

# Lekce 01

## Harmonogram

- Úvod - o robotovi
- Rychlý kvíz na Python
- Praktická cvičení
- Volný souboj robotů

## Rychlý kvíz na Python

Až budete mít chvílku, tak prosím vyplňte krátký kvíz:

[Odkaz na kvíz \(https://forms.gle/V96ChH11spFH2go49\)](https://forms.gle/V96ChH11spFH2go49)

## O robotovi

### Názvosloví

- API: aplikační rozhraní
- Gimbal: hlava robota - otáčecí hlaveň s kamerou
- Chassis: tělo robota - může se otáčet nezávisle
- LED: osvětlení. Robot má osvětlenou hlavu, tělo a diodu pod hlavní
- Marker: piktogram s čísly/písmeny/znaky
- Python: proměnná, funkce, cyklus, podmínka

### Zdroje

- [Online příručka, Scratch a Python \(https://www.dji.com/cz/robomaster-s1/programming-guide\)](https://www.dji.com/cz/robomaster-s1/programming-guide)
- [Oficiální příklady v Pythonu \(https://github.com/ROBOMASTER-S1/ROBOMASTER-S1-Python-Examples\)](https://github.com/ROBOMASTER-S1/ROBOMASTER-S1-Python-Examples)
- [Seznam příkazů v Pythonu \(https://github.com/ROBOMASTER-S1/ROBOMASTER-S1-Python-Examples/blob/master/Robomaster%20S1%20Python%20Commands.py\)](https://github.com/ROBOMASTER-S1/ROBOMASTER-S1-Python-Examples/blob/master/Robomaster%20S1%20Python%20Commands.py)
- [Manuál v češtině \(https://www.dji.com/cz/robomaster-s1/programming-guide\)](https://www.dji.com/cz/robomaster-s1/programming-guide)
- [Manál v angličtině \(https://dl.djicdn.com/downloads/robomaster-s1/20200324/RoboMaster\\_S1\\_User\\_Manual\\_v1.8\\_EN.pdf\)](https://dl.djicdn.com/downloads/robomaster-s1/20200324/RoboMaster_S1_User_Manual_v1.8_EN.pdf)
- [Kurz na Python \(https://naucese.python.cz/course/pyladies/\)](https://naucese.python.cz/course/pyladies/)

## Úkoly

Plňte zadané úkoly. Ideálně se snažte na řešení přijít a nedívejte se rovnou na výsledek (je uvedený vždy na další stránce). Pokud se vám budou zadané úkoly zdát moc jednoduché, plňte i bonusové úkoly. Nebojte se ptát, pokud něco nepůjde :).

### #1 Rozblikat robota

#### Zadání

Diody na hlavě (gimbal) robota rozsviňte postupně červeně, zeleně a modře. Každá barva bude rozsvícená 0.5 sekundy. Toto 3x opakujte.

#### Postup

Veškerý kód budeme psát do těla funkce `start()`, tedy takto:

```
def start():  
    # zde bude kód
```

Pro rozsvícení LED použijeme funkci:

```
led_ctrl.set_top_led(rm_define.armor_top_all, R, G, B, rm_define.effect_always_on)  
# R, G, B jsou kombinace barev
```

Kde je proměnná `led_ctrl` objekt ovládající diody. Objekt `led_ctrl` má metodu (funkci) `set_top_led`, kterou využijeme pro rozsvícení požadovaných diod. První parameter `rm_define.armor_top_all` odkazuje na objekt `rm_define`, což je soubor definic *konstant*, pomocí kterých adresujeme prvky robota. `rm_define.armor_top_all` adresuje horní diody.

Proměnné `R`, `G` a `B` jsou intenzity *červné*, *zelené* a *modré* (v počítači jsou barvy definované intenzitou těchto tří složek/barev). `R`, `G` a `B` mohou být od 0 do 255. (víte proč?) (bonus: kolik barevných kombinací takto můžeme reprezentovat?)

#### Čekání

Aby bylo vidět, že jsme požadovanou barvu rozsvítili, tak než dáme příkaz pro rozsvícení další barvy, musíme s rozsvícením počkat (spinkat = `sleep`).

```
time.sleep(x) # kde x je počet sekund, kolik se má čekat.  
# Může být i desetinné číslo - např. 0.1, pak se jedná o zlomky vteřiny
```

Nyní poskládejte rozsvícení tří požadovaných barev a čekání před každým rozsvícením.

### Opakování rozsvícení - cyklus

Výsledek příkazů opakujte v cyklu. Příklad cyklu:

```
for i in range(3):  
    print(i) # vypíše 0,1,2 - tzn. cyklus se 3x provede
```

### Řešení

```
def start():
    cekani = 0.5 # půl sekundy
    cyklu = 3
    for i in range(cyklu):
        led_ctrl.set_top_led(rm_define.armor_top_all, 255, 0, 0, rm_define.effect_always_on)
        time.sleep(cekani)
        led_ctrl.set_top_led(rm_define.armor_top_all, 0, 255, 0, rm_define.effect_always_on)
        time.sleep(cekani)
        led_ctrl.set_top_led(rm_define.armor_top_all, 0, 0, 255, rm_define.effect_always_on)
        time.sleep(cekani)
```

Bonus: zvládnete tento kód napsat přehledněji (nebo jinak)?.

## #2 Rozpoznávat číslice

Nyní se můžeme vrhnout na další úkol. Lze rozšířit předchozí příklad, ale také můžeme začít od nuly.

### Zadání

Rozpoznejte robotem, která číslice je ukázána.

### Postup

Nejprve zapneme rozpoznávání markerů (marker = piktogram, neboli jednoduchý QR kód):

```
vision_ctrl.enable_detection(rm_define.vision_detection_marker)
```

Definujeme funkci start, kde pouze budeme čekat. Po dobu tohoto čekání se budou detekovat markery.

```
def start():
    time.sleep(10)
```

Dále definujeme tzv. callback (funkce která se zavolá, když se něco stane). Tento callback se zavolá, když se rozpozná marker s číslem:

```
def vision_recognized_marker_number_all(msg):
    # tady si zadefinujeme, co se má stát, když detekujeme marker s číslem
```

A tento callback rozšíříme o vypsání rozpoznané číslice

```
def vision_recognized_marker_number_all(msg):
    print(vision_ctrl.get_marker_detection_info())
```

Co se nám vypsalo?

```
[1, 11, 0.5, 0.5666..., 0.123..., 0.277...]
```

Vypsal se nám objekt `detection_info`, viz <https://www.dji.com/cz/robomaster-s1/programming-guide> -> Smart -> 14. Identified marker info. Jednotlivé prvky v poli značí [počet detekovaných markerů, ID detekovaného markeru, souřadnice X, Y, šířka markeru, výška]. Pro zjištění o jakou číslici se jedná budeme potřebovat to 2. číslo. V příkladu výše je tam hodnota 11. Ale pozor, jedná se o tzv. ID markeru, nejedná se o jeho hodnotu (co je na obrázku). ID jsou obrázkům přiřazena následujícím způsobem (viz předchozí odkaz):

```
ID=0: emit sounds
ID=1: stop
ID=2: dice
ID=3: target
ID=4: left arrow
ID=5: right arrow
ID=6: forward arrow
ID=8: red heart
ID=10-19: 0-9
ID=20-45: A-Z
```

Pokud tedy chceme zjistit detekované číslo, musíme od ID markeru odečíst 10.

### Detekované číslo - řešení

```
vision_ctrl.enable_detection(rm_define.vision_detection_marker)
```

```
def vision_recognized_marker_number_all(msg):  
    cislo = vision_ctrl.get_marker_detection_info()[1] - 10  
    print(cislo)
```

Řešení otestujte na dostupných obrázcích (markerech). Rozpozná toto řešení i jiné markery než čísla?

Bonus: rozsviďte robota barvou v závislosti na ukázaném markeru.

Bonus bonusu: číslo na markeru bude definovat "sytost" barvy (např. lineární škálou).

### #3 Rozpoznávat číslíce - střelba

#### Zadání

Definujte si funkci `strelba()` (nebo anglicky `fire()`), která bude simulovat střelbu bliknutím diody pod hlavní. Pokud robotovi ukážete marker s číslem, robot "vystřelí" na marker tolikrát kolik je na markeru číselná hodnota.

#### Postup

Vyjděte z předchozího příkladu.

Diodu pod hlavní ovládáme funkcemi `led_ctrl.gun_led_on()` a `led_ctrl.gun_led_off()`. Zvolte tedy vhodné čekání mezi rozsvícením a zhasnutím aby byla frekvence blikání vhodná pro simulaci střelby. Nezapomeňte po detekování číslíce chvíli počkat na další číslici. Detekce totiž probíhá kontinuálně a to několikrát za vteřinu, robot by vám tedy neustále "střílel",

#### Řešení

```
def fire():
    led_ctrl.gun_led_on()
    time.sleep(0.2)
    led_ctrl.gun_led_off()
    time.sleep(0.2)

def vision_recognized_marker_number_all(msg):
    cislo = vision_ctrl.get_marker_detection_info()[1] - 10
    for _ in range(cislo):
        fire()
    time.sleep(4) # počkáme před detekcí dalšího markeru

def start():
    vision_ctrl.enable_detection(rm_define.vision_detection_marker) # nezapomeneme zapnout detekci markerů
    time.sleep(60) # program poběží minutu
```

Bonus: při střelbě přehrajte zvuk.

### #3 Pohybovat hlavou dokola a střílet na číslice

#### Zadání

Úkolem je nyní rozpohybovat hlavu robota. Cílem je, aby se hlava (gimbal) otáčela, postupně na jednu a pak na druhou stranu (hlava se nemůže točit neomezeně dokola). Hlava se bude otáčet o půl otočky dozadu vždy na jednu a pak na druhou stranu. Pokud v průběhu otáčení bude detekován marker s číslem, otáčení hlavy se zastaví a robot vystřelí. Poté bude v otáčení pokračovat.

#### Postup

Pro otáčení hlavy (gimbalu) použijte následující funkce:

- `robot_ctrl.set_mode(rm_define.robot_mode_free)` - Nastaví volný pohyb hlavy (tělo nebude následovat otáčení hlavy).
- `gimbal_ctrl.set_rotate_speed(speed)` - Nastaví rychlost otáčení hlavy (můžete zkusit různé stupně rychlosti 0-540, avšak otestujte, že i ve větší rychlosti robot rozpozná markery)
- `gimbal_ctrl.yaw_ctrl(degree)` - Otočí hlavu o degree stupňů (povolené hodnoty jsou od -250° do 250°).
- `gimbal_ctrl.suspend()` - Pozastaví otáčení
- `gimbal_ctrl.resume()` - Pokračuje v otáčení
- `gimbal_ctrl.yaw_pitch(degree)` - Náklon hlavy ve svislém směru o degree stupňů (0 je vodorovně).

Bonus: místo čekání na další rozpoznání markeru s číslem si vedte evidenci již "rozstřílených" číslic a na takové markery pak nestřílejte.

### #4 Bonus - Hlídkování

#### Zadání

Rozšiřte předchozí úkol tak, aby se robot po otočce na každou stranu přesunul cca metr do strany a tam hledal makery stejným způsobem. Následně se zase vraťte na původní místo a opakujte hledání.