PRŮVODCE TEORIÍ

Počítačové sítě

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

Počítačové sítě dělíme dle přenosového média na:

- Drátové
- Bezdrátové
  - Wi-Fi
  - Bluetooth
  - Radio
  - 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:

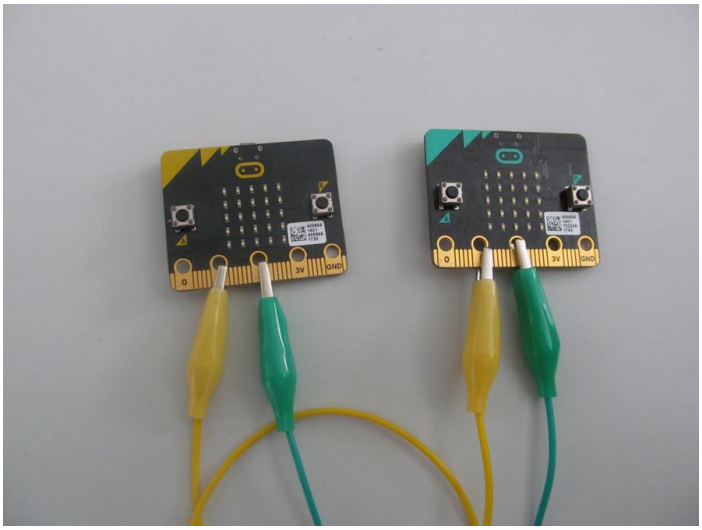
```
from microbit import *
while True:
    if button_a.was_pressed():
        display.show("A")
        pin1.write_digital(1)
    else:
        pin1.write_digital(0)
    if button_b.was_pressed():
        display.show("B")
        pin2.write_digital(1)
    else:
        pin2.write_digital(0)
    display.clear()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

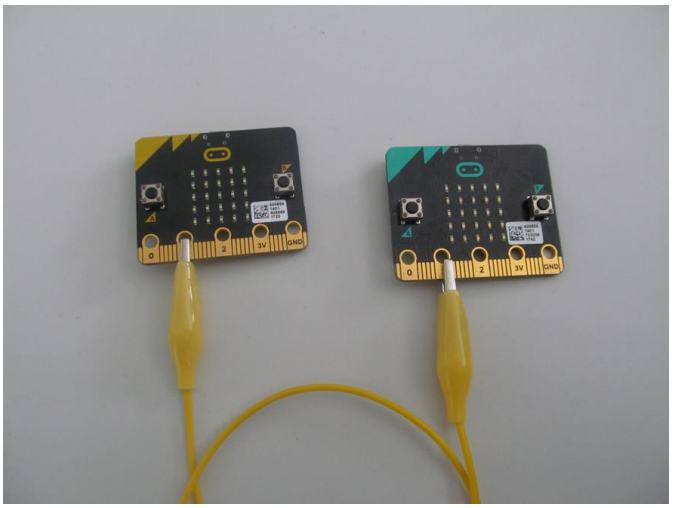
- žádný signál (00)
- signál na prvním vodiči (10)
- signál na druhém vodiči (01)
- signál na obou vodičích (11)

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 použijeme datový shield. 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.



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 nějaká událost, 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

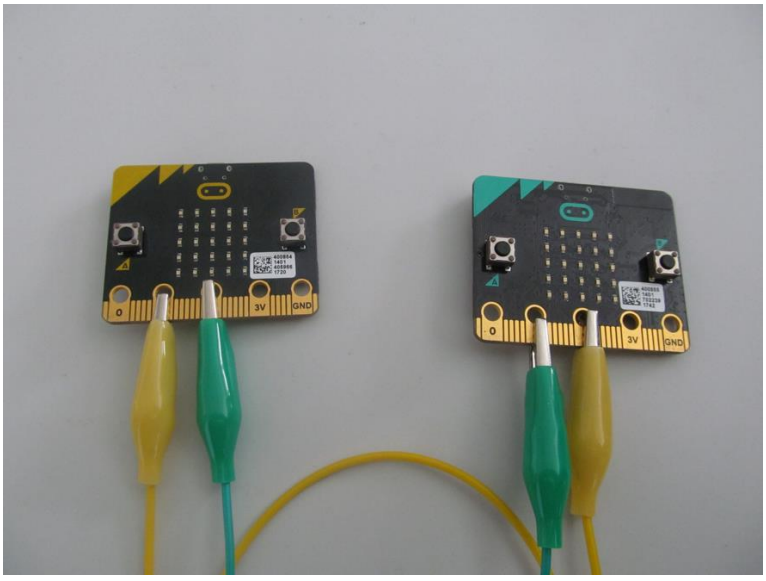
například přenášet i Morseovu abecedu, pokud si například domluvíme, že kratší znak je tečka, delší znak je čárka.

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 říkal *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 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)
radio.off()
```

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

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 odpověd