Gymnázium Přírodní škola, o.p.s.
Profilová práce — třída Omíkron
Nižší stupeň studia
2022/2023

Robin Kovář

Ozobot a Edison ve škole

Vedoucí práce: Filip Hložek

Datum odevzdání: 18. listopadu 2023

Úvod

Moje profilová práce pojednává o vytvoření návodu na naprogramování robotů Ozobota a Edisona. Píšu jí, protože programování je podle mě důležitá dovednost a je nejlepší s ním začít co nejdřív. IT technologiím se v dnešním světě už prakticky nevyhneme a je potřeba se naučit je používat. Programování, nejen že rozvíjí mozek (především logické a abstraktní uvažování – klíčové pro kterýkoliv obor lidské činnosti), ale je to další směr, kterým se mohou studenti vydat. Domnívám se, že je velmi důležité jim tuto cestu též nabídnout.

Aby programování mělo šanci zaujmout, je klíčové s ním začít co nejdříve. Nečekat až do chvíle, kdy má většina žáků jasno a už je zajímá jenom jejich budoucí obor. Na dvou robotech – Ozobot a Edison - studenti přímo uvidí, jak jejich kód funguje a lépe pochopí základní principy algoritmizace a programování. Žáci mohou snáze pochopit základní principy algoritmizace a programování na robotech, např. Ozobot a Edison, kterými se zabývám ve své profilové práci. Navíc to s roboty může

být i zábavnější. Moje profilová práce může sloužit jako brána k zábavnému programování i ukázka toho, co se dá v programování opravdu dělat. Hlavní cílem je představit zájemcům tyto dva roboty a ukázat jim tak možnou cestu, kterou se lze vydat. Moje práce je určená zájemcům z řad studentů od jedenácti let a také učitelům informatiky, kteří ji mohou využít ve svých hodinách.

Cíle

Stanovené cíle v záměru profilové práce byly tyto:

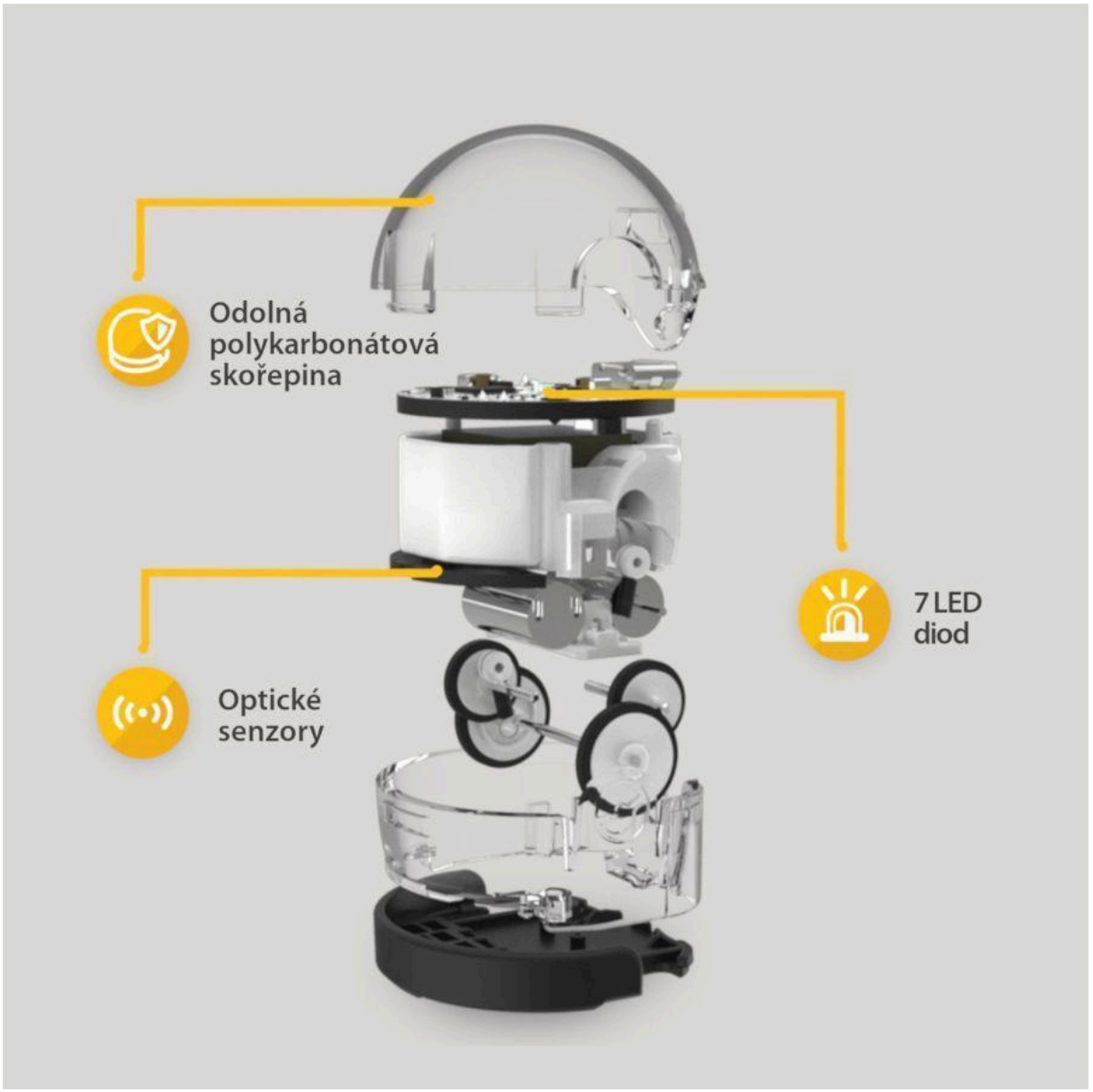
1) Vytvořit dva ukázkové kódy pro každého robota tak, aby z nich byly patrné možnosti obou robotů s využitím hlavních programovacích konstruktů: užití proměnných, cyklů, rozhodování a uživatelem definovaných funkcí.
2) Připravit materiál pro učitele na školách nebo samouky rozdělený do 5 výukových lekcí, a to včetně příkladů k samostatnému řešení.

LEKCE 1 – Seznámení s OZOBOTem

Popis

Ozobot je malý a jednoduchý robot. Lze ho programovat dvěma způsoby:

- Barevnými kódy
- Programovacím jazykem OzoBlockly

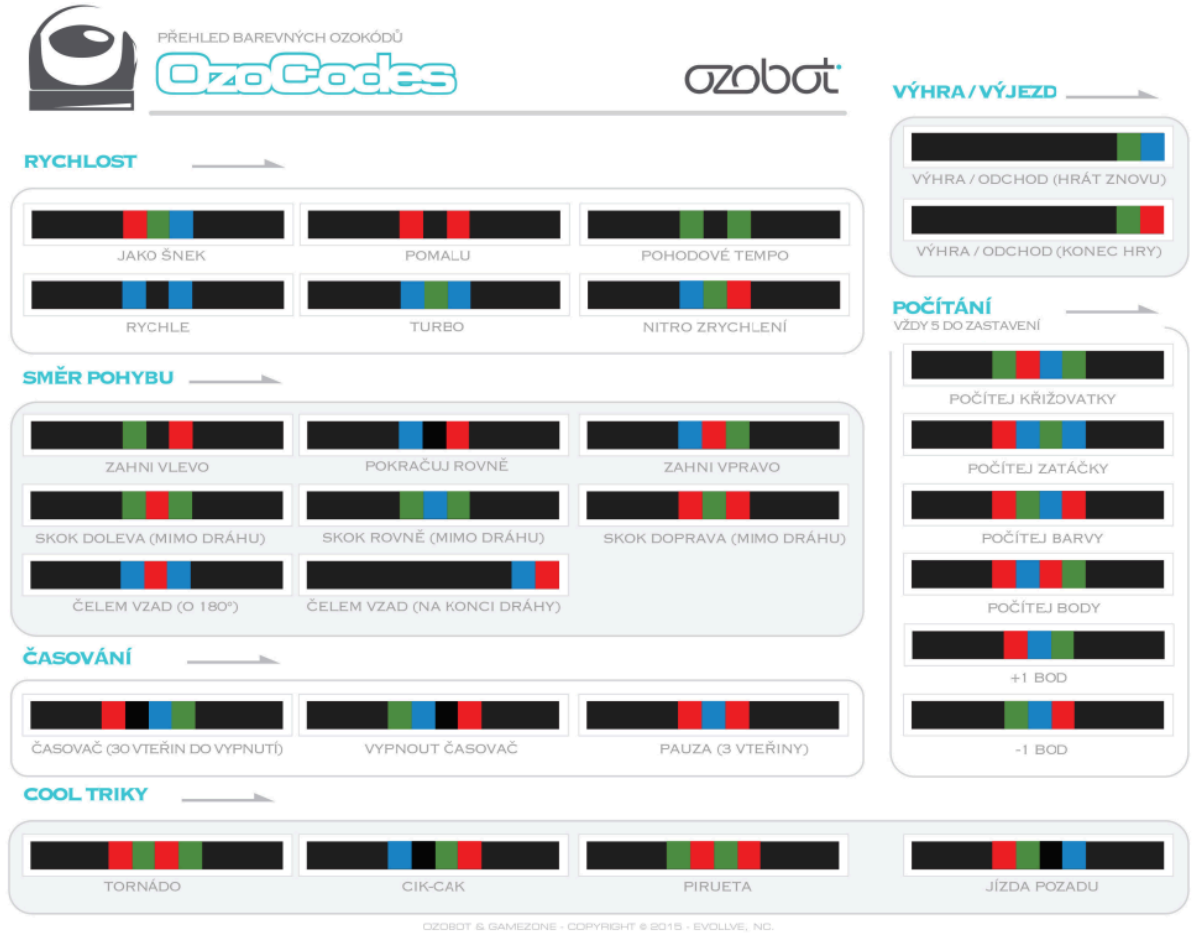


Obrázek 1 - Popis Ozobota

Ozobot má jeden senzor na snímání barev. Podle toho, na jaké barvě stojí, mění barvu jeho LED dioda, která je umístěna na jeho vršku. Pohybuje se díky sadě kol, které jsou poháněné bateriemi. Ozobot se nabíjí pomocí kabelu USB mini. Má také jedno tlačítko, aby ho bylo možné ovládat. Pokud se stiskne jednou, tak bude následovat černou čáru a reagovat na barevný kód. Pokud se stiskne dvakrát, začne plnit naprogramovaný kód.

Barevné kódy

Ozobot snímá barvy a podle nich se dá ovládat příkazy. Automaticky jezdí po černých čarách, pokud na cestě rozpozná barevný kód, vykoná příslušný příkaz. Možné příkazy jsou na obrázku č. 2.

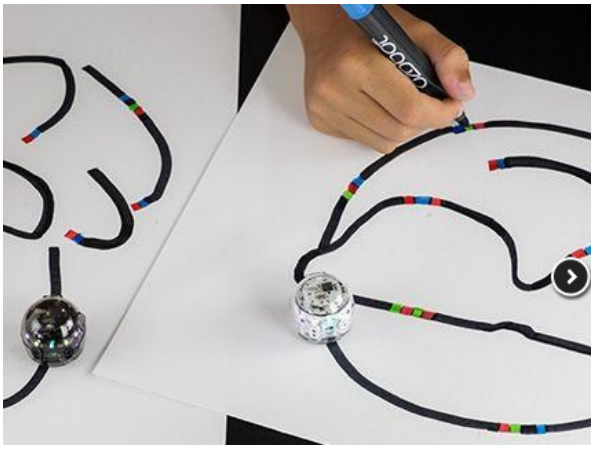


Obrázek 2 - barevné kódy [1]

V kódech se používá od dvou do čtyř barev: zelená, modrá, černá a červená. Pro tvorbu čárových kódů a černých čar je vhodný libovolně dlouhý papír, nejlépe v roli, a čtyři tlusté fixy se seříznutými hroty (například od Centropenu).

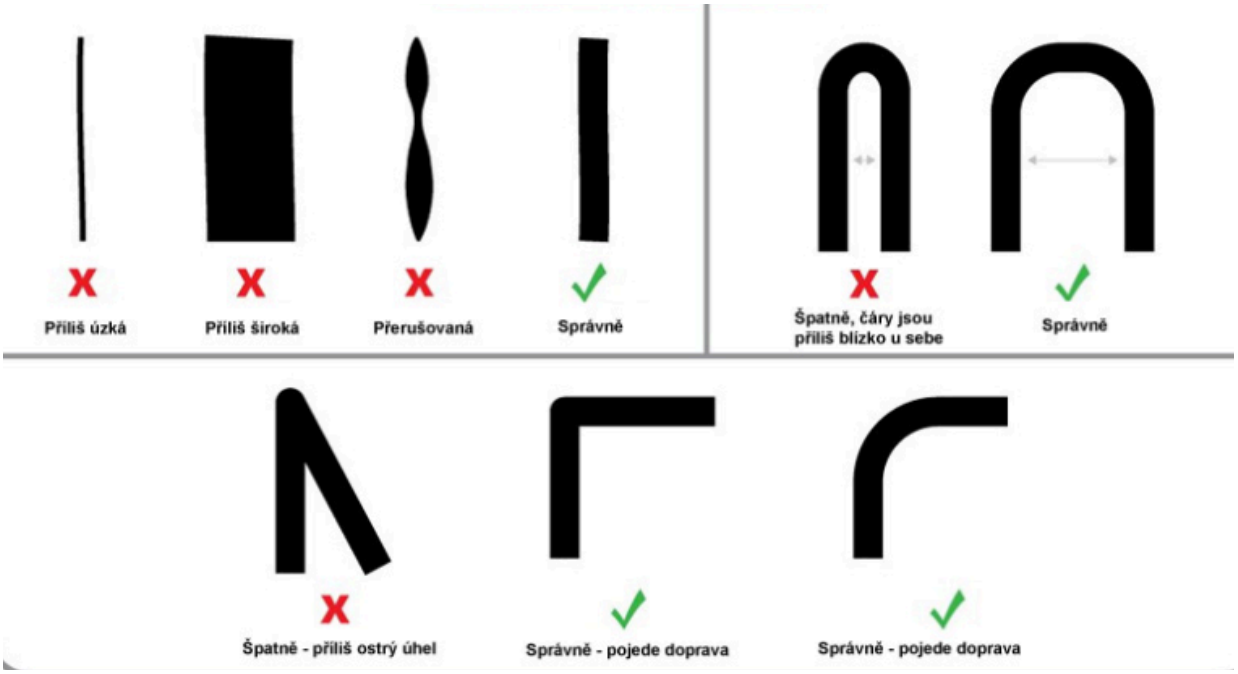
Kreslení vodící čáry s barevnými kódy

Při tvorbě vodících čar je nutné dodžovat několik málo podmínek při jejich kreslení, aby Ozobot kvalitně rozpoznával nejen směr jízdy, ale i předávané příkazy.



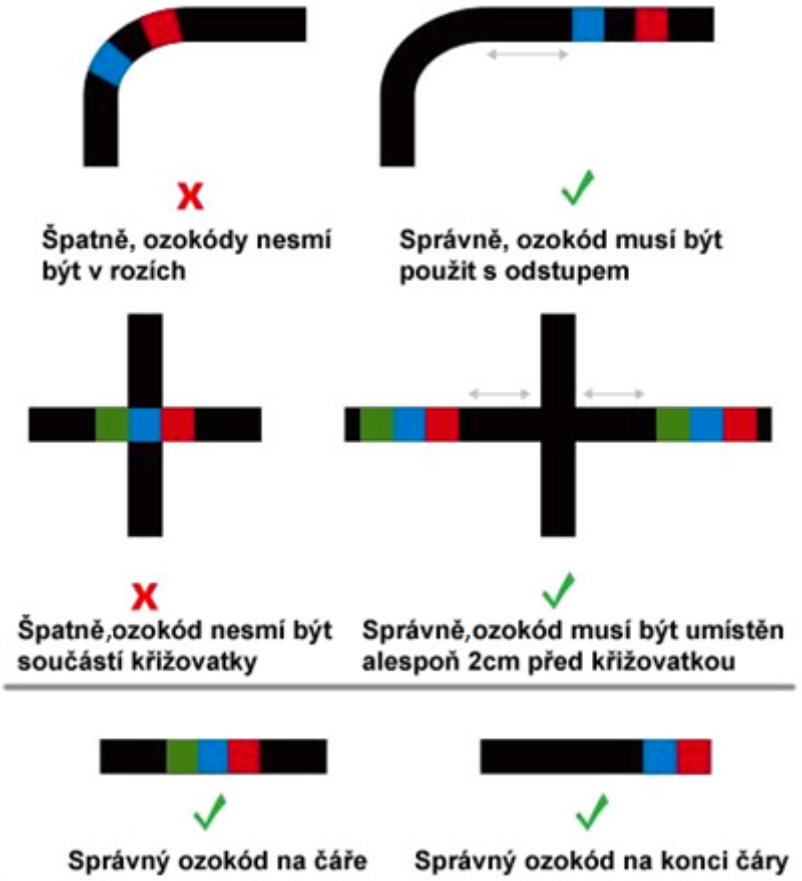
Obrázek 3 - kreslení vodící čáry s kódy

Níže přikládám návod pro správné kreslení vodící čáry a barevných kódů.



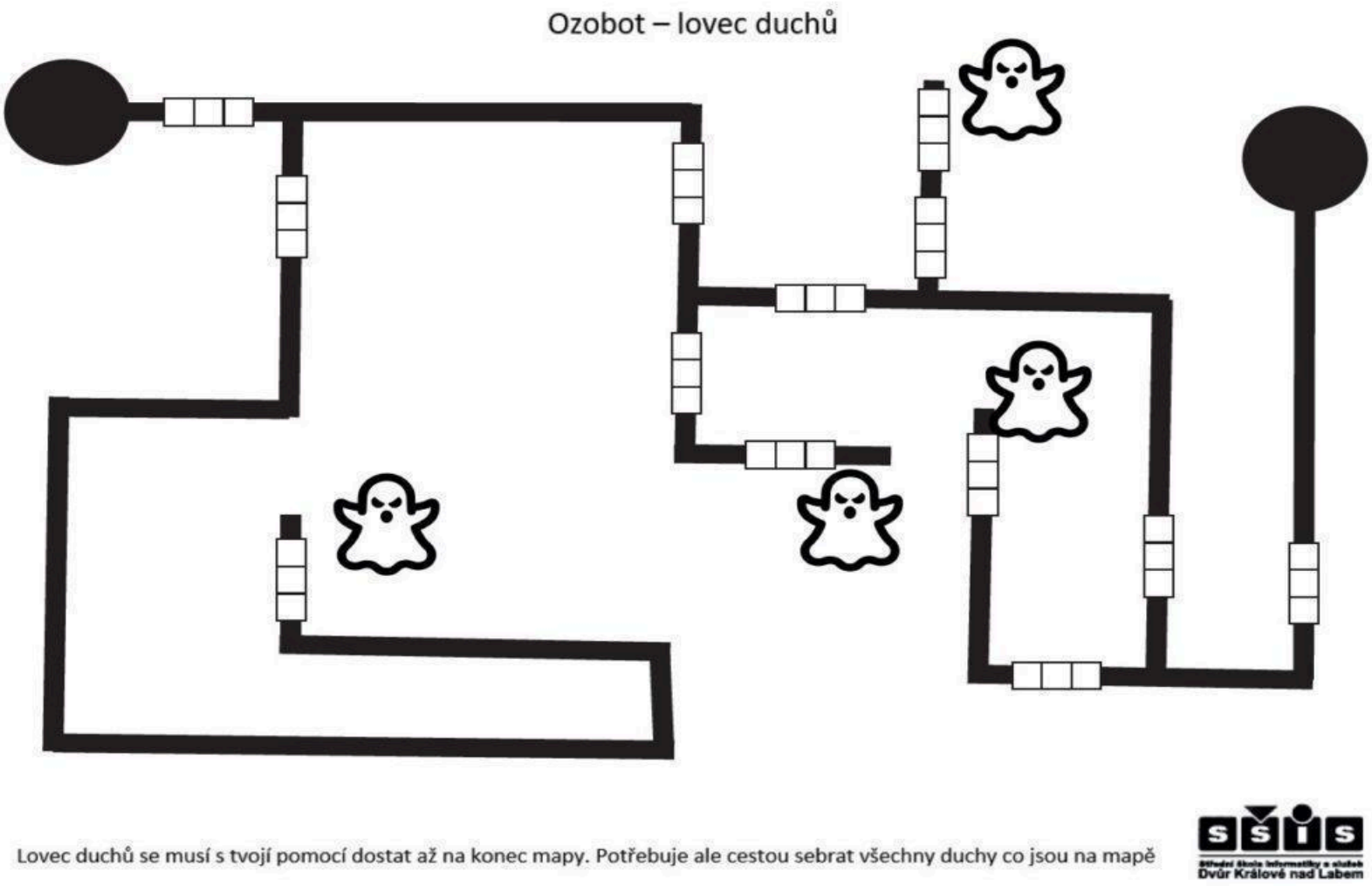
Obrázek 4 – Návod pro kreslení čáry [1]

Obrázek 5 – Správný ozokód [1]

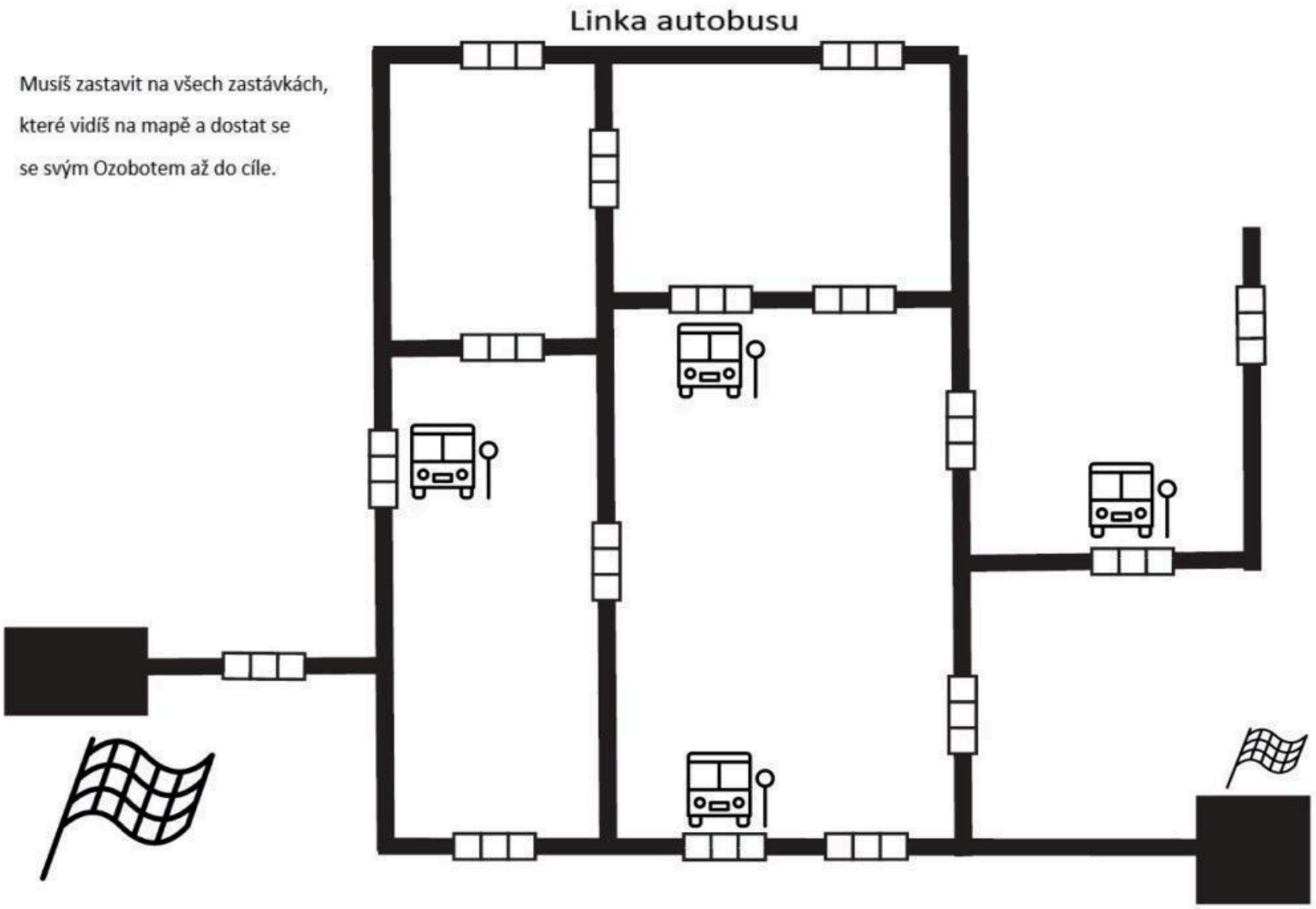


Obrázek 6 – Správný ozokód [1]

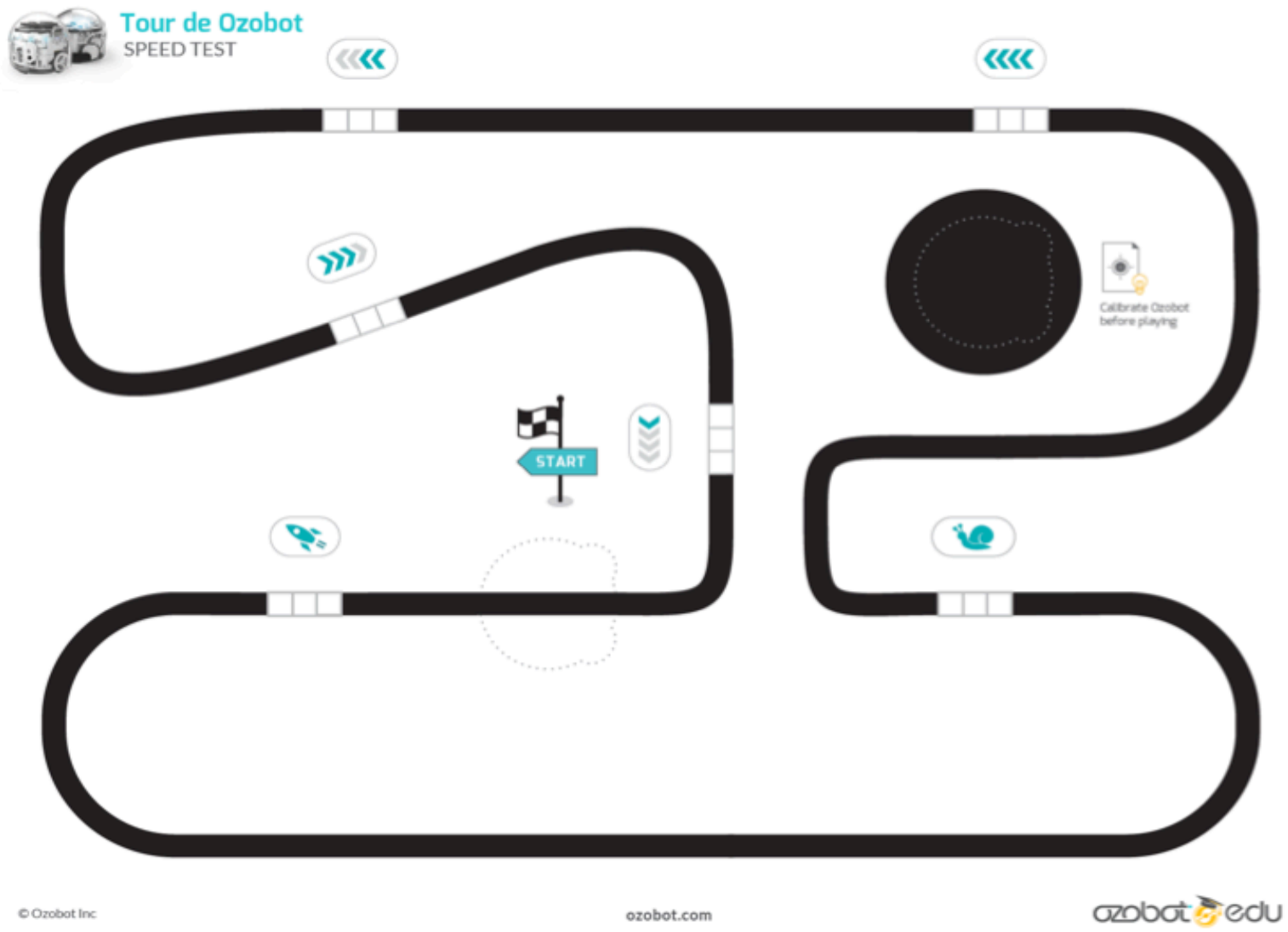
Úkoly pro Ozobota - barevné kódy



Obrázek 7 - Úkol 1 [2]



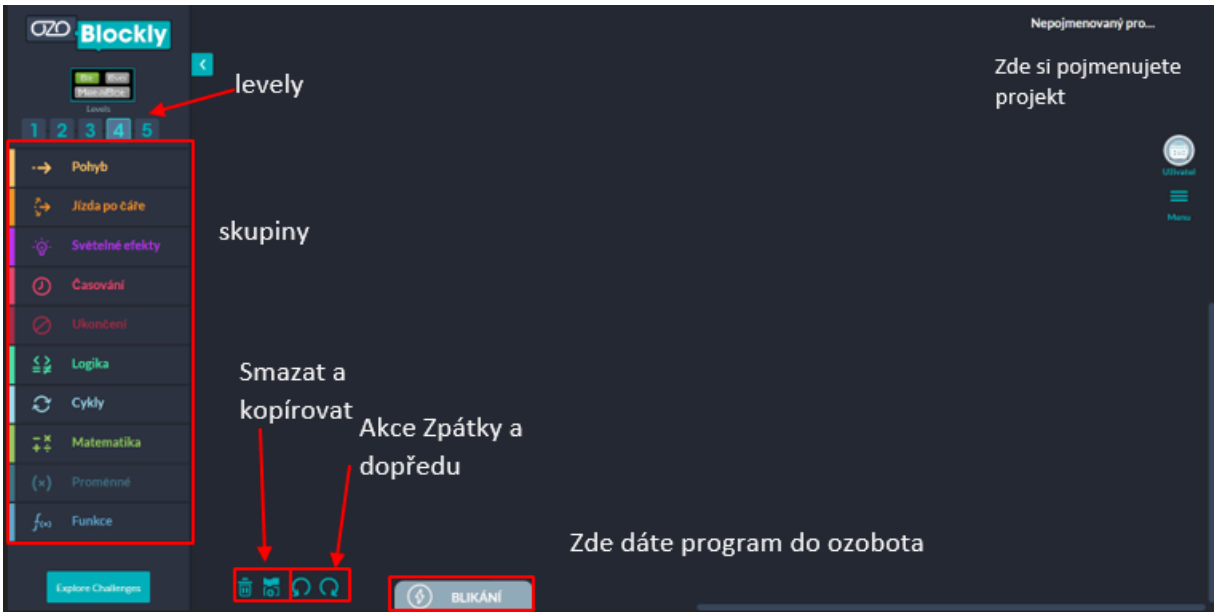
Obrázek 8 - Úkol 2 [3]



Obrázek 9 - Úkol 3 [4]

LEKCE 2 – OzoBlockly

Editor Ozoblockly



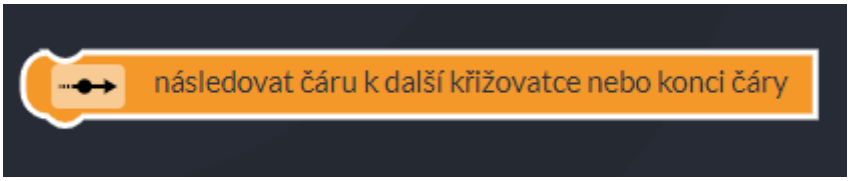
Obrázek 10 – Popis prostředí OzoBlockly [7]

OzoBlockly je programovací jazyk pro Ozobota. Velice se podobá Scratchi. Programuje se pomocí „bloků“, jak název napovídá. Otevřete si program editor Ozoblockly (https://ozoblockly.com/editor) a vyberte level 4. Nalevo vidíte 10 skupin bloků: pohyb, jízda po čáře, světelné efekty, časování, ukončení, logika, cykly, matematika, proměnné a funkce. Vždy vyberete jednu skupinu, kde máte nabídku bloků ve skupině. Blok vezmete a přetáhnete do pracovní plochy. Bloky se do sebe zacvakávají jako lego. Kód bude probíhat ze shora dolů. Když chcete blok odstranit, tak ho musíte vzít a přetáhnout zpět doleva (s blokem posouváte i všechny bloky, co se nachází pod ním, ale už ne ty, co jsou nad ním).

Druhy bloků

V OzoBlockly máme k dispozici tři hlavní druhy bloků, pomocí kterých vytváříme program pro Ozobota – normální blok, zasouvací blok a zacvakávací blok.

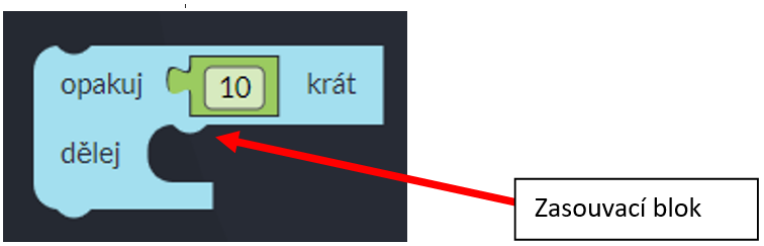
Normální blok



Obrázek 11 - Normální blok

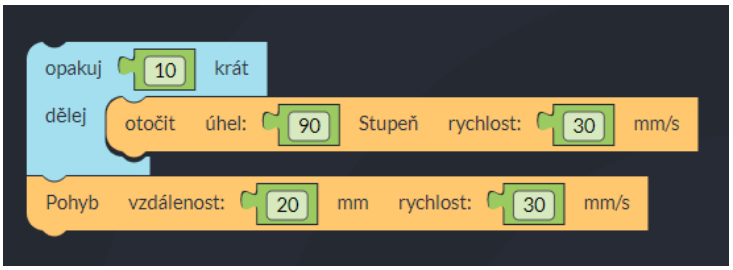
Tento typ bloku není ničím speciální. Vykonává popsaný úkol. Může v sobě obsahovat zacvakávací bloky, které představují parametry, např. dobu trvání akce nebo rychlost pohybu apod.

Zasouvací blok



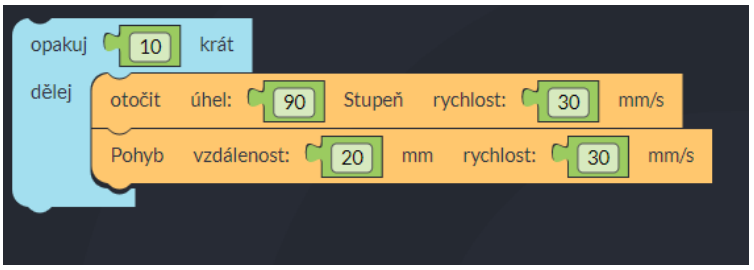
Obrázek 12 - Zasouvací blok

Tento typ bloku je speciální tím, že se do něj dají zacvakávat další bloky. Jeho efekt bude mít vliv jenom na bloky, co jsou v něm, tedy, když bude něco opakovat, tak bude opakovat jenom věci, co jsou v něm zacvaknuté, ale už ne ty, co jsou pod ním. Jakmile se jeho funkce dokončí, tak kód pokračuje dál.



Obrázek 13 – Příklad užití zasouvacího a normálního bloku

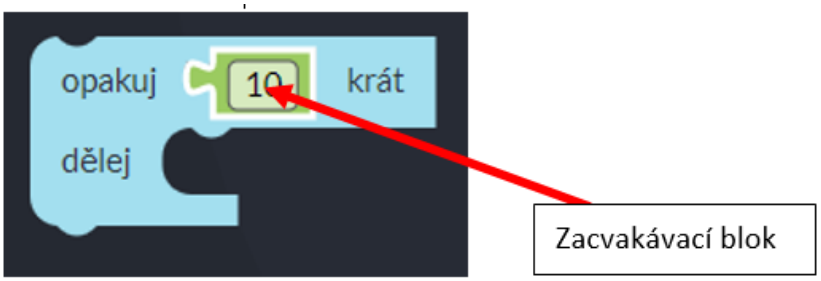
Co bude kód z obrázku č. 13 dělat? Tento velice jednoduchý kód bude dělat to, že Ozobot se 10 krát otočí o 90° a poté popojede 20mm dopředu.



Obrázek 14 – Příklad užití zasouvacího a normálního bloku

Zatímco v kódu na obrázku č. 14 se otočí o 90° a poté popojede 20mm. Tato sekvence se bude opakovat 10 krát.

Zacvakávací blok



Obrázek 15 - Zacvakávací blok

Tento typ bloku se zacvakává do míst, co vypadají jako puzzle. Dodává informaci – konkrétní hodnotu (např. kolikrát se má něco stát).

Skupiny bloků na čtvrté úrovni

Pohyb

V této skupině programujete Ozobota, kam má jít a jak se pohybovat (jakou rychlostí, dopředu, dozadu atd.). V 1. bloku nastavíte, jakou vzdálenost urazí a jak rychle (pokud uvedete číslo záporné, pojede dozadu). Ve 2. nastavíte, jak se má otočit. Ve 3. bloku můžete nastavit různé rychlosti pro pravé a levé kolo. 4. blok ho zastaví. 5. blok říká, že pojede, dokud nenalezne černou čáru.

Jízda po čáře

Pokud programujeme Ozobota pomocí vlastního programu v OzoBlockly, pak automaticky čáru nesleduje. Někdy však chceme, aby ji sledoval a tak lze použít bloky z této skupiny. Nabízí se několik možností: s 1. blokem sleduje čáru, dokud neskončí nebo nedojede na křižovatku. Ve 2. bloku rozhodnete, jakým směrem se má vydat (odpovídají barevným kódům na obrázku níže). V 3. bloku nastavujete rychlost, s jakou se bude na čáře pohybovat.

Obrázek 16 – Příklady barevných kódů pro směr pohybu

Světelné efekty

Bloky z této skupiny umožní volit, jakou barvou má Ozobot svítit. První blok vypne světla, s druhým blokem nastavíte, jakou barvou má svítit. Můžete si také navolit svojí barvu pomocí RGB.

Ukončení

V této skupině se nachází bloky, se kterými ukončíte program různými způsoby:

- ukončit + vypnout Ozobota,
- ukončit program + nečinný stav,
- ukončit program + následovat čáru.

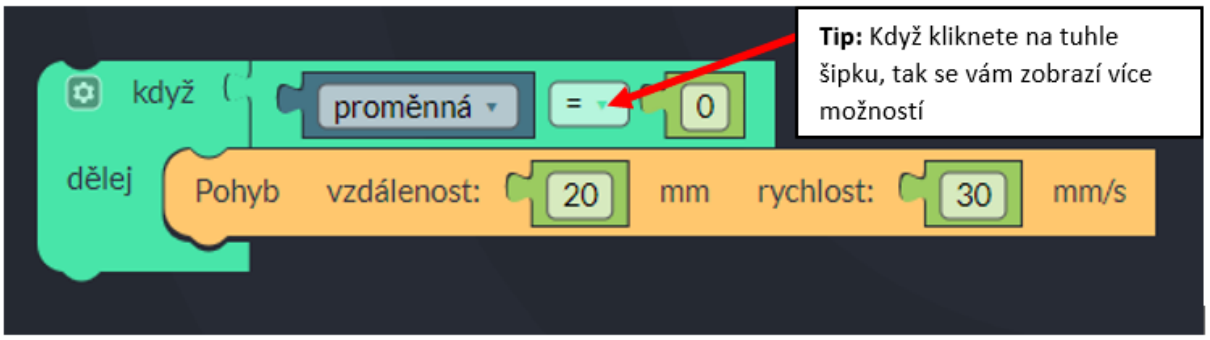
Časování

Zde si nastavíte, jaká doba uplyne mezi jinými bloky. Je k dispozici jenom jeden blok.

Logika

Ve skupině Logika jsou bloky pro logické rozhodování robota. Bloky zde mají místa, kam se mohou dodávat jiné bloky. Nelezneme zde bloky pro:

- Rozhodování (když – dělej)
- Tvorbu podmínky (testování platnosti výrazu)
- Logický součin a součet (AND a OR)
- Negaci
- Boolean hodnotu (Pravda/Lež)
- Podmíněné rozodování (když pravda – dělej větev 1, když lež – dělej větev 2)



Obrázek 17 – Příklad užití logických bloků

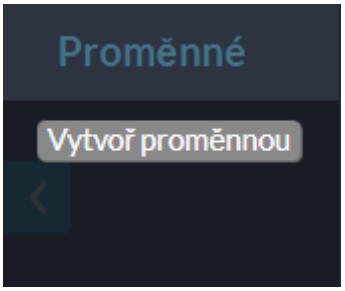
Cykly

Cykly jsou od toho, chceme-li něco opakovat. K dispozici máme následující bloky:

- Blok s předem definovaným počtem opakování
- Blok s opakováním, dokud je splněna podmínka
- Blok s parametricky definovaným počtem opakování
- Blok pro přerušení cyklu

Proměnné

Proměnné jsou hodnoty se jménem. Hodnoty v nich můžete v průběhu programu upravovat. Nejdříve si musíte proměnnou vytvořit. Kliknete na skupinu proměnné a poté kliknete na tlačítko vytvořit proměnnou.



Obrázek 18 – Vytvoření proměnné

Zadáte jméno, podle kterého se na ní budete odkazovat, může se jmenovat jakkoliv. Je vhodné si zavést pravidla pro pojmenování, abyste se v programu dobře orientovali. Jakmile si vyberete jméno, tak se vám tam hned objeví další dva bloky. Jeden vám umožní vaší proměnnou upravovat a druhý jí zase vyvolávat a někam jí zacvaknout.

Matematika

Tyto bloky mají funkci matematických operací (+, - , *, :). První blok je zacvakávací kostička s číslem.

Obrázek 19 - blok s číslem

Tento blok je v této skupině nejpoužívanější. Ostatní bloky mohou sčítat, odečítat, generovat náhodná čísla atd. Do těchto bloků se nemusí nutně zacvakávat jen bloky s číslem, ale i proměnné, které v sobě mají nějakou hodnotu.

Blikání

Výše jsme se seznámili se všemi skupinami ze čtvrté úrovně. Pořád jsme si ale neřekli jednu z nejpodstatnějších věcí. Jak ten kód vlastně do Ozobota dostat. Uděláte to tak, že si rozkliknete kartu Blikání, kterou najdete ve spodní části obrazovky.

Obrázek 20 – Červeně kalibrace, zeleně nahrávání Ozobota, žlutě načíst bit

Otevře se vám okno pro nahrávání kódu. Nastavte jas obrazovky na 100 % (důležité, nahrávání probíhá opto-elektronickou cestou). Poté si musíte Ozobota zkalibrovat. To uděláte tak, že ho přidržíte u obrazovky na bílém kruhu, přitom budete držet spouštěcí tlačítko, dokud Ozobot nezačne blikat bíle. Následně ho přidržíte v místě označeném zeleným obdélníkem a stisknete tlačítko načíst bit. Ukazatel vedle tlačítka bude ukazovat, jak dlouho bude ještě nahrávání trvat. To, že nahrávání proběhlo úspěšně, poznáte podle toho, že Ozobot začne blikat zeleně. Pokud ale bliká červeně, tak musíte nahrávání opakovat. Poté, co máte program nahraný v Ozobotovi, tak dvakrát po sobě stiskněte spouštěcí tlačítko a program se rozběhne. Tedy hlavně Ozobot se rozeběhne.

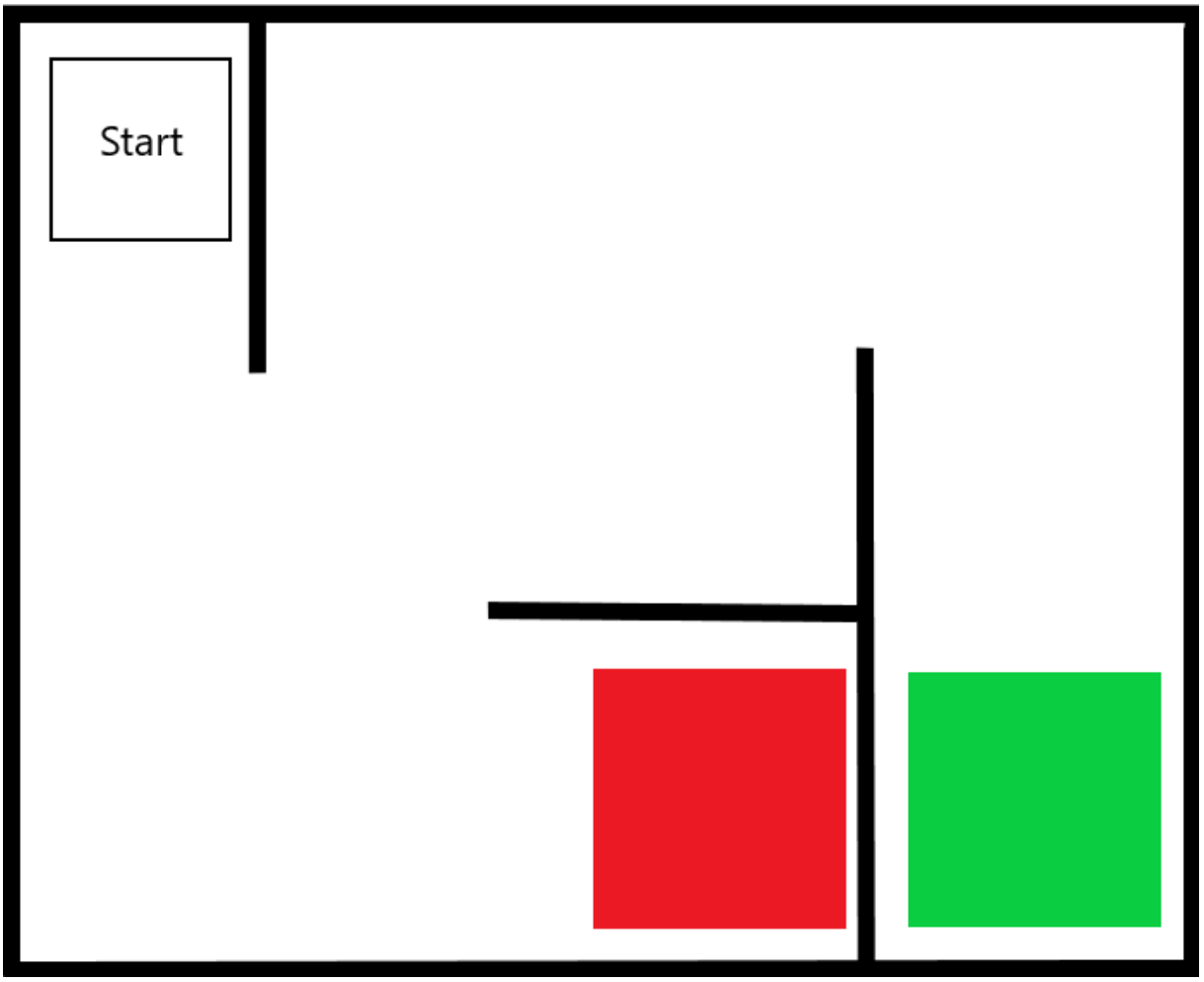
Úkoly pro Ozobota - OzoBlockly

Úkol 1

Obrázek 21 - Zadání úkolu 1 [5]

Ozobot se rozjede ve směru šipky z pozice Start. Když dojede k černé čáře, rozsvítí se červeně (a zastaví se). Po zastavení si Ozobot couvne a natočí se o náhodný úhel od 1° do 127°, když nebude u „ohradníku“ pokračuje vpřed a svítí modře. Pokud narazí na ohradník, rozsvítí se červeně, couvne, natočí se, pokračuje stále dokola.

Úkol 2

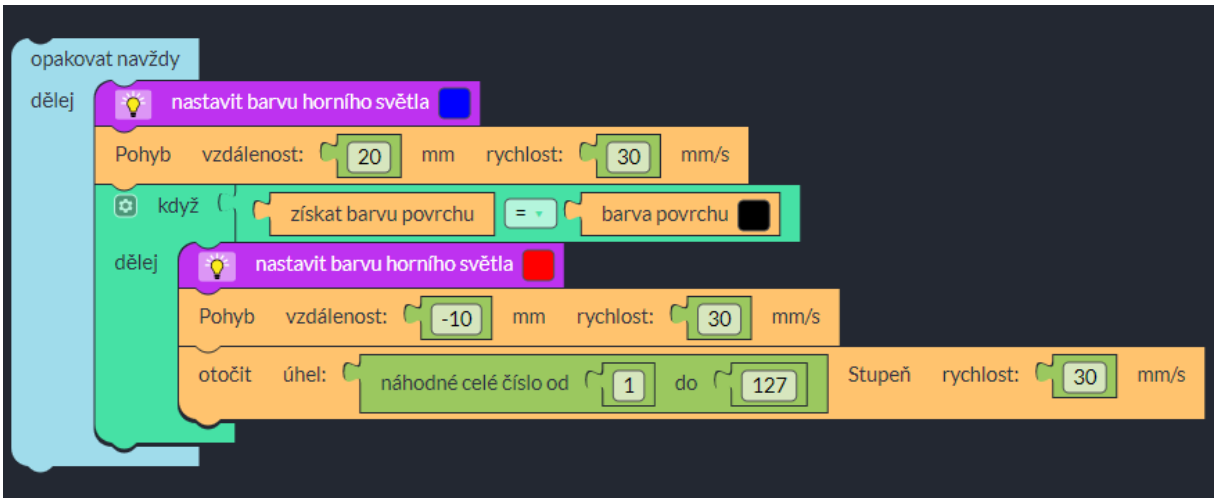


Obrázek 22 - Zadání úkolu 2

Tento úkol je podobný tomu předchozímu. Ozobot jede rovně, ale když narazí na černou čáru, tak se zastaví, popojede dozadu, otočí se o náhodný úhel od 1° do 127° a jede dál. Když Ozobot narazí do zeleného čtverce, tak končí program. Pokud ale narazí do zeleného čtverce předtím, než narazí do červeného čtverce, tak se nic neděje. Takže nejdřív musí do červeného a až potom do zeleného čtverce.

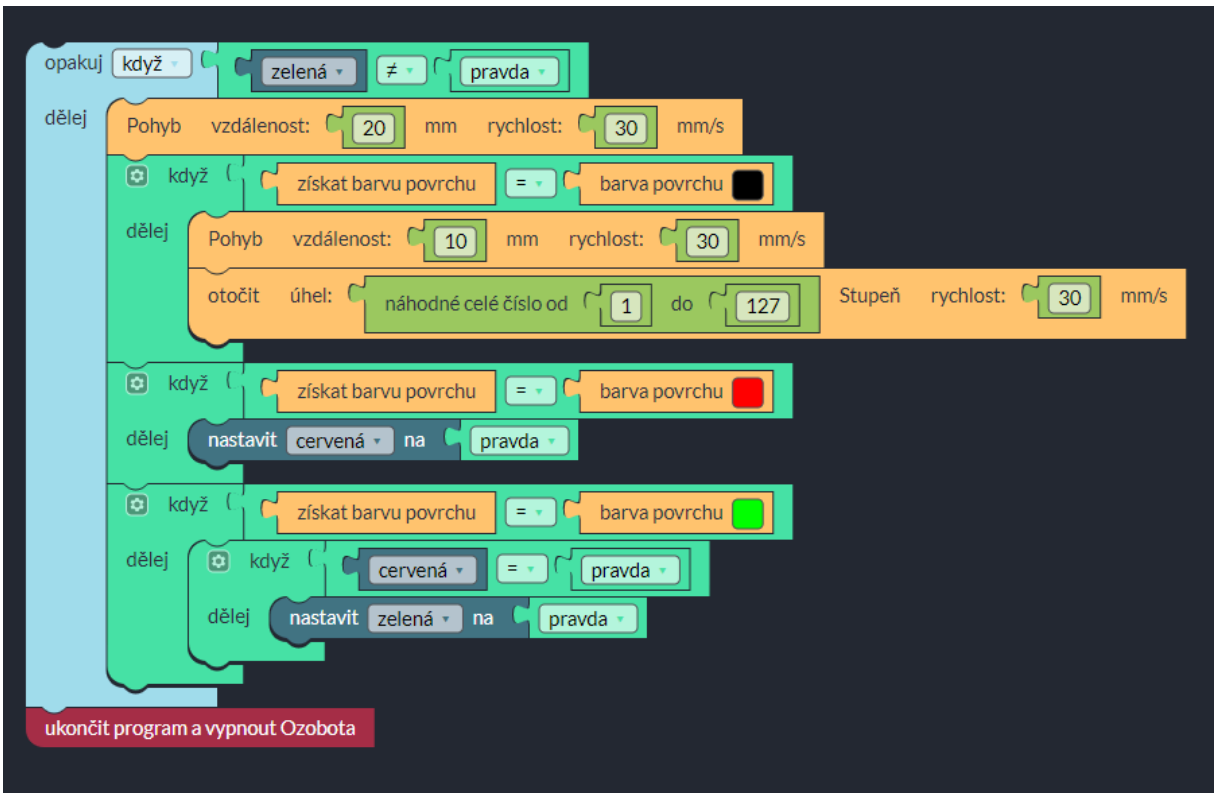
Řešení

Řešení úkolu 1



Obrázek 23 – Řešení úkolu 1

Řešení úkolu 2

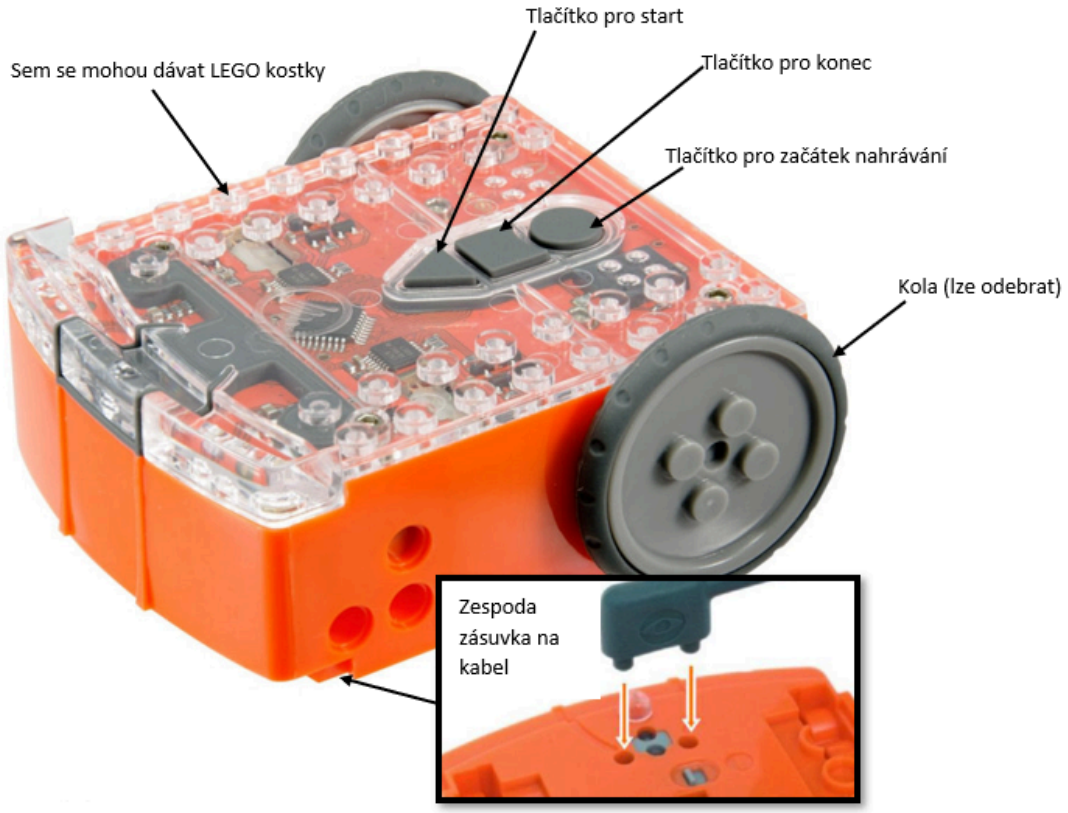


Obrázek 24 - Řešení úkolu 2

LEKCE 3 – Seznámení s Edisonem

Popis

Edison je složitější robot. Neprogramuje se pomocí bloků ani barevných kódů, ale pomocí programovacího jazyka EdPy. EdPy je modifikace jazyka Python určená speciálně pro Edisona. Je to tedy vstup do opravdového programování.

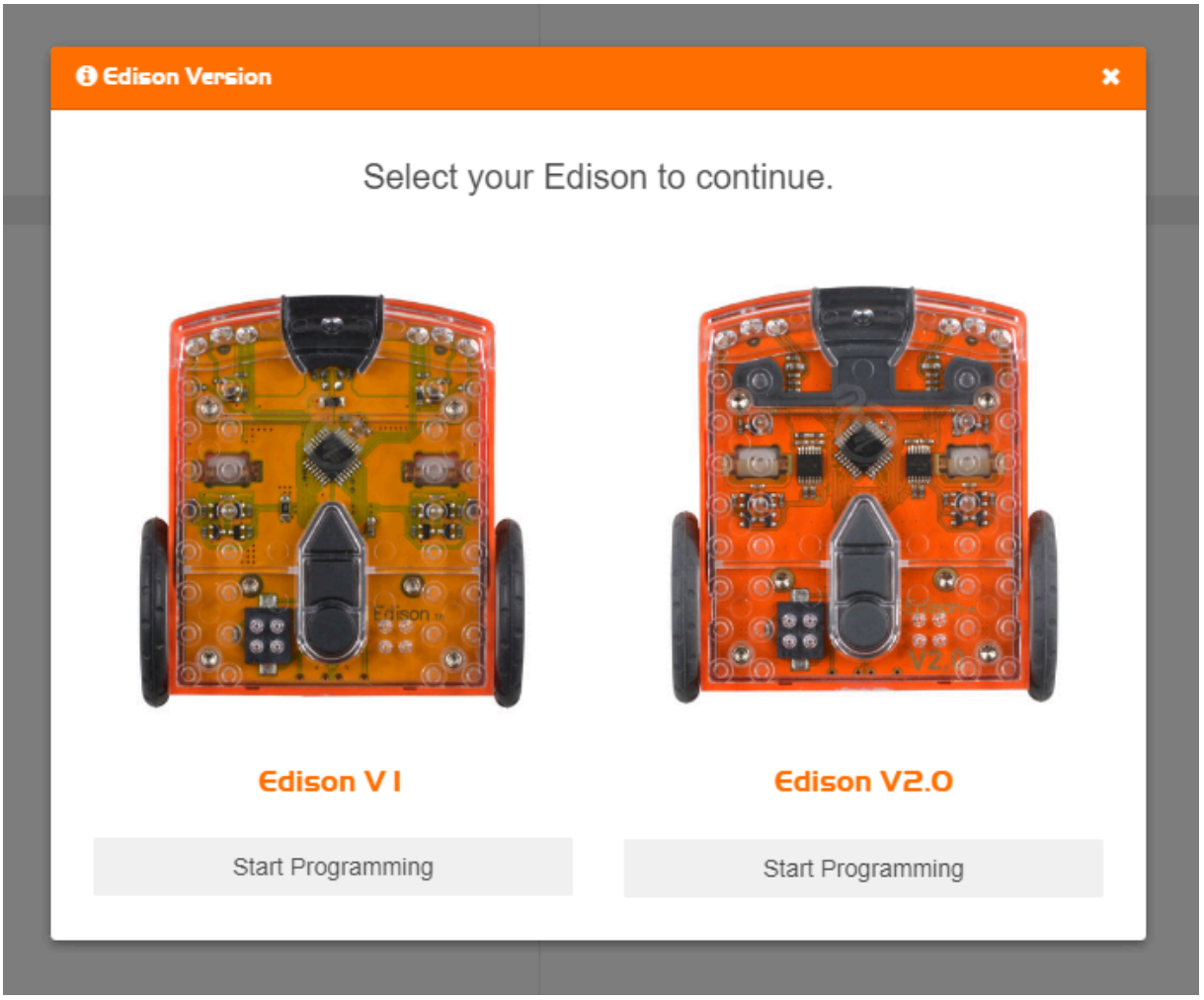


Obrázek 27 – Popis Edisona

LEKCE 4 – EdPy

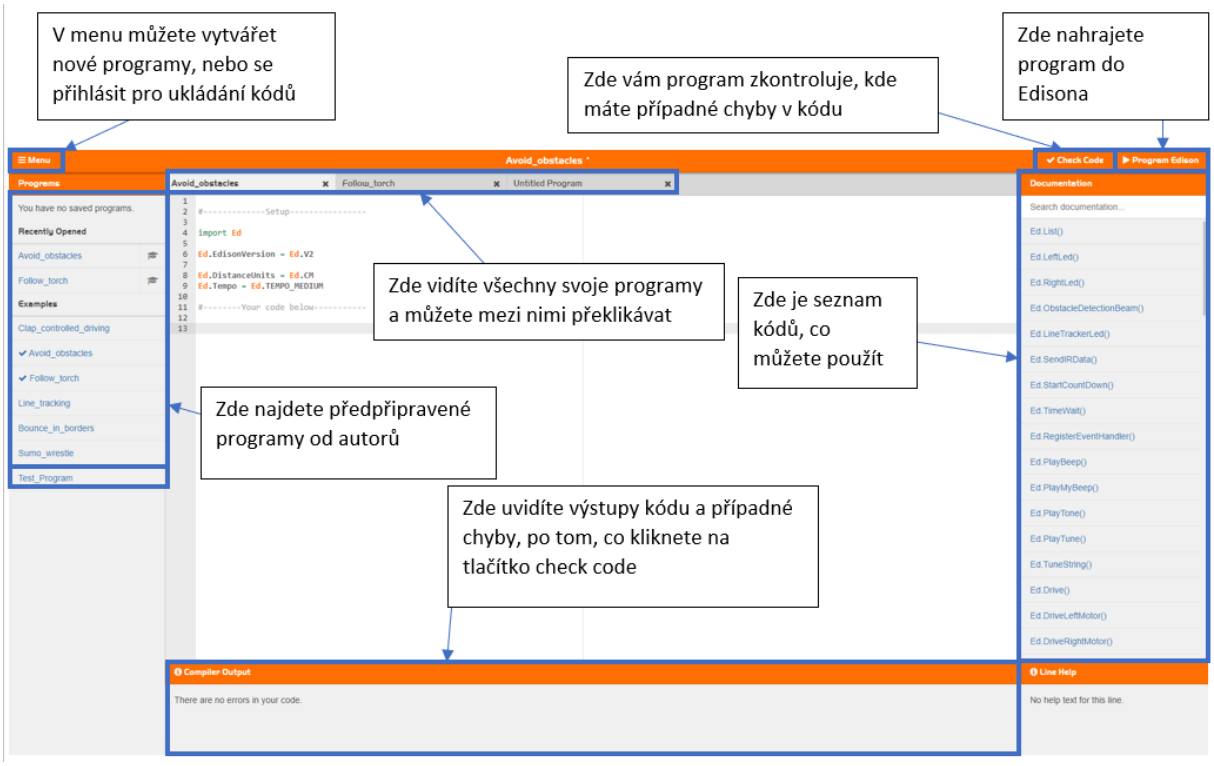
Jak v programu pracovat

Program běží v internetovém prohlížeči, najdete jej na této stránce: edpyapp.com Když si program otevřete, objeví se vám toto okno:



Obrázek 28- Výběr verze Edisona

Vyberte, jakou verzi Edisona máte k dispozici, a můžeme začít. S největší pravděpodobností budete mít druhou verzi.



Obrázek 29 – Popis grafického rozhraní EdPy [6]

Jak jste si již mohli všimnout, tak toto je již „textový“ program. Bloky jsou zde nahrazeny příkazy. Uprostřed máte řádkový editor, na kterém budete psát svůj kód.

Na ukázku si můžete otevřít testovací program, který naleznete v okénku předpřipravených programů. Jmenuje se Test_Program.

Nahrání kódu do Edisona

K Edisonovi jste dostali kabel, který do něj připojíte ze spodu. Druhý konec kabelu připojíte do počítače přes vstup pro sluchátka. Poté dejte hlasitost na maximum a stiskněte tlačítko program Edison. Až to doběhne, tak můžete kabel odpojit a program spustit.

Test_Program | Untitled Program

```
1
2  #-------------Setup----------------
3
4  import Ed
5
6  Ed.EdisonVersion = Ed.V2
7
8  Ed.DistanceUnits = Ed.TIME
9  Ed.Tempo = Ed.TEMPO_MEDIUM
10
11 #--------Your code below-----------
12
13
14
15 while True:
16     Ed.PlayBeep()
17     Ed.LeftLed(Ed.OFF)
18     Ed.RightLed(Ed.ON)
19     Ed.Drive(Ed.SPIN_RIGHT, 5, 350)
20     Ed.TimeWait(20, Ed.TIME_MILLISECONDS)
21     Ed.PlayBeep()
22     Ed.LeftLed(Ed.ON)
23     Ed.RightLed(Ed.OFF)
24     Ed.Drive(Ed.SPIN_LEFT, 5, 350)
25     Ed.TimeWait(20, Ed.TIME_MILLISECONDS)
26
```

Obrázek 31 - testovací program

Na první pohled si můžete na obrázku č. 28 všimnout, že před všemi příkazy musíte napsat Ed. Také, že po všech příkazech musíte psát závorky.

Ale otázka je, k čemu to je? „Ed.“ píšete proto, že využíváte procedury a fukce vyvinuté přímo pro Edisona a dostupné v modulu ED, který jsme na začátku programu importovali příkazem „import Ed“.

Moduly

Co to je modul? Modulů je spousta v pythonu. Je to takzvaně modifikace, která obsahuje více kódů a jeden z nich je EdPy. Když chcete naimportovat modul do pythonu, tak na začátku kódu napíšete příkaz **import** a poté jméno daného modulu. Například jak můžete vidět na obrázku: import Ed. Jakmile máte svůj modul naimportovaný, tak můžete používat jeho příkazy tím, že před ně napíšete název modulu, potom tečka a poté příkaz. To je důvod, proč musíte psát Ed. před každým příkazem.

Závorky

Proč ale ty závorky? Závorky v kódu píšeme vždy při volání funkce nebo procedury, i když v nich nic není. Pokud procedura nebo funkce přijímá parametry, vložíte je právě do těch závorek. Jako

v červeně označeném příkazu máte **Ed.LeftLed(Ed.OFF)**. Máte příkaz, který ovládá levou diodu. Pak závorku, ve které je detail (parametr) pro její vypnutí. Do závorek se dá psát i více parametrů, musíte je ale oddělit čárkami. Ve všech případech musíte do závorek psát parametry v definovaném pořadí, jako v příkazu označeném modře. **Ed.Drive(Ed.SPIN_RIGHT, 5, 350)**. Tento příkaz dělá to, že Ozobot pojede v nějakém směru. Ale v závorce je detail, že nepojede ani dopředu, ani dozadu, ale že se otočí doprava. Poté je čárka, která říká rychlost, jakou se bude otáčet, další čárka a stupeň otočení.

Seznam možných funkcí a procedur je rozsáhlý a pro jejich použití, včetně parametrů, je nezbytné jít do příručky pro vývojáře.

Proměnné

Jak vytvořit proměnnou? V EdPy proměnnou vytvoříte tak, že se nejdřív rozhodnete, jak se proměnná bude jmenovat, třeba **x**. Do kódu napíšeme: **x =**. Poté si řekneme, co chceme, aby naše proměnná obsahovala. Může to být číselná hodnota, **True**, **False** atd. Takže jednoduše napíšeme **x = 1** nebo **x = True/False** (neboli inicializujeme proměnnou).

Cykly

V EdPy jsou samozřejmě také cykly. Fungují stejně jako v OzoBlockly. V OzoBlockly jste vzali cyklus a zacvakávali do něj, co jste chtěli, aby se opakovalo. Tady to funguje tak, že když chcete, aby to bylo v cyklu, tak řádek odrazíte tabulátorem. Cyklus nepůsobí na řádky, co nejsou odražené. Cyklus ale nemůžete přerušit. Nemůžete neodrazit jeden příkaz a poté mít zase odražené řádky.

```
x = 1
while x != 10:
    x = x + 1
    Ed.PlayBeep()
    Ed.LeftLed(Ed.OFF)
    Ed.RightLed(Ed.ON)
    Ed.Drive(Ed.SPIN_RIGHT, 5, 350)
    Ed.TimeWait(20, Ed.TIME_MILLISECONDS)
Ed.PlayBeep()
    Ed.LeftLed(Ed.ON)
    Ed.RightLed(Ed.OFF)
    Ed.Drive(Ed.SPIN_LEFT, 5, 350)
    Ed.TimeWait(20, Ed.TIME_MILLISECONDS)
```

Obrázek 32 – Příklad chybného odražení

Kód na obrázku č. 29 by probíhal následovně: co je označené zeleně by se opakovalo 10x. Pak by jednou robot pípnul, a to, co je označené červeně by způsobilo chybu. Museli byste pod pípnutí napsat další cyklus.

Cyklus While

While cyklus je vykonáván, dokud řídící podmínka je pravdivá. Použijeme-li **while True**, jedná se o nekonečný cyklus.

Takže když teď víme, jak vytvořit proměnnou, tak můžeme vytvořit cyklus s určeným počtem opakování. Nejdříve si vytvoříme proměnnou (v našem případě **x**), které dáme jako hodnotu nula. Poté použijeme cyklus **while** a za to napíšeme řídící podmínku. **While** v angličtině znamená dokud, takže musíme napsat, že cyklus se opakuje, **dokud** se proměnná nerovná počtu opakování (v našem případě 10). Nerovná se píše **!=**. Vždy musíme na konec napsat dvojtečku, tím ukončíme podmínku cyklu. Jdeme na další řádek a dáme tabulátor. Vždy, když proběhne jedno opakování, tak se k **x** přičte 1, abychom věděli, že kolo proběhlo. Jakmile se to zopakuje 10x tak **x** se bude rovnat 10, takže cyklus přestane.

```
x = 0
while x != 10:
    x = x + 1
    Ed.PlayBeep()
    Ed.LeftLed(Ed.OFF)
    Ed.RightLed(Ed.ON)
    Ed.Drive(Ed.SPIN_RIGHT, 5, 350)
    Ed.TimeWait(20, Ed.TIME_MILLISECONDS)
```

Obrázek 33 – WHILE cyklus

Cyklus For

Jednodušeji, jak definovat cyklus s určitým počtem opakování je použít jiného typu cyklu - cyklus **for**. Napíšete ho například: **for x in range(10)**. Nejdříve napíšete **for**, poté náhodnou proměnnou, poté **in range()** a do závorky počet opakování. Pod to zase napíšete, co chcete opakovat. Takže to nakonec vypadá nějak takhle.

```
for x in range(10):
    Ed.RightLed(Ed.ON)
    Ed.LeftLed(Ed.ON)
    Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
    Ed.RightLed(Ed.OFF)
    Ed.LeftLed(Ed.OFF)
    Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
```

Obrázek 34 - FOR cyklus

Přerušení cyklu

Přerušení cyklu se vynutí příkazem *Pass.*

Rozhodování

Potom jsou tady i příkazy na rozhodování. Fungují podobně jak v OzoBlockly. Vypadají takto:

```
x = 0
for i in range(10):
    if x == 2:
        pass
    else:
        x = x + 1
```

Obrázek 35 – Rozhodování: if, else

Na obrázku č. 32 můžeme vidět naše dva nové příkazy. **If** a **else**. **If** se používá, když chcete napsat: Když **x** se rovná 2, tak udělej. Když ale použijete **if**, tak musíte použít i **else**. Do **else** nepíšete podmínku. Je to k tomu, že když podmínka v **if** není pravdivá (v našem případě, když se x nerovná 2), tak se stane to, co je v **else**. Samozřejmě, jak je vidět i na obrázku, tak aby se na kód příkazy vztahovaly, tak musí být odražené tabulátorem. Pak je tu ale i příkaz **elif**.

```
x = 0
for i in range(10):
    if x == 3:
        pass
    elif x == 1:
        x = x + 2
    else:
        x = x + 1
```

Obrázek 36 – Rozhodování elif

Na obrázku č. 33 můžete vidět, jak se **elif** například používá. Nejdříve před ním musí být **if** a za ním **else**. Nejdříve se zeptáte, jestli **x** se rovná 3. To, když nevyjde, tak se přejde k **elif**, kde se ptáte, jestli se **x** aspoň rovná 1. To, když také nevyjde, tak to nakonec přejde k **else**.

Matematické operace

Rovná se	**==**	(pozor znaménko = představuje přiřazení hodnoty)
Nerovná se	**=!**	
Modulo (zbytek z dělení)	**%**	
Násobení	*****	
Plus	**+**	
Mínus	**-**	

Úkoly pro Edisona

Když jste v programu EdPy, tak na pravé straně máte přehled všech příkazů, funkcí a procedur z modulu Ed. Je tam uvedeno, co dělají, jaké psát argumenty do závorek a v jakém pořadí. Podívejte se tam. Důležitá poznámka: v programu jsou automaticky nastavené cm jako jednotka vzdálenosti.

Úkol 1

Jak napsat, aby robot jel dopředu 5 cm?

Úkol 2

Jak napsat, aby robot jel dopředu 5 cm a zablikal oběma světly? Toto se bude opakovat 3x s použitím cyklu.

Úkol 3

Jak napsat, aby robot jel dopředu libovolnou vzdálenost v cyklu, co se opakuje 6x, ale vždy, když je číslo sudé, tak místo aby jel dopředu, tak zapípal a jel dozadu?

Řešení

Řešení úkolu 1

```
Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 5)
```

Obrázek 37 - řešení úkolu 1

Řešení úkolu 2

```
for x in range(3):
    Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 5)
    Ed.LeftLed(Ed.ON)
    Ed.RightLed(Ed.ON)
    Ed.TimeWait(50, Ed.TIME_MILLISECONDS)
    Ed.LeftLed(Ed.Ed.OFF)
    Ed.RightLed(Ed.Ed.OFF)
```

Obrázek 38 - řešení úkolu 2

Řešení úkolu 3

```
for x in range(6):
    if x % 2 == 0:
        Ed.PlayBeep()
        Ed.Drive(Ed.BACKWARD, Ed.SPEED_5, 5)
    else:
        Ed.Drive(Ed.FORWARD, Ed.SPEED_5, 5)
```

Obrázek 39 - řešení úkolu 3

Závěr

Moje profilová práce je návodem na programování robotů Ozobota a Edisona a přeneseně obecným úvodem do programování a tvorby algoritmů. Je určena pro samouky nebo učitele na školách, které by vyučování pomocí těchto dvou robotů zajímalo.

V práci jsem připravil základní popis, jak začít s programováním robotů Ozobota a Edisona. Sepsal jsem důležité kroky pro začátek práce, vytvořil jsem vzorové úkoly a snažil se hlouběji vysvětlit některé programátorské kroky – např. rozdíl v cyklech FOR a WHILE, začátečnické problémy se špatným odražením v Pythonu, způsob tvorby cyklů v OzoBlockly.

Budu velmi rád, pokud má práce přispěje k zavedení pravidelných hodin nebo i jen částí z nich, kdy se studenti budou věnovat programování. Jsem přesvědčený, že dovednost programovat jednoduché úlohy je klíčová pro rozvoj logického myšlení. A dobré logické myšlení je nezbytné téměř v každém oboru lidské činnosti.

Zdroje

[1] *Ozokódy – Ozocodes* [online]. [cit. 2022-11-13]. Dostupné z: https://ondranauci.cz/ozokody-ozokodes/

[2] *Ozobot - lovec duchů* [online]. [cit. 2022-11-13]. Dostupné z: http://sposdk.cz/wp-content/uploads/2020/04/lovec-duch%c5%af.pdf

[3] *Ozobot - linka autobusu* [online]. [cit. 2022-11-13]. Dostupné z: http://sposdk.cz/wp-content/uploads/2020/04/autobus.pdf

[4] *Robotika - Speed test* [online]. [cit. 2022-11-13]. Dostupné z: https://www.nextech.sk/a/Robotika--E2-80-93-miniaturny-robot-Ozobot-so-zaujimavymi-moznostami-programovania

[5] *Zvíře v ohrádce* [online]. [cit. 2022-11-13]. Dostupné z: http://Ozobot.sandofky.cz/wp-content/uploads/BW_OzoOhrada.pdf

[6] *EdPy - programovací prostředí* [online]. [cit. 2022-11-13]. Dostupné z: https://www.edpyapp.com/

[7] *Ozoblockly - programovací prostředí* [online]. [cit. 2022-11-13]. Dostupné z: https://ozoblockly.com/