Vyšší odborná škola a
Střední průmyslová škola elektrotechnická
Božetěchova 3, 772 00 Olomouc

ZADÁNÍ VÝSTUPNÍHO PROJEKTU

Z PŘEDMĚTU VYT

Název práce: **Programování se zařízením micro:bit**

Zpracovatel: MARTIN LUCZKA

Třída: 2A

Hodnotitel: Mgr. Jaroslav Krbec

Školní rok: 2022/2023

- Formou formátovaného dokumentu představení zařízení micro:bit a jeho funkcí. Seznámení se s programovacím jazykem MicroPython. Vysvětlení základních principů a jejich možná aplikace.
- Porovnání cen zařízení micro:bit u nás a v zahraničí pomocí tabulky a grafu.
- Propagace a představení micro:bitu a MicroPythonu v rámci webové stránky. Ucelené prostředí pro jednotlivá cvičení a jejich řešení.
- Propagační leták o micro:bitu, jeho možnostech a využití v praxi.
- Video shrnující celý projekt a jeho části. Vysvětlení a prezentace nejzajímavějších cvičení + video dokumentace vlastního micro:bitu.

V Prostějově dne 5. 5. 2023

1 Obsah

2 Úvod

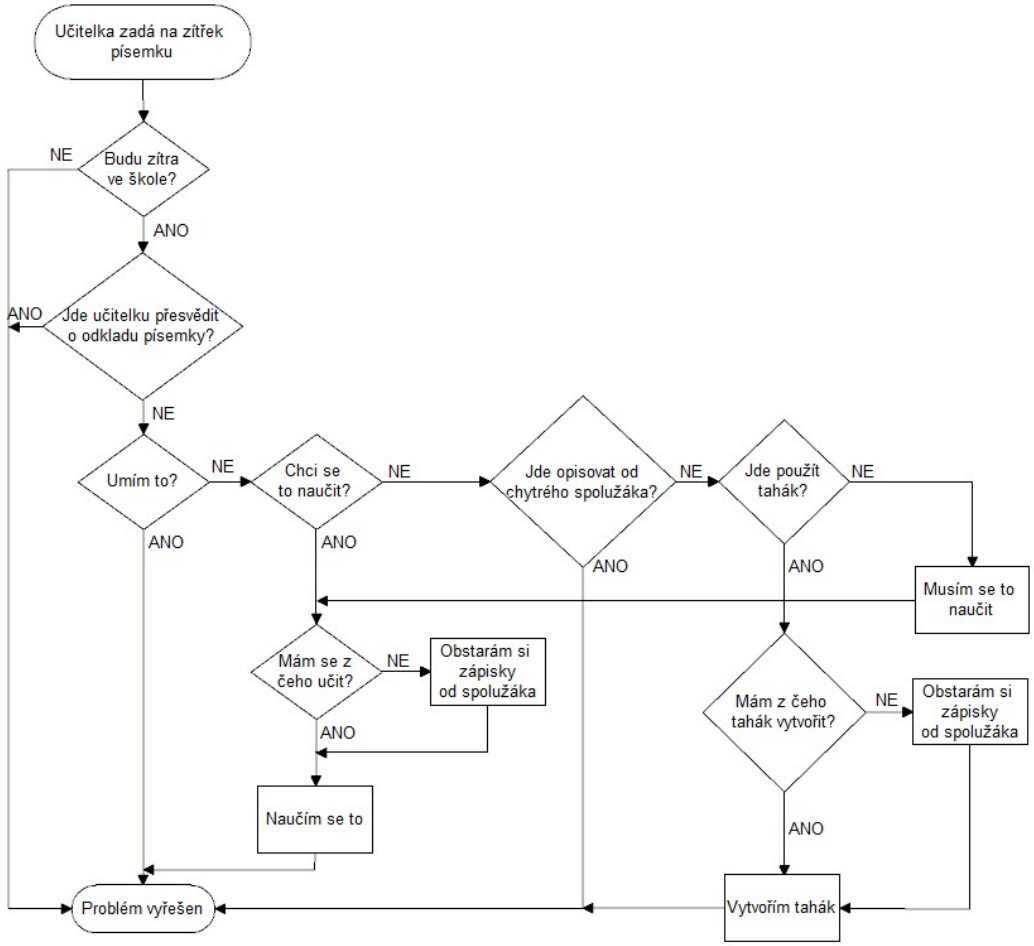
Pro svůj závěrečný projekt jsem si vybral programování se zařízením micro:bit. S micro:bitem jsem se setkal již v prvním ročníku na střední škole. Tehdy jsem ještě s programováním neměl jakékoliv zkušenosti. Micro:bit mě s programováním seznámil velice poutavou a zábavnou formou. Při mých prvních zkušenostech jsem micro:bit programoval pomocí tzv. bloků. Tato forma programování mě svého času velice chytla, protože byla, a stále je, velice intuitivní. Studenta takto micro:bit právě nehodí rovnou do kódu, ale dá mu i jiné způsoby tvoření v tomto prostředí. V druhém ročníku na střední škole jsem se micro:bitem potkal znovu, tentokrát již v rámci kódu. V hodinách jsem se učil micro:bit programovat za pomoci programovacího jazyka MicroPython. Programovat na úrovni kódu jsem se takto učil poprvé. Takovéto programování v kódu mě velice zaujalo a vždy jsem se těšil, až se dozvím něco nového, něco co budu moct ve svém řešení problémů využít. V tomto projektu se budu snažit vytvořit 5 úloh, které budou obsahovat, snad co nejvíce, funkcí. Micro:bit je nesmírně komplexní zařízení a celý jeho popis by se hodil spíše na nějakou dlouhodobou práci. Budu se snažit čtenáře seznámit se základními funkcemi programování i samotného Micro:bitu.

3 Programování se zařízením micro:bit

3.1 Proč programujeme?

Programování je v současnosti poměrně dost skloňované slovo, ale co že vlastně znamená? Programování je řešení problémů, vytváření určitého prostředí, které po aktivaci nějakých vstupů, provede nějaký úkon, činnost. V programování je dost velkou částí také analýza problému a hledání řešení. V souvislosti s programováním se také setkáme s pojmem algoritmus. **Algoritmus** je nějaký návod či postup. Tímto návodem se poté řídíme při řešení problému a vytváření kódu.

S algoritmem je také spjat další pojem a tím je vývojový diagram. **Vývojový diagram** je právě takový druh algoritmu. Právě ten návod či postup je v takovém diagramu vyjádřen graficky. Na příkladu určitě pochopíte, jak takový vývojový diagram. Na internetu takových vývojových diagramů najdete bezpočet. Na ukázku jsem si dovolil použít vývojový diagram, který jsem sám v minulosti vytvořil. Popisuje situaci ze školního prostředí, takže to by některým mohlo být velmi blízké.



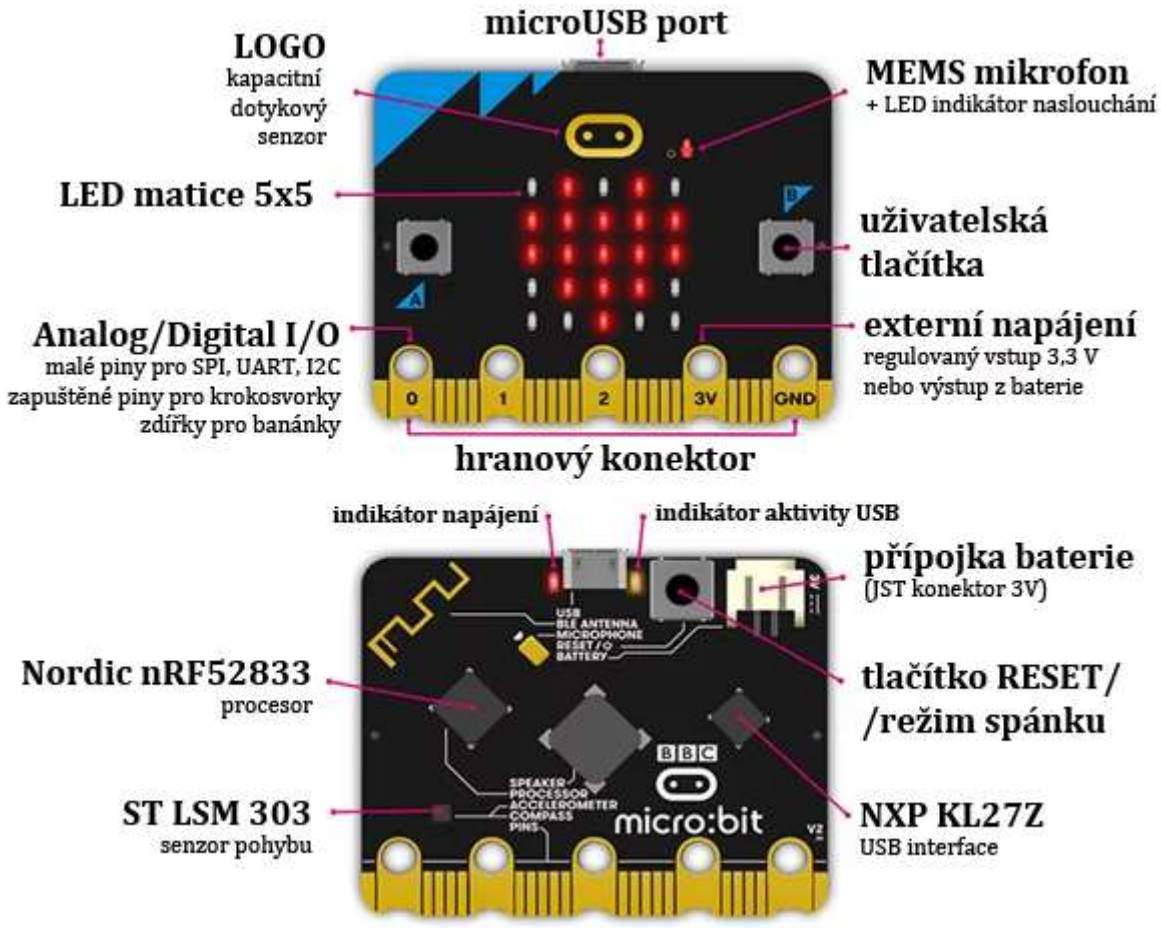
Obrázek 1 - Vývojový diagram

Zdroj: Vlastní

3.2 Co je to micro:bit?

Micro:bit je v podstatě takový kapesní počítač. Je to zařízení, se kterým se můžeme seznámit se základními principy programování. Můžeme jej programovat jak v blocích, tak i v samotném kódu – MicroPython či JavaScript. Toto zařízení bylo vytvořeno na podporu výpočetní techniky na školách ve Velké Británii. Micro:bit jako takový vyšel ve dvou verzích. První verze v1 vyšla v únoru roku 2016 a druhá verze v2 vyšla v roce 2020, přesněji v říjnu. Druhá verze se oproti té první liší mírně většími rozměry a také přidáním některých funkcí. Navíc je tu mikrofon na snímání zvuku, reproduktor, dotykový senzor a úsporný režim.

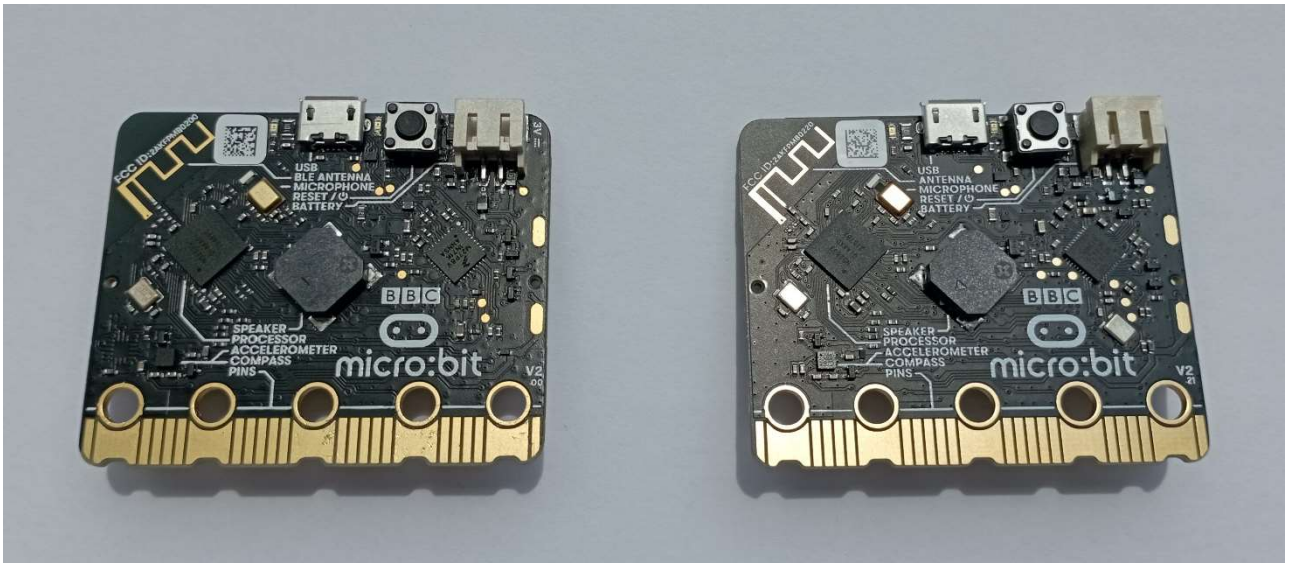
Celé zařízení v podstatě stojí a běží na dvou věcech. Je to displej, který je tvořen 25 SMD LED diodami, ty jsou uspořádány do čtverce 5 x 5, a také dvě programovatelná tlačítka na bocích displeje. Dále by byla škoda opomenout také programovatelné piny, díky kterým lze micro:bit připojovat k různým jiným zařízením, micro:bit takto funguje jako taková řídící jednotka.



Obrázek 2 - Struktura zařízení micro:bit *Zdroj: HW KITCHEN*

Obrázek 3 - micro:bity zepředu

Zdroj: Vlastní



Obrázek 4 - micro:bity zezadu

Zdroj: Vlastní

3.3 Programování v blocích

Než přejdeme přímo do kódu, bylo by dobré se nejdříve seznámit s tímto způsobem programování. Sám jsem v něm začínal a dost mi pomohl pochytit základní principy programování. Celé prostředí je navrženo velice dobře. Bloky jsou podle funkce barevně odlišeny a dají se i různě upravovat. Toto prostředí je skvělé pro jednoduché programy, které lze s micro:bitem vytvořit. Pokud bychom chtěli nějaký složitější program, toto prostředí by nám už nevyhovovalo. Celý program z bloků by byl již moc velký a pravděpodobně i poměrně nepřehledný. Pro tvoření takového programu by bylo mnohem vhodnější přesunout se do kódu. Právě třeba do MicroPythonu, ve kterém bychom složitější program vytvořili jistě elegantněji. V této kapitole bych Vás rád seznámil s tímto prostředím. Ukážeme si jednotlivé funkce a také jejich využití v některých programech. Jsou to naše první programy. Začneme něčím jednoduchým a postupně se budeme věnovat složitějším úkolům. Můžeme zde také z bloků přejít na kód, takže se můžeme podívat, jak by náš program vypadal v kódu. K dispozici jsou zde 2 programovací jazyky: JavaScript a MicroPython.

Toto prostředí najdeme na tomto odkazu: https://makecode.microbit.org/, můžeme si zde vytvořit vlastní projekty, ale také se můžeme nechat inspirovat projekty, které jsou zde volně k dispozici.

V prostředí se také nachází simulátor micro:bitu, takže pokud micro:bit nemáme ve fyzické podobě, tak si i tak můžeme zkusit něco naprogramovat. V tomto simulátoru jsou zahrnuty všechny funkce micro:bitu, takže o žádnou část micro:bitu používáním tohoto digitálního simulátoru nepřicházíme.

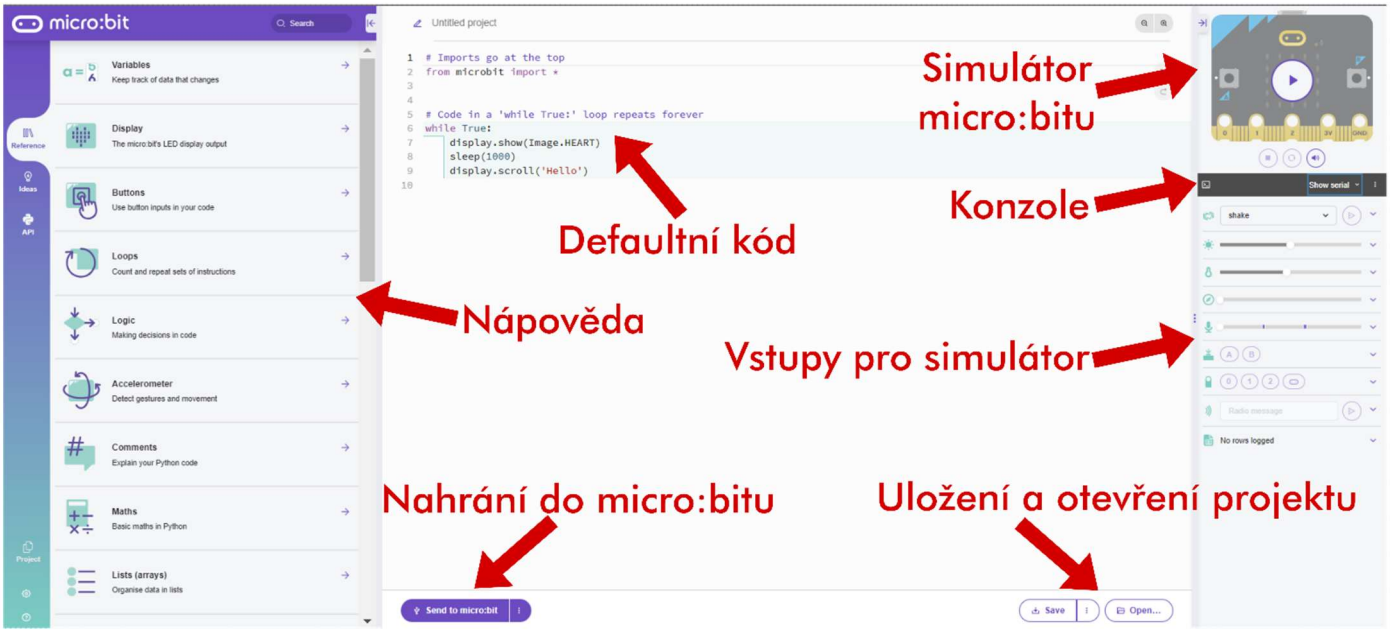
Obrázek 5 - Prostředí MakeCode *Zdroj: Vlastní*

3.4 Programování v MicroPythonu

Po seznámení se s bloky si také představíme jiný způsob programování se zařízením micro:bit. Na velice krátké programy jsou bloky poměrně vhodné, program je takto velice přehledný a dobře se v něm orientuje. Ovšem po naprogramování několika jednoduchých programů bychom se chtěli přesunout na něco složitějšího. To lze v bloku také uskutečnit, ale program se stává již nepřehledný a špatně se v něm hledá případná chyba či nesrovnalost. V takovém případě se přesouváme do kódu, kde jsme i během několika řádků schopni vytvořit poměrně komplexní řešení nějakého problému. Programování v kódu se na první pohled zdá velmi složité, ale jsem si jist, že na příkladech vše pochopíte. Programovat budeme v tzv. MicroPythonu. **Ale co že ten MicroPython vlastně je?**

MicroPython je programovací jazyk, který je do značné míry kompatibilní s **Pythonem 3**. Řekl bych, že o Pythonu jste už možná někdy slyšeli. MicroPython je právě optimalizován na práci s mikrokontroléry. Tento jazyk byl vytvořen australským programátorem **Damienem Goeorgem** po úspěšné kampani v roce **2013** na crowdfundingové platformě **Kickstarter**. První vydání se uskutečnilo 3. května roku 2014. Verze pro micro:bit byla vytvořena v roce 2016.

Micro:bit má také vlastní prostředí pro programování v MicroPythonu. Nachází se zde simulátor micro:bitu a také nápověda a představení příkazů tohoto programovacího jazyka. Toto prostředí najdete na tomto odkazu: https://python.microbit.org/v/3/reference



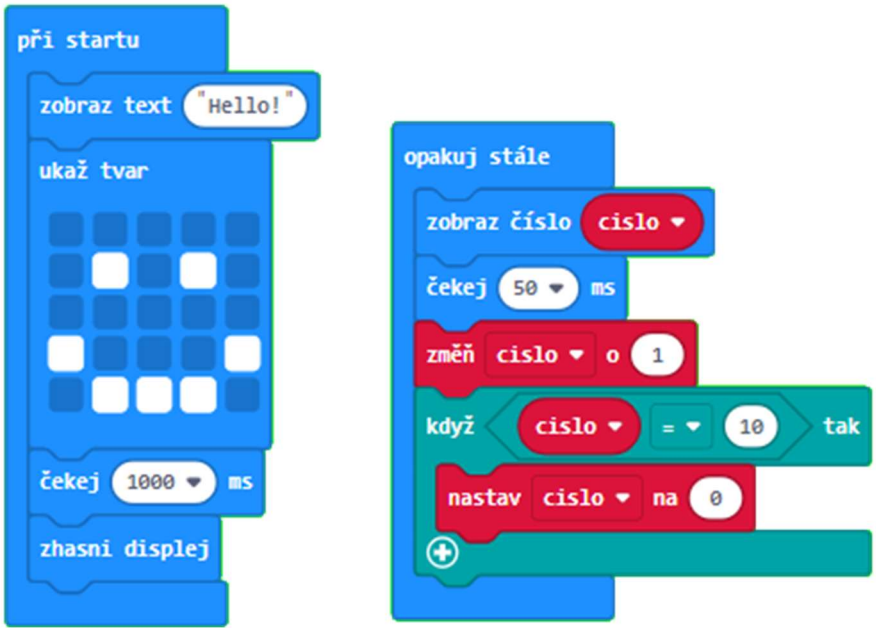
Obrázek 6 - Prostředí Python Editor *Zdroj: Vlastní*

3.5 Základní principy programování doprovázené úlohami

V této části dokumentu se seznámíme se základními principy, které si ihned ukážeme na nějaké úloze. Vždy bude před námi stát nějaký problém, který si rozebereme a budeme ho postupně řešit. Při řešení těchto problému budu postupně představovat jednotlivé funkce a zákonitosti programování. Ze začátku začneme opravdu jednoduchými programy a postupně bychom se měli propracovat k nějakému mírně komplexnějšímu příkladu, který by mohl mít i nějaké využití v praxi. Ale hlavní je spíše prezentace a správné uchopení jednotlivých principů.

3.5.1 Čítání od 0 do 9

Ze začátku budeme od micro:bitu požadovat velice jednoduchou záležitost. Při startu budeme chtít, aby nás micro:bit pozdravil a usmál se na nás. Smajlíka si vytvoříme sami, ale mohli bychom také využít sadu předdefinovaných obrázků. Tuto sadu kroků bychom mohli přeskočit, ale chceme se naučit a využít co nejvíce funkcí. Na druhou stranu ale tyto kroky ze startu nejsou zbytečné, protože si spuštěním těchto kroků ověříme funkčnost našeho micro:bitu. Pokud se zobrazí příkazy ze startu, ale dále již nic, tak víme, že máme problém někde v kódu. Pokud bychom ale tyto kroky do svého kódu nezahrnuly, tak bychom kód mohli mít napsaný dobře, ale chyba by mohlo být někde na straně připojení či funkčnosti samotného micro:bitu. Dále budeme chtít, aby nám zobrazovaná čísla čítala od 0 do 9 a poté aby se vše opakovalo. Nejdříve se podíváme, jak by se taková úloha dala řešit v blocích a poté si vše vysvětlíme.



Obrázek 7 - Čítání od 0 do 9 (bloky) *Zdroj: Vlastní*

Nejdříve začněme blokem „**při startu“**, příkazy v tomto bloku se spustí pouze na začátku spuštění daného programu. Máme zde zobrazení textu, který se zobrazí tak, že obrazovku postupně přejede. Poté zobrazí tvar, v našem případě je to usměvaví smajlík, kterého jsme si sami vytvořili, ale mohlo by to být cokoliv. Smajlíka necháme na obrazovce 1 s použitím funkce „**čekej“** (v kódu poté často používané „**sleep“**). Dále si také většinou při startu definujeme tzv. **proměnné**.

Tyto **proměnné** jsou většinou nějaké hodnoty, které se průběžně mění. Většinou s nimi sčítáme, odčítáme, násobíme je nebo se na ně ptáme při podmínkách. V našem případě je naší proměnnou číslo, které se nám postupně mění.

Hodnotu této proměnné jsme si mohli nastavit na 0 ještě v rámci bloku při startu, ale v tomto případě nemusíme, protože se hodnota proměnné, bez předešlého nastavení, bere jako 0.

Dále zde máme další „nadřazený“ blok a tím je blok „**opakuj stále“**. Tento blok neustále opakuje příkazy a vyhodnocuje podmínky, které se v něm nacházejí.

Ze začátku chci vždy zobrazit proměnnou číslo, poté nechám číslu nějaký čas, aby se mohlo zobrazit (příkaz „čekej“ by v tomto případě zde ani být nemusel, protože příkaz „**zobraz číslo“** již nějaký čas zobrazování zahrnuje, ale bude dobré si tento zvyk osvojit už v blocích).

Dále naši proměnnou změníme o číslo 1 (pokud bychom měnili o číslo -1, tak bychom odečítali).

Poté zde máme použitou **podmínku**. Podmínky jsou v programování asi jeden z nejzákladnějších pilířů pro tvorbu nějakého funkčního programu. Budeme se s nimi setkávat dnes a denně, takže by bylo dobré je pochopit, co nejdříve a co nejlépe.

S touto konkrétní podmínkou našemu programu říkáme, že když bude naše číslo rovno 10, tak má nastavit hodnotu čísla na 0. Když v našem programu tato podmínka není splněna, tak jí náš „cyklus“ přeskočí a začíná zase od začátku příkazů.

Většinou ale program podmínky nepřeskakuje, ale pokud podmínka není splněna, tak se něco děje. To co se děje se samozřejmě mění od programu k programu.

Program jsme tedy takto záměrně „zacyklili“ a úloha je vyřešena.

Na další straně se podíváme, jak by se tato úloha dala vyřešit v kódu za pomoci MicroPythonu, nejdříve zase ukáži řešení a poté si jednotlivé příkazy rozebereme.

```
1
2  from microbit import *
3
4  #definování proměnné
5  cislo = 0
6
7  #uvítací část kódu
8  display.scroll("Hello!")
9  display.show(Image.HAPPY)
10 sleep(1000)
11 display.clear()
12
13 #opakující se část kódu (čítání)
14 while True:
15     display.show(cislo)
16     sleep(500)
17     cislo = cislo + 1
18     if cislo == 10:
19         cislo = 0
20
```

Obrázek 8 - Čítání od 0 do 9 (kód) *Zdroj: Vlastní*

Ze začátku našeho kódu si nejdříve musíme definovat naši **proměnnou**. V našem případě je tímto číslem nula.

Tento **„datový typ“** je tzv. **Integer**, tento typ seskupuje **celá čísla**.

Dále je zde tzv. **Float**, hodnotami tohoto typu jsou **reálná čísla**. Do reálných hodnot patří také desetinná čísla. Bude dobré zmínit, že MicroPython, potažmo Python, využívá jako desetinnou čárku **desetinnou tečku**. Čárkou se oddělují jednotlivé položky.

Poté také **Textový řetězec**, častěji používaný **String**, ten jsme právě použili v našem případě pro zobrazení našeho pozdravu u příkazu ***display.scroll("Hello!")***. Tento datový typ se vkládá do uvozovek.

Dále můžeme v MicroPythonu vytvořit seznamy, které můžou obsahovat různé datové typy. Tyto seznamy jsou ohraničeny **hranatými závorkami**. Tento typ je tzv. **List**.

Ještě bych tu zmínil tzv. **Booleovské hodnoty**. Jsou to v podstatě pravdivostní hodnoty, se kterými jste se možná setkali, nejspíše v matematice. Tyto hodnoty jsou dvě: **True** a **False**. V číslicové logice se používají pro vyjádření těchto hodnot **1** a **0**.

Po definování naší proměnné zobrazíme naší uvítací sekvenci. Příkazy začínající *display*… pracují s displejem. Většinou jde o nějaké zobrazení čísla či textu, zhasnutí obrazovky atd. Při příkazu display.scroll text či číslo po obrazovce plynule přejede. Při příkazu ***display.show()*** se daný argument pouze ukáže. Po tomto příkazu je důležité, aby následoval příkaz ***sleep()***, který program na určitou dobu pozastaví. Poté program zhasne obrazovku příkazem ***display.clear()***.

Dále potřebujeme udělat část kódu, která bude čítat čísla a opakovat se. Toto opakování zařídíme příkazem ***while True***. Následující příkazy, které chceme, aby do tohoto elementu patřili, musíme odsadit a příkaz ***while True*** ukončit dvojtečkou.

Číslo nejdříve zobrazíme a poté proměnnou upravíme připočítáním čísla 1. Na konec kódu vložíme podmínku, která zajistí opakování se čítání od 0 do 9. Pokud by číslo mělo být 10, tak se proměnná v našem případě „resetuje“ a nastaví se zpátky do nuly.

Zmínili jsme zde **Podmínky**. Ty jsou v programování jedním z nejzákladnějších pilířů. Programu vlastně říkáme, co má dělat, pokud je něco splněno. Pokud program vyhodnotí podmínku jako pravdivou, tak daný příkaz vykoná. Pokud ovšem podmínka nebude splněna, program ji vyhodnotí jako nepravdivou a příkaz se nevykoná. Program nepravdivou podmínku v podstatě přeskočí a pokračuje dál.

```
1
2  from microbit import *
3
4  cislo = 0
5
6  while True:
7      cislo = 0
8      while cislo < 10:
9          display.show(cislo)
10         sleep(500)
11         cislo = cislo + 1
12
```

Obrázek 9 - čítání od 0 do 9 (jiné řešení)
Zdroj: Vlastní

Opakující se část kódu by se také dala udělat jiným způsobem. Zde bychom využili příkazu **while**. Tento příkaz opakuje určitou část kódu, dokud je splněna nějaká podmínka. V našem případě se bude zobrazovat proměnná číslo, dokud bude číslo menší než 10. V případě, že už tato podmínka neplatí, program přeskočí do nadřazeného příkazu ***while True***. Proměnná se zde nastaví zpátky do 0 a čítání může začít od znovu.

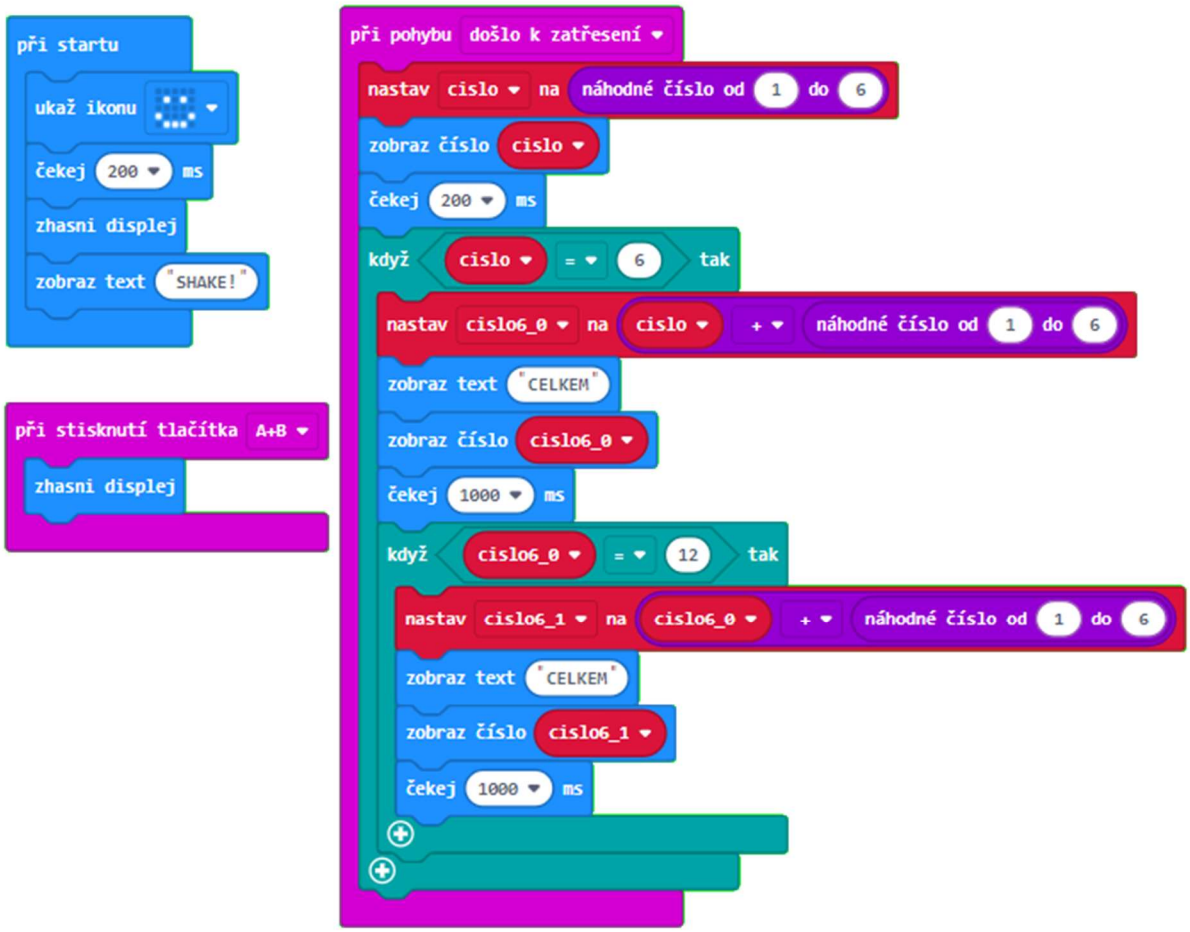
Ukázku tohoto programu prostřednictvím samotného micro:bitu můžete zhlédnout na tomto odkazu: https://youtu.be/8yI3x4ZHioo

3.5.2 Hrací kostka

V následující úloze si ještě trochu osvěžíme proměnné a podmínky. Pro řešení také využijeme některých funkcí micro:bitu. V této úloze si vytvoříme hrací kostku. Už je to úloha, která by se dala využít i v praxi. Když si chcete s přáteli zahrát nějakou deskovou hru, tak se Vám může stát, že se v balení nenachází hrací kostka. Takové hrací kostky jsou poměrně drobné a v průběhu let se jich bezpočet poztrácí. Ovšem pokud budete mít ve svém micro:bitu tento program, tak si hru budete moci přesto zahrát. Takovou kostku si také v průběhu hry můžete přeprogramovat, pokud například změníte pravidla či začnete hrát hru jinou.

Na naší kostce budou hodnotu od 1 do 6. Tato hodnota bude náhodně vygenerována a zobrazena. Takový program by byl velice jednoduchý, ale my si ho ještě trochu vylepšíme. Při hodu čísla 6 budeme „házet“ znovu. Číslo 6 budeme moci „hodit“ až 3 krát po sobě. Tudíž největší možné číslo bude 18.

Nejdříve si ukážeme řešení v blocích, v tom si popíšeme jednotlivé kroky a pochopíme princip. V kódu se poté podíváme na jednotlivé příkazy.



Obrázek 10 - Hrací kostka (bloky) *Zdroj: Vlastní*

Ze startu opět provedeme určitou uvítací sekvenci, abychom věděli, jestli se nám program vůbec spustil. Naše kostka se spustí zatřesením micro:bitu. Při tomto úkonu nastaví do proměnné náhodné číslo od 1 do 6 a zobrazí ho. Ale program se ptá poté dále. Jestliže daným číslem byla 6, tak nastaví další proměnnou, k té původní poté přičítá opět náhodné číslo od 1 do 6 (reprezentace dalšího hodu). Pokud se součet rovná 12 (hodnota druhého hodu byla také 6), tak program hází znovu a naposledy. Jak při druhém, tak při třetím hodu kostka vypíše celkovou hodnotu hodu. Nejvíce je možno hodit číslo 18 v případě, že by dotyčný hodil 6 třikrát za sebou. Pro zajímavost, šance na tento hod je 1 ku 216. Program jsem zkoušel několikrát, i při vytváření, ale tento hod se mi nepodařil uskutečnit. Ještě je zde přidaná funkce, že při stisknutí tlačítek A i B najednou se zhasne display. To se hodí při hodu do hodnoty 10, jednotky totiž na displeji zůstanou. Dvoumístná čísla totiž přes obrazovku „projedou“.

```
1
2  from microbit import *
3  import random
4
5  # uvítací sekvence
6  display.show(Image('00000:'
7                     '09090:'
8                     '00400:'
9                     '60006:'
10                    '08880'))
11 sleep(1500)
12 display.clear()
13 display.scroll("SHAKE!", delay = 100)
14
15 while True:
16     # zajištění resetu
17     if button_a. is_pressed() and button_b. is_pressed():
18         display.clear()
19     # logika házení kostky
20     if accelerometer.was_gesture('shake'):
21         cislo = random.randint(1,6)
22         display.show(cislo)
23         sleep(500)
24         if cislo == 6:
25             cislo6_0 = cislo + random.randint(1,6)
26             display.scroll("CELKEM", delay = 75)
27             display.scroll(cislo6_0)
28             sleep(1000)
29             if cislo6_0 == 12:
30                 cislo6_1 = cislo6_0 + random.randint(1,6)
31                 display.scroll("CELKEM", delay = 75)
32                 display.scroll(cislo6_1)
33                 sleep(1000)
```

Obrázek 11 - Hrací kostka (kód) *Zdroj: Vlastní*

Dále zde máme řešení v kódu. Nejdříve zobrazíme smajlíka. To udělám trošku jinak než v předchozím případě. Kód umožňuje zápis, při kterém si můžeme nastavit jednotlivé LED. Můžeme určit, které budou svítit a hlavně jak moc. Číslo od 0 do 9 představuje intenzitu osvětlení LED. Po nějaké době se obrazovka vypne a poté po obrazovce přeběhne hláška „SHAKE!“. U té si můžeme nastavit délku trvání pomocí ***delay***.

Dále zde máme část kódu, u které chceme, aby platila vždy. Nejdříve zde máme zajištění resetu pomocí podmínky. Příkazu jsou v této podmínce spojeny logickým součinem **AND**. Tato podmínka se splní pouze za předpokladu, že obě dvě tvrzení budou **True**.

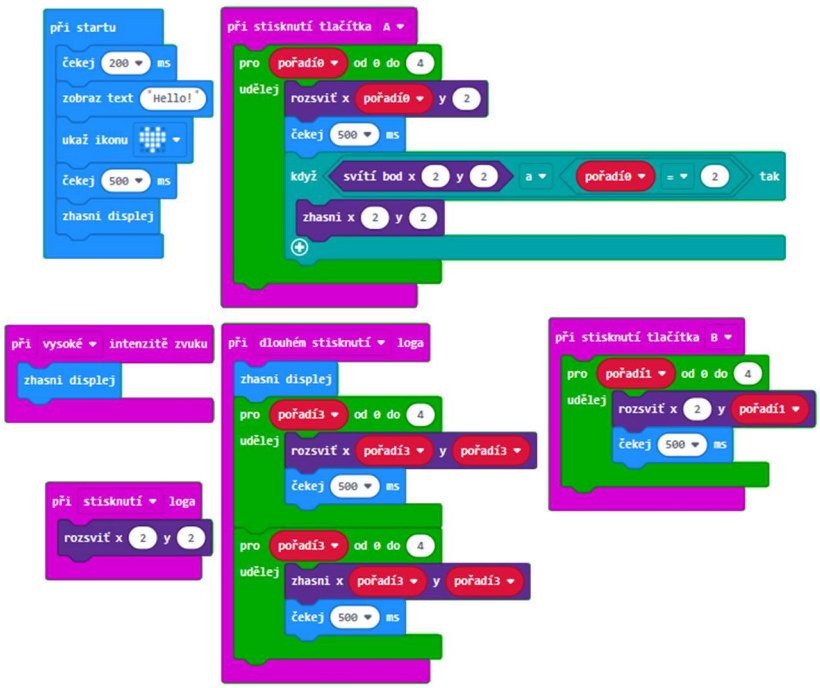
Pokud by zde byl logický součet **OR**, tak by se podmínka splnila při splnění jakéhokoliv tvrzení, což by znamenalo, že se obrazovka micro:bitu vypne při zmáčknutí libovolného tlačítka.

Potom už máme samotné házení kostky. Pokud micro:bit zaznamená otřes, tak se opět nastaví proměnná. Zde využíváme funkce ***random.randint()***. Do závorky poté píšeme rozsah hodnot, ze kterých chceme náhodně vybírat. Tato funkce se nám „odemkla“ díky importu knihovny ***random*** (3 řádek kódu). Dále číslo opět zobrazujeme a následují podmínky v podstatě identické s bloky. Video zde: https://youtu.be/fBPHRY1wY_g

Pojmenování proměnných je čistě na Vás, ale je dobré, aby název alespoň trochu napovídal funkci samotné proměnné. Pokud například ukládáme číselnou hodnotu, tak je to jasné, proměnnou pojmenujeme ***cislo***. V případě, že ukládáme například pravdivostní hodnotu, tak je dobré uvést, k čemu se ta daná pravdivostní hodnota váže. Pokud budeme sledovat třeba zapnutí/vypnutí motoru, tak bychom takovou proměnnou mohli pojmenovat například ***chod_motoru*** nebo jednoduše ***motor***.

3.5.3 Animace displeje

V tomto úkolu si trochu pohrajeme s displejem a představíme si příkaz **for**. Chceme si zde hlavně představit, co nejvíc možností micro:bitu. Budeme chtít, aby při stisknutí tlačítka A se postupně rozsvítila vodorovně prostřední řada a při stisknutí tlačítka B se zase postupně vyplní prostřední sloupec. Je zde také podmínka, že se bude zhasínat prostřední LED. Tu pak rozsvítíme stisknutím loga. Při dlouhém stisknutí loga či zatřesení (rozdíl bloků a kódu) se LED diody postupně rozsvítí do úhlopříčky a poté se do té samé úhlopříčky pozhasínají. Dále je zde jen taková „srandička“, při vysoké intenzitě zvuku okolo micro:bitu se micro:bit zhasne.



Obrázek 12 - Animace displeje (bloky) *Zdroj: Vlastní*

Myslím si, že na bloky už si pomalu zvykáte, takže jejich popis začneme postupně zkracovat. Tady nás určitě nejvíce budou zajímat ty tmavě zelené bloky, přesněji jak fungují. Jsou to tzv. **For cykly**. Fungují tak, že po určitý počet kroků zopakují určitou sekvenci, ale v každém zopakování změní vloženou proměnnou. Tvoří tak určitý spád příkazů. Koukněme se náš příklad. Pro ***pořadí0*** od 0 do 4 má příkaz něco udělat. Jelikož se jedná o rozsvícení LED a proměnná je na souřadnici x, tak se bude postupně rozsvěcovat prostřední řada. Souřadnice y je zde konstantní, hodnota 2. Teď jsme tu popisovaly for cyklus pro tlačítko A. Pro tlačítko B je situace v podstatě stejná akorát se zde mění souřadnice y, tudíž se vykresluje prostřední sloupec. Dále je zde akorát vložená podmínka, která zhasíná prostřední bod. Pokud je vykreslený prostřední sloupec a začneme vykreslovat prostřední řadu, tak je zde efekt určitého proboření či vymazání políčka. Toto políčku poté můžeme doplnit stisknutím loga. Dále tady máme příkaz při dlouhém stisknutí loga. Tato část funguje tak, že se v cyklu mění jak souřadnice x, tak souřadnice y. LED diody se tedy rozsvěcují do úhlopříčky. Dále následuje v podstatě totožná část kódu, která LED akorát zhasíná.

```
from microbit import *
poradi0 = 0
poradi1 = 0
poradi2 = 0
# uvítací sekvence
sleep(200)
display.show(Image.HEART, delay = 500)
display.clear()

while True:
    # při hlasitém zvuku obrazovka zhasne
    if microphone.current_event() == SoundEvent.LOUD:
        display.clear()
    # akce při stisknutí tlačítka a
    if button_a. is_pressed():
        for poradi0 in range (0, 5, 1):
            display.set_pixel(poradi0, 2, 9)
            sleep(500)
            if display.get_pixel(2,2) == 9 and poradi0 == 2:
                display.set_pixel(2,2,0)
    # akce při stisknutí tlačítka b
    if button_b. is_pressed():
        for poradi1 in range (0, 5, 1):
            display.set_pixel(2, poradi1, 9)
            sleep(500)
    # akce při stisknutí loga
    if pin_logo.is_touched():
        display.set_pixel(2, 2, 9)
    # akce při zatřesení
    if accelerometer.was_gesture('shake'):
        display.clear()
        for poradi3 in range (0, 5, 1):
            display.set_pixel(poradi3, poradi3, 9)
            sleep(500)
        for poradi3 in range (0, 5, 1):
            display.set_pixel(poradi3, poradi3, 0)
            sleep(500)
```

Obrázek 13 - Animace displeje (kód) *Zdroj: Vlastní*

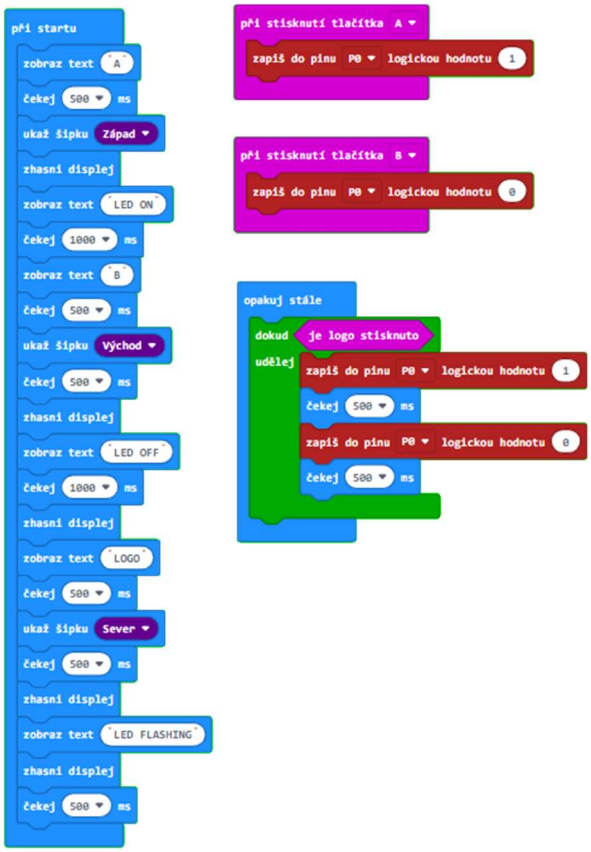
Na prvním řádcích si opět definujeme proměnné a napíšeme si zde nějakou uvítací část. Dále vyřešíme zhasnutí displeje při hlasitém zvuku. V kódu poté po příkazu ***for x in range*** následuje závorka se třemi hodnotami. První hodnota je počáteční proměnná cyklu. Druhou hodnotu poté volíme o jedno menší než jaké číslo chceme vyvolat jako poslední. Poslední číslo je určitá posloupnost. Pokud je tam 1, jako v našem případě, cyklus for bere každé číslo rozsahu (pro 1 se nemusí uvádět). Pokud bychom ovšem do této hodnoty napsali číslo 2, tak by se vyvolávalo každé druhé číslo z daného rozsahu.

V kódu také využíváme příkazu ***display.set_pixel(x, y, z)***. Tento příkaz nám rozsvěcuje konkrétní LED, ***x*** a ***y*** jsou souřadnice, ***z*** je intenzita osvětlení (0 až 9). Displej micro:bit je dán tak, že souřadnice **0, 0** jsou **vlevo nahoře**. Souřadnice **4, 4** jsou poté **vlevo dole**. Funkčnost a možnosti této animace ukazuji v tomto videu: https://youtu.be/KqzDni-IHU4

3.5.4 LED ON/OFF

Další úkol jsem pojmenoval LED ON/OFF, protože v něm budeme postupně rozsvěcovat LED diodu, ale ne na displeji, ale na nepájivém poli. Budeme se totiž seznamovat s piny. V našem případě si na nepájivé pole připojíme LED diodu s ochranným rezistorem (požívejte 100 – 1k Ω). Pin GND půjde na katodu LED (kratší vývod) a pin0 půjde na anodu LED (delší vývod). Na uchycení pinů jsem použil „krokodýlky".

V tomto programu si napřed vytvoříme jakýsi popis a návod při startu našeho programu. Když stiskneme tlačítko A, tak se LED rozsvítí. Pokud stiskneme tlačítko B, tak se má LED zhasnout. Když nás klikání již přestane bavit, tak můžeme využít cyklu, který nám bude LED rozsvěcovat a zhasínat po určitých intervalech. Tento cyklus poběží, dokud nepustíme zmáčknutí loga micro:bitu.



Obrázek 14 - LED ON/OFF (bloky) *Zdroj: Vlastní*

Sekvence při startu vždy zobrazí název vstupu, ukáže na něj pomocí šipek a heslovitě napíše, co daný vstup dělá. LED funguje tak, že svítí, pokud je na katodě logická nula (ta tam bude vždy, protože je připojená na pin GND) a na anodě je logická jednička. Tato logická jednička se na anodu dostanu při stisku tlačítka A.

V bloku opakuj stále je poté cyklus, který se bude opakovat, pokud bude stisknuto logo micro:bitu. Stav LED diody se budou měnit každou polovinu sekundy. Pojďme se ještě podívat, jak by tato situace vypadala v kódu.

```
1
2  from microbit import *
3
4  # popis funkce programu
5  display.show("A")
6  sleep(1000)
7  display.show(Image.ARROW_W)
8  sleep(1000)
9  display.clear()
10 display.scroll("LED ON", delay = 100)
11 display.show("B")
12 sleep(1000)
13 display.show(Image.ARROW_E)
14 sleep(1000)
15 display.clear()
16 display.scroll("LED OFF", delay = 100)
17 sleep(200)
18 display.scroll("LOGO", delay = 100)
19 display.show(Image.ARROW_N)
20 sleep(1000)
21 display.clear()
22 display.scroll("LED FLASHING", delay = 100)
23 display.clear()
24
25 # logika ON/OFF
26 while True:
27     if button_a.is_pressed():
28         pin0.write_digital(1)
29     if button_b.is_pressed():
30         pin0.write_digital(0)
31     while pin_logo.is_touched():
32         pin0.write_digital(1)
33         sleep(500)
34         pin0.write_digital(0)
35         sleep(500)
36
```

Obrázek 15 - LED ON/OFF (kód) *Zdroj: Vlastní*

Při zobrazování textu a obrázků je potřeba si dávat pozor na jednu věc. Pokud máme příkaz ***display.show()***, tak se něj musíme dát příkaz ***sleep*** o nějaké hodnotě. Pokud bychom dali pouze nějaký delay jako u ***display.scroll()***, tak se nám daný prvek ani nezobrazí.

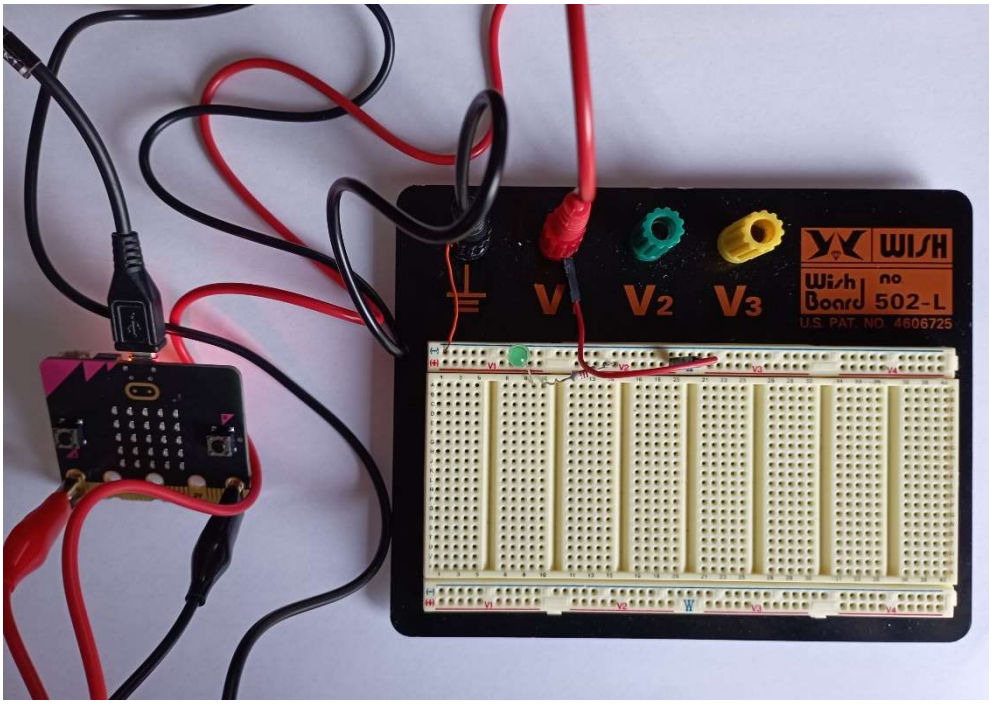
Dále zde máme logicku samotného vypínání a zapínání LED. Tady je důležité uvést správné číslo pinu a také logickou hodnotu. Toto jsou celkem stěžejší parametry tady těchto příkazů.

U příkazu while se poté neustále střídá vysoká a nízká úroveň napětí.

Těch pinů se může využít mnohem víc a můžou vzniknout velice komplexní program s několika vstupy a výstupy. Naším hlavním cílem je si tuto funkci hlavně představit a odzkoušet na nějkém poměrně jednoduchém příkladu.

Při zacházení s těmito piny je také důležité, abychom nepropojili GND s nějakým pinem, na kterém bude vysoká úroveň napětí (logická 1) bez jekékoli zátěže, v takovémto případu by mohlo dojít k tzv. zkratu, tento zkrat by mohl mít za následek poškození zařízení.

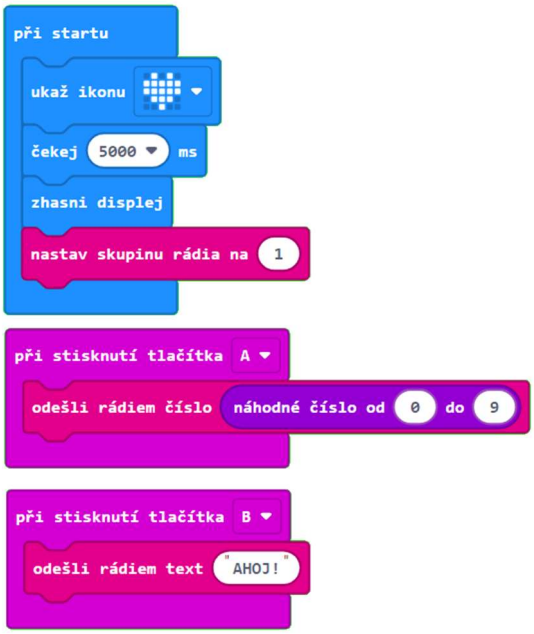
Na řešení, funkci a zapojení tohoto úkolu se můžete kouknout na tomto odkazu: https://youtu.be/sMX9eoo5lo4



Obrázek 16 - Zapojení micro:bitu a nepájivého pole *Zdroj: Vlastní*

3.5.5 Představení rádia

V tomto úkolu si představíme funkce rádia. Na tento úkol je potřeba mít micro:bity dva, což může být problém. Z tohoto důvodu jsem tento úkol zařadil na konec našeho micro:bitového pátrání. Já naštěstí dva micro:bity mám, takže Vám tento úkol můžu zprostředkovat. K tomuto úkolu budou tedy potřeba dva kódu přijímací a odesílací. Při stisku tlačítka A se pošle náhodné číslo od 0 do 9. Při stisku tlačítka B se pošle zpráva „AHOJ!“.



Obrázek 17 - rádio vysílač (bloky) *Zdroj: Vlastní*

Obrázek 18 - rádio přijímač (bloky) *Zdroj: Vlastní*

Při řešení takového úkolu s rádiem je důležité se ujistit, že oba dva micro:bity jsou nastaveny na stejné skupině rádia, v našem případě je to skupina 1. Oba dva micro:bity mají za úkol ze začátku zobrazit jiný obrázek, to je proto, abychom věděli, který vysílá a který přijímá. Přijímač po přijetí zprávy zprávu zobrazí a poté ještě ukáže určitou ikonu. Vše Vám bude popřípadě jasné po zhlédnutí videa.

```
1
2  from microbit import *
3  import radio
4
5  # uvítací sekvence pro rozlišení VYSÍLAČ
6  radio.on()
7  display.show(Image.HEART)
8  sleep(5000)
9  display.clear()
10 radio.config(group=1)
11
12 # posílání zpráv
13 while True:
14     if button_a.is_pressed():
15         radio.send("AHOJ!")
16
```

Obrázek 19 - rádio vysílač (kód) *Zdroj: Vlastní*

```
1
2  from microbit import *
3  import radio
4
5  # uvítací sekvence pro rozlišení PŘIJÍMAČ
6  radio.on()
7  display.show(Image.HAPPY)
8  sleep(5000)
9  display.clear()
10 radio.config(group=1)
11
12 # přijímání zpráv
13 while True:
14     zprava = radio.receive()
15     if zprava:
16         display.scroll(zprava)
17         display.show(Image.HAPPY)
18         sleep(1000)
19
```

Obrázek 20 - rádio přijímač (kód) *Zdroj: Vlastní*

Rádio v kódu si uděláme mírně jednodušší. Nejdříve musíme funkce rádia vůbec importovat do našeho kódu. Poté je také zásadní rádio zapnout. Obě rádia opět naladíme na stejnou skupinu, aby se byly schopni domluvit. Pokud u vysílače zmáčkneme tlačítko A, tak se pošle zpráva „AHOJ!“.

Kód u přijímače už je trochu o trochu zajímavější. Zde je začátek v podstatě totožný jako u vysílače. Změna, logicky, nastane u přijímání samotné zprávy. Vytvoříme si zde proměnnou ***zprava***. Pokud rádio opravdu nějakou zprávu přijme, tak se tato proměnná vyhodnotí jako True. Tímto se splní podmínka a zpráva se na displeji postupně zobrazí. Po této správě již micro:bit akorát zobrazí smajlíka, kterým potvrdí správné dokončení spojení.

Do takového rádia bychom toho mohli poslat nesmírně moc. Mohli bychom například využít funkcí hudby, které micro:bit nabízí a navzájem si posílat různě upravenou hudbu a melodie.

Na tuto úlohu se můžete podívat v tomto videu: https://youtu.be/BW_6vqW_TSA

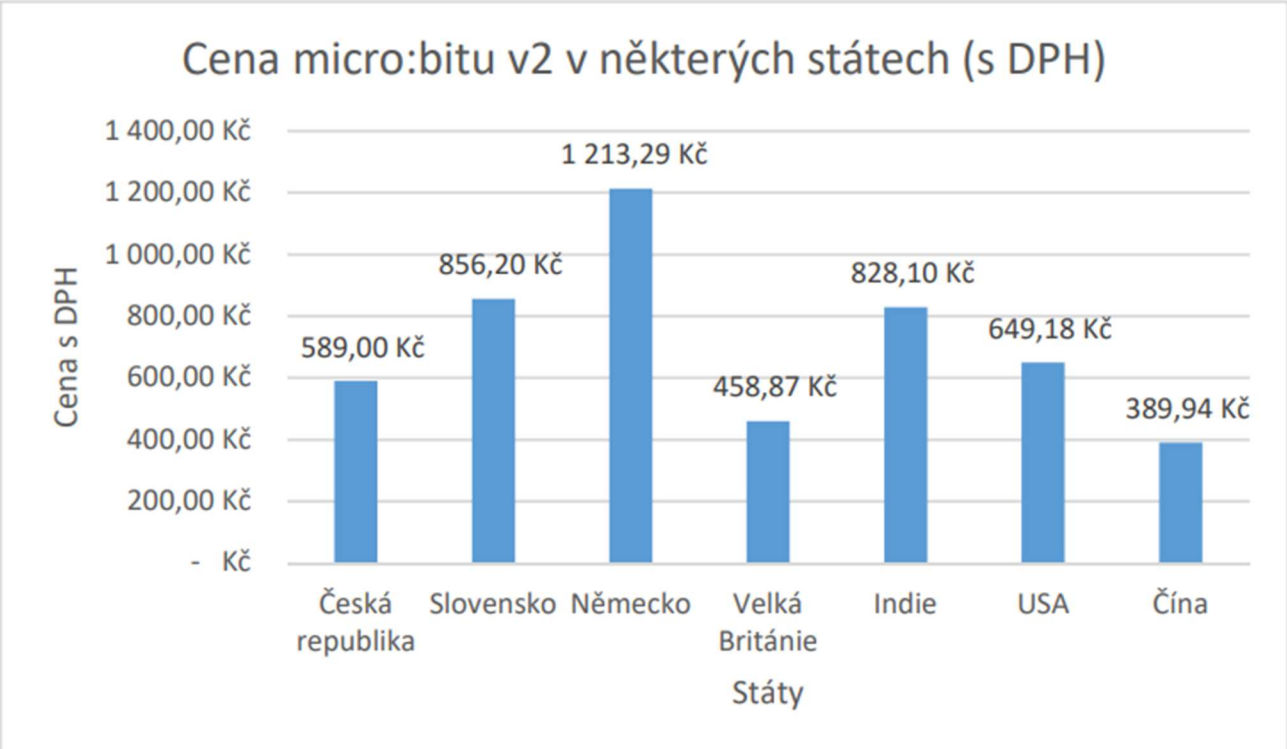
Touto úlohou jsou naše cvičení úspěšně ukončené, snad Jste se něco dozvěděli a naučili. A hlavně doufám, že Vás to **bavilo**.

3.6 Porovnání cen micro:bitu u nás a v zahraničí (tabulka a graf)

Ceny budu vybírat na různých webech, které působí v daném státě. Uvedu cenu s DPH a bez DPH. Uvedené státy budou ty největší a pro nás nejzajímavější. Hledané produkty jsou micro:bity verze 2. Ceny ze zahraničí převedu na českou korunu a poté s nimi budu pracovat. Zvýrazním největší a nejmenší ceny a uvedu některé průměry. DPH u USA je jakýsi průměr všech států. Z grafu krásně vyplívá, že nejdražší micro:bit je v Německu a nejlevnější v Číně.

Stát	Cena s DPH (měna státu)	Cena s DPH (česká měna)	DPH	Cena bez DPH (česká měna)
Česká republika	589,00 Kč	**589,00 Kč**	21%	486,78 Kč
Slovensko	35,99 €	**856,20 Kč**	20%	713,50 Kč
Německo	51,00 €	**1 213,29 Kč**	19%	1 019,57 Kč
Velká Británie	£ 16,50	**458,87 Kč**	20%	382,39 Kč
Indie	₹ 3 185,00	**828,10 Kč**	18%	701,78 Kč
USA	$ 29,93	**649,18 Kč**	5%	617,83 Kč
Čína	¥ 128,27	**389,94 Kč**	13%	345,08 Kč
Průměr	-	**712,08 Kč**	-	**609,56 Kč**
Nejvyšší hodnota	-	**1 213,29 Kč**	-	**1 019,57 Kč**
Nejnižší hodnota	-	**389,94 Kč**	-	**345,08 Kč**

Tabulka 1 - Tabulka cen micro:bitu u nás i v zahraničí *Zdroj: Vlastní*



Graf 1 - Graf porovnávající ceny micro:bitu u nás i v zahraničí *Zdroj: Vlastní*

4 Závěr

Tento projekt má sloužit jako studijní materiál a je určen pro ty, kteří ještě s programováním neměli tu čest začít, nebo opravdu jen trochu. Když se zpětně kouknu na svou odvedenou práci na tomto projektu, tak mám pocit, že to co jsem chtěl vypracovat, tak jsem vypracoval. Práce s micro:bitem mě bavila. Projekt takového rozsahu jsem ještě nikdy nevytvořil a jsem rád, že jsem se do toho s motivací dobře odvedené práce v předmětu VYT mohl pustit. Nejvíc mě naplňuje fakt, že tato práce, doufám, měla nějaký smysl a bude relevantní i v delším časovém horizontu. Bylo skvělé pracovat se všemi znalostmi z různých programů, které jsem v hodinách výpočetní techniky pochytil za poslední dva roky. Je pro mě až neuvěřitelné, jak moc jsem se od základní školy posunul ve zvládání digitálních technologií. Znalosti, které jsem při vytváření tohoto projektu získal, si rozhodně ponesu do jiných předmětů na střední škole, popřípadě i dál.

V projektu jsem využil tyto programy: Microsoft Word, Microsoft Excel, Inkscape, Adobe Photoshop, DaVinci Resolve, Visual Code Studio.

Celý projekt poté shrnuji v tomto videu: https://youtu.be/AmIxPrUWv3M

5 Seznam obrázků, tabulek a grafů

6 Použité zdroje

H, B. Micro:bit ve výuce: programujeme s nadšením ve škole i v kroužku. *Microbiti* [online]. [cit. 2023-06-18]. Dostupné z: https://www.microbiti.cz/

BRICKHACKER, EMMET. BBC MICRO:BIT V2 UŽ KONCEM ROKU 2020. *Bastlírna HW Kitchen* [online]. 17. 10. 2020 [cit. 2023-06-18]. Dostupné z: https://bastlirna.hwkitchen.cz/bbc-microbit-v2-uz-koncem-roku-2020/

User guide: Python guide. *Micro:bit* [online]. [cit. 2023-06-18]. Dostupné z: https://microbit.org/get-started/user-guide/python/

Micro Bit. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 15. 5. 2023 [cit. 2023-06-18]. Dostupné z: https://en.wikipedia.org/wiki/Micro_Bit

MicroPython. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 1. 3. 2023 [cit. 2023-06-18]. Dostupné z: https://en.wikipedia.org/wiki/MicroPython