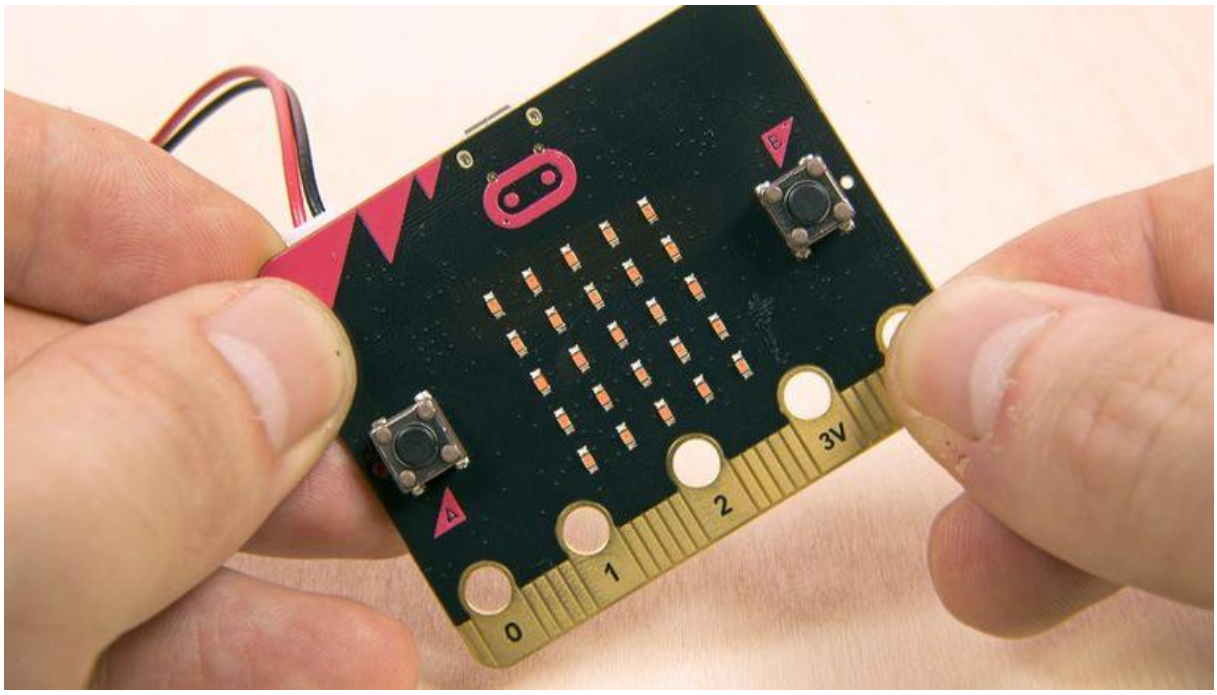
Micro:bit – programování v Pythonu

Ľuboslav Lacko

Python je interpretovaný programovací jazyk, určený pro všeobecné programování. Hlavnou výhodou je jeho srozumitelnost, takže zdrojovému programu v jazyku Python porozumí i začátečník a velice brzy v něm dokáře realizovat své projekty. Pokud ovšem začátečníkovi ukážete kód v jazyku C, který se hemží směrníky, nejspíš ho od programování odradíte.



Jazyk Python byl navržen jako vysoce modulární. K dispozici je několik stovek programových modulů, což z něj činí jazyk použitelný pro jakýkoli projekt. Cílem autorů Pythonu je, aby vytváření aplikací bylo zábavné. To se odráží i v názvu jazyka, který se jmenuje podle britské komediální skupiny Monty Python.

Pokud chcete tento programovací jazyk používat na počítači, musíte si nainstalovat Python verze 3.x [2]. Jako vývojové prostředí doporučujeme PyCharm [3].

Webová aplikace

K programování aplikací pro desku Micro:bit můžete použít webový nástroj [4], pak už nebudete muset nic instalovat do místního počítače.

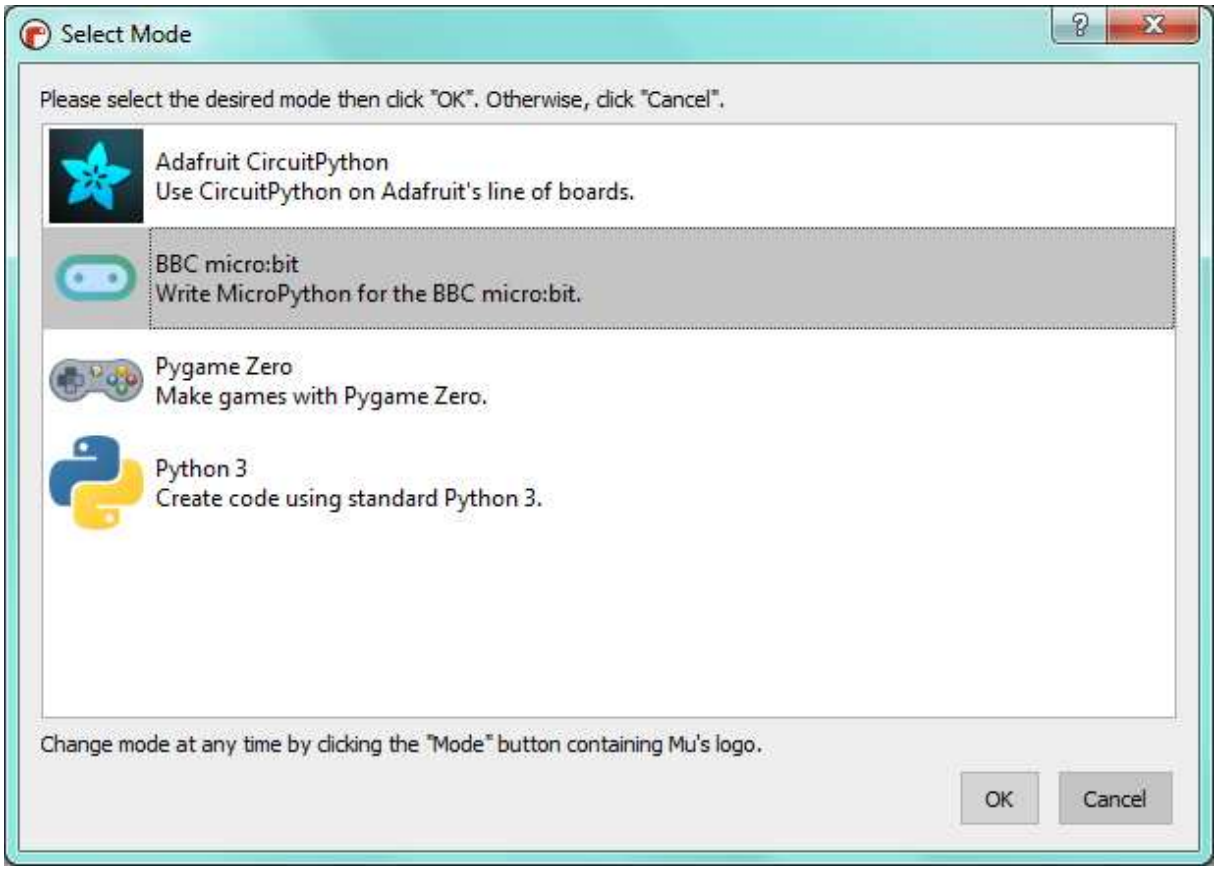
Po vytvoření programu v prostředí stáhnete binární kód z webu v souboru s příponou hex. Micro:bit připojený k počítači kabelem USB se zobrazí jako klasické paměťové zařízení USB,

do kterého stačí přetažením nebo jiným obvyklým způsobem zkopírovat stažený soubor programu. Program se pak v zařízení automaticky spustí. V případě aplikace Python obsahuje tento hexadecimální soubor nejen kód aplikace, ale také engine interpretu MicroPython.

Off-line editor

To není příliš praktické, takže kód můžete psát, kompilovat a nahrát do desky Micro:bit přímo na vývojářském počítači. K tomu potřebujete pouze editor **mu** [5].

Pomocí tlačítka Flash nahrajete program Python přímo do desky Micro:bit připojené přes USB. Stejně jako u nástrojů pro vývoj webových stránek se nemusíte zabývat stahováním souboru s příponou hex a jeho kopírováním do micro:bitu.



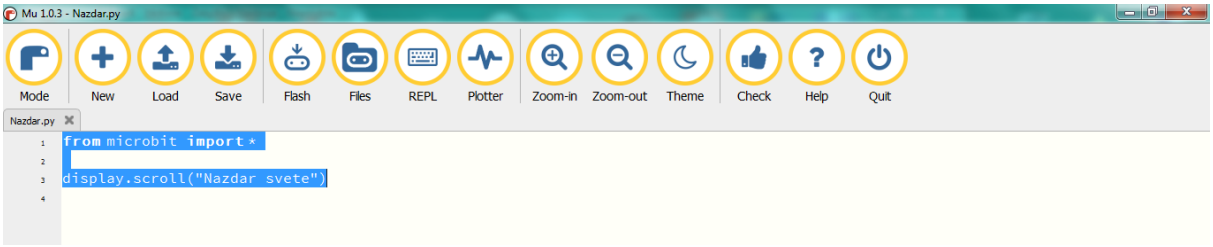
Podporované režimy nástroje Mu

Začínáme

Nebudeme se primárně zabývat syntaxí jazyka Python, ale jen specifiky implementace jazyka Micro Python na mikropočítačové desce BBC micro:bit, tj. fungováním jejích hardwarových komponent; displejem, tlačítky, vestavěnými senzory a samozřejmě možností práce se sběrnicí, ke které lze připojit různá zařízení a externí moduly. BBC micro:bit má jednoduchý maticový displej s 25 LED, tlačítka, senzory pohybu, náklonu, teploty a intenzity okolního světla. Podporuje dva typy bezdrátové komunikace - Bluetooth Low Energy (BLE) pro

komunikaci s mobilními zařízeními a rádiový přenos pro komunikaci s jinými zařízeními BBC micro:bit. Sériová komunikace s počítačem prostřednictvím kabelu USB je také možná.

V jazyku Python není bohužel komunikace přes BLE podporována, takže nám zbývá jen rádiový přenost. Ale o tom později.



Příkaz `from microbit import *` importuje knihovny obsahující funkce specifické pro micro:bit. Hvězdička označuje, že jsou importovány všechny knihovny.

Zobrazování

Zobrazovací schopnosti desky Micro:bit jsou poměrně skromné, k dispozici je maticový displej s LED, rozložnými v rastru 5 x 5. Jednotlivé LED jsou adresovány individuálně od (0,0) v levém horním rohu až po (4,4). Třetím parametrem funkce `set_pixel()` je jas příslušné LED (0 nesvítí, 9 nejvyšší jas).

```
from microbit import *
display.set_pixel(0, 4, 9)
```

Další příklad ukazuje cyklické rozsvícení LED maticového displeje:

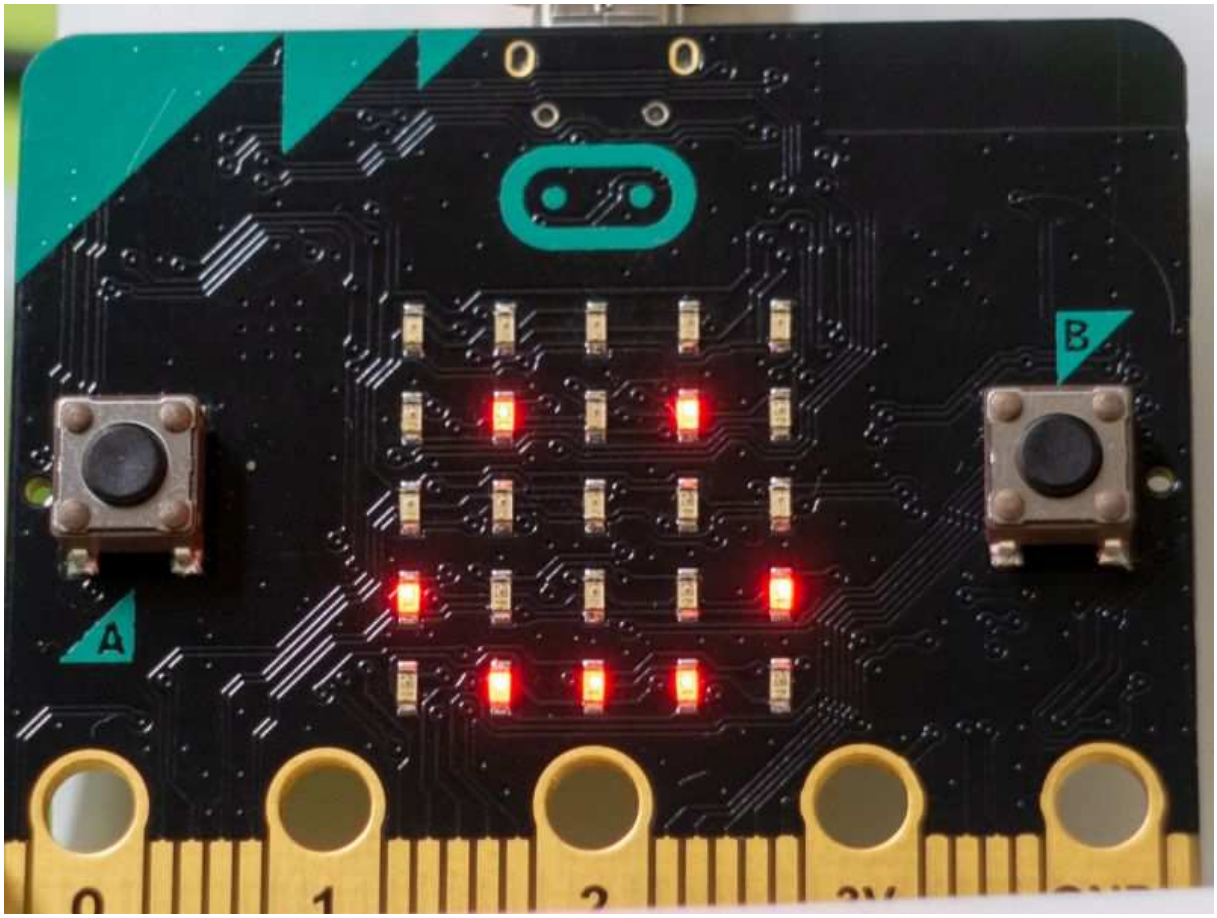
```
from microbit import *
display.clear()
for x in range(0, 5):
    for y in range(0, 5):
        display.set_pixel(x, y, 9)
```

Na displeji lze staticky zobrazit jeden znak nebo jiný grafický symbol. Víc se jich tam nevejde. I na takovém displeji je však možné zobrazovat textový řetězec o několika znacích. Trik spočívá v tom, že se text vodorovně roluje.

```
from microbit import *
display.scroll("Nazdar svete!")
```

Chcete-li zobrazit statickou ikonu předdefinovaného obrázku, použijte příkaz ve tvaru:

```
from microbit import *
display.show(Image.HAPPY)
```



K dispozici je velké množství obrázků:

HEART, HEART_SMALL, HAPPY, SMILE, SAD, CONFUSED, ANGRY, ASLEEP, SMILE, SURPRISED, SILLY, FABULOUS, MEH, YES, NO, TRIANGLE, TRIANGLE_LEFT, CHESSBOARD, DIAMOND, DIAMOND_SMALL, SQUARE, SQUARE_SMALL, RABBIT, COW, MUSIC_CROTCHET, MUSIC_QUAVER, PITCHFORK, XMAS, PACMAN, TARGET, TSHIRT, ROLLERSKATE, DUCK, HOUSE, TORTOISE, BUTTERFLY, STICKFIGURE, GHOST, SWORD, GIRAFFE, SKULL, UMBRELLA, SNAKE

Lze zobrazit ikony šipek: ARROW_N, ARROW_NE, ARROW_E, ARROW_SE, ARROW_S, ARROW_SW, ARROW_W, ARROW_NW, kde je směr určen zkratkami světových stran na podtržítku a dokonce i v analogové podobě pro vyjádření času s přesností na hodiny: CLOCK1 - CLOCK12.

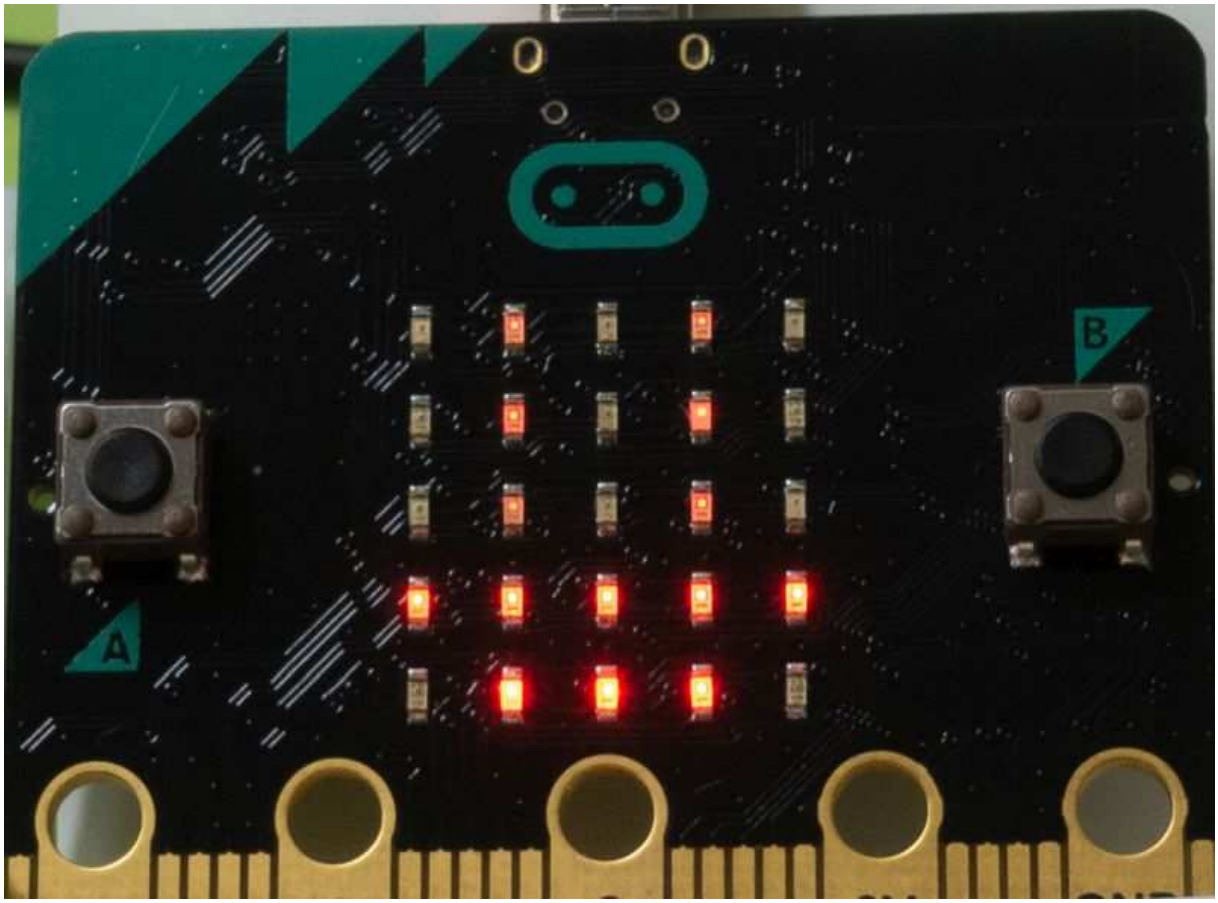
Hodiny a šipky lze také zobrazit jako animovanou sekvenci pomocí definovaných objektů polí ALL_ARROWS a ALL_CLOCKS. Tímto způsobem můžete vytvořit například animovaný ukazatel hodin, který uživateli naznačuje, že se na něco čeká:

```
from microbit import *
display.show(Image.ALL_CLOCKS, loop=True, delay=200)
```

Pokud vám nevyhovuje žádná ikona z uvedené poměrně bohaté nabídky, můžete si definovat vlastní obrázek v matici 5 x 5 LED, které tvoří pixely. V pětimístných textových řetězcích označuje číslice 0 vypnutou LED a čísla 1 až 9 určují jas rozsvícených LED.

Například:

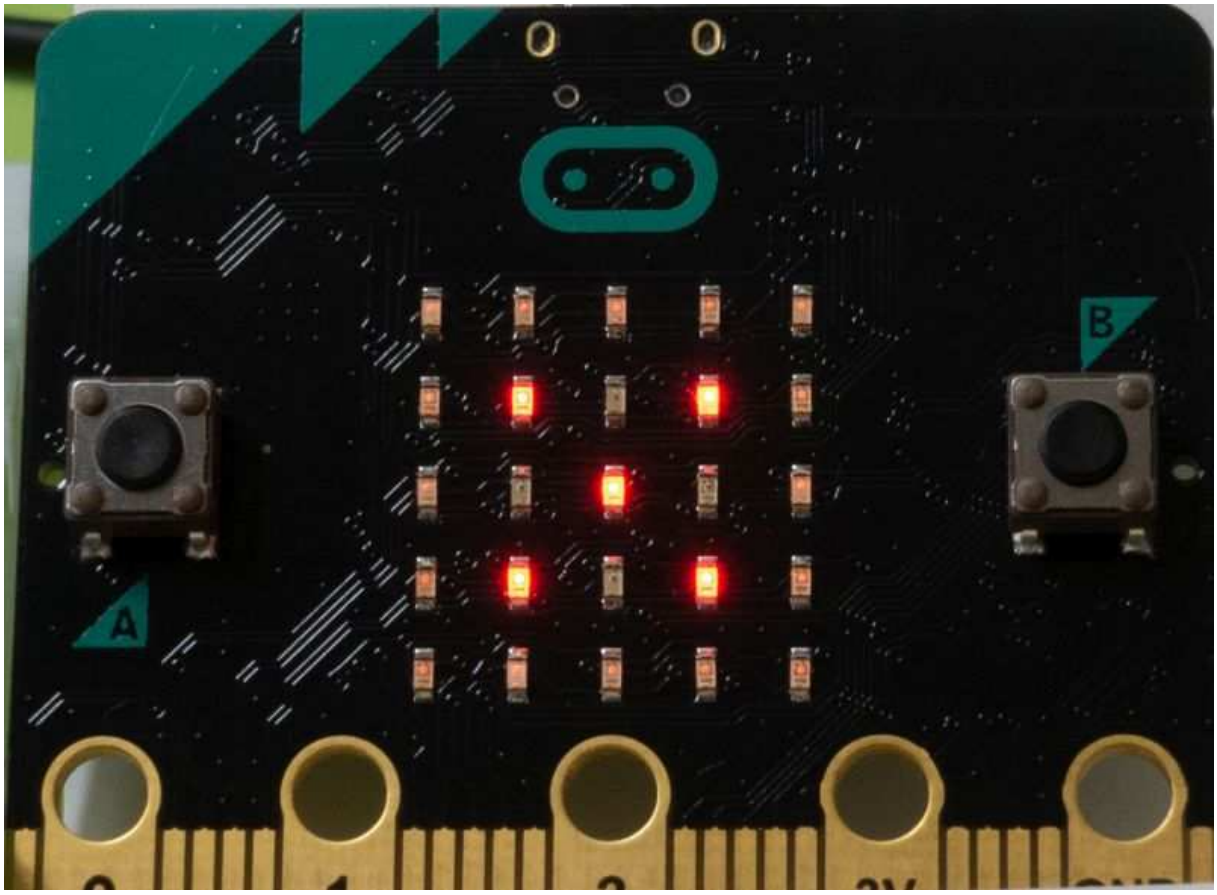
```
from microbit import *
lodka = Image("05050:05050:05050:99999:09990")
display.show(lodka)
```



Ne každý má takovou představivost, aby dokázal navrhnout obrázek tímto způsobem, nebo aby si dokázal představit, co se z takového záznamu zobrazí. Mnohem přehlednější je takový zápis, kde číslice v řetězcích tvoří matici 5 x 5 bodů.

```
from microbit import *
obr = Image("33333:"
            "39093:"
            "30903:"
            "39093:"
            "33333")
display.show(obr)
```

Hádáte správně, je to hrací kostka s číslem 5, a vidět je i obrys kostky. Animaci je možno vytvořitt i z vlastních obrázků, stačí je uspořádat do pole.



Například byste mohli zobrazit hod kostkou (pro zjednodušení v jednoduché sekvenci) pomocí animace.

```
from microbit import *
dice1 = Image("33333:"
              "30003:"
              "30903:"
              "30003:"
              "33333")

dice2 = Image("33333:"
              "30093:"
              "30003:"
              "39003:"
             "33333")

dice3 = Image("33333:"
              "39003:"
              "30903:"
              "30093:"
              "33333")

dice4 = Image("33333:"
              "39093:"
```

```
                "30003:"
                "39093:"
                "33333")

dice5 = Image("33333:"
                "39093:"
                "30903:"
                "39093:"
                "33333")

dice6 = Image("33333:"
                "39093:"
                "39093:"
                "39093:"
                "33333")

all_obr = [dice1, dice2, dice3, dice4, dice5, dice6]
display.show(all_obr, delay=500)
```

Tlačítka

Budeme rozlišovat mezi prvky pro interakci desky s uživatelem a piny na sběrnici, určenými pro interakci s připojenými zařízeními. Primární tlačítka A a B jsou k dispozici pro interakci uživatele. V kódu jsou reprezentovány objekty button_a a button_a. Migranty z jiných jazyků upozorňujeme, že tělo cyklu `while True:` v jazyce Python se skládá ze všech příkazů pod příkazem definujícím cyklus, které jsou odsazeny od levého okraje začátku příkazu o jeden tabulátor, nebo jak často používají vývojáři, odsazeny o 4 mezery. V jednom souboru kódu však můžete použít pouze jeden druh odsazení, a to buď tabulátory, nebo mezery.
Příkazy v podmínce typu `if` jsou odsazeny stejným způsobem.

```
from microbit import *
while True:
    if button_a.is_pressed() and button_b.is_pressed():
        display.scroll("AB")
        break
    elif button_a.is_pressed():
        display.scroll("A")
    elif button_b.is_pressed():
        display.scroll("B")
sleep(100)
```

Funkce `is_pressed()` zjišťuje, zda bylo tlačítko v daném okamžiku stisknuto. V mnoha případech je užitečná funkce `was_pressed()`, která vám řekne, zda bylo tlačítko stisknuto v době, kdy kód dělal něco jiného.

```
from microbit import *
```

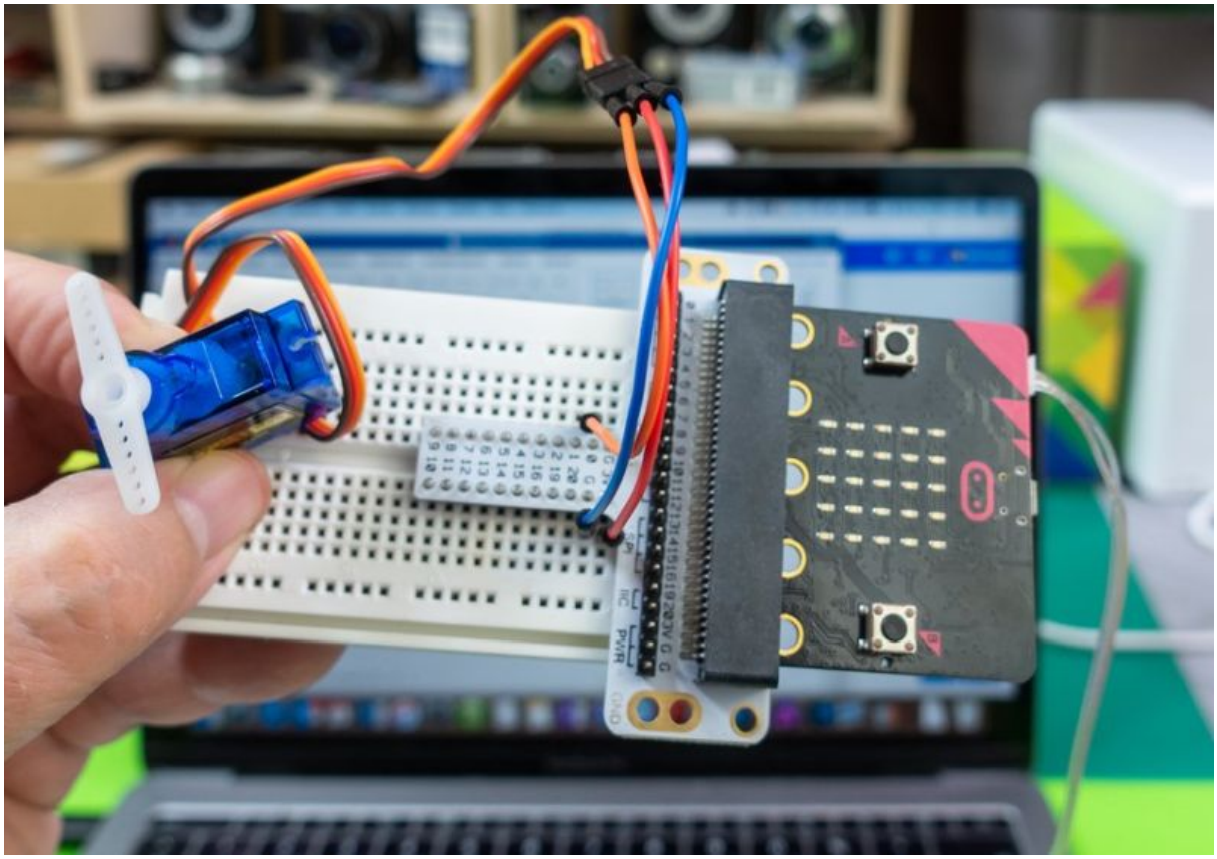
```
while True:
    if button_a.was_pressed():
       display.scroll("A")
    else:
       display.scroll(Image.ASLEEP)
   sleep(1000)
```

Pokud potřebujete zjistit, kolikrát bylo tlačítko stisknuto během určité doby, kdy byla deska uspána, ale přesto zaznamenala události stisknutí tlačítka, použijete funkci `get_presses()`.

Tento příklad uvádí počet stisknutí tlačítka A během trojsekundového intervalu.

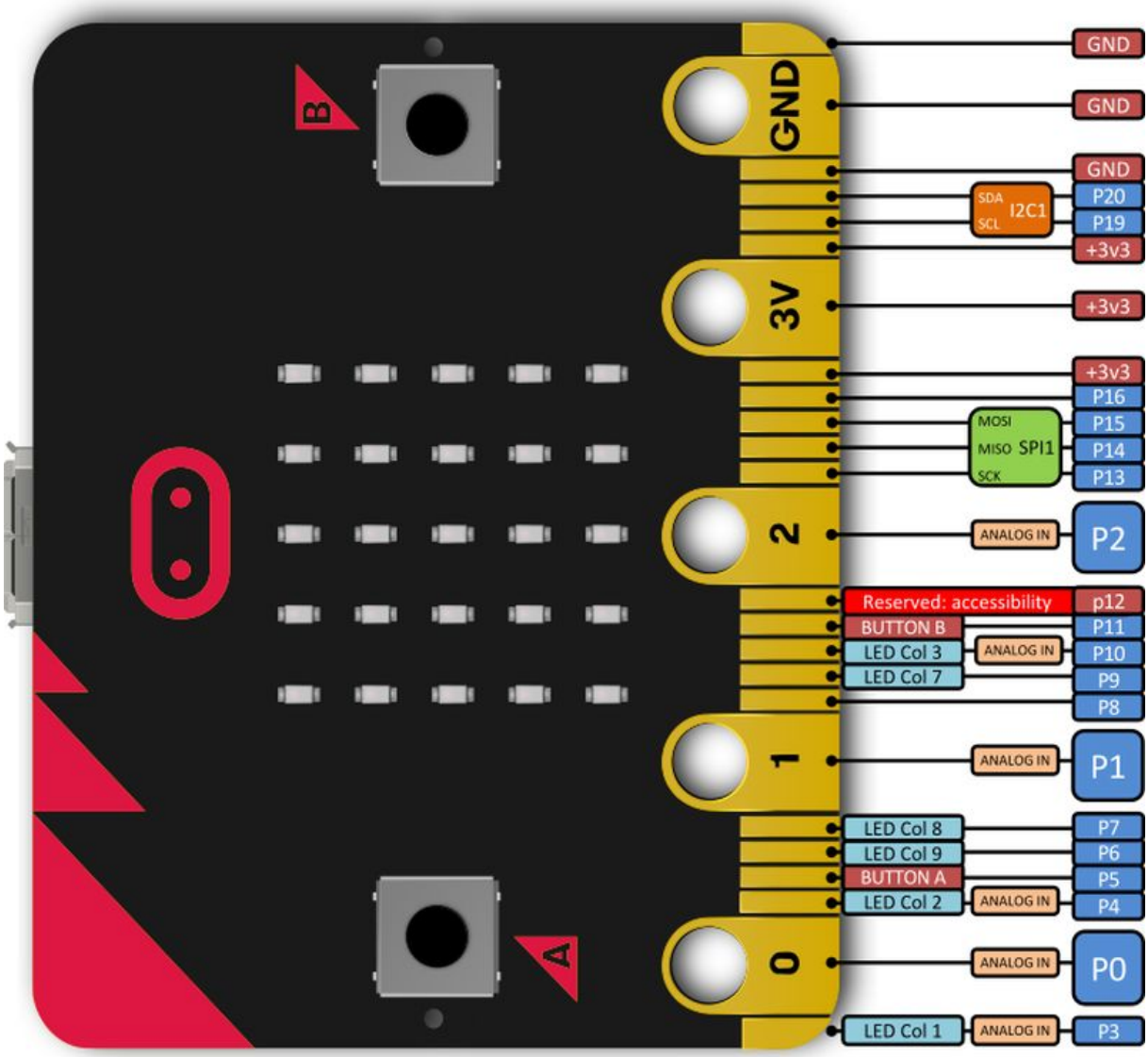
```
from microbit import *
while True:
    sleep(3000)
    count = button_a.get_presses()
display.scroll(str(count))
```

Obsluha hardware a soubory



Piny sběrnice a PWM

Vstupní/výstupní piny sběrnice, z nichž mnohé mají alternativní funkce, jsou adresovány jako `microbit.pin0 - microbit.pin20`.



Piny 0,1 a 2 mají 10 megaohmové rezistory připojené na +3,3 V, které slouží jako dotykový senzor. Funguje to tak, že jednou rukou uchopíte pin GND a prstem druhé ruky se dotknete pinů 0, 1 nebo 2. Rezistory 10 k jsou připojeny na piny 5 a 11, aby byla umožněna funkce tlačítek A a B. Pro zobrazení se používají také sběrnicové piny 3, 4, 6, 7, 9 a 10. Pokud je chcete použít k jinému účelu, musíte displej vypnout příkazem `microbit.display.off()`.

V tomto příkladu využijeme kontaktní políčka pro dotyk prstu. Zda se osoba dotýká pinu, lze zjistit pomocí příkazu `is_touched()`, který vrací hodnoty True nebo False.

```
from microbit import *
while True:
```

```
    if pin0.is_touched():
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
```

Pokud je pin nastaven jako digitální vstup, úroveň napětí, a tedy logická hodnota, se načte pomocí příkazu `read_digital()`. Příkaz vrací hodnoty 0 nebo 1.

```
from microbit import *
while True:
    if pin0.read_digital():
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
```

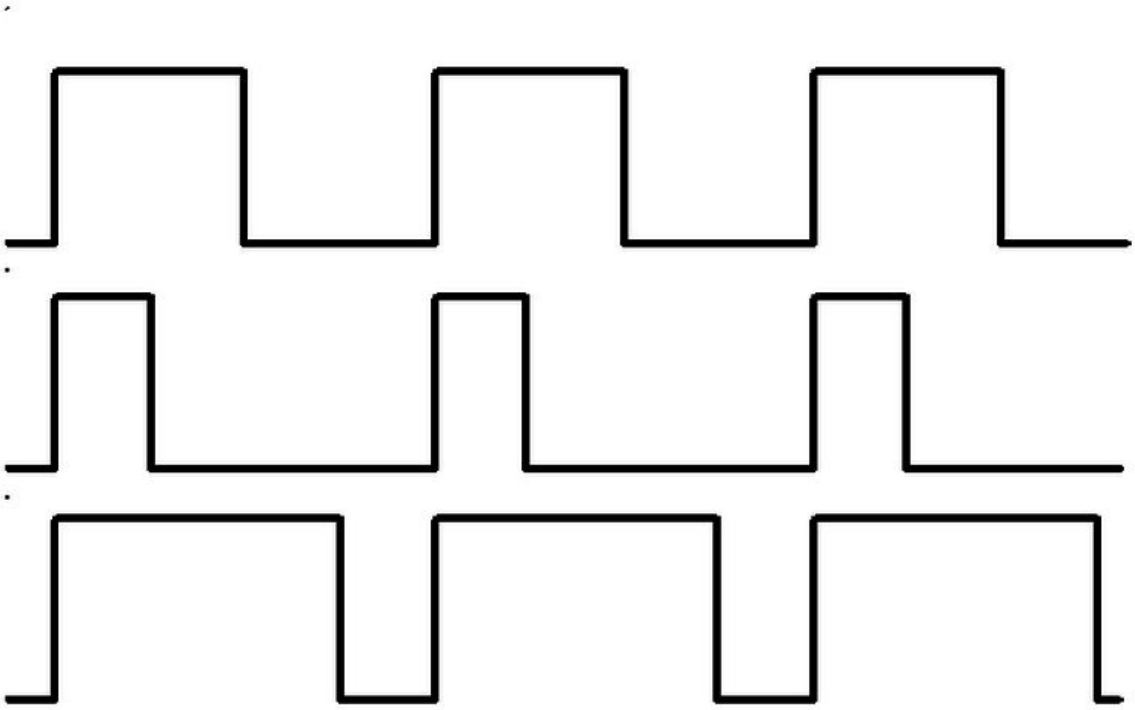
Hodnota na příslušném pinu se nastavuje příkazem `write_digital(parametr)`, přičemž parametr může nabývat hodnot 0 nebo 1.

```
from microbit import *
while True:
    pin0.write_digital(1)
    sleep(500)
    pin0.write_digital(0)
    sleep(500)
```

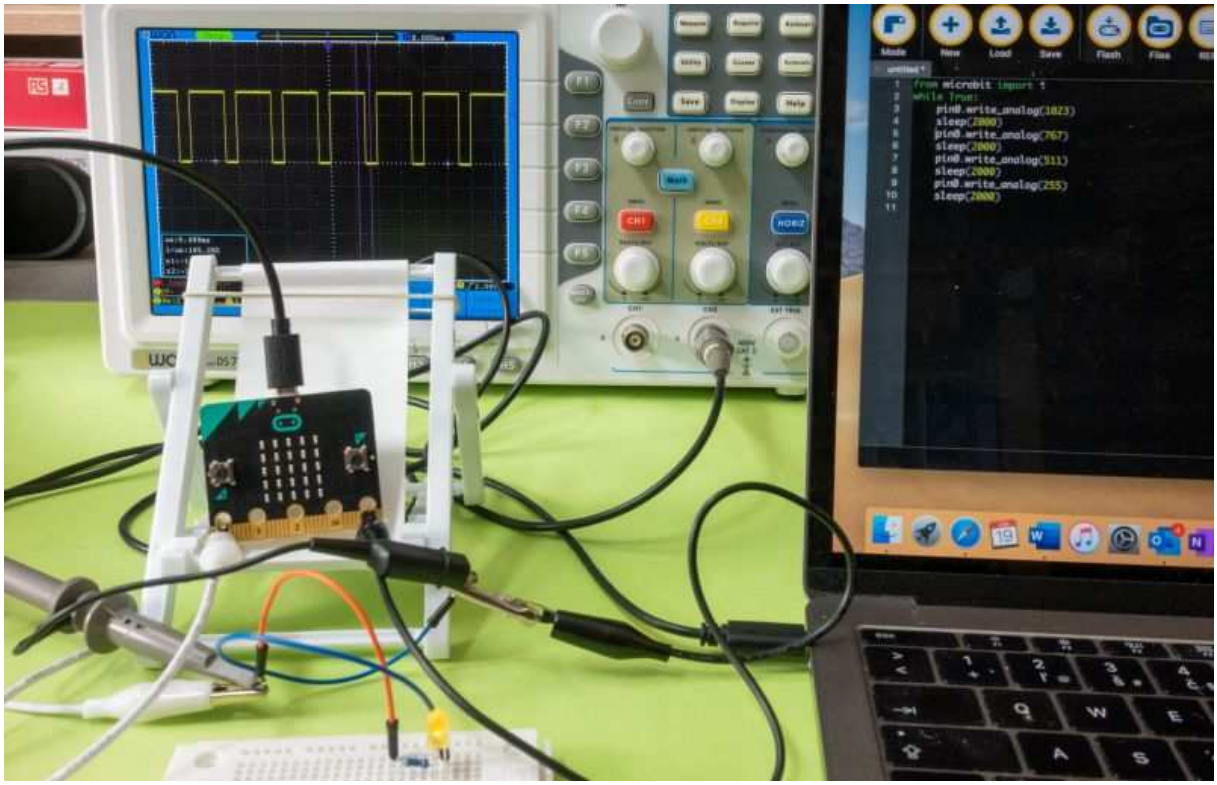
Pokud používáte piny v analogovém režimu, použije se k jejich čtení příkaz `read_analog()` a rozsah napětí na vstupu 0 až 3,3 V bude odpovídat hodnotám od 0 do 1023.
Na pinu sice můžete nastavit i analogovou hodnotu `write_analog(parametr)`, ale ta se projeví jako PWM signál, tj. šířkově pulsně modulovaný průběh. Hodnoty parametrů se pohybují od 0, což je 0 % signálu v logické úrovni 1, do 1023, což je 100 % signálu v logické úrovni 1.

```
from microbit import *
while True:
   pin0.write_analog(1023)
   sleep(2000)
   pin0.write_analog(767)
   sleep(2000)
   pin0.write_analog(511)
   sleep(2000)
   pin0.write_analog(255)
   sleep(2000)
```

Obrázek ukazuje tři průběhy s šířkovou modulací:



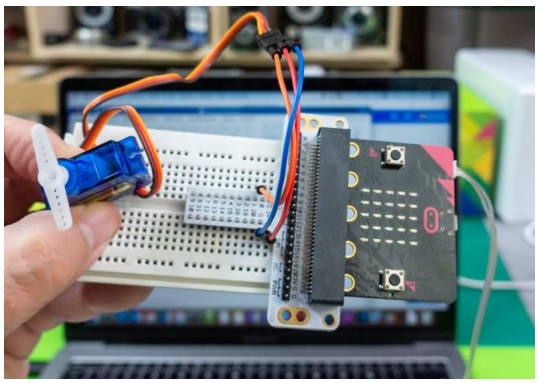
- První by byl generován pomocí write_analog (511), protože má přesně 50% periodu - polovinu času každé periody je napětí na pinu 3,3 a polovinu času je na úrovni GND. Výsledkem je, že celková energie tohoto signálu je stejná, jako kdyby bylo nastaveno 1,65 V místo 3,3 V.
- Druhý signál má 25% pracovní cyklus a lze jej generovat pomocí write_analog(255). Má to podobný účinek, jako kdyby na tomto pinu bylo trvalé napětí 0,825 V.
- Třetí signál má pracovní cyklus 75 % a je generován pomocí write_analog (767). Má třikrát větší energii než druhý signál a odpovídá napětí 2,475 V.
- PWM dobře funguje se zařízeními, jako jsou žárovky nebo motory, které mají setrvačnost, případně využívají setrvačnost lidského oka, typickým příkladem jsou LED.

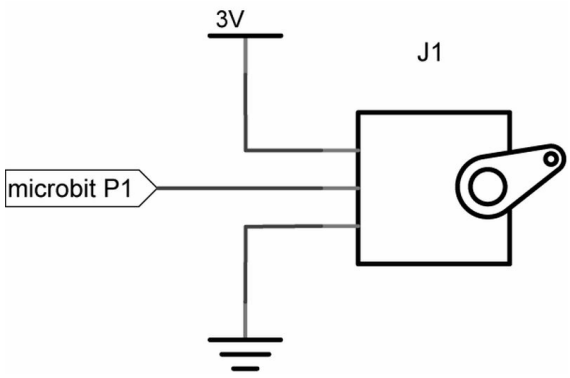


PWM s přesnou šířkou impulsu se často používá k řízení úhlu natočení servomotorů nebo k řízení výkonu stejnosměrných motorů, například u dronů. Proto je nutné, aby bylo možné přesně nastavit periodu impulzu.

K nastavení se používají příkazy `set_analog_period(count_milliseconds)` nebo `set_analog_period_microseconds(count_microseconds)`. Pro mikrosekundy je minimální hodnota parametru 256 µs.

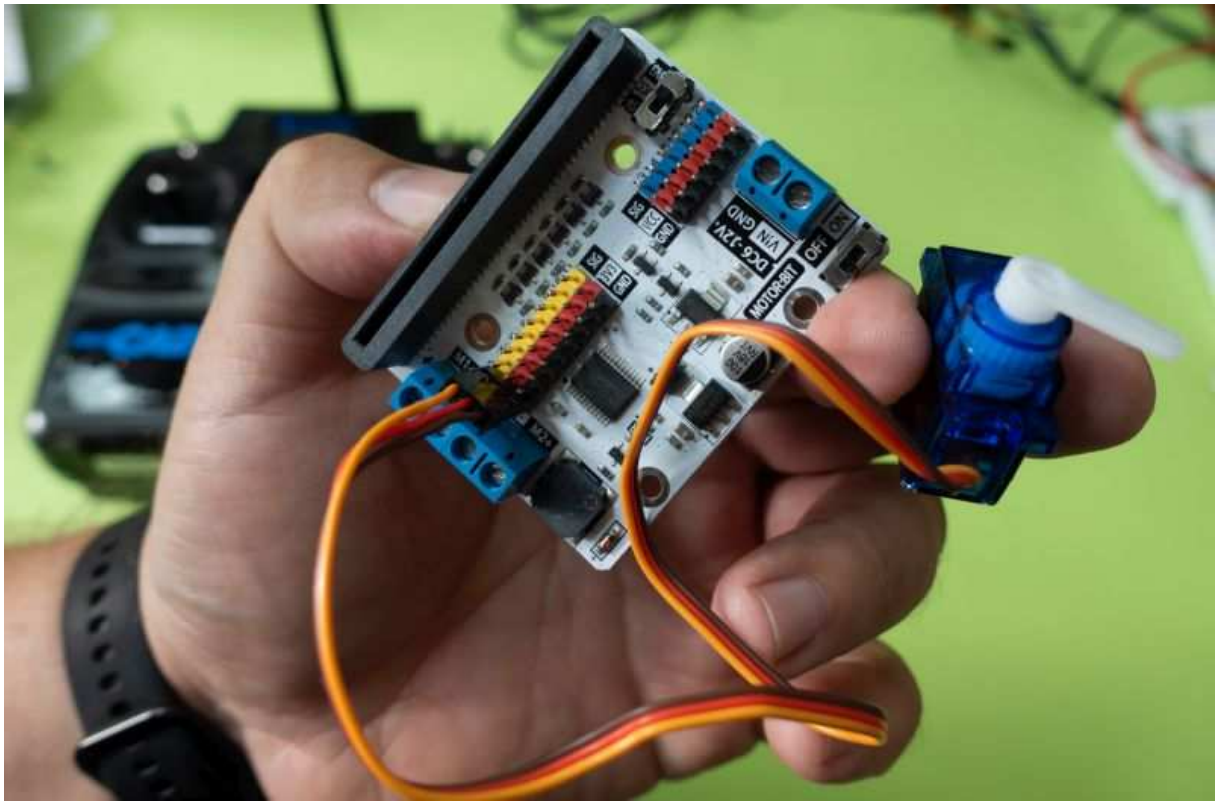
Pro servomotory se používá frekvence signálu PWM 50 Hz, což je 20 milisekund.





Zapojení serva

Servomotory a stejnosměrné motory lze připojit také pomocí přídavné desky.



Program, který pootočí servem z 0 na 180° a zpět:

```
from microbit import *
pin0.set_analog_period(20)
while True:
    pin0.write_analog(180)
    sleep(1000)
    pin0.write_analog(1)
    sleep(1000)
```

Zvuk

Pokud mezi piny 0 a GND připojíte malý reproduktor, můžete micro:bitem generovat tóny. Zdůrazňuji reproduktor, nikoli bzučák, protože bzučák po přivedení napětí generuje pouze jeden tón. Případně můžete frekvenci tónu plynule měnit. Na obrázku je reproduktor ze sady Grove.


```
from microbit import *
import music
while True:
    for freq in range(880, 1760, 16):
       music.pitch(freq, 6)
    for freq in range(1760, 880, -16):
       music.pitch(freq, 6)
```

Můžete například nechat přehrát jednu z předdefinovaných melodií:

```
from microbit import *
import music
music.play(music.BIRTHDAY)
```

Existují melodie:
DADADUM, ENTERTAINER, PRELUDE, ODE, NYAN, RINGTONE, FUNK, BLUES, BIRTHDAY, WEDDING, WEDDING, FUNERAL, PUNCHLINE, PYTHON, PYTHON, BADDY, CHASE, BA_DING, WAWAWAWAWAA, JUMP_UP, JUMP_DOWN, POWER_UP a POWER_DOWN.
Nebo můžete pomocí tónů definovat vlastní melodii. Každá nota je definována textovým řetězcem ve tvaru NOTA[oktáva]:[trvání].

```
from microbit import *
import music
tune = ["C4:4", "D4:4", "E4:4", "C4:4", "C4:4", "D4:4", "E4:4", "C4:4",
       "E4:4", "F4:4", "G4:8", "E4:4", "F4:4", "G4:8"]
music.play(tune)
```

nebo jednodušeji:

```
from microbit import *
import music
tune = ["C4:4", "D", "E", "C", "C", "D", "E", "C", "E", "F", "G:8",
       "E:4", "F", "G:8"]
music.play(tune)
```

Měření teploty

Deska micro:bit má integrovaný teplotní senzor, takže vytvořit z micro:bitu teploměr, který bude zobrazovat teplotu na displeji jako rolující údaj, je hračka.

```
from microbit import *
while True:
   teplota = temperature()
  display.scroll(str(teplota) + 'C')
  sleep(500)
```

Akcelerometr

Akcelerometr na desce BBC micro:bit snímá pohyb ve třech osách:

- X - naklápění doleva a doprava,
- Y - naklápění dopředu a dozadu,
- Z - pohyb nahoru a dolů.

Funkce `accelerometer.get_x()` vrací kladnou nebo zápornou hodnotu zrychlení v daném směru v jednotkách, které lze označit jako milig. Předpona mili v tomto případě pro snadnější implementaci neznamená 1/1000, ale 1/1024. Gravitační zrychlení v ose Z má tedy hodnotu 1024 milig.
Pomocí kódu můžete zobrazit zrychlení v každé ose:

```
from microbit import *
while True:
    x = accelerometer.get_x()
    y = accelerometer.get_y()
    z = accelerometer.get_z()
   print("x, y, z:", x, y, z)
   sleep(500)
```

Pokud držíte desku micro:bit v klidu, displejem směrem nahoru, hodnoty zrychlení X a Y se pohybují kolem nuly a zrychlení Z je přibližně -1024. To vám říká, že gravitace působí směrem dolů k desce. Pokud desku otočíte displejem dolů, měla by se hodnota změnit na +1024

milig. Pokud deskou zatřesete dostatečně silně, uvidíte, že zrychlení dosahuje hodnot až ±2048 mili-G. To proto, že tento akcelerometr je nastaven na měření maximálně ±2048 milig. Skutečné zrychlení ovšem může být větší než to naměřené.

Příklad pro určení směru náklonu:

```
from microbit import *
while True:
   reading = accelerometer.get_x()
    if reading > 20:
       display.show("R")
    elif reading < -20:
       display.show("L")
    else:
       display.show("-")
```

Protože je akcelerometr poměrně citlivý, je v příkladu uvedena podmínka tolerance. Pokud se hodnota zrychlení pohybuje mezi -20 a 20 milig, nepovažuje se ještě za pohyb, ale za toleranci měření.

Gesta

Deska BBC micro:bit dokáže pomocí akcelerometru zachytit a analyzovat i složitější gesta. V kódu se jejich názvy zadávají jako textové řetězce. Jejich seznam je následující:

- up
- down
- left
- right
- face up
- face down
- freefall
- 3g
- 6g
- 8g
- shake

Gesta 3g, 6g a 8g jsou indikována, když zrychlení desky v jakémkoli směru překročí odpovídající hodnotu násobku gravitačního zrychlení. Následující příklad zobrazí ikonu veselého smajlíka pouze v případě, že je zařízení v poloze displejem nahoru.

```
from microbit import *
while True:
   gesture = accelerometer.current_gesture()
    if gesture == "face up":
```

```
        display.show(Image.HAPPY)
    else:
        display.show(Image.ANGRY)
```

Kompas

Snímač magnetického pole umožňuje zjistit polohu micro:bitu vůči světovým stranám. Před měřením je nutné elektronický kompas zkalibrovat, jinak nebudou následná měření přesná. Metoda `compass.calibrate()` spustí minihru, ve které máte za úkol naklánět desku micro:bit, dokud se celý displej nezaplní svítícími pixely. Metoda `compass.heading()` zadává úhel v rozsahu 0 až 360 stupňů ve směru hodinových ručiček, přičemž 0 je sever. Pro grafické zobrazení směru se výpočtem v následujícím příkladu určí index hodin tak, aby malá ručička zobrazených hodin na maticovém displeji ukazovala sever.

```
from microbit import *
compass.calibrate()
while True:
    smer = ((15 - compass.heading()) // 30) % 12
    display.show(Image.ALL_CLOCKS[smer])
```

Údaje v souborech

I u malého mikroprocesorového systému je užitečné mít možnost ukládat data do trvalého úložiště, tj. do paměti takového typu, kde zůstanou i po vypnutí napájení. MicroPython na platformě micro:bit podporuje jednoduchý souborový systém uložený v paměti flash, kde se nachází programový kód. Pro ukládání dat je k dispozici přibližně 30 kilobajtů. Připomínáme, že Microbit má 256 kB paměti flash a 16 kB paměti RAM. Při nahrávání kódu v souboru HEX jste si jistě všimli, že micro:bit připojený přes USB se tváří jako USB klíč k počítači. Soubory, které vytváříte a do kterých zapisujete data, však na této připojitelné jednotce nebudou. Souborový systém na desce micro:bit je jednoduchý, takže není možné uspořádat soubory do složek.

V prvním příkladu uložíme do souboru nějaká data, pro zjednodušení krátký text. Příkaz `open()` otevře, nebo pokud ještě nebyl vytvořen, vytvoří soubor a přiřadí jej k objektu. Všimněte si, že parametr 'w' slouží k nastavení objektu soubor do režimu zápisu. Parametr 'r' slouží k nastavení objektu do režimu čtení, což je také výchozí nastavení.

V příkladu funkce otevře (vytvoří) soubor s názvem udaje.txt a přiřadí jej objektu soubor. V bloku kódu odsazeném pod příkazem with se pak pro zápis použije objekt soubor.

```
from microbit import *
with open('udaje.txt', 'w') as soubor:
    soubor.write("Lorem ipsum")
```

V dalším příkladu tato data načteme a zobrazíme. Obsah načtený ze souboru se přiřadí k textovému objektu. Všimněte si, že řádek obsahující příkaz `print()` není odsazen. Blok kódu

spojený s příkazem with je pouze jeden řádek, který načte soubor. Jakmile je blok kódu uzavřen, Python soubor automaticky uzavře.

```
from microbit import *
with open('udaje.txt') as soubor:
    text = soubor.read()
   display.scroll(text)
```

Podobně jako v klasických operačních systémech může Python na desce micro:bit také manipulovat se soubory nebo je vypisovat. K tomu slouží modul os. Funkce `listdir()` slouží k zobrazení seznamu souborů.

```
from microbit import *
import os
with open('udaje.txt', 'w') as soubor:
   soubor.write("Lorem ipsum")
with open('pokus.bin', 'w') as soubor:
   soubor.write("Lorem ipsum")
soubory = os.listdir()
display.scroll(str(soubory))
```

`print()` vypíše seznam souborů do konzoly vývojového prostředí.


```
from microbit import *
import os
with open('udaje.txt', 'w') as soubor:
   soubor.write("Lorem ipsum")
with open('pokus.bin', 'w') as soubor:
   soubor.write("Lorem ipsum")
```

```
soubory = os.listdir()
print(str(soubory))
```

metoda `remove()` slouží k odstranění souboru:

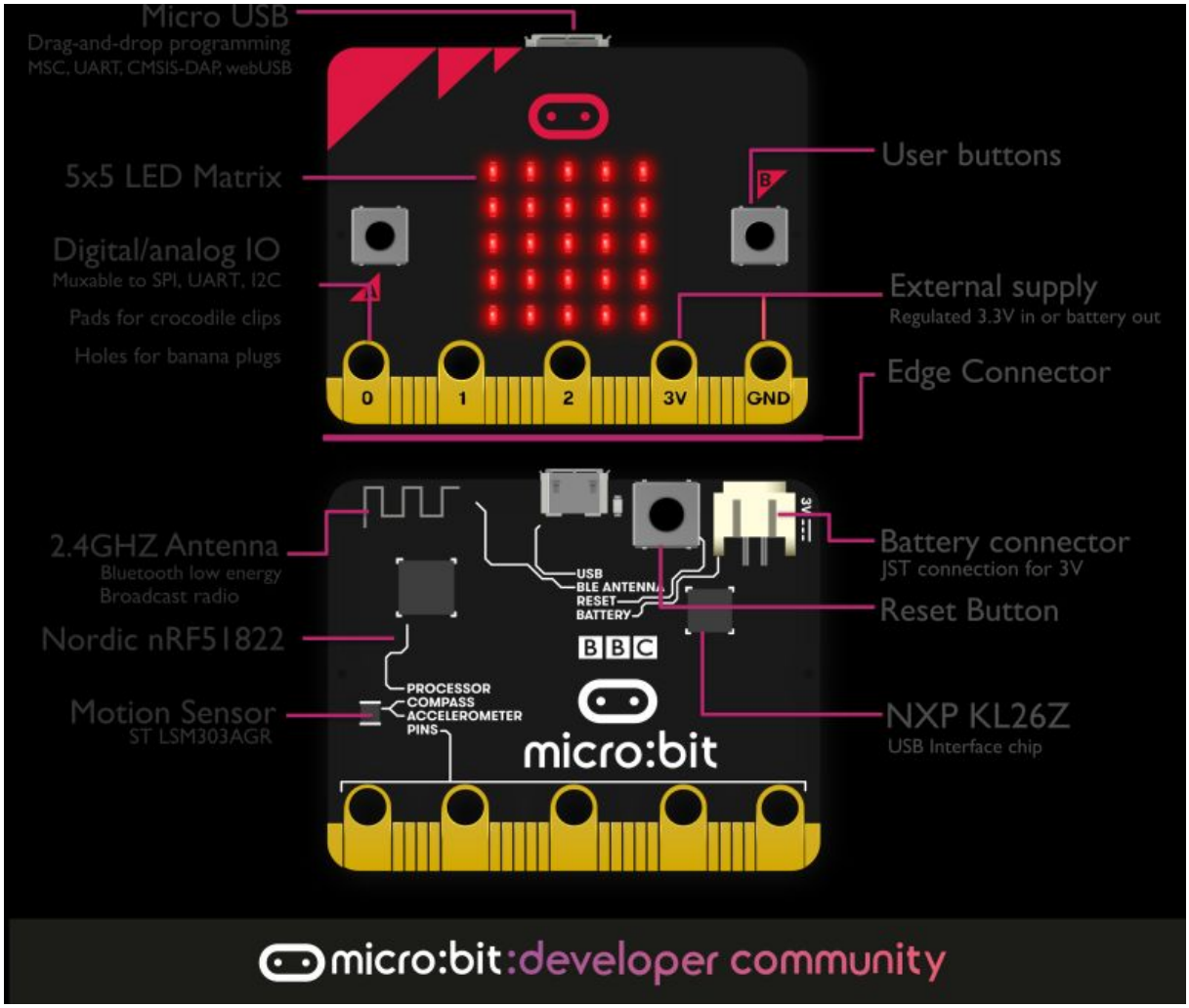
```
from microbit import *
import os
os.remove('pokus.bin')
```

můžete zjistit velikost souboru pomocí této metody:

```
from microbit import *
import os
with open('udaje.txt', 'w') as soubor:
   soubor.write("Lorem ipsum")
velikost = os.size('udaje.txt')
print(str(velikost))
```

Bezdrátová komunikace

Micro:bit podporuje dva typy bezdrátové komunikace - Bluetooth Low Energy (BLE) pro komunikaci s mobilními zařízeními a rádiový přenos pro komunikaci s jinými zařízeními BBC micro:bit. Rádiovou komunikaci lze naladit na různé kanály (číslované 0-83). Všichni zúčastnění se naladí na stejný kanál a všichni slyší, co na tomto kanálu vysílají ostatní. Můžete také nastavit vysílací výkon. Čím vyšší je výkon, tím větší je komunikační dosah. Vzhledem k tomu, že se používá stejný čip a stejný výkon jako pro komunikaci Bluetooth, bude maximální dosah v otevřeném prostoru přibližně 50 až 70 m. Při konfiguraci rádiové komunikace můžete nastavit délku zprávy, výchozí délka je nastavena na 32 bajtů, maximální délka může být 251 bajtů.



Můžete posílat binární zprávy s libovolným obsahem, při učení programování bude nejjednodušší posílat textové řetězce. Přijaté zprávy jsou řazeny do fronty s nastavitelnou velikostí, odkud jsou čteny. Výchozí velikost fronty jsou tři zprávy. Přečtení zprávy ji odstraní z fronty. Pokud se fronta zaplní, tj. zprávy jsou přijímány rychleji, než je váš kód dokáže přečíst, nové zprávy jsou ignorovány. Protože příkazy radio.RATE_250KBIT, radio.RATE_1MBIT nebo radio.RATE_2MBIT lze použít k nastavení poměrně vysoké rychlosti komunikace, od 250 kBitů za sekundu až po dva megabity za sekundu, může se fronta přijatých zpráv při špatně napsaném kódu poměrně rychle zaplnit. Ve výchozím nastavení je přenosová rychlost nastavena na 1 Mbit/s.

Rádiový modul umožňuje zadat několik parametrů a také adresu a skupinu. Skupina může mít hodnotu 0 až 255, ve výchozím nastavení je nastavena hodnota 0. Důležité je, že radiomodem odfiltruje přijaté zprávy, které neodpovídají vaší adrese a skupině. Proto je důležité předem nastavit adresu a skupinu, kterou bude vaše aplikace používat. Micro:bit samozřejmě stále přijímá zprávy broadcast pro jiné kombinace adres a skupin.

Pomocí kódu můžete odesílat a přijímat zprávy na vybraném kanálu:

```
from microbit import *
import radio

radio.on()
```

```
radio.config(channel=19)
radio.config(power=7)

my_message = "Voz ker minie ker voz nie krava o ker nos trie"

while True:
    radio.send(my_message)
    incoming = radio.receive()
    if incoming is not None:
        display.show(incoming)
        print(incoming)
        sleep(500)
```

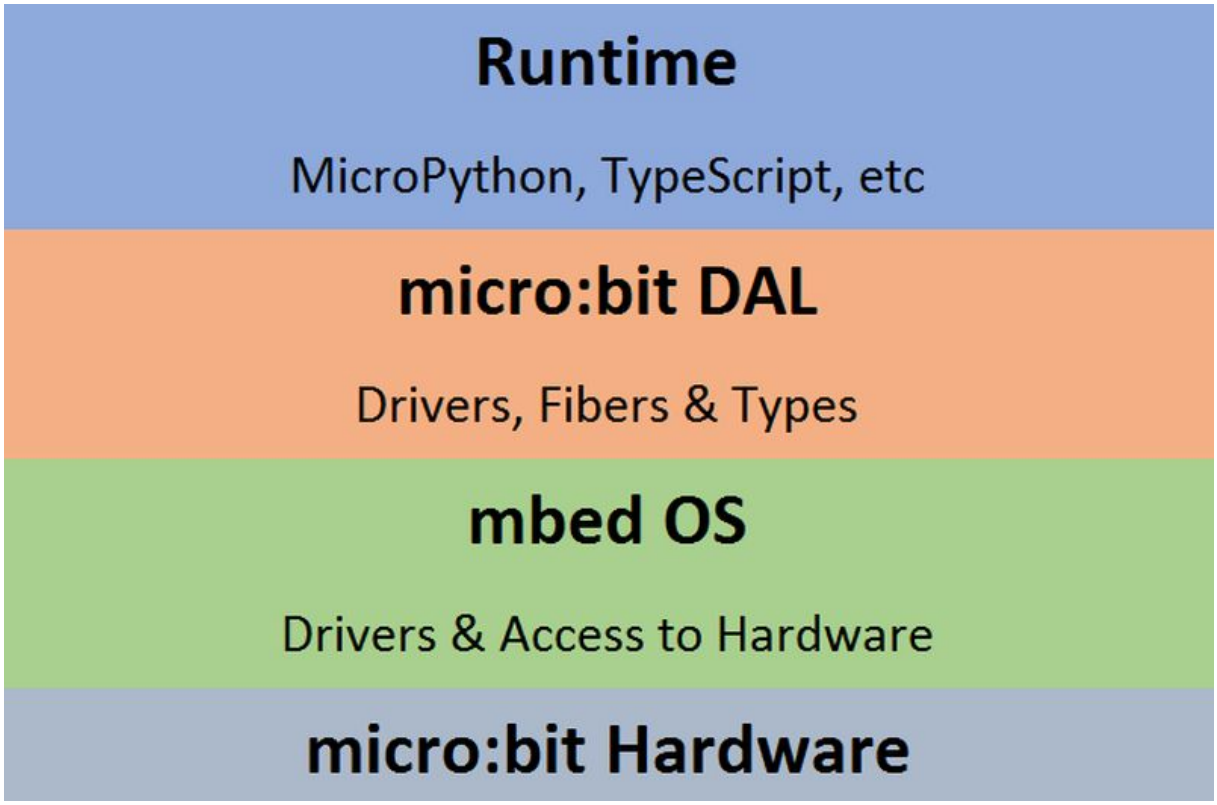
Pokud kanál nenastavíte, můžete použít hromadnou komunikaci na výchozím kanálu 7. Následující kód odešle zprávu. Na všech deskách, které ji obdrží, se zobrazí srdíčko. Každá deska však navíc s pravděpodobností 1:3 odešle stejnou zprávu, jejíž přijetí ostatní desky potvrdí, a některé z nich, teoreticky každá třetí, odešlou zprávu znovu.

```
from microbit import *
import radio
import random

radio.on()

while True:
    if button_a.was_pressed():
        # vysielanie.
        radio.send('x')
        # príjem.
    incoming = radio.receive()
    if incoming == 'x':
        display.show(Image.HEART)
        sleep(500)
        display.clear()
        # obcas odpovedz
        if random.randint(0, 3) == 0:
           sleep(500)
           radio.send('x')
```

Abychom to shrnuli trochu technicky, na počítači vývojáře nebo na webu, kde programujete online v MicroPythonu, se vytvoří soubor s příponou HEX, který obsahuje engine MicroPythonu a váš kód, který je k němu přibalen. Motor kód předem zkompiluje a spustí. Ještě hlouběji do hardwarové architektury se nad hardwarem nachází vrstva mini operačního systému mbed OS, který je určen (také) pro systémy s mikrokontroléry Cortex M0. Na vyšší vrstvě jsou ovladače hardwarových komponent a na horní vrstvě běhové prostředí TypeScript nebo Micro Python.



A nakonec ještě několik triků. Programovací jazyk Python je modulární a objektově orientovaný. Jistě jste si všimli, že všechny příklady začínají importem modulů pro hardwarovou implementaci MicroPythonu na desce BBC:micro:bit.

```
from microbit import *
```

Otevřete konzolu REPL v editoru Mu a zadejte příkaz

```
>>> import microbit
>>> dir (microbit)
```

Zobrazí se seznam importovaných modulů:

```
['__name__', 'Image', 'display', 'button_a', 'button_b', 'accelerometer', 'compass',
'i2c', 'uart', 'spi', 'reset', 'sleep', 'running_time', 'panic', 'temperature',
'pin0', 'pin1', 'pin2', 'pin3', 'pin4', 'pin5', 'pin6', 'pin7', 'pin8', 'pin9',
'pin10', 'pin11', 'pin12', 'pin13', 'pin14', 'pin15', 'pin16', 'pin19', 'pin20']
```

Můžete také například snadno zjistit, do jaké třídy funkce patří:

```
>>> type(display)
>>> type(pin0)
>>> type(pin9)
```

Odkazy

[1] – doprovodné video https://youtu.be/7Sdaop4Vh5Y
[2] - https://www.python.org
[3] - https://www.jetbrains.com/pycharm/
[4] - https://python.microbit.org/v/1.1
[5] - https://codewith.mu