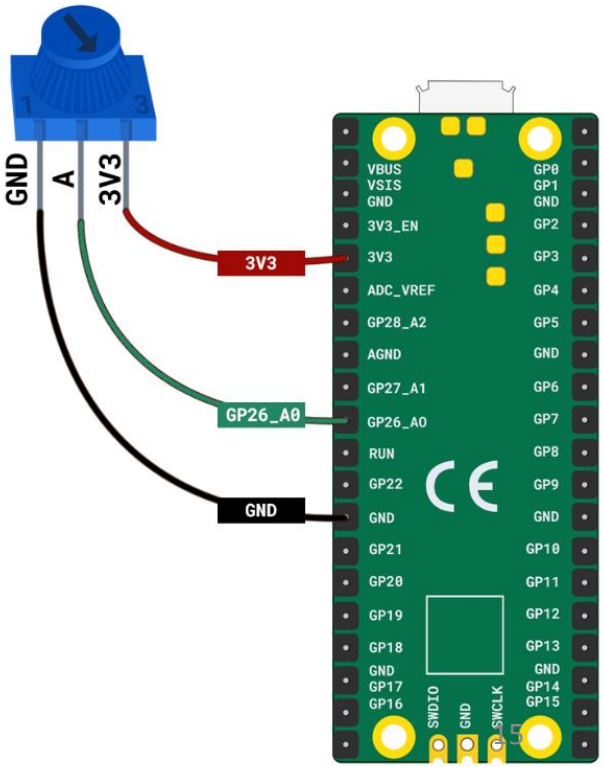
ADC – měření napětí

```
from machine import Pin, ADC
voltage = ADC(26)
voltage_value = voltage.read_u16()   #přečti hodnotu
```



$$U_2 = \frac{R_A}{R_T} \cdot U_1 = k_t \cdot U_1$$





Řízení programu - cyklus

```
from machine import Pin
from utime import sleep, sleep_ms
inbuiltLed = 25   #číslo GPIO pinu
while True:
        led.value(1)          # 3V3 LED svítí (LED přes R..)
        sleep_ms(200)         # čekání 200 ms
        led.value(0)          # 0V
        sleep(1)              # čekání 1 s
```

„nekonečná smyčka – hlavní část programu která se neustále vykonává“ **pozor na odsazení příkazů....**

Řízení programu – daný počet cyklů

```
x = 0
for y in range(0, 9):
    x += 1
print(x)
```

```
for x in range(2, 6):
  print(x)
```

```
x = 0
while x < 9:
    x += 1
print(x)
```

(vyzkoušejte si co jednotlivé programy dělají...)

Řízení programu - podmínka

```
from machine import Pin
from utime import sleep, sleep_ms
pin5 = Pin(5, Pin.IN, Pin.PULL_UP)
inbuiltLed = 25   #číslo GPIO pinu
led = Pin(inbuiltLed, Pin.OUT)

while True:
        if pin5.value():        #pokud není tlačítko stisknuté
                led_pin.value(1)        #tak LED svítí
        else:                    # pokud je stisknuté, zhasni...
                led_pin.value(0)
        sleep_ms(10)            # čekání 10 ms
```

Tisk do konzole (na sériový port – do programu Thonny)

```
print("Hello ETC22")
```

Co si máte vyzkoušet

1) blikat s LED (i s různou svítivostí LED - PWM)

2) blikat s LED jen když je stisknuté tlačítko

3) zahrát nějakou melodii

4) svítit s LED tak moc, jak velké napětí je na odporovém trimru

5) měřit napětí na odporovém trimru a vypisovat ho každých 200 ms do konzole, pokud je větší než 2V tak rozsvítit LED

6) zobrazit naměřené napětí pomocí programu DataPlotter

https://github.com/jirimaier/DataPlotter/blob/master/documentation/Data%20protocol%20guide%20cz.pdf

Domácí úkol

vytvořte program, který na stisknutí tlačítka bude 5 sekund plynule rozsvěcet a zhasínat LED a přitom bzučet pomocí PIEZO měniče, jehož tón se bude měnit, tak jak se bude měnit jas LED

pokud byste měli hodně času, vytvořte program na měření rychlosti reakce na stisk tlačítka, s výpisem času na terminál

(PICO pípne a rozsvítí LED, vy musíte co nejrychleji stisknout tlačítko, po stisku se vypíše na sériový port, kolik ms to trvalo... od začátku signálu po stisk..)

Timer – časovač – periodické vykonávaní nějaké akce

```
timer = Timer()                 #potřebujeme třídu časovač
def ledblink(timer):            #definice funkce
    led.toggle()
timer.init(freq=2.5, mode=Timer.PERIODIC,
callback=ledblink)


#použije třídu Timer, aby se 2.5x za sekundu zavolala
#funkce ledblink, která blikne s LED
#stačí nastavit na začátku programu a již běží samo...
```

Přerušení – rychlá reakce na nějaký podnět

```
from machine import Pin
pin_button = Pin(14, mode=Pin.IN, pull=Pin.PULL_UP)
pin_led = Pin(16, mode=Pin.OUT)
def intr_handler(pin):
        pin_led.value(not pin_led.value())       #nějaká akce

pin_button.irq(trigger=Pin.IRQ_FALLING,handler=intr_handler)

while True:                          # hlavní smyčka programu
   ...                               # něco dělej, třeba čekej...
```

Tisk desetinných čísel

```
x , y = 64 , 128.4096
y_string = "y„
# 1. varianta - oddělení čárkami, Tato varianta je vhodná na jednoduché a rychlé zobrazení textu, Jednotlivé prvky jsou odděleny mezerou
print ("x =", x , ",", y_string , "=", y )  # x = 64 , y = 128.4096
# 2. varianta - použití f-stringu
# Tato varianta umožňuje formátovat float čísla d označuje typ integer, s typ string a f typ float určuje počet zobrazených desetinných míst
print(f"x = {x:d} , { y_string :s} = {y:.2f}")    # x = 64 , y = 128.41
print("$$P",time_stamp,",",sum_voltage,",",trimV,";",sep='',end='')   #DataPlotter format
```

Dataplotter – měření napětí a další

```
from machine import Pin, Timer, PWM, ADC
from utime import sleep, sleep_ms

inbuiltLed = 25
led = Pin(inbuiltLed, Pin.OUT)
button = Pin(16,Pin.IN,Pin.PULL_UP)
ledT = Pin(4, Pin.OUT)
voltage = ADC(28)         #creating potentiometer object
trimvoltage = ADC(27)
timerr = Timer()
voltages = [0.0, 0.0, 0.0,0.0]
array_position = 0
time_stamp = 0
voltage_threshold = 2.1
alarm_threshold = 3.1
buzzer = PWM(Pin(15))
buzzer.freq(500)

def meas(timerr):
    sum_voltage = 0
    global time_stamp
    global array_position
    global voltages
    trimV=(3.3 * trimvoltage.read_u16())/65536
    voltages[array_position] = (3.3 * voltage.read_u16())/65536
    array_position +=1
    if array_position == 4:
        array_position =0
        a = 0
        sum_voltage = 0
        while a < 4:
            sum_voltage = voltages[a]
            a += 1
        sum_voltage /=4
        time_stamp += 0.04
        print("$$P",time_stamp,",",sum_voltage,",",trimV,";",sep='',end='')   #DataPlotter format
        led.toggle()
        if sum_voltage > trimV:
            ledT.value(1)
        else:
            ledT.value(0)
        if sum_voltage > trimV:
            buzzer.duty_u16(32768)
        else:
            buzzer.duty_u16(0)

timerr.init(freq=100, mode=Timer.PERIODIC, callback=meas)

while True:
    if button.value() == 0:
        sys.exit()
    sleep_ms(10)
```