iMYŠLENÍ!

ROBOTIKA PRO STŘEDNÍ ŠKOLY: PROGRAMUJEME MICRO:BIT POMOCÍ PYTHONU

Jiří Pech
Milan Novák

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Poděkování

Michal Kočer, Klára Pechová, Jan Wenig, Štěpán Bartoš, Karel Vlna, Vladimír Čížek, Jana Kalová

Jihočeská univerzita
v Českých Budějovicích
University of South Bohemia
in České Budějovice

ROBOTIKA PRO STŘEDNÍ ŠKOLY: PROGRAMUJEME MICRO:BIT POMOCÍ PYTHONU

Mgr. Jiří Pech, Ph.D., PhDr. Milan Novák

Recenzent:
Mgr. Martin Cápay, PhD.

Vydavatel:
Jihočeská univerzita v Českých Budějovicích, Pedagogická fakulta

Obálka:
Mgr. Pavel Pfauser

Rok vydání: 2020

Úvod

Tato učebnice si klade za úkol dát učitelům a žákům do rukou materiál, s jehož pomocí se naučí základy a principy elektrotechniky (robotiky) pomocí jednočipové vývojové platformy **BBC micro:bit**[1]. Současně nenásilnou formou vyučuje či opakuje programovací jazyk **Python** ve verzi **MicroPython** a jeho některé konstrukce.

Je lepší, pokud již studenti z předchozího studia mají nějaké zkušenosti s Pythonem, ale není to bezpodmínečně nutné.

Učebnice je určena především žákům netechnických oborů středních škol a učilišť a může být použita i pro práci v kroužcích elektrotechniky a programování i u mladších dětí (tak od sedmé třídy ZŠ).

Učebnice je stavěna tak, aby žáci v naprosté většině lekcí a příkladů vystačili pouze s micro:bitem a nemuseli sestavovat žádné obvody, nebo jen velmi jednoduché obvody pomocí kabelů s krokodýlky. To je třeba pro připojení reproduktoru (sluchátek) při přidání audio výstupu ve třetí kapitole a dále pro propojení dvou micro:bitů v páté kapitole. V závěrečné kapitole pak učebnice obsahuje volitelné části, kde se již obvody sestavují, ale tyto části je případně možné projít pouze teoreticky.

Poznámka – **Micro:bit** byl původně navržen pro děti ve věku 11–12 let. Předpokládalo se však programování v grafickém režimu **Microsoft MakeCode**, podobném Scratchi. V této učebnici použitý **Python** ve verzi **MicroPython** vyžaduje o něco zkušenější (a starší) uživatele.

Struktura učebnice

Učebnice je šířena jako pdf soubor s úvodem, závěrem a šesti výukovými kapitolami. Rovněž existuje pdf soubor s metodickými materiály pro učitele a pracovními listy.

K učebnici dále existují webové stránky na GitHubu:

https://github.com/jipech/PRIM-microbit

Zde jsou navíc umístěny všechny zdrojové kódy a ukázkové prezentace k jednotlivým hodinám. Vše je zde i ve formátu pro Word nebo PowerPoint, takže každý učitel si může vše upravit, jak uzná za vhodné.

Struktura na webu je následující:

Každá kapitola učebnice má čtyři nebo pět částí (adresářů):

Pro učitele – obsahuje kompletní text kapitoly včetně všech částí, návrhy výukových prezentací a průvodce hodinou s radami, jak vést výuku, seznamem potřebného materiálu, a odhad nutného času pro výuku. Ke všemu jsou k dispozici zdrojové kódy, učitel si vše může upravit dle svých potřeb.

Pro žáky – pracovní listy k jednotlivým hodinám. Až na výjimky se vejdou na jeden list papíru (oboustranně) a je možné je tak žákům vytisknout anebo dát k dispozici jako pdf soubor.

1 Micro:bit je ochranná známka Micro:bit Educational Foundation

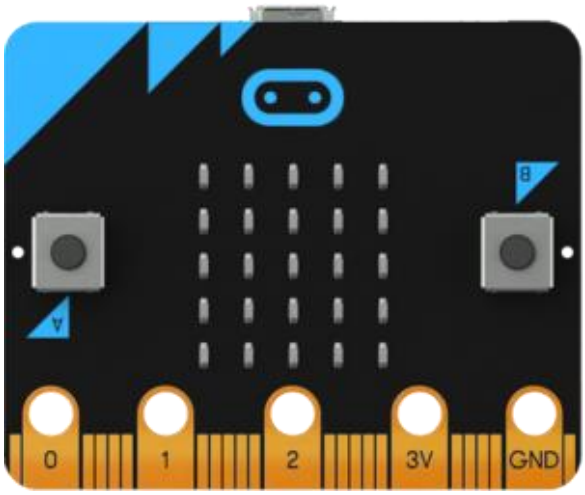
Samostudium – teoretický úvod k jednotlivým kapitolám, který opakuje a rozšiřuje probíranou látku a umožňuje žákům i učitelům hlouběji uchopit daná témata. Spojením těchto kapitol vznikl text nazvaný „ucebnice“. Pokud by se např. zajímali rodiče o to, co děti probírají, je možné jim tento text rovněž doporučit.

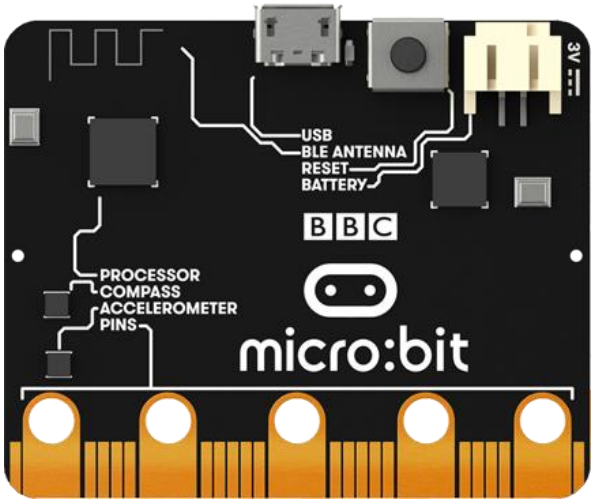
Zdrojové kódy **–** všech řešených příkladů. Díky nim zejména rozsáhlejší programy není nutné opisovat.

Různé – fotografie, videa, obvody atd.

CO JE TO MICRO:BIT

Micro:bit je open-source vývojová platforma vyvinutá ve Velké Británii za podpory BBC určený primárně pro výuku informačních technologií.





Obrázky převzaty z http://microbit.org

Micro:bit obsahuje:

- 5x5 matici LED diod
- dvě programovatelná tlačítka (označení A a B)
- kompas
- tříosý akcelerometr (gyroskop)
- 17 GPIO pinů, z nichž 3 jsou snadno přístupné pomocí např. krokodýlových kabelů

Dále umožňuje:

- zjišťovat intenzitu osvětlení, magnetického pole a teplotu
- komunikaci pomocí Bluetooth, která bohužel není možná pomocí MicroPythonu
- komunikaci pomocí radia
- snadné připojení sluchátek či repráku

Micro:bit je možné programovat pomocí několika programovacích jazyků – Microsoft **MakeCode** (grafický jazyk podobný Scratchi), **JavaScript** a **MicroPython**. Mezi MakeCode a JavaScriptem lze při programování online přepínat a pracovat tak střídavě v grafickém a textovém režimu. Na druhou stranu online uložené programy nemusí být při problémech se sítí dostupné.

V této učebnici bude vysvětlován pouze **MicroPython**, který umožňuje psát programy i offline a ukládat je lokálně.

CO BUDETE POTŘEBOVAT

Ve většině kapitol učebnice si vystačíte s následujícím vybavením:

BBC micro:bit, nejlépe pro každého studenta.

Poznámka – micro:bity se prodávají v různých barvách. Kromě barev se ale ničím neliší.

USB kabel s micro USB zakončením. Pokud nebude váš USB kabel fungovat, vyzkoušejte jiný. Stává se to. Obecně platí, čím kratší kabel, tím lépe bude fungovat pro datový přenos.

Micro:bit můžete napájet buď prostřednictvím zmíněného USB kabelu, anebo potřebujete **držák na baterie** (obvykle dvě AAA) s odpovídajícím kabelem.

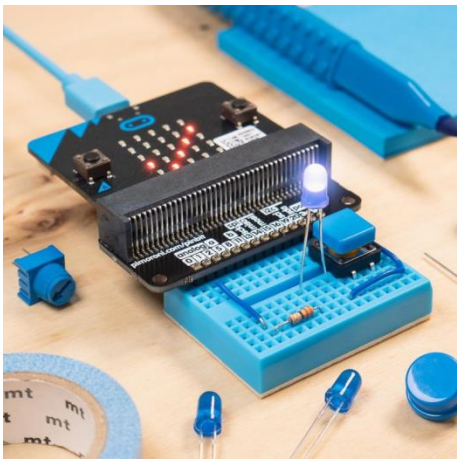
Počítač s libovolným operačním systémem (Windows, Linux, Mac OS, Chrome OS) a nainstalovaným **Mu** editorem (https://codewith.mu/). Pro Windows a Mac OS stáhněte mu editor z těchto stránek, v Linuxu obvykle existuje balík mu-editor a pro Chrome OS jej stáhněte ze Store. Lze mít tedy Mu i jako rozšíření pro prohlížeč Chrome. Alternativně, pokud nechcete (nemůžete) nic instalovat do vašich počítačů, lze pracovat i vzdáleně ve webovém prohlížeči na výše uvedených webových stránkách.

Rovněž lze takto pracovat i na stránkách https://python.microbit.org/v/2.0.

Pokud máte micro:bit V1 a chcete aby vydával zvuky, budete potřebovat libovolný **reproduktor** (sluchátka) vybavený jackem.

Dva vodiče: Pro drátové propojení dvou micro:bitů nebo připojení nějaké periférie se hodí několik vodičů vybavených na konci „krokodýlky“ – viz následující obrázek[2]. Pořiďte si těchto vodičů více – cca čtyři na jeden micro:bit v různých barvách, z toho jeden by měl být červený (pro plus) a jeden černý (pro mínus – zem), budou se vám dobře rozlišovat.

Předchozí bod samozřejmě můžete nahradit nepájivým polem a propojovacími vodiči. Potřebujete rovněž rozhraní pro propojení micro:bitu s nepájivým polem. Viz obrázek:



Pokud budete probírat i kapitolu 6 – Periférie, potřebujete též následující součástky:

Tříbarevnou diodu se společnou katodou (zem).

Teplotní čidlo pracující s napětím 3 V, např. TMP36.

Rovněž doporučuji si z internetu stáhnout dokument *BBC micro:bit MicroPython Documentation* v aktuální verzi. (https://microbit-micropython.readthedocs.io/en/latest/)

[2] Není-li uvedeno jinak jsou fotografie dílem autorů textu

PŘEDPOKLÁDANÉ VSTUPNÍ ZNALOSTI

Přestože, se snaží autoři o vysvětlení funkce použitých programových struktur jazyka MicroPython, je rozhodně lepší, pokud již studenti mají nějaké zkušenosti s programováním, např. pokud znají učebnici *Základy programování v jazyce Python pro střední školy* z projektu imysleni.cz (https://imysleni.cz). Některé struktury jsou rovněž blíže rozebrány v příloze této učebnice.

Co se týče znalostí elektroniky a zapojování obvodů, nejsou žádné speciální znalosti vyžadovány, vše je probíráno od základů.

ZDROJOVÉ KÓDY PROGRAMŮ

```
from microbit import *
while True:  # Nekonecny cyklus
    display.scroll("Ahoj svete")
    sleep(1000)
```

Abychom předešli různým nedorozuměním, přidáváme ukázku programu v MicroPythonu spolu s vysvětlením jeho struktury:

- Jednotlivé řádky buď začínají hned prvním písmenem příkazu (jako na řádcích 1 a 2), nebo jsou odsazené (jako řádky 3 a 4), anebo v programu pro optické oddělení částí mohou být i prázdné řádky.
- Prázdný řádek by neměl obsahovat žádný znak kromě Enteru (konec řádku).
- Řádek s programem začíná hned prvním znakem prvního příkazu.
- Je-li řádek odsazený (podmínka, cyklus atd.), pak editor Mu striktně vyžaduje odsazení o čtyři mezery. Každá další úroveň (vnořené cykly, podmínky atd.) je odsazena o další čtyři mezery. Druhá úroveň je tedy odsazena od začátku řádku o osm mezer, třetí o dvanáct atd.
- Je-li na nějakém řádku pokračovací komentář (za příkazem – jako na řádku 3), pak před jeho uvozujícím znakem # musí být právě dvě mezery a za ním nejméně jedna.
- Nemusíte se ale obávat, editor Mu vás bude hlídat, abyste vše psali správně. Stačí vždy stisknout tlačítko Check a dozvíte se, co máte špatně zapsáno.
- Ve výpisech zdrojových kódů neuvádíme číslování řádků, některé studenty to mátlo a čísla přidávali ke kódu.

VÝPIS NA LED DISPLAY

Co se naučíte

- Ovládat matici 5x5 LED diod na micro:bitu
- Zobrazit běžící text nebo jeden statický znak
- Zobrazit přednastavený obrázek
- Vytvořit jednoduchou animaci

Co budete potřebovat

- PC s nainstalovaným editorem Mu
- Propojovací USB kabel s micro USB koncovkou
- Micro:bit

Časová náročnost

3 až 5 vyučovacích hodin po 45 minutách

První program – Hello world

Základní ovládání editoru Mu již znáte z úvodu, proto přistoupíme přímo k programování.

Otevřete si na počítači editor Mu a stiskněte tlačítko New. Měli byste vidět následující text:

```
from microbit import *

# Write your code here :-)
```

Na řádku 1 se zavádí systémová knihovna, která obsahuje potřebné funkce a metody pro práci s micro:bitem. Tímto řádkem **musí** začínat všechny vaše programy.

Znak # na začátku třetího řádku znamená, že se jedná o komentář. Tento řádek můžete klidně smazat a kód programu psát místo něj. Je možné i psát hned na řádek 2.

Zkuste pro začátek následující kód:

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

Popis: řádek 2 znamená, že po displeji micro:bitu má běžet text uvedený v uvozovkách. Stiskněte tlačítko Save a text programu uložte do počítače. Programu bude automaticky přiřazena přípona py – rozpoznávací znamení pythonovských programů.

Nyní připojte svůj micro:bit pomocí USB kabelu k počítači. Počkejte asi pět vteřin a pak stiskněte tlačítko Flash. Vyčkejte, až přestane blikat žlutá LED dioda na micro:bitu, a pak byste měli vidět, jak přes displej micro:bitu přeběhne váš text.

Nyní můžete vyzkoušet následující modifikaci kódu (přepsat text, uložit a nahrát na micro:bit):

```
from microbit import *
while True:
    display.scroll("Ahoj svete")
    sleep(1000)
```

Popis: Na řádku 2 je nyní zaveden tzv. nekonečný cyklus. Jeho příkazy jsou odsazené o čtyři mezerníky od začátků řádků. Pozor – je třeba dodržet na každém řádku cyklu stejný počet mezer (může být i vyšší, ale pouze násobek čtyř). Některé pythonovské editory mají, co se tohoto pravidla týče syntaxi volnější a umožňují použití tabulátoru, některé naopak nikoliv, a proto je lepší se toto neučit. Na řádku 4 je pak příkaz `sleep(1000)` – čekej 1000 milisekund – čekej 1 sekundu.

Program v nekonečné smyčce vypíše text, počká jednu sekundu a opakuje.

Další příklady

Zadání: Napište program, který vypíše čísla od jedné do deseti pomocí příkazu `for` a pak skončí.

Řešení:

```
from microbit import *
for i in range(1, 11):
    display.scroll(i)
```

Vyzkoušejte si i jiné možnosti výpisu pomocí range. Plná syntaxe je: `range(pocatek, konec, krok)` , kde krok může být i záporný, pak ale by měl být počátek větší než konec. Více v příloze této učebnice.

Popis: Na řádku 2 je zaveden cyklus s pevným počtem opakování. Hodnota proměnné i se mění dle rozsahu intervalu `range(a, b)` od a do b-1. Chceme-li tedy vypsat čísla od 1 do 10, musíme příkaz zapsat takto. Za čárkou v intervalu musí být v editoru Mu mezera. Pozor, na konci řádku je dvojtečka, tady se také často dělá chyba. Na řádku tři je pak výpis aktuálního čísla dle iterace cyklu.

Zadání: Řešte předchozí příklad pomocí příkazu `while`

Řešení:

```
from microbit import *
i =1
while (i < 11):
    display.scroll(i)
    i = i + 1
```

Popis: Na řádku 2 do proměnné i přiřadíte hodnotu 1. Pozor, okolo znaku = jsou v editoru Mu vyžadovány mezery. Na řádku 3 je cyklus, který se opakuje, dokud je i menší než 11. Pozor, kolem znaku nerovnosti musí být mezery a na konci řádku je dvojtečka. Na řádku 5 zvyšujeme hodnotu proměnné i o jedničku. Pozor opět na chybějící mezery. U všech nutných mezer se jedná o syntaktická pravidla Mu, která v jiných editorech nemusí být vyžadována.

Zadání: Po dobu jedné vteřiny zobrazte na displeji písmeno X.

Řešení:

```
from microbit import *
display.show("X")
sleep(1000)
display.clear()
```

Popis: Na řádku 2 program zobrazí znak X. Na řádku 3 čeká program jednu sekundu (tedy přesněji 1000 milisekund). Příkaz na řádku 4 pak smaže displej.

Přednastavené obrázky

MicroPython obsahuje asi padesát připravených obrázků. Ukázka jejich použití je v následujícím kódu:

```
from microbit import *
display.show(Image.SAD)
sleep(1000)
display.show(Image.SMILE)
sleep(1000)
display.show(Image.HAPPY)
sleep(1000)
display.clear()
```

Zobrazení obrázků je na řádcích 2, 4 a 6. Jak je vidět, jedná se o konstanty začínající slovem Image.

Seznam všech obrázků naleznete v příloze A a v dokumentaci MicroPythonu pro micro:bit.

Příklad: Pomocí konstant obrázků `Image.HEART` a `Image.HEART_SMALL`, simulujte údery srdce.

Řešení:

```
from microbit import *
while True:
    display.show(Image.HEART)
    sleep(400)
    display.show(Image.HEART_SMALL)
    sleep(400)
```

Pauza `sleep(400)` je zvolena tak, aby frekvence odpovídala zhruba 75 úderům za minutu.

Vlastní obrázky

Příklad: Zobrazte na displeji obrázek rakety

```
from microbit import *
raketa = Image("00900:"
               "05550:"
               "05550:"
               "09990:"
               "90909:")
display.show(raketa)
```

Popis: Struktura (proměnná) raketa na řádcích 2 až 6 popisuje obrázek. Pětice čísel ukončených dvojtečkou a uzavřených do apostrofů popisuje vždy jeden řádek displeje shora dolů. Číslo pak znamená intenzitu světla od 0 (dioda nesvítí) po 9 (dioda svítí naplno). 5 tedy znamená svítí zhruba poloviční intenzitou. Na řádku 7 je pak příkaz pro zobrazení obrázku.

Je možná i syntaxe se zápisem Image do jednoho řádku:

```
from microbit import *
raketa = Image("00900:05550:05550:09990:90909:")
display.show(raketa)
```

Nyní si na základě tohoto příkladu sestrojíme pohyblivý obrázek startující rakety. Zdrojový kód je následující:

```
from microbit import *
raketa1 = Image("00900:"
                "05550:"
                "05550:"
                "09990:"
                "90909:")
raketa2 = Image("00900:"
                "05550:"
                "05550:"
                "09990:"
                "99999:")
raketa3 = Image("05550:"
                "05550:"
                "09990:"
                "99999:"
                "00000:")
raketa4 = Image("09990:"
                "99999:"
                "00000:"
                "00000:"
                "00000:")
raketa5 = Image("99999:"
                "00000:"
                "00000:"
                "00000:"
                "00000:")
raketa6 = Image("00000:"
                "00000:"
                "00000:"
                "00000:"
                "00000:")
raketa = [raketa1, raketa2, raketa3, raketa4, raketa5, raketa6]
display.show(raketa, delay = 500)
```

Na řádcích 2 až 33 je postupně vytvořeno šest obrázků, označených raketa1 až raketa6. Na předposledním řádku je z těchto obrázků sestavena struktura (proměnná) raketa. Tato struktura se nazývá list (seznam). Struktura raketa je pak na posledním řádku postupně zobrazována, s pauzou půl sekundy mezi jednotlivými snímky.

Práce s konkrétní diodou

Příklad: Sestrojte program, který bude náhodně rozsvěcet jednotlivé diody s různou intenzitou světla.

Řešení:

```
from microbit import *
import random
while True:
    x = random.randint(0, 4)
    y = random.randint(0, 4)
    intenzita = random.randint(0, 9)
    display.set_pixel(x, y, intenzita)
    sleep(10)
```

Program používá generátor náhodných čísel. Pro jeho použití je nutné načíst knihovnu `random` na řádku 2. Na řádcích 4 až 6 je pak tento generátor volán funkcí `random.randint`, která má dva parametry (a, b) a vrací náhodné celé číslo z uzavřeného intervalu <a,b>. Zde je postupně získána x-ová a y-ová souřadnice rozsvěcené diody a intenzita světla dané diody.

Funkce na řádku 7 `display.set_pixel` má tři parametry (x, y, intenzita), získané v předchozím kroku, a nastavuje podle nich na souřadnicích x (sloupec) a y (řádek) diodu na intenzitu (0 až 9). Bod (0, 0) je vlevo nahoře, vpravo dole pak (4, 4). Opět platí, že intenzita je 0 (nesvítí) až 9 (svítí naplno).

Použití funkce `sleep` je nutné, jinak dochází k příliš rychlému „blikání“.

Příklad: Upravte předchozí zadání tak, že budete nastavovat pouze dvě úrovně intenzity (0 a 9).Budete náhodně vybírat souřadnice a pokud dioda na dané souřadnici nebude svítit, tak ji rozsvítíte, a naopak pokud svítí, zhasnete jí.

Řešení:

```
from Microbit import *
import random
while True:
    x =random.randint(0, 4)
    y =random.randint(0, 4)
    if (display.get_pixel(x, y)):
        display.set_pixel(x, y, 0)
    else:
        display.set_pixel(x, y, 9)
    sleep(10)
```

Popis: Zde pouze vybíráme náhodně souřadnice diody. Na řádku šest je použita funkce `display.get_pixel(x, y)`. Ta vrací hodnotu svícení dané diody.

Syntaxe (zjednodušená) příkazu if je následující:

```
if (podmínka):
    blok příkazů 1
else:
    blok příkazů 2
```

Je-li podmínka splněná (její hodnota je True) provede se blok příkazů 1 jinak blok příkazů 2. Jako splněná je podmínka vyhodnocena i tehdy pokud je výsledkem podmínky číslo různé od nuly. Toho s úspěchem využíváme zde, neboť funkce `get_pixel(x, y)` v případě, že daná dioda svítí vrátí hodnotu 9. V opačném případě vrátí 0 a tedy podmínka není splněna a provede se část za `else`.

Pozor, druhá úroveň odsazení musí být opět násobek čtyř, je tedy osm mezer.

PŘÍLOHA – SEZNAM PŘIPRAVENÝCH OBRÁZKŮ

Význam většiny obrázků je podle nás jasný z jejich názvu, pokud si nejste jisti, určitě obrázky vyzkoušejte.

- `Image.HEART`
- `Image.HEART_SMALL`
- `Image.HAPPY`
- `Image.SMILE`
- `Image.SAD`
- `Image.CONFUSED`
- `Image.ANGRY`
- `Image.ASLEEP`
- `Image.SURPRISED`
- `Image.SILLY`
- `Image.FABULOUS`
- `Image.MEH`
- `Image.YES`
- `Image.NO`
- `Image.CLOCK12`, `Image.CLOCK11`, `Image.CLOCK10`, `Image.CLOCK9`, `Image.CLOCK8`, `Image.CLOCK7`, `Image.CLOCK6`, `Image.CLOCK5`, `Image.CLOCK4`, `Image.CLOCK3`, `Image.CLOCK2`, `Image.CLOCK1`
- `Image.ARROW_N`, `Image.ARROW_NE`, `Image.ARROW_E`, `Image.ARROW_SE`, `Image.ARROW_S`, `Image.ARROW_SW`, `Image.ARROW_W`, `Image.ARROW_NW`
- `Image.TRIANGLE`
- `Image.TRIANGLE_LEFT`
- `Image.CHESSBOARD`
- `Image.DIAMOND`
- `Image.DIAMOND_SMALL`
- `Image.SQUARE`
- `Image.SQUARE_SMALL`
- `Image.RABBIT`

- Image.COW
- Image.MUSIC_CROTCHET
- Image.MUSIC_QUAVER
- Image.MUSIC_QUAVERS
- Image.PITCHFORK
- Image.XMAS
- Image.PACMAN
- Image.TARGET
- Image.TSHIRT
- Image.ROLLERSKATE
- Image.DUCK
- Image.HOUSE
- Image.TORTOISE
- Image.BUTTERFLY
- Image.STICKFIGURE
- Image.GHOST
- Image.SWORD
- Image.GIRAFFE
- Image.SKULL
- Image.UMBRELLA
- Image.SNAKE

2 PRÁCE S TLAČÍTKY

Co se naučíte

- Ovládat obě programovatelná tlačítka
- Psát programy reagující na stisk tlačítka
- Význam logických spojek and a or

Co budete potřebovat

- PC s nainstalovaným editorem Mu
- Propojovací USB kabel s micro USB koncovkou
- Micro:bit

Časová náročnost

Jedna vyučovací hodina 45 minut

Micro:bit obsahuje celkem tři tlačítka. Tlačítko umístěné na zadní straně mezi vstupy pro USB kabel a napájecí kabel je reset. To nás teď nebude zajímat. Na přední straně jsou umístěna dvě programovatelná tlačítka, A a B, jimž se budeme v této kapitole věnovat.

Začněte jednoduchým příkladem:

```
from microbit import *
while True:
    if button_a.is_pressed():
        display.show(Image.HAPPY)
    if button_b.is_pressed():
        display.show(Image.SAD)
    sleep(100)
    display.clear()
```

Práce s tlačítky je ukázána na řádcích 3 a 5. Jedná se vlastně o dotaz, zda je tlačítko stisknuté. MicroPython má, jak vidíte připraveny dvě proměnné `button_a` a `button_b`. Funkce `button_a.is_presed()` vrací 1, pokud je tlačítko stisknuté, jinak vrací 0. Existuje ještě funkce `button_a.was_presed()`, která testuje, zda tlačítko bylo stisknuté od minulé kontroly nebo od zapnutí micro:bitu. Obdobné funkce jsou i pro tlačítko b.

Chcete-li testovat současný stisk obou tlačítek použijte následující programové konstrukci:

```
from microbit import *
while True:
    if(button_a.is_pressed())and(button_b.is_pressed()):
        display.show(Image.HEART)
        sleep(100)
        display.clear()
```

Mezi oběma testovacími funkcemi na řádku 3 je použita logická spojka `and`, která znamená, že celkově podmínka platí, pouze pokud platí obě dílčí podmínky – jsou stisknuta obě tlačítka současně.

Naopak, pokud testujete, zda je stisklé libovolné tlačítko (A nebo B), použijte následující konstrukci se spojkou `or` (nebo):

```
from microbit import *
while True:
    if ((button_a.is_pressed()) or (button_b.is_pressed()):
        display.show(Image.HEART)
    sleep(100)
    display.clear()
```

Kromě uvedených funkcí `is_presed` a `was_presed`, je pro objekty `button_a` a `button_b` definována ještě funkce `get_presses()`. Tato funkce zjistí počet stisknutí tlačítka od posledního testování a nastaví jej na nulu.

Následující příklad vyčká po zapnutí (nebo stisku reset) micro:bitu deset sekund a pak zobrazí počet stisků tlačítka A během této doby (od zapnutí nebo resetu):

```
from microbit import *
sleep(10000)
display.show(button_a.get_presses())
```

3 HUDBA

Co se naučíte

- Připojit k micro:bitu reproduktor, buzzer nebo sluchátka
- Přehrát přednastavený zvuk
- Naučíte micro:bit mluvit
- Vytvořit vlastní melodii

Co budete potřebovat

- PC s nainstalovaným editorem Mu
- Propojovací USB kabel micro USB koncovkou
- Micro:bit
- Dva vodiče nejlépe s krokodýlky na obou koncích
- Reproduktor nebo sluchátka s jackem, popřípadě piezzo buzzer

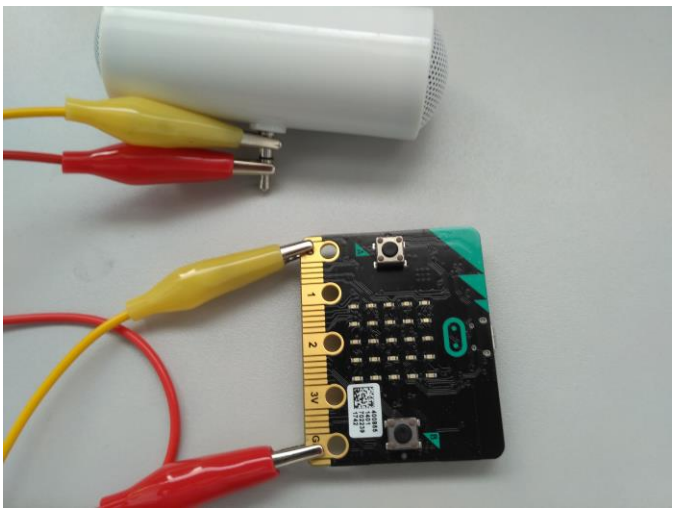
Časová náročnost

Dvě až čtyři (v případě samostatné práce) vyučovací hodiny pro 45 minutách.

Microbit V2 vydaný v závěru roku 2020 již obsahuje piezzo buzzer a hardwarové konstrukce popsané v této kapitole tak již nejsou třeba. Všechny zde uvedené programy nadále fungují beze změny.

Připojení audio výstupu

Micro:bit nemá přímý audio výstup, přesto je připojení externího reproduktoru velmi snadné. Budete nyní potřebovat dva vodiče nejlépe opatřené na koncích krokodýly. Ty na dolní straně micro:bitu připněte jeden na GND a druhý na 0. Druhý konec vodičů připojte na jack libovolného reproduktoru či sluchátek. Nezáleží na tom, který vodič připojíte, na který kontakt na jacku. Má-li váš jack tři vstupy, pak jeden z vodičů připojte na prostřední a druhý na libovolný z krajních vodičů. Má-li čtyři vstupy, pak by měly fungovat buď oba krajní nebo oba vnitřní (možná budete muset



trochu experimentovat). Také můžete použít jako výstup piezzo buzzer, pak prostě připojíte každý vodič k jednomu z pinů. Vše popisuje obrázek.

Počítejte s tím, že v případě kvalitních reproduktorů může být výstup poměrně hlasitý a nastavte výstup hlasitosti na repráku nebo sluchátkách na nižší úroveň.

Na základě zkušeností navíc můžeme říct, že micro:bit hraje (a často lépe) i při zapojení vodiče prvního vodiče k pinu 1 nebo 2. Druhý vodič každopádně ponechte na GND.

Přehrávání připravených melodií

MicroPython obsahuje asi dvacet předem připravených melodií, jejichž seznam najdete v dokumentaci MicroPythonu a příloze této kapitoly. Ukázka použití je v následujícím příkladu:

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

Tento zvuk je poměrně dlouhý, takže budete mít čas správně nastavit reproduktor, nasadit si sluchátka apod.

Všimněte si, že na řádku 2 je nutné zavést knihovnu pro přehrávání hudby. Samotný příkaz pro přehrání melodie je pak na řádku 3.

Připravené melodie lze dobře kombinovat s připravenými obrázky, které jsme již probírali. To nám dobře demonstruje následující příklad:

```
from microbit import *
import music
while True:
    if button_a.is_pressed():
        display.show(Image.HAPPY)
        music.play(music.POWER_UP)
    if button_b.is_pressed():
        display.show(Image.SAD)
        music.play(music.POWER_DOWN)
    display.clear()
```

Význam jednotlivých příkazů už by vám měl být jasný a proto neuvádíme žádný další popis.

Micro:bit mluví

Micro:bit umí i mluvit. Naneštěstí pro nás pouze anglicky. Knihovna pro mluvení je navíc zatím označena jako vývojová, takže se můžete potkat s chybami. Mluvení je velmi jednoduché:

```
from microbit import *
import speech
speech.say("Hello",speed = 100)
```

Na řádku 2 se zavádí knihovna pro mluvení a na řádku 3 je zadán vlastní příkaz pro řeč. Zde micro:bit pozdraví. Parametr `speed = 100` je nepovinný. Defaultní hodnota je 72, ale přijde

nám, že při této hodnotě mluví micro:bit příliš rychle. Čím vyšší číslo, tím je řeč pomalejší a naopak. Nezapomeňte pro slova použít anglickou transkripci např. "Yoseph" pro Josef.

Dokumentace doporučuje zapojit pro řeč sluchátka (repráky) mezi piny 0 a 1 (a ne 0 a GND jako u hudby). A skutečně zvuk je v tomto případě obvykle silnější a čistší.

Přehrání not

Micro:bit dovede přehrát noty. Následující program přehraje melodii „Ovčáci, čtveráci“. Zápis programu trochu připomíná vytváření animovaných obrázků.

```
from microbit import *
import music
nota = ["C4:4", "R:1", "E4:4", "R:1", "G4:4", "R:4","C4:4",
        "R:1", "E4:4", "R:1", "G4:4", "R:4",
        "E4:2", "R:1", "E4,2", "R:1", "D4:2", "R:1", "E4:2",
        "R:2", "F4:2", "R:1", "D4:2", "R:1",
        "E4:2", "R:1", "E4,2", "R:1", "D4:2", "R:1", "E4:2",
        "R:2", "F4:2", "R:1", "D4:2", "R:1",
        "E4:4", "R:1", "D4:4", "R:1", "C4:4"]
music.play(nota)
```

Proměnná nota je seznam (list) s významem zápis jednotlivých tónů. Např. C4:4 znamená nota C ve čtvrté oktávě (0 – nejnižší, 8 – nejvyšší) o délce 4. Nota R znamená pauzu (rest). Příkaz music.play(nota) pak daný záznam přehraje.

PŘÍLOHA – SEZNAM PŘIPRAVENÝCH MELODIÍ

Doporučujeme melodie vyzkoušet, na první pohled nemusí být jasné o co jde.

```
music.DADADADUM
music.ENTERTAINER
music.PRELUDE
music.ODE
music.NYAN
music.RINGTONE
music.FUNK
music.BLUES
music.BIRTHDAY
music.WEDDING
music.FUNERAL
music.PUNCHLINE
music.PYTHON
music.BADDY
music.CHASE
music.BA_DING
music.WAWAWAWAA
music.JUMP_UP+
music.JUMP_DOWN
music.POWER_UP
music.POWER_DOWN
```

4 POLOHA

Tato kapitola se věnuje orientaci micro:bitu a jeho pohybu v prostoru

Co se naučíte

- Pracovat s vestavěným akcelerometrem, pochopit jeho možnosti a využít jej
- Používat gesta
- Pracovat s vestavěným kompasem
- Používat detekci magnetického pole

Co budete potřebovat

- PC s nainstalovaným editorem Mu
- Propojovací USB kabel s micro USB koncovkou
- Micro:bit
- Dva vodiče nejlépe s krokodýlky na obou koncích
- Reproduktor nebo sluchátka s jackem, popřípadě piezzo buzzer

Časová náročnost

Čtyři vyučovací hodiny po 45 minutách

Akcelerometr

Akcelerometr (Accelerometer) je zařízení pro zjištění aktuální orientace, směru pohybu, zrychlení, popř. volného pádu nebo třesení.

Micro:bit obsahuje přímo integrovaný akcelerometr. Jak je patrno z uvedené definice můžete v našem programu použít akcelerometr pro zjištění polohy micro:bitu nebo typu aktuálního pohybu. S velkým úspěchem jej lze využít v případě, že micro:bit použijete jako ovladač pro jiné zařízení (např. vozítko) řízené jiným micro:bitem. Jak spolu mohou komunikovat vám prozradí následující kapitola.

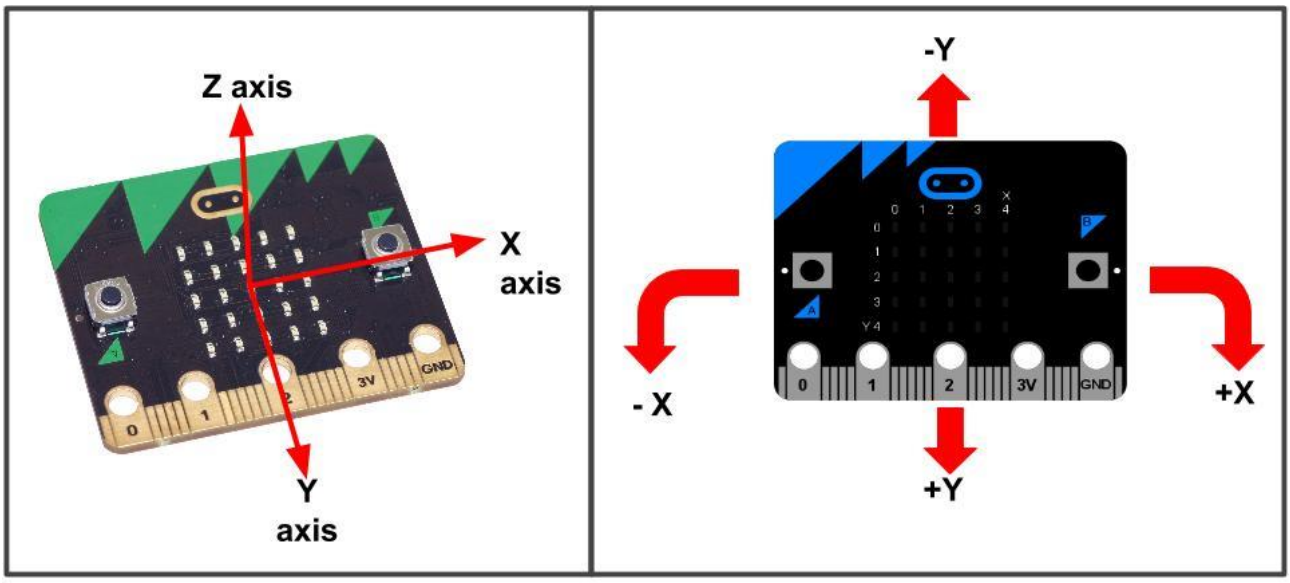
Nejprve se naučíte, jak akcelerometr využít pro určení orientace micro:bitu v prostoru. Pro zjištění této pozice slouží trojice příkazů:

`accelerometer.get_x()`, `accelerometer.get_y()` a `accelerometer.get_z()`

Tyto tři příkazy vrací hodnoty od -2000 do + 2000.

Pokud micro:bit položíte na stůl diodami nahoru konektory k sobě, pak směr doprava doleva je osa x a směr od sebe k sobě osa y. Náklon vpravo je dán kladnými hodnotami u `accelerometer.get_x()` a vlevo pak zápornými. Náklon vpřed nám dá kladné znaménko u `accelerometer.get_y()` a náklon vzad záporné.

Pokud micro:bit postavíte na konektory diodami k sobě, pak náklon k sobě vám dá kladné hodnoty u `accelerometer.get_z()` a náklon vzad záporné. Natočení vpravo a vlevo po hraně pak obdobně u `accelerometer.get_x()`.



Poloha os a význam plus a minus ve směru os.

Zdroj: http://fibasile.fabcloud.io/microbit-notebook/accelerometer/

Vše si můžete dobře vyzkoušet na následujícím příkladu:

```
from microbit import *
mez = 400
while True:
    naklon = accelerometer.get_x()
    if naklon > mez:
        display.show("P")
    elif naklon < -mez:
        display.show("L")
    else:
        display.show("-")
```

Tento program po nahrání do micro:bitu zobrazuje pozici micro:bitu vůči ose x. Je-li micro:bit vodorovně, zobrazuje na displeji znak -, pokud micro:bit nakloním doprava zobrazí P, pokud doleva zobrazí L. Proměnná `mez`, určuje, jak velký sklon bereme jako naklonění vpravo či vlevo.

Nejprve zkuste experimentovat s hodnotou proměnné mez a zkoumejte, jak je nutné naklonit micro:bit pro projevení změny.

Nyní změňte hodnotu x postupně na y a z a prohlédněte si, jak reaguje hodnota proměnných na jednotlivé polohy micro:bitu. Protože zde nemá smysl pravá a levá strana, nahraďte např. hodnoty (P, -, L) za (+, 0, -). Ideálně máte-li k dispozici tři micro:bity zkuste do každého nahrát program pro jednu osu a zkoumejte, jak se mění hodnoty. Jak sami vidíte lze takto velice dobře sestrojit bezdrátový ovladač, reagující na změnu orientace v prostoru.

V následujícím příkladu si představíme jednoduchý simulátor nástroje *theremin*. Jedná se o nástroj, na který se hraje, aniž by se ho hráč dotýkal, kdy jednou rukou určuje výšku tónu a druhou dobu trvání. Takovýto nástroj postavit nedokážeme, ale budeme jej simulovat tak, že náklonem micro:bitu ve směru osy x budeme určovat výšku tónu a náklonem ve směru osy y délku tónu.

Pro jednoduchost se spokojíme s tóny v rozsahu jedné oktávy od C4 do C5 a čtyřmi různými délkami 1, 2, 4 a 8 (viz předchozí kapitola). K micro:bitu V1 si připojte repráčky nebo sluchátka na piny 0 a GND a nahrajte následující program:

```
from microbit import *
import music
while True:
    x = accelerometer.get_x()
    y = accelerometer.get_y()
    if (x < -1000):
        ton = "C4"
    elif (x < -700):
        ton = "D4"
    elif (x < - 400):
        ton = "E4"
    elif (x < -100):
        ton = "F4"
    elif (x < 200):
        ton = "G4"
    elif (x < 500):
        ton = "A4"
    elif (x < 800):
        ton = "B4"
    else:
        ton = "C5"
    if (y < -500):
        nota = ton
    elif (y < 0):
        nota = ton + ":2"
    elif (y < 500):
        nota = ton + ":4"
    else:
        nota = ton + ":8"
    music.play(nota)
```

Všimněte si, jak postupně vzniká řetězec, tvořený výškou a délkou noty, který je vzápětí přehrán. Experimentujte se zvětšením (zmenšením) rozsahu výšek a délek tónů.

Gesta

Gesta (gesture) u micro:bitu jsou nějaké pohybové činnosti, které se s ním v daném okamžiku dějí. Může se jednat o otočení požadovaným směrem, zrychlení, zatřesení a volný pád. Seznam gest naleznete v příloze k této kapitole.

Začneme jednoduchým programem, který zobrazí na displeji úsměv, pokud je micro:bit otočen displejem vzhůru (face_up) a naopak smutný obličej, je-li tomu jinak:

```
from microbit import *
while True:
    gesture = accelerometer.current_gesture()
    if gesture == "face up":
        display.show(Image.HAPPY)
    else:
        display.show(Image.ANGRY)
```

Můžete zkusit i jiné gesto z dostupných. Pouze pozor na volný pád, aby nedošlo k poškození micro:bitu. Abyste stihli gesto, můžete obrázek nechat na displeji nějakou dobu např. 3 sekundy.

Rovněž lze použít následující metody. GESTO nahraďte libovolným gestem dle přílohy.

`microbit.accelerometer.current_gesture()` – vrací jméno právě použitého gesta

`microbit.accelerometer.is_gesture(GESTO)` – vrací True nebo False podle toho, zda právě probíhá GESTO

`microbit.accelerometer.was_gesture(GESTO)` – vrací True nebo False podle toho, zda od posledního dotazu bylo GESTO

`microbit.accelerometer.get_gestures()` – vrací seznam gest od posledního zjišťování gesta nebo od spuštění programu

Následující příklad vám poskytne věšteckou odpověď na problém, který vás trápí. Myslete usilovně na problém, s jehož řešením si nevíte rady, pak zatřeste micro:bitem a on vám poradí.

```
from microbit import *
import random
odpovedi = [
    "Jiste",
    "Urcite",
    "Pravdepodobne",
    "To vypada dobre",
    "Ano",
    "Zeptej se pozdeji",
    "Ted nevim",
    "Nelze urcit",
    "Urcite ne",
    "Ne",
    "Nikdy",
]
while True:
    display.show('8')
    if accelerometer.was_gesture("shake"):
        display.clear()
        sleep(1000)
        display.scroll(random.choice(odpovedi))
        sleep(10)
```

Samozřejmě si upravte odpovědi dle sebe. Jedná se vlastně o úpravu hry magic 8ball. Proto v klidovém stavu zobrazuje micro:bit číslo 8.

Kompas

Micro:bit obsahuje integrovaný kompas, který současně lze použít jako čidlo intenzity magnetického pole.

Základní použití si můžete ukázat na následujícím programu:

```
from microbit import *
compass.calibrate()
while True:
    display.scroll(compass.heading())
    sleep(1000)
```

Před použitím jakýchkoliv funkcí týkajících se magnetického pole je pro správnou funkci nutná kalibrace micro:bitu.

Pro kalibraci je nutno otáčet micro:bitem tak dlouho, než displej zaplníme svítícími diodami. Na Micro:bitu vám vždy před kalibrací proběhne instrukce, jak postupovat. Po zaplnění displeje je třeba několik vteřin (cca. 5) počkat, než se na displeji objeví smajlík.

Pokud přesto kompas micro:bitu ukazuje nečekanou hodnotu, stiskněte na něm klávesu reset a opakujte kalibraci. To obvykle pomůže.

Micro:bit položte na rovnou plochu nebo jej držte co nejvíce rovně. Micro:bit nyní ukáže na displeji azimut (úhel svírající se směrem na sever ve směru hodinových ručiček). Směr azimutu je přímo od displeje nahoru.

V astronomii se úhel určuje od jihu ve směru hodinových ručiček. Upravte program tak aby ukazoval astronomický azimut.

Vyzkoušejte si rovněž, co se stane, když kolem micro:bitu pohybujeme magnetem nebo zmagnetizovaným předmětem (nůžky, šroubovák …).

Nyní program upravíme tak, aby micro:bit ukazoval symboly světových stran S, V, J, Z. Za sever budeme považovat intervaly úhlů (0,45) a (316, 359), za východ (46, 135), za jih (136, 225) a za západ (226, 315).

```
from microbit import *
compass.calibrate()
while True:
    uhel = compass.heading()
    if (uhel < 46):
        display.show("S")
    elif (uhel < 136):
        display.show("V")
    elif (uhel < 226):
        display.show("J")
    elif (uhel < 316):
        display.show("Z")
    else:
        display.show("S")
    sleep(1000)
```

Program nyní upravíme tak, aby ukazoval na displeji micro:bitu směr na sever. Využijeme při tom obrázek Image.ALL_CLOCKS. Jedná se vlastně o pole dvanácti obrázků, které se volají Image.ALL_CLOCKS[uhel], kde uhel je číslo od 0 do jedenácti. Na displeji pak ukazují čáru (lépe křivku) od středu micro:bitu ve směru malé hodinové ručičky ukazující danou hodinu o hodnotě proměnné uhel. Pozor namísto 12 směr nahoru ukazuje hodnota 0. Můžete si to ověřit následujícím programem:

```
from microbit import *
for uhel in range(0, 12):
    display.show(Image.ALL_CLOCKS[uhel])
    sleep(1000)
    display.clear()
```

Nyní zůstává otázkou, jak zajistit, aby na displeji byl zobrazen směr k severu. Máme celkem dvanáct poloh ručičky (ukazatele). To znamená 30° na každou polohu, např. sever je od -15°do 15°. Pozici ručičky pak dostaneme na první pohled krkolomným vzorcem:

```
Pozice = ((15 - Azimut) // 30) % 12
```

Máme zde pro vás možná neznámé operace // a %. Jejich význam je následující:

// je celočíselné dělení – dělení beze zbytku. Např. `7 //2 = 3` a `7 // 3 = 2`.

% je zbytek po dělení. Např. `7 % 2 = 1`, `7 % 3 = 1`.

Výpočtem (`15 - Azimut) // 30` dostaneme číslo od 0 do -11. (pro 0°dostaneme 0 a pro 359° dostaneme -11). Tyto hodnoty převedeme na kladné pomocí operace % 12 a dostaneme správný index pro pozici ukazatele.

Vzorec lze samozřejmě upravit. Například místo 15 lze použít hodnotu 375 a místo % 12 lze použít + 12. V těchto případech, je ale výsledek v intervalu 1 až 12 a hodnotu 12 je třeba převést na 0 (hodnota indexu 12 je mimo rozsah).

Program tak vypadá následovně:

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

Intenzita magnetického pole

Na závěr si ukážeme ještě další vlastnost, kterou má kompas. Umožňuje rovněž měřit hodnotu magnetického pole v jednotkách nT (nano tesla).

Můžeme tedy napsat následující program, který sleduje, zda magnetické pole v okolí překročí určitou hodnotu (zde 5000 nT) a pak zobrazit na určitou dobu smajlík.

```
from microbit import *
hodnota = 5000
compass.calibrate()
pocatek = compass.get_field_strength()
while True:
    sleep(100)
    sila = compass.get_field_strength()
    if abs(sila - pocatek) > hodnota:
        display.show(Image.HAPPY)
        sleep(3000)
        display.clear()
```

Vyzkoušejte, v okolí, kterých přístrojů se nachází magnetické pole. Např. počítače, mobily, tablety. Rovněž také zmagnetizované nůžky, nože anebo šroubováky.

S pomocí tohoto programu můžete předvést následující kouzlo. V ruce ukryjete malý silný magnet a přejedete touto rukou nad micro:bitem. Micro:bit zobrazí úsměv. Řekněte neznalému, že micro:bit se rozsvítí pouze v okolí lidí s magnetickým potenciálem a nechte je pohyb zopakovat. Bez magnetu samozřejmě k ničemu nedojde.

PŘÍLOHA – SEZNAM PŘIPRAVENÝCH GEST

- up – Micro:bit je otočen nahoru
- down – Micro:bit je otočen dolů
- left – Micro:bit je otočen vlevo
- right – Micro:bit je otočen vpravo
- face up – Micro:bit leží otočen diodami nahoru
- face down – Micro:bit leží otočen diodami dolů
- freefall – Micro:bit padá volným pádem
- 3g – Micro:bit se pohybuje zrychlením 3g
- 6g – Micro:bit se pohybuje zrychlením 6g
- 8g – Micro:bit se pohybuje zrychlením 8g (pravděpodobně se nalézá ve startující raketě)
- shake – Micro:bitem je třeseno

5 SÍŤ

Tato kapitola se věnuje propojení dvou nebo více Micro:bitů pomocí sítě ať drátové či bezdrátové

Co se naučíte

- Základní principy sítí
- Propojit dva Micro:bity pomocí drátu a přenést informaci
- Totéž bezdrátově, pomocí rádia

Co budete potřebovat

- PC s nainstalovaným editorem Mu
- Propojovací USB kabel s micro USB koncovkou
- Micro:bit
- Dva vodiče nejlépe s krokodýlky na obou koncích

Časová náročnost

Jedna až čtyři hodiny po 45 minutách v závislosti na tom, zda a jak podrobně chcete probírat drátové propojení.

Počítačové sítě

Než se pustíme do práce se sítí uvedeme si pár teoretických definic, abychom později lépe porozuměli tomu, co stavíme a programujeme.

Počítačová síť – umožňuje vzájemné propojení dvou nebo více počítačů, které si tak mohou vzájemně posílat zprávy.

Počítačové sítě dělíme dle přenosového média na:

- Drátové
- Bezdrátové (vzduchem)
    - Wi-Fi
    - Bluetooth
    - Radio (rádiové vlny)
    - Infra

My budeme micro:bity propojovat, jak *drátově* pomocí kabelů, tak *bezdrátově* pomocí radia. Micro:bit sice obsahuje i Bluetooth přijímač a vysílač, ale jak bude dále vysvětleno v MicroPythonu jej nelze použít.

Dle směru vysílání dělíme sítě na:

Simplexové – vysílání směřuje pouze jedním směrem, na jedné straně se nachází vysílač (sender nebo transmitter) a na druhé přijímač (receiver).

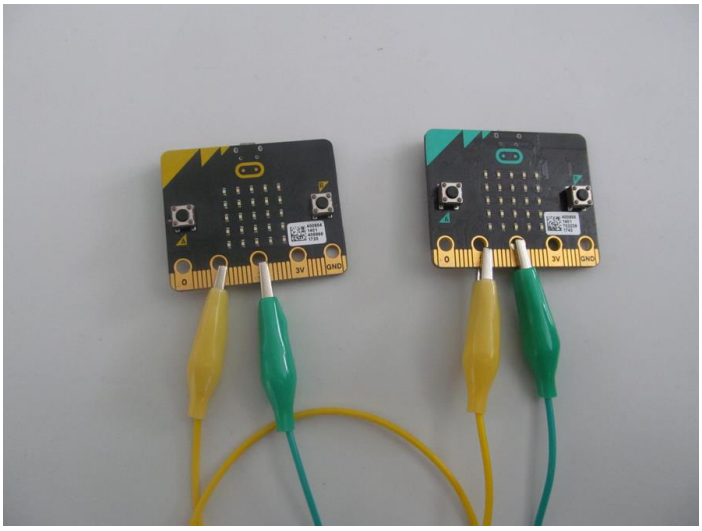
Half duplex (poloviční duplex) – vysílání může směrovat oběma směry, ale v daném okamžiku pouze jedním směrem.

Duplex – vysílání může směřovat v daném okamžiku oběma směry současně

Síťový protokol – soubor předem domluvených pravidel, kterými se řídí daný síťový přenos.

Drátový přenos

Dva micro:bity propojíme pomocí dvou kabelů tak, že propojíme vzájemně piny 1 a piny 2 na obou micro:bitech. Pin1 na prvním micro:bitu s pin1 na druhém micro:bitu a stejně tak i oba pin2. Opět jako u přehrávání zvuku můžeme s výhodou použít kabely s „krokodýlky“ na koncích.



Je třeba stanovit, který micro:bit bude „Vysílač“ a který „Přijímač“. Jedná se tedy o simplexový přenos. Funkce programů je následující na prvním micro:bitu stiskneme tlačítko A nebo B a na druhém micro:bitu se vzápětí rozsvítí symbol A nebo B. Jedná se tedy o binární stav, jeden ze symbolů může reprezentovat jedničku (pravdu – true) a druhý nulu (nepravdu – false). Je snadné místo A a B rozsvítit i 1 nebo 0. Pomocí síťového protokolu lze domluvit, co v daném případě který symbol znamená a o jakou informaci se jedná.

Na micro:bitu „Vysílač“ nahrajte a odlaďte následující program:

```
from microbit import *
while True:
    if button_a.was_pressed():
        display.show("A")
        pin1.write_digital(1)
    else:
        pin1.write_digital(0)
    if button_b.was_pressed():
        display.show("B")
        pin2.write_digital(1)
    else:
        pin2.write_digital(0)
    display.clear()
    sleep(10)
```

Program v nekonečném cyklu sleduje, zda bylo stisknuté tlačítko A, nebo tlačítko B a pokud ano vysílá jedničku na daném propojení. Pro kontrolu zobrazí rovněž odpovídající písmeno.

Na micro:bitu „Přijímač“ nahrajte a odlaďte následující program:

```
from microbit import *
while True:
    if pin1.read_digital():
        display.show("A")
    elif pin2.read_digital():
        display.show("B")
    sleep(1000)
    display.clear()
```

Nyní micro:bity "Přijímač" a "Vysílač" propojte a pro jistotu oba resetujte. Po stisku kláves na Vysílači by měl Přijímač zobrazovat písmena.

Tomuto případu, kdy signál putuje po více kabelech se říká *paralelní přenos* – signály posíláme vedle sebe. Bylo by možné využít tohoto zapojení k přenosu 4 znaků, s následujícími možnostmi. V daném okamžiku je:

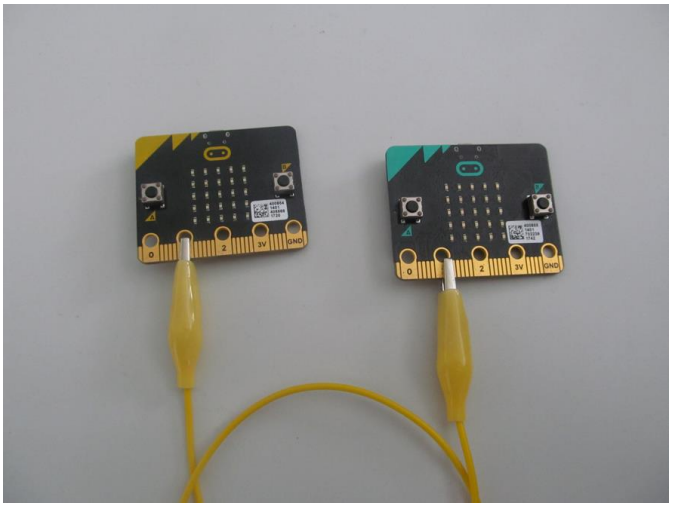
- žádný signál (00)
- signál na prvním vodiči (10)
- signál na druhém vodiči (01)
- signál na obou vodičích (11)

Kolik znaků bychom mohli přenést při maximálním možném počtu vodičů, který umožňuje micro:bit? Odpověď závisí na tom, zda použijeme tři vodiče v běžném režimu (pomocí krokodýlků) anebo až 17 vodičů, pokud micro:bit připojíme pomocí nějakého shieldu k nepájivému poli. Vždy je to však 2^n.

Princip sériového přenosu

Pokud chceme propojit micro:bity pouze jedním kabelem, a i pak přenášet jiný signál než zapnuto vypnuto, je nutno se domluvit na nějakém protokolu. V následujícím příkladu odlišujeme přenesený signál dle jeho délky – 500 ms nebo 1500 ms. Měříme jeho délku pomocí funkce running.time() a pokud je signál kratší než jedna sekunda, považujeme jej za první stav a pokud je delší než jedna sekunda za druhý stav. Takovémuto způsobu přenosu se říká *sériový* – signály posíláme za sebou.

Micro:bity propojíme pouze jedním kabelem mezi piny 1. Viz následující obrázek.



Na micro:bit „Vysílač nahrajeme následující program:

```
from microbit import *
while True:
    if button_a.was_pressed():
        display.show("A")
        pin1.write_digital(1)
        sleep(500)
        pin1.write_digital(0)
    if button_b.was_pressed():
        display.show("B")
        pin1.write_digital(1)
        sleep(2000)
        pin1.write_digital(0)
    display.clear()
```

Program v nekonečné smyčce kontroluje stisk kláves A a B a při stisku vyšle signál odpovídající délky a pro kontrolu zobrazí i kód stisknuté klávesy. Na micro:bit „Přijímač“ nahrajeme následující program:

```
from microbit import *
while True:
    if pin1.read_digital():
        start = running_time()
        while pin1.read_digital():
            pass
        konec = running_time()
        cas = konec - start
        if cas < 1000:
            display.show("A")
        else:
            display.show("B")
        sleep(1000)
        display.clear()
```

Program v nekonečné smyčce hlídá, zda se objeví signál na pinu1. Pokud ano zaznamená si jeho čas. Funkce running.time() vrací čas v milisekundách od spuštění Micro:bitu. Nyní se čeká, dokud je na pinu1 signál a po jeho ukončení opět změříme čas. Spočítáme dobu signálu odečtením začátku od konce. Je-li signál kratší, než jedna sekunda považujeme jej za první typ znaku zde např. A. Je-li delší pak za B.

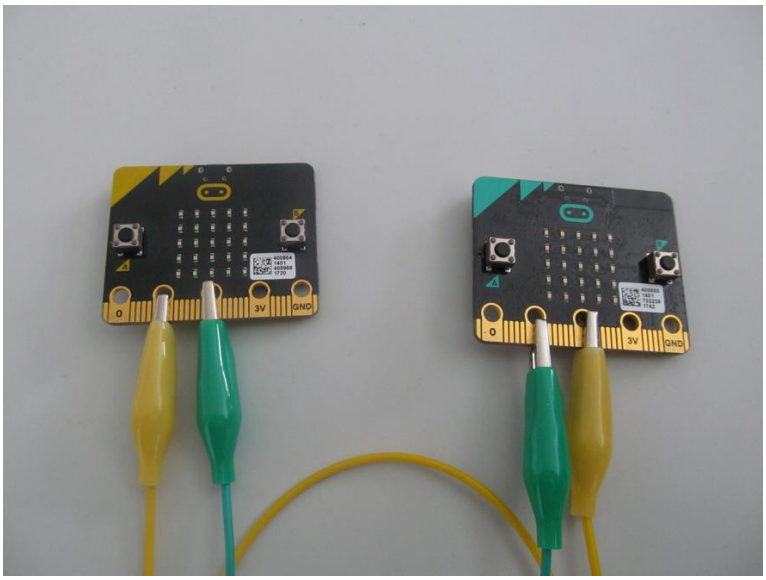
Příkaz `pass` se používá tehdy, pokud máme cyklus, který pouze čeká, až skončí nebo nastane nějaká událost, programovací jazyk MicroPython totiž nepovoluje prázdný cyklus.

Tímto způsobem si můžeme přenášet binární signál, namísto znaků A a B, můžeme použít 1 a 0 (nebo naopak). Tímto způsobem je možné například přenášet ASCII kód, když se domluvíme, že osm za sebou přenesených znaků je kód jednoho ASCII znaku. Lze takto například přenášet i Morseovu abecedu, pokud si například domluvíme, že kratší znak je tečka, delší znak je čárka.

Zájemci si mohou zkusit obě tyto úlohy řešit. Vyřešení úlohy s přenosem kódu Morseovy abecedy naleznete na internetových stránkách MicroPythonu.

(https://microbit-micropython.readthedocs.io/en/latest/tutorials/network.html#the-end-result)

Nyní opět změníme zapojení. Propojíme micro:bity tak, že kabel bude na jednom micro:bitu připojen na pin1 a na druhém na pin2. S druhým kabelem to uděláme naopak. Micro:bity budou tedy zapojeny do kříže na pinech 1 a 2. Oba micro:bity budou nyní současně „Přijímač“ i „Vysílač“ na oba tedy nahrajeme stejný kód.



```
from microbit import *
while True:
    if pin1.read_digital():
        display.show(Image.HAPPY)
    else:
        display.clear()
    if button_a.was_pressed():
        pin2.write_digital(1)
    else:
        pin2.write_digital(0)
    sleep(100)
```

Význam kódu je velice jednoduchý. Program v nekonečné smyčce provádí následující. Pokud je na pinu 1 signál, zobrazí se smajlík. Pokud je stisknutá klávesa A vyšleme signál na pin 2. To samé se děje i na druhé straně. Jedná se o *duplexní signál*, současně přenášíme informaci oběma směry.

Možná vás nyní napadlo, že by mohl stačit jen jeden kabel. V tom případě musíme vyřešit následující dva problémy:

- Pokud by Micro:bit vyslal signál na vodič, současně by se na něm rovněž při testu signálu na témže vodiči načetla jednička a měl by za to, že signál i přijímá. Této situaci se dá zabránit tak, že v případě vysílání signálu se netestuje stav signálu na vodiči.
- Další problém by vznikl by pokud oba Micro:bity vysílaly současně. Takovémuto případu se říká *kolize*. Řešením je, že Micro:bit před vysláním signálu nejprve ověří, zda na kabelu již není signál a pak teprve začne vysílat. Je třeba si uvědomit, že to není dokonalé řešení, oba Micro:bity mohou testovat a začít vysílat ve stejném okamžiku. Je to málo pravděpodobné, ale ne zcela vyloučené. I tento případ má řešení, po začátku vysílání Micro:bit po náhodné době na malou chvilku přeruší signál a ověří, zda na vodiči není signál z druhé strany a pak buď počká nebo vysílá dál. Doba, po které se to testuje, nesmí být vždy stejná, aby opět obě strany nedělaly totéž.

Takovémuto přenosu, jak již víme by se říkalo *half duplex*. Můžete si jej zkusit naprogramovat, včetně řešení kolize. Toto je zjednodušení způsobu, jakým je přenášen internet po běžných ethernetových kabelech.

```
from microbit import *
while True:
    if pin1.read_digital():
        display.show(Image.HAPPY)
    else:
        display.clear()
    if button_a.was_pressed():
        pin2.write_digital(1)
    else:
        pin2.write_digital(0)
    Sleep(100)
```

Bluetooth přenos

Ačkoliv Micro:bit obsahuje přijímač i vysílač **bluetooth** signálu, **nelze** jej **v MicroPythonu využít**. Na vině je to, že do paměti Micro:bitu nelze současně umístit překladač MicroPythonu i kód pro obsluhu bluetooth kvůli jeho paměťové náročnosti.

Problémem je i to, že pokud jsme na Micro:bitu pracovali s MicroPythonem, nelze jej pomocí bluetooth spárovat s jiným zařízením. Jediné možné řešení je nahrát na Micro:bit libovolný program vytvořený grafickým programovacím jaztykem MakeCode. Tím dojde k uvolnění paměti a

zpětnému nahrání potřebného firmware. Pak již Micro:bit můžeme spárovat s jiným zařízením prostřednictvím bluetooth.

Dodatek – micro:bit V2 již obsahuje dostatek paměti, ale implementace Bluetooth do MicroPythonu je dosud (2021) ve vývoji.

Rádiový přenos

Velice zajímavou možností, jak mohou spolu dva micro:bity komunikovat je bezdrátový rádiový přenos. Je možné použít celkem 84 kanálů označených 0 až 83. Jedná se o frekvenci 2400 MHz (která odpovídá kanálu 0), každý kanál má rozsah cca. 1 MHz.

Práce s tímto přenosem je velmi jednoduchá, MicroPython obsahuje knihovnu, která má v sobě metody umožňující přímo přenos textového řetězce (nebo čísla). Ukázka je v následujícím příkladě. Na straně odesílatele:

```
from microbit import *
import radio
kanal = 23
radio.on()
radio.config(channel = kanal)
while True:
    if button_a.is_pressed():
        radio.send("Zprava")
    sleep(1000)
```

A na straně příjemce:

```
from microbit import *
import radio
kanal = 23
radio.on()
radio.config(channel = kanal)
while True:
    zprava = radio.receive()
    if (zprava):
        display.scroll(zprava)
        zprava = ""
```

Parametr channel je nepovinný, pokud jej neuvedete, pak je použit předvolený kanál 7. Na druhou stranu, pokud si má spolu povídat větší množství micro:bitů, je nutné, aby ty, které k sobě patří, používaly nezávislý kanál a nepletly se tak ostatním.

Na tomto příkladě je rovněž vidět potenciální nebezpečí. Pokud útočník ví, na kterém kanále si naše micro:bity povídají, může je buď odposlouchávat anebo podstrkovat vlastní zprávy. Jedná se o typ útoku *Man in the middle*.

Možné použití této technologie se nabízí v následujících možnostech:

- *Dálkový ovladač*. Jeden z micro:bitů můžeme ovládat pomocí gest a přenášet pokyny k druhému, který může něco řídit. Je takto možné například ovládat autíčko nebo nějakou stavebnici, kterou lze ovládat pomocí micro:bitu
- *Zabezpečení*. Jeden z micro:bitů může sledovat např. pohyb, intenzitu světla, magnetického pole atd. a při neobvyklém stavu poslat signál jinému, který je umístěn na místě, kde vyhlásí poplach.

Pro přímou komunikaci se Micro:bity příliš nehodí, protože není jednoduché zadat zprávu, kterou chceme odeslat. Smysluplnou se zdá možnost výběru z nabídky připravených zpráv a odpovědí.

6 PŘIPOJOVÁNÍ PERIFÉRIÍ

Tato kapitola se věnuje připojení periférií. Pod pojmem periférie zde rozumíme další součástky, ze kterých budeme získávat data nebo takové jejichž chování budeme ovládat pomocí micro:bitu.

Každý uživatel micro:bitu po nějaké době zjistí, že mu nepostačují jeho funkce a vlastnosti a chce jej použít i pro další účely. To lze dokázat připojením dalších zařízení (periférií). Jako typický příklad si ukážeme připojení trojbarevné diody jako výstupního zařízení a teplotního čidla jako vstupního zařízení. Pokud pochopíte princip připojení těchto zařízení, pak lze říct, že jste schopni připojit jakékoliv možné zařízení.

Kapitola je koncipována jako volitelná, autoři původně chtěli skončit minulou kapitolou, neboť se předpokládalo, že studenti nebudou pro studium této učebnice potřebovat nic jiného než micro:bit. Jelikož jsme však již připojovali sluchátka a propojovali micro:bity vzájemně, je nám líto nezařadit ještě tuto kapitolu. Ponecháváme zcela na vyučujícím, zda se jí rozhodne zařadit. Znamená rovněž další finanční náklady, které však jsou oproti ceně micro:bitu minimální.

Následující úlohy lze samozřejmě řešit i alternativně pomocí nepájivého pole a přiblížit se tak více elektrotechnické praxi. To ponecháváme na rozhodnutí vyučujícího. Pro vyučující, kteří se rozhodnou nepájivé pole používat, jistě nebude problém si úlohy upravit.

Co se naučíte

- Připojit a ovládat tříbarevnou diodu
- Připojit teplotní čidlo a zjistit pomocí něj teplotu
- Pomocí dvou micro:bitů sestrojit jednoduchý bezdrátový teploměr s jedním čidlem

Co budete potřebovat

- PC s nainstalovaným editorem Mu
- Propojovací USB kabel s micro USB koncovkou
- Micro:bit
- Čtyři vodiče nejlépe s krokodýlky na obou koncích
- Tříbarevnou diodu se společnou katodou (záporným pólem)
- Teplotní čidlo pro napětí 3 V např. TMP-36

Časová náročnost

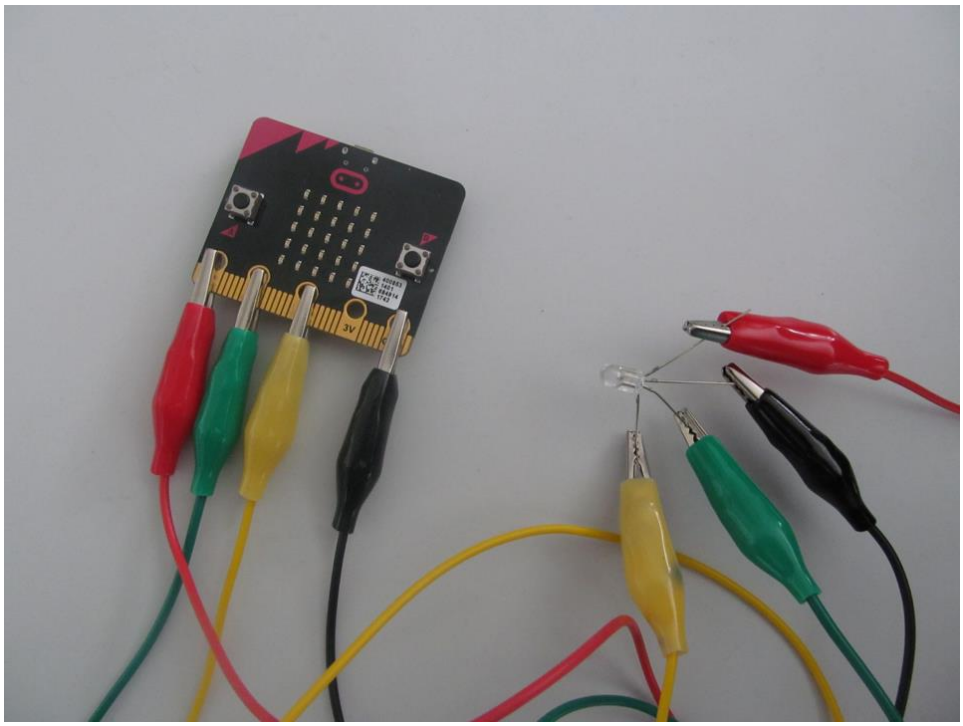
Dvě až čtyři vyučovací hodiny po 45 minutách, dle toho jaké periferie budete připojovat.

Tříbarevná dioda

Tříbarevná dioda je taková dioda, která umí svítit modře, zeleně a červeně a kombinací těchto barev. U každé z těchto barev lze nastavit i intenzitu. Takováto dioda má obvykle čtyři vstupy (piny). Nejdelší z nich je společný a připojujeme na něj buď napětí (pak mluvíme o společné anodě) nebo zem (pak mluvíme o společné katodě). Na jedné straně pak je jeden pin (který obvykle ovládá červenou) a na druhé straně dva piny (od středu obvykle zelená a modrá).

Pro naše použití se bude hodit dioda se společnou katodou, která se mnohem snadněji zapojuje a ovládá. Dejte si pozor při nákupu, ať si koupíte tento druh diody.

Diodu zapojte podle následujícího obrázku. Dejte si pozor, aby se jednotlivé piny nebo krokodýlky vodičů vzájemně nedotýkali. **Správně bychom samozřejmě ještě měli zapojit předřadný odpor 330 Ohmů**, ale napětí na výstupech Micro:bitu není takové, aby došlo k okamžité likvidaci diody, a navíc by nám to významně komplikovalo zapojení celého systému. V této učebnici se snažíme o co nejjednodušší zapojení bez použití např. nepájivého pole, bez kterého bychom se v tomto případě zřejmě neobešli.



Funkci a pořadí barev si nyní můžete vyzkoušet pomocí jednoduchého programku:

```
from microbit import *
pin0.write_digital(1)
sleep(2000)
pin0.write_digital(0)
pin1.write_digital(1)
sleep(2000)
pin1.write_digital(0)
pin2.write_digital(1)
sleep(2000)
pin2.write_digital(0)
```

který můžeme významně zjednodušit a zpřehlednit takto:

```
from microbit import *
A = [pin0, pin1, pin2]
for I in range(0, 3):
    A[I].write_digital(1)
    sleep(2000)
    A[I].write_digital(0)
```

Ten postupně na dvě vteřiny rozsvítí barvy dle pinů připojených na pin0, pin1, a nakonec na pin2 micro:bitu. Takto si můžete nastavit pořadí barev dle svého přání.

V tomto příkladu se jedná o digitální dvoustavový (binární) výstup. Na výstup zapisujeme vždy jedničku nebo nulu, barva buď svítí naplno nebo nesvítí. Můžete si vyzkoušet rozsvítit také kombinaci barev nebo všechny tři barvy najednou.

Nyní si vyzkoušejte i analogový výstup, když budete postupně rozsvěcet a zhasínat jednu barvu. Ve skutečnosti se jedná o diskretizaci analogového prostoru. Máme k dispozici 1024 úrovní (0 až 1023).

```
from microbit import *
while True:
    for I in range(0, 1024):
        pin0.write_analog(I)
        sleep(2)
    sleep(1000)
    for I in range(1023, 0, -1):
        pin0.write_analog(I)
        sleep(2)
```

Pro jistotu ještě jednou vysvětlíme rozdíl mezi digitálním a analogovým signálem:

- digitální signál se skládá z posloupnosti 0 a 1 (ano – ne). Např. CD nebo DVD.
- analogový se skládá ze spojitého signálu, který může nabývat jakékoliv hodnoty mezi dvěma danými stavy (tedy např. i mezi 0 a 1). Např. gramofon.

Následuje ukázkový program, který je nazván „Magická lampa“. Tento program diodu plynule rozsvěcí a zhasíná náhodnou barvou, přičemž při zhasínání se již rozsvěcí barva následující. Je ošetřeno, aby se barvy neopakovaly.

Tentokrát je použit analogový zápis, který na příslušný pin posílá hodnotu od 0 (nesvítí) do 1023 (svítí naplno). Tak dosáhneme postupně všech poloh mezi úplným zhasnutím a úplným rozsvícením dané barvy.

Proměnná `minula` obsahuje informaci o naposledy nastavené barvě (0 až 2). Na počátku je nastavena na např. na 2, takže začneme tím, že se zháší barva 2 a současně rozsvěcí nově vybraná barva. Doba svitu po rozsvícení je stanovena opět náhodně. Na závěr cyklu současnou barvu nastavíme jako minulou. Můžete příklad různě upravovat, např. zvětšit časovou prodlevu.

```
from microbit import *
import random
A = [pin0, pin1, pin2]
minula = 2
while True:
    barva = random.randint(0, 2)
    while (barva == minula):
        barva = random.randint(0, 2)
    delka = random.randint(1000, 5000)
    for I in range(0, 1024):
        A[barva].write_analog(I)
        A[minula].write_analog(1023-I)
        sleep(2)
    sleep(delka)
    minula = barva
```

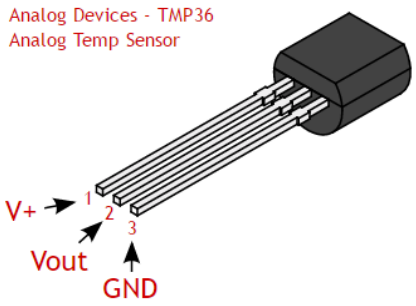
Lampu si můžete skutečně vyrobit. Lze použít papírovou konstrukci, lampion anebo nějakou lampu vyrobenou na 3D tiskárně.

Pozor: Pokud chcete tuto lampu nechat svítit delší dobu než několik vteřin je nutné zapojit předřadný odpor 330 Ohmů, aby nedošlo k poškození diody. Napětí na pinech při plném zatížení je přibližně 2,8 V.

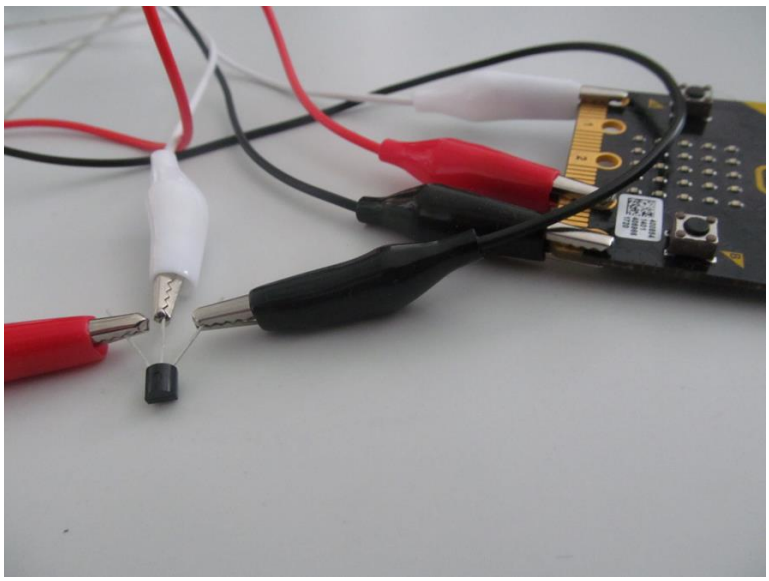
Teplotní čidlo

Pro tuto kapitolu potřebujete nějaké levné teplotní čidlo, pracující s napětím 3 V. Autoři použili čidlo TMP-36, ale je samozřejmě možné použít libovolné které je k dispozici. K použitému čidlu, potřebujete dokumentaci (zvanou datasheet), kterou buď dostanete spolu s čidlem nebo si jí stáhnete z webu prodejce nebo odjinud z internetu.

První věc, kterou si musíte zjistit je zapojení čidla. Například čidlo TMP-36 se zapojuje dle následujícího schématu:



Zde V+ je napájení, připojte na něj 3V, GND (zem) připojte na GND a Vout je výstup, který zapojte na libovolný pin, například na pin0:



Všimněte si na fotografii, že plochá hrana čidla je dole (opačně než na obrázku). Dejte si pozor abyste nespletli zapojení napájení a země, mohli byste snadno teplotní čidlo zničit (autoři s tím již mají zkušenosti).

Čidlo po připojení napájení a země začne měřit teplotu a výsledek sděluje úrovní napětí na výstupním pinu (Vout). Je zde napětí od 0 do 1023 mV. Toto napětí ukazuje procento ze vstupního napětí, které je u micro:bitu 3.17 V.

Proto je výpočet napětí:

$$napeti = \frac{Vout \cdot 3.17}{1024}$$

Odtud pak již vypočtete teplotu (ve stupních celsia):

$$teplota = \frac{napeti - 500}{10}$$

Vzorec je převzatý z datasheetu k čidlu TMP 36 a může se lišit, pokud máte jiné čidlo než popisované TMP 36. V takovém případě si nalezněte potřebné vzorce v odpovídajícím datasheetu.

Pokud vše přepíšete do programu, dostanete následující kód:

```
from microbit import *
while True:
    hodnota = pin0.read_analog()
    napeti = hodnota * (3170 / 1024)
    teplota = (napeti - 500) / 10
    display.scroll(round(teplota, 1))
    sleep(10000)
```

Jak je vidět mezi jednotlivými měřeními je pauza 10 sekund. Tu si samozřejmě můžete upravit, dle vlastního přání.

Počítejte s tím, že po zapojení chvíli trvá, než se teplotní čidlo srovná na teplotu měřeného okolí. Zejména pokud jste jej před tím drželi delší dobu v ruce. První dva až tři výsledky doporučujeme ignorovat. Všimněte si, jak se teplota postupně ustaluje na určité hodnotě.

Ověřte si výsledek vůči nějakému teploměru, kterému důvěřujete. Pokud se hodnoty významně liší, pokuste se zjistit, kde je problém v tomto pořadí:

- ověřte program, zejména výpočty
- ověřte zapojení
- pokud se liší u všech žáků od spolehlivého teploměru pak ověřte, zda máte správný datasheet a používáte správný vzorec pro výpočet
- ověřte voltmetrem napětí

Pro úplnost je třeba dodat, že samotný micro:bit obsahuje teplotní čidlo, které však není zcela spolehlivé, kvůli tomu, že jej ovlivňuje teplota procesoru.

Program, který využívá toto čidlo je velmi jednoduchý:

```
from microbit import *
while True:
    teplota = temperature()
    display.scroll(teplota)
    sleep(10000)
```

Máte-li více micro:bitů můžete si srovnat změřené teploty oběma čidly anebo střídavě měřit teplotu oběma způsoby.

Závěr

Nyní jste již prošli celou učebnici a měli byste micro:bit a MicroPython ovládat. Učebnice i kvůli některým úmyslným zjednodušením nemůže nahradit manuál, a proto pokud vám něco chybělo anebo se chcete dozvědět více, navštivte stránky MicroPythonu pro micro:bit.

Na úplný závěr si nyní můžete ještě vyzkoušet závěrečný příklad. Pro jeho zvládnutí potřebujete nejméně dva micro:bity a doporučena je práce v nejméně dvoučlenné skupině. Zadání příkladu naleznete v Průvodci hodinou anebo v Pracovním listě.

Co dál

Možná vás nyní napadne, co s micro:bitem dále. Zde bychom poradili zkusit žákům obstarat nějaký „microbití“ hardware. Dají se sehnat například vozítka připravená pro micro:bit nebo simulace herní konzole. Můžete si třeba také zkusit ovládat pomocí micro:bitu například stále více populárního Otto bota. Nebudeme zde uvádět odkazy na konkrétní produkt nebo e-shop, věříme, že si svůj kus hardware spolu se studenty sami vyberete.

Odkazy pro další studium

https://microbit.org/

https://www.microbiti.cz/

http://robotika.sandofky.cz/otto-bot/

PŘÍLOHA

Syntaxe některých použitých programových konstrukcí Pythonu.

PODMÍNĚNÝ PŘÍKAZ

Provede se pouze je-li splněna podmínka

Základní tvar

```
if (podmínka):
    příkaz1
    příkaz2
příkaz3
```

Příkazy 1 a 2 se provedou pouze je-li splněna podmínka, příkaz 3 vždy. Pro odsazení se používají **čtyři mezery**. Obvykle je možné nahradit čtyři mezery tabelátorem, ale nemusí to být vždy pravidlem, naopak použití tabelátoru může někdy způsobit chybu. Většina editorů pozná podmíněný příkaz díky dvojtečce na konci podmínky a odsazení zařídí sama. Konec podmínky je dán ukončením odsazování.

Příklad:

```
if (a<>0):
    b = 1 /a
    print(b)
```

Předchozí podmínku lze psát pouze

```
if (a):
```

Pokud výraz v závorce (hodnota a) má hodnotu různou od nuly je podmínka brána jako splněná. Pokud naopak je roven nule pak jako nesplněná.

Používají se relační (porovnávací) znaménka:

- < - menší
- > - větší
- == - rovno (neplést s = přiřazovací příkaz)
- != - nerovno
- <= - menší nebo rovno
- >= - větší nebo rovno

Logické spojky:

- **and** (a zároveň)
- **or** (nebo)
- **not** (negace , ne)

Příklad:

```
if ((a>1) and (a<2)):
```

totéž je:

```
if not((a<=1) or (a>=2)):
```

Poznámka – závorky nejsou povinné, ale pomáhají v přehlednosti

Rozšířený podmíněný příkaz

```
if (podminka1):
    příkazy
else:
    příkazy
```

Není-li splněna podmínka provedou se příkazy za `else`. Příklad

```
if (a):
    b = 1/a
    print("b je ",b)
else:
    print("Nelze dělit nulou")
```

Příkaz lze dále rozvětvit pomocí části `elif`, kterou lze opakovat libovolněkrát:

```
if (podminka1):
    příkazy
elif (podminka2):
    příkazy
elif(podminka3):
    příkazy
else:
   příkazy
```

Poznámky

- část s `else` není povinná
- nezapomínat na dvojtečku za podmínkou nebo `else` – častá chyba
- namísto části s `elif` lze použít vícenásobné opakování s podmínkou `if`

CYKLUS WHILE

Jedná se o cyklus s proměnným počtem opakování – před začátkem nevíme kolikrát proběhne. Nemusí proběhnout ani jednou.
Syntaxe:

```
while (podmínka):
     příkazy
```

Př:

```
i = 0
while (i < 5):
    i = i+1
    print(i)
```

Je možné použít příkazy

- `continue` – skok na začátek cyklu s opětovným vyhodnocením podmínky.
- `break` – vyskočení z cyklu

Příklad:

```
i = 0
while (i < 5):
    i = i+1
    if (i == 2):
        continue
    if (i == 6):
        break
    print(i)
```

Nekonečný cyklus – jako podmínku použijeme True. Např:

```
while True:
    vstup = input("Zadej matematicky vyraz: ")
    print(eval(vstup))
```

Jediná možnost, jak takový cyklus ukončit je pomocí příkazu `break`.

CYKLUS FOR

Jedná se o cyklus s pevným počtem opakování – víme kolikrát proběhne
Syntaxe:

```
for promenna in (iterovatelný objekt):
     příkazy
else:
     příkazy
```

Část za else se provede, pokud není podmínka splněna. Tato část je nepovinná a obvykle se nepoužívá.
iterovatelný objekt – lze jej rozložit na části, MicroPython podporuje:
`range(A)` – od 0 do A-1, po jedné
`range(A,B)` – od A do B-1, po jedné
`range(A,B,C)` – od A do B-1 po C

Příklady:

```
for I in range (4):
    print(I)
```

(provede se pro 0,1,2,3)

```
for I in range (2,4):
    print(I)
```

provede se pro 2,3

```
for I in range (2,8,2):
    print(I)
```

provede se pro 2,4,6

```
for I in range (6,3,-1):
    print(I)
```

provede se pro 6,5,4