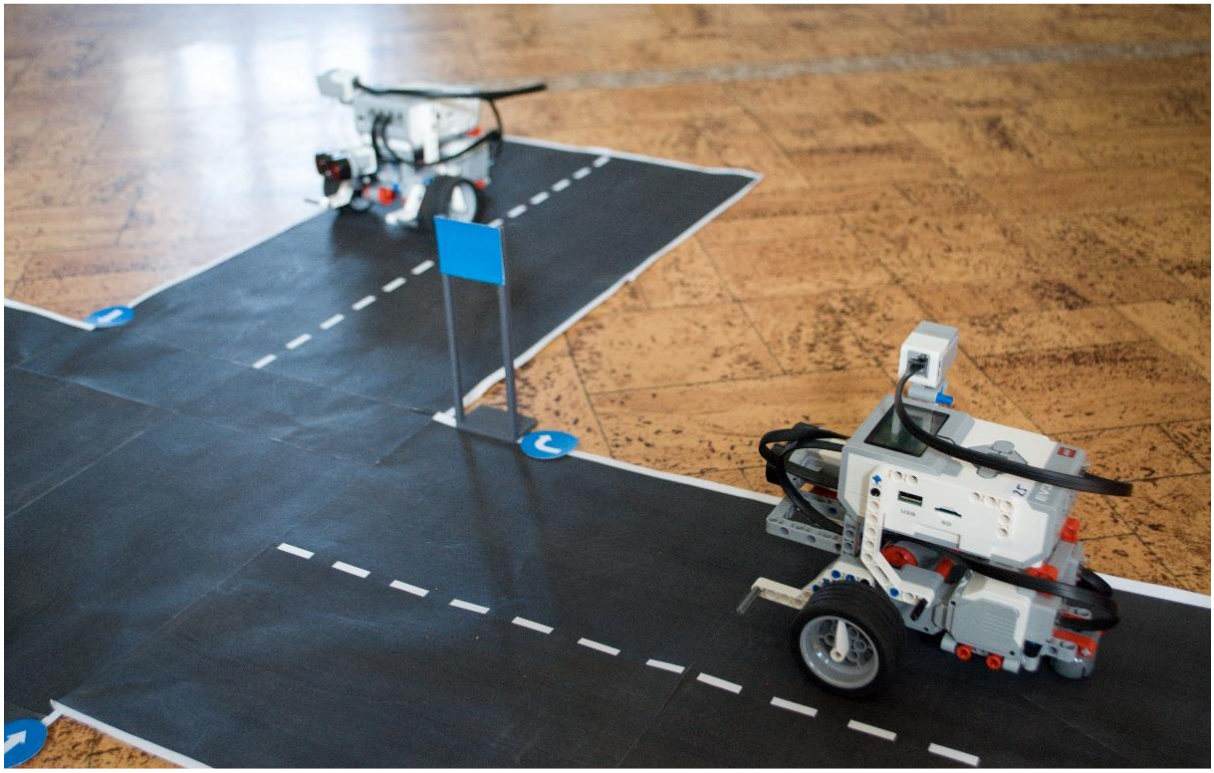
Úlohy pro robotickou stavebnici Lego MINDSTORMS EV3



Mgr. Kristýna Holečková

5 Sada úloh se zaměřením na fyzikální měření

Hlavním cílem závěrečné práce bylo vytvoření úloh pro stavebnici LEGO MINDSTORMS EV3, které na sebe budou tematicky navazovat a zároveň si žáci osvojí práci s roboty od základního po náročnější programování. Úlohy jsem navrhovala tak, aby žáci mohli aplikovat své znalosti z fyziky v informatice. Aby práce byla co nejvhodnější právě pro roboty, zvolila jsem téma Dynamika hmotného bodu, což je ve fyzice téma prvního ročníku SŠ. V úlohách se konkrétně zabývali rovnoměrným zrychleným a zpomaleným pohybem robota. V oblasti informatiky se seznámili kromě stavebnice LEGO EV3 také s novou aplikací pro programování a s pojmy jako např. proměnná, cyklus a podmínka. Důležitou součástí většiny úloh byly senzory, a to senzor barev a ultrasonický senzor.

Úlohy jsou koncipované tak, aby žáci pracovali badatelskou formou výuky ve dvojicích, případně ve trojicích z důvodu vyšší absence některých žáků. Všechny skupinky pracovali na stejných úlohách a měli tak možnost si navzájem pomáhat. Zadání úloh pro žáky jsou k náhledu na konci tohoto dokumentu jako přílohy.

5.1 Zjištění rychlosti robota

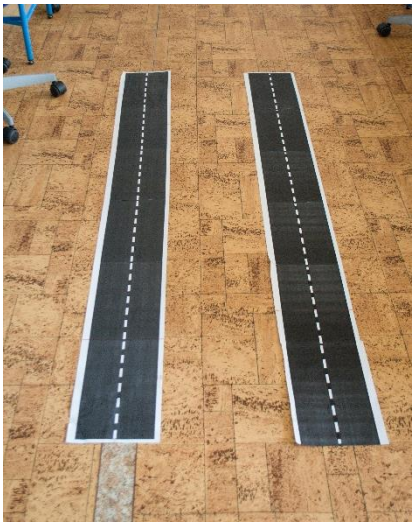
Cílem této úlohy je zjistit na připravené dráze maximální rychlost přímočarého rovnoměrného pohybu robota. V platformě MakeCode se jeho rychlost udává v procentech, což je z fyzikálního hlediska nepřesné. Abychom mohli později pracovat s konkrétními hodnotami, potřebujeme znát jeho okamžitou rychlost *v*.

5.1.1 Cíle

- připravit program na běh motorů v určitém časové úseku
- změřit vhodný úsek na dráze vhodný na měření rychlosti – jeden metr
- změřit čas
- vypočítat rychlost
- pro kontrolu zopakovat měření hodnot

5.1.2 Postup

Žáci měli připravenou dráhu (runway) vyrobenou z papíru, na které měli zjistit rychlost robota v metrech za sekundu.



Obrázek 10: Dráha na zjištění rychlosti robota

Po sestavení robota v programu MakeCode použili žáci pohybový příkaz *run*, ve kterém nastavili maximální rychlost v procentech (100 %) pro motory připojených na portech B a C a zároveň v příkaze nastavili čas 5 s, po který se bude robot pohybovat.

Následně žáci změřili svinovacím metrem část runwaye a vyznačili jeden metr na dráze. Poté robota pustili a měřili čas, za který robot dráhu ujel. Pomocí naměřených hodnot vypočítali rovnoměrnou rychlost přímočarého pohybu.

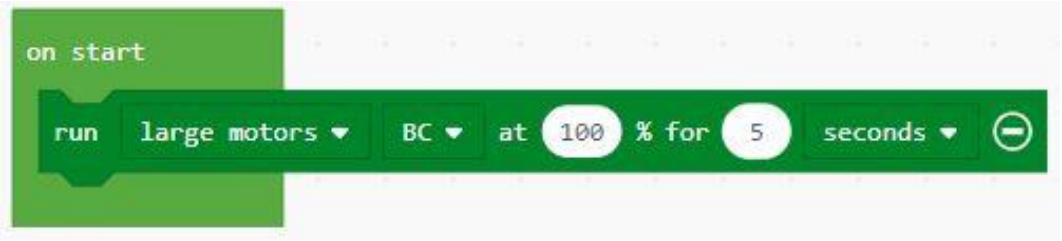
Měření opakovali vícekrát, aby si byli jisti, že naměřené hodnoty jsou shodné. Zároveň zkoušeli měřit čas při rychlosti nastavené na 50 % a 25 %, aby tak zjistili, zda je například při čase sníženém na polovinu i rychlost v m/s poloviční, tedy zda platí přímá úměrnost mezi procenty v MakeCode a skutečnou rychlostí.

5.1.3 Problémy při řešení úlohy

U některých robotů nešla změřit konstantní hodnota času nebo čas nevycházel u poloviční rychlosti oproti maximální rychlosti apod. Nakonec jsme zjistili, že potíže vznikly málo nabitými akumulátory nebo tužkovými bateriemi použitými v robotech.

5.1.4 Řešení úlohy

Skutečné rychlosti z fyzikálního pohledu se u robotů různí. U robotů s menšími koly byla rychlost naměřená na 0,4 m/s, u ostatních robotů 0,48 m/s a 5 m/s.



Obrázek 11: Nastavení rychlosti motorů v platformě MakeCode

5.2 Plynulý rozjezd robota při stálém zrychlení

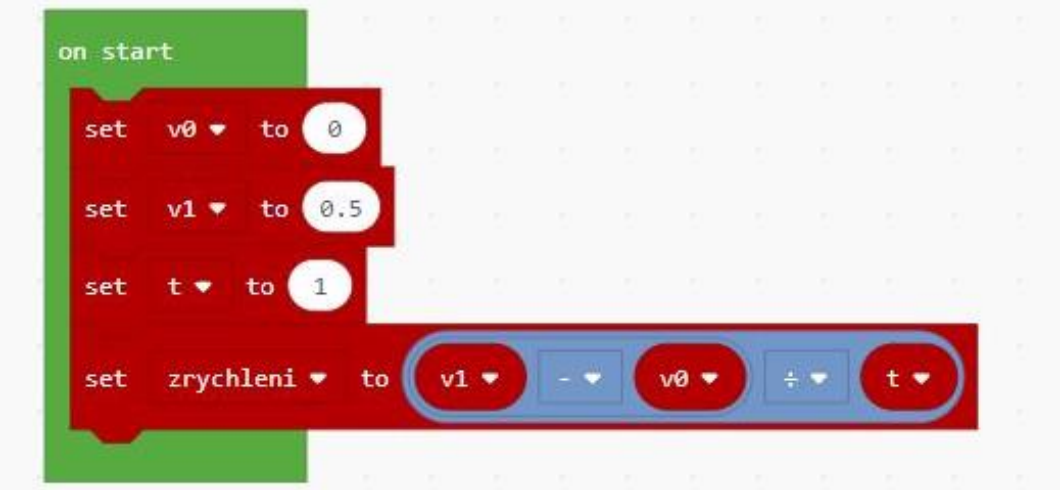
Po změření rychlosti řešili žáci úlohu na rovnoměrné zrychlení robota z nulové rychlosti *v0* na maximální rychlost *v1* za jednu sekundu. Aby robot s hodnotami fyzikálních rychlostí uměl pracovat, musí je žáci převést na procenta. V této úloze použili žáci simulátor pohybu robota na kontrolu hodnot, se kterými má kostka pracovat.

5.2.1 Cíle

- seznámit se s pojmem proměnné, umět je zavést do kódu a upravovat jejich hodnoty
- nastavovat a kombinovat matematické operace
- seznámit se s pojmem cyklus a podmínka, pochopit jejich funkčnost
- hledat a opravit chyby v kódu pomocí vyhodnocování zobrazujících se hodnot na simulátoru robota

5.2.2 Postup

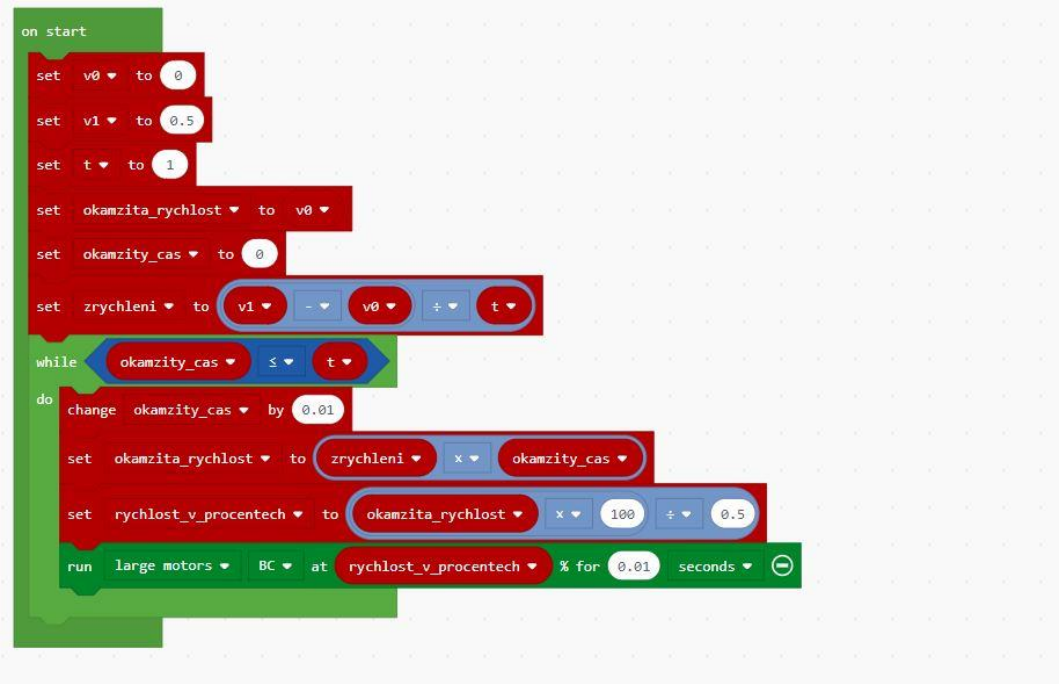
Nejprve jsem se žáky prošla postup krok po kroku, jak se zvětšuje rychlost tak, aby se zvyšovala rovnoměrně v čase jedné sekundy. Následně jsme přešli na tvorbu programu. Nejprve jsem žáky seznámila s proměnnými, vysvětlila jejich funkci a poté jsme vytvořili/deklarovali proměnné pro počáteční rychlost *v0*, konečnou rychlost *v1*, čas *t* a *zrychleni* a nastavili jsme je na příslušné hodnoty. Proměnnou *zrychleni* není vhodné nastavit na konkrétní hodnotu z důvodu možné změny rychlosti *v0* a *v1*. Je tedy třeba, aby proměnná *zrychleni* se počítala automaticky.



Obrázek 12: Deklarace základních proměnných

Dále museli žáci vyřešit zvyšování rychlosti za jednu sekundu a přijít na to, že se musí určitý sled událostí opakovat, tedy že budou muset použít cyklus pro zkrácení kódu. Vysvětlili jsme si, jaký je rozdíl mezi běžným cyklem a cyklem s podmínkou. Žáci zjišťovali, jaké další hodnoty musí znát kromě těch, které jsme zavedli na začátku programu. V průběhu kódu potřebují pracovat s okamžitou rychlostí a s okamžitým časem. Okamžitá rychlost se bude zvyšovat vlivem narůstajícího času a zrychlení (z aplikace vzorečku $a = \frac{v}{t} \rightarrow v = a \cdot t$). Pro správnou komunikaci robota s rychlostí je zároveň třeba nastavit proměnnou, která bude převádět okamžitou rychlost na procenta. Žáci zjistili, že mohou použít trojčlenku, neboť znají rychlost při 100 %. Okamžitá rychlost se dopočítá pomocí zrychlení a času. Hodnota okamžité rychlosti v procentech je x, tedy $x = \frac{v \cdot 100}{0{,}5}$.

Aby čas počítaný programem odpovídal reálnému času zrychlení, musí robot pracovat s motory vždy stejně dlouhou dobu jako přičítaný čas v cyklu.



Obrázek 13: Použití cyklu s podmínkou a měnícími se proměnnými

Pro kontrolu správných výpočtů jsem žákům doporučila zavést příkazy *show value*, které zobrazí text na displeji kostky. Hodnoty mohou kontrolovat i na simulátoru robota, na kterém vidí kromě těchto příkazů i práci motorů, tedy jejich rychlost v procentech.

5.2.3 Problémy při řešení úlohy

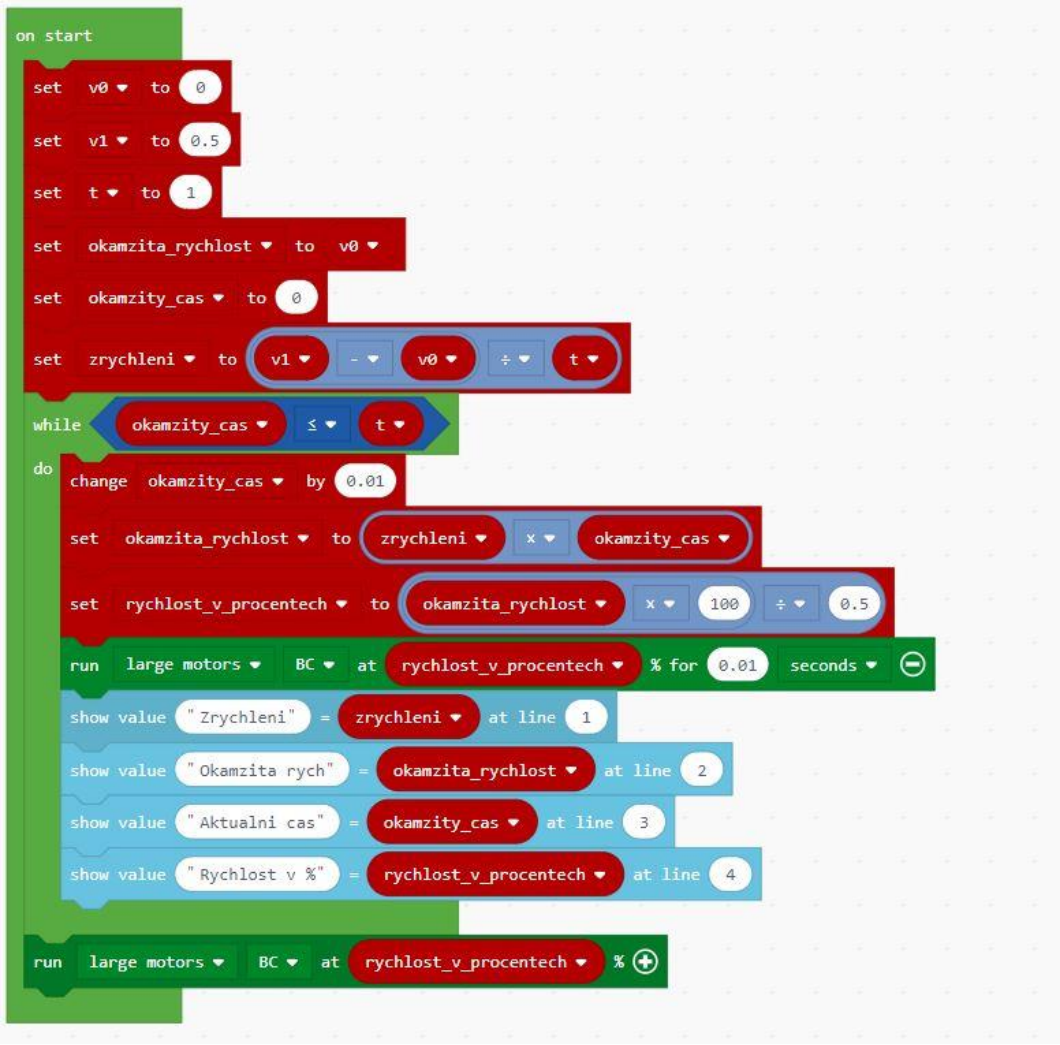
Při tvorbě programu se žákům vyskytlo mnoho problémů, které museli řešit. Nejobtížnější bylo navést žáky na algoritmické myšlení tak, aby byli schopni postupně vytvářet program, který bude funkční. V této úloze jsem musela žákům radit, respektive je navádět více, protože neznali ani základy programování. Dále se vyskytl problém s aplikací vzorečku pro zrychlení a přímé úměrnosti, tedy s použitím trojčlenky.

Simulátor často ukazoval hodnotu víc jak 100 %, což logicky při výkonu robota není možné. Buď byla chyba v matematickém vzorci, nebo v chybném pořadí příkazů. V takovém případě jsem žákům poradila, aby si napsali postup na papír, aby viděli, jak

robot s hodnotami pracuje. Ze zapsaných hodnot z cyklu žáci pak zjistili, jak se postupně mění, a nakonec na chyby přišli.

5.2.4 Řešení úlohy

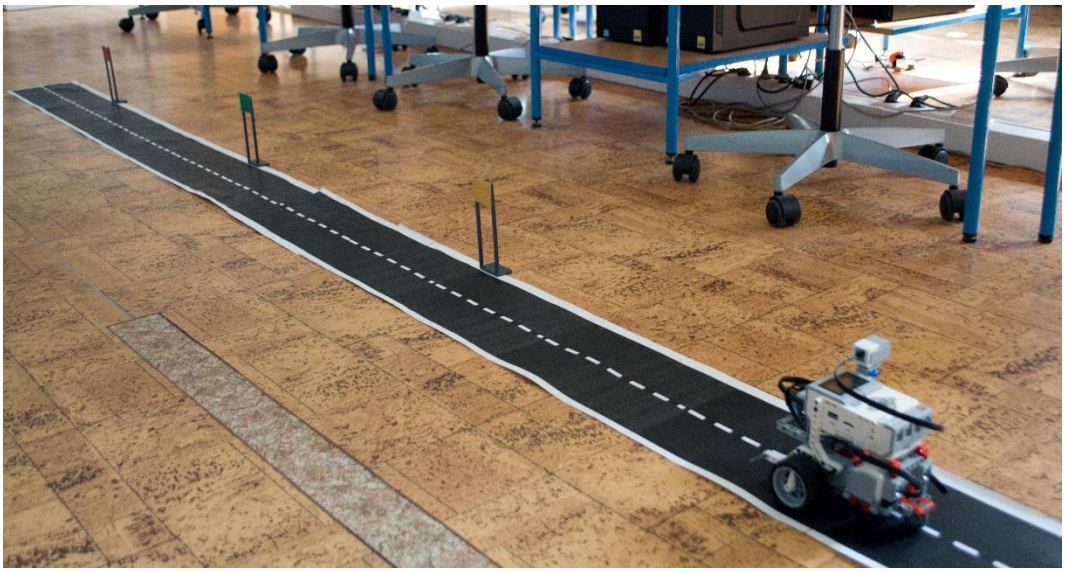
Změna proměnné *okamzity_cas* v cyklu musí souhlasit s aktuálním pohybem robota, aby při měření skutečného času při rozjíždění jsme neměli mnohem delší časový úsek, než který byl stanoven. Hodnotu proměnné *okamzity_cas* si můžeme stanovit podle sebe, já jsem zvolila malé hodnoty z důvodu plynulejšího rozjezdu. Při dosazení větších hodnot (např. 0,5) by byl rozjezd robota „sekavý“. Žáci však zadávali i hodnotu 0,05 a robot se rozjížděl plynule.



Obrázek 14: Výsledný kód pro rovnoměrné zrychlení robota

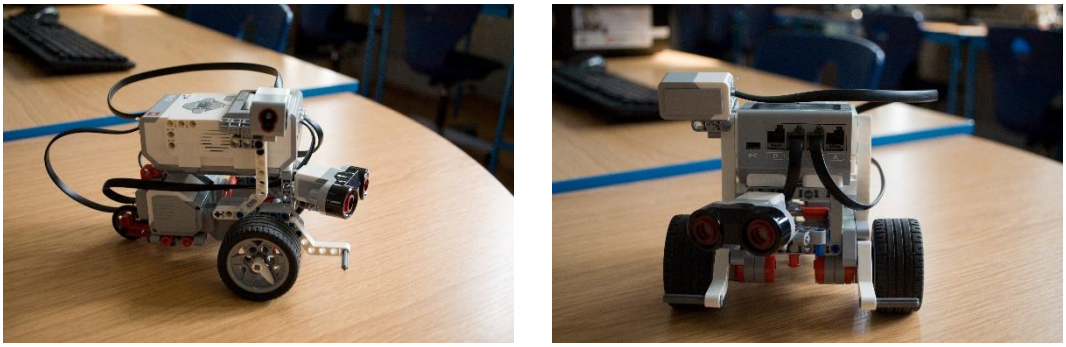
5.3 Rovnoměrné zrychlení a zpomalení při detekci barev

V této úloze měli žáci za úkol pomocí barevných značek regulovat rychlost robota formou zrychlení nebo zpomalení. Barevné značky byly přilepené na stojanech, které byly vytištěny na 3D tiskárně. Značky by bylo možné umístit i na podložku/dráhu, po které se robot pohybuje, ale z důvodu regulace vzdálenosti mezi značkami a rychlé reakci pro případné selhání při načtení barvy jsem zvolila tuto variantu. Jízda robota je ovlivňována zelenou, žlutou a červenou barvou. Při detekci zelené barvy má robot zrychlit na maximální rychlost, pokud detekuje žlutou barvu, robot znatelně zpomalí a při červené barvě má postupně zastavit.



Obrázek 16: Značky na signalizování rychlosti robota

Ke stavebnici museli žáci připojit senzor detekující barvy do dostatečné výšky, aby značky dokázal detekovat.



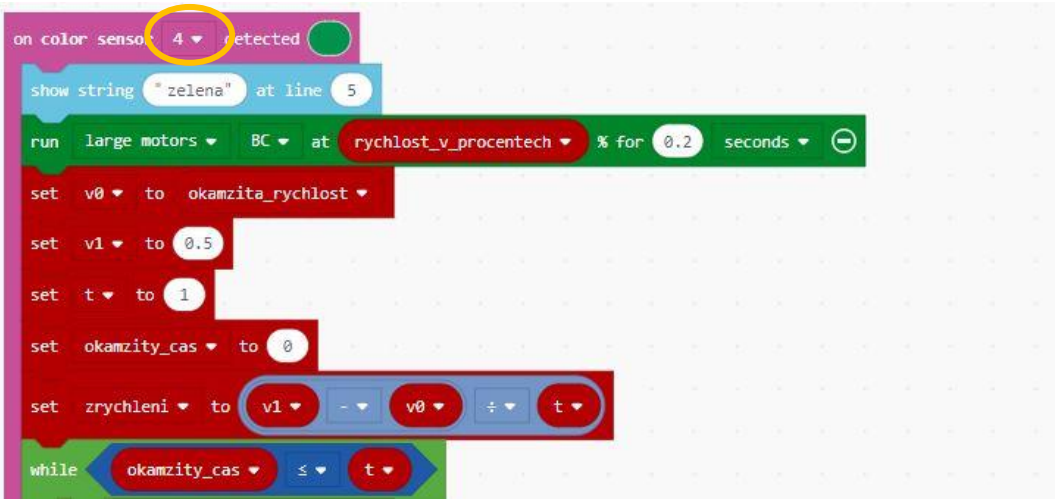
Obrázek 15: Připojení senzoru na detekci barev

5.3.1 Cíle

- nastavení detekce barev v aplikaci MakeCode
- aplikace kódu pro konkrétní barvu
- práce s proměnnými – deklarace, nastavení a zapojení do kódu
- hledání a opravení chyb v kódu pomocí vyhodnocování zobrazujících se hodnot na simulátoru robota
- orientace v textovém kódu JavaScript, práce s proměnnými, podmínkou a cyklem

5.3.2 Postup

Úlohu jsem rozdělila do více vyučovacích hodin z důvodu obsáhlosti. Detekce zelené barvy byla poměrně jednoduchá, protože zrychlení na maximální rychlost žáci programovali již v minulé úloze. Museli však dbát na to, aby bylo správně nastavené číslo portu pro barevný senzor.



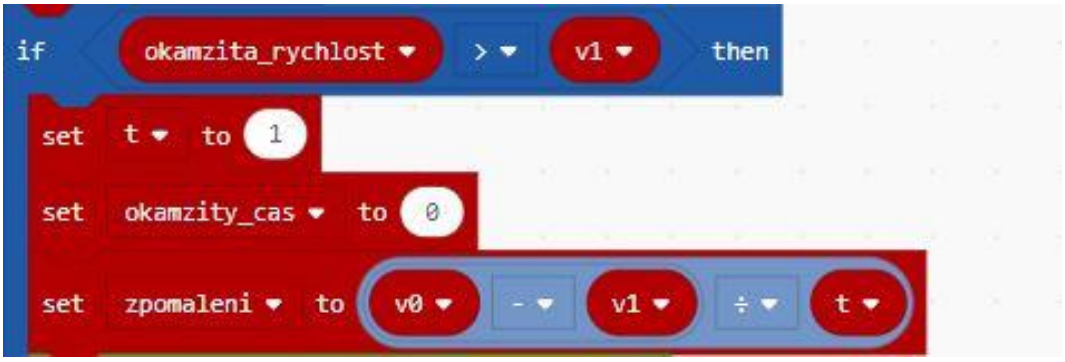
Obrázek 17: Zrychlení při detekci zelené barvy

Pro zjištění správné detekce jsem žákům doporučila připojit do kódu příkaz na zobrazení názvu barvy na displeji. Po načtení barvy je třeba, aby robot popojel kousek dál z důvodu opakující se detekce barvy. Robot by měl „sekavý“ pohyb, dokud by se od značky neoddálil. Žáci si museli také uvědomit, že počáteční rychlost *v0* nemůže

být nulová. Robot, než dojede k barevné značce, se pohybuje určitou rychlostí, kterou má v kódu již zaznamenanou – proměnná *okamzita_rychlost*.

Následně řešili žáci detekci žluté barvy, tedy zpomalení. Začátek této části programu je velmi podobný. Je třeba nastavit počáteční rychlost *v0*, konečnou rychlost *v1* a proměnnou *zrychleni*, případně *zpomaleni*. I tomto případě musí být počáteční rychlost nastavená pomocí proměnné *okamzita_rychlost*.

Aby nedocházelo ke kolapsu a robot nezačal couvat, bylo třeba zkontrolovat pomocí podmínky, zda jeho okamžitá rychlost není už při detekci nižší než rychlost, na kterou by měl dle instrukcí zpomalit. Následně si žáci buď zvolili novou proměnnou *zpomaleni*, kterou kostka opět spočítá pomocí matematických operací, nebo znovu zvolí proměnnou *zrychleni,* ale musí prohodit proměnné pro rychlosti, aby hodnota nevycházela v záporných číslech.



Obrázek 18: Podmínka na kontrolu rychlosti, proměnná zpomaleni

V této fázi jsem žákům doporučila, aby si na papír napsali, jak by měl robot postupovat, aby se jeho rychlost postupně snižovala až na uvedenou hodnotu.

Žáci zjistili, že *okamzitou_rychlost* musí zmenšovat nejen pomocí narůstajícího času, ale i její původní hodnotou. Další postup je stejný jako při zrychlování – převod *okamzite_rychlosti* na procenta, dosazení této proměnné pro pohyb motorů a zobrazování hodnot na displeji kostky.

Obrázek 19: Nastavení proměnné okamzita_rychlost

Pro naprogramování detekce červené barvy použili žáci místo blokového prostředí JavaScript. Prošli jsme společně zápis kódu po detekci zelené a žluté barvy.

Žáci následně zkopírovali celý kód pro žlutou barvu a upravili ho tak, aby souhlasil se zadáním a robot za 1 sekundu rovnoměrným zpomalením zastavil.

```
sensors.color4.onColorDetected(ColorSensorColor.Yellow, function () {
    brick.showString("zluta", 5)
    motors.largeBC.run(rychlost_v_procentech, 0.2, MoveUnit.Seconds)
    v0 = okamzita_rychlost
    v1 = 0.2
    if (okamzita_rychlost > v1) {
        t = 1
        okamzity_cas = 0
        zpomaleni = (v0 - v1) / t
        while (okamzity_cas <= t) {
            okamzity_cas += 0.01
            okamzita_rychlost = okamzita_rychlost - zpomaleni * okamzity_cas
            rychlost_v_procentech = okamzita_rychlost * 100 / 0.5
            motors.largeBC.run(rychlost_v_procentech, 0.01, MoveUnit.Seconds)
            brick.showValue("Zpomaleni", zpomaleni, 1)
            brick.showValue("Okamzita rych", okamzita_rychlost, 2)
            brick.showValue("Aktualni cas", okamzity_cas, 3)
            brick.showValue("Rychlost v %", rychlost_v_procentech, 4)
        }
    }
    motors.largeBC.run(rychlost_v_procentech)
})
```

Obrázek 20: JavaScript pro detekci žluté barvy

```
sensors.color4.onColorDetected(ColorSensorColor.Red, function () {
    brick.showString("cervena", 5)
    motors.largeBC.run(rychlost_v_procentech, 0.2, MoveUnit.Seconds)
    v0 = okamzita_rychlost
    v1 = 0
    if (okamzita_rychlost > v1) {
        t = 1
        okamzity_cas = 0
        zpomaleni = (v0 - v1) / t
        while (okamzity_cas <= t) {
            okamzity_cas += 0.01
            okamzita_rychlost = okamzita_rychlost - zpomaleni * okamzity_cas
            rychlost_v_procentech = okamzita_rychlost * 100 / 0.5
            motors.largeBC.run(rychlost_v_procentech, 0.01, MoveUnit.Seconds)
            brick.showValue("Zpomaleni", zpomaleni, 1)
            brick.showValue("Okamzita rych", okamzita_rychlost, 2)
            brick.showValue("Aktualni cas", okamzity_cas, 3)
            brick.showValue("Rychlost v %", rychlost_v_procentech, 4)
        }
    }
})
```

Obrázek 21: Změny v JavaScriptu pro detekci červené barvy

5.3.3 Problémy při řešení úlohy

Problémů při sestavování a vylepšování kódu bylo několik. Podobně jako v předchozí úloze nastala situace, že zpomalování nebylo plynulé nebo při zastavení robot couval. Vždy šlo o chybné poskládání kódu (například změna hodnoty v proměnné *okamzity_cas* byla změněna až na konci cyklu).

Dalším problémem bylo chybné rozpoznání barvy senzorem, konkrétně místo zelené barvy někdy senzor zaznamenal modrou barvu. Zkoušeli jsme i změnu místa kvůli lepšímu světlu, ale problém tímto způsobem zcela vyřešen nebyl. Proto jsem doporučila zkopírovat kód jak pro zelenou, tak i pro modrou barvu, což však kolidovalo s následující úlohou, kde se modrá barva používá pro signalizaci křižovatky.

V jedné skupině dívek nastal problém s nerovnoměrným pohybem jednoho motoru vůči druhému, proto jim robot na rovině zatáčel. Žákyně zkoušely vypojit a zapojit kabely nebo je vyměnit, nakonec však použily motory z volné stavebnice. Problém stejně přetrval, ačkoliv se nám zdálo, že se projevoval méně často.

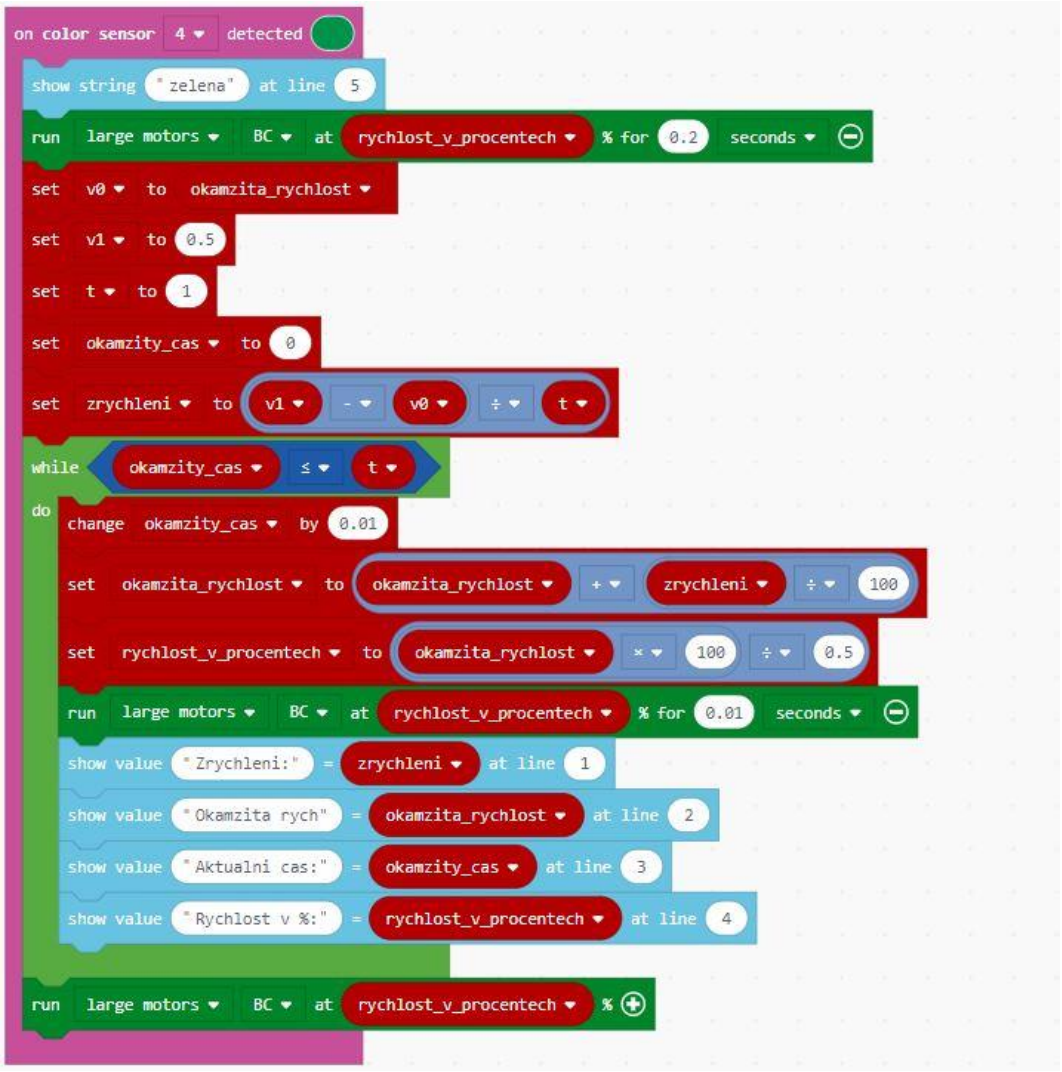
5.3.4 Doporučení učitelům pro přípravu dráhy a barevných značek

Ačkoliv detekování barev pomocí stojanů je dle mého názoru zajímavý nápad, může být toto použití v některých situacích nepraktické. Pokud se robot nepohybuje zcela rovně a trochu vybočuje z dráhy, je třeba se značkami pohotově posouvat, aby mezi senzorem a barvou nebyla příliš velká vzdálenost. Zajímavá by mohla být pro žáky i tvorba stojanů v aplikaci pro 3D modelování a následně samotný tisk, což však ve školách není zcela běžné. Výhodou těchto značek je, že učitel může libovolně měnit pořadí barev pro důkladnější otestování úlohy.

Doporučila bych dráhu připravit tak, aby barvy byly vyznačeny na podložce, ale zároveň aby bylo možno je přesouvat, kombinovat, zvětšovat mezi značkami vzdálenost atd.

5.3.5 Řešení úlohy

Detekce zelené barvy – zrychlení robota na maximální rychlost:



Obrázek 22: Detekce zelené barvy – zrychlení na maximální rychlost

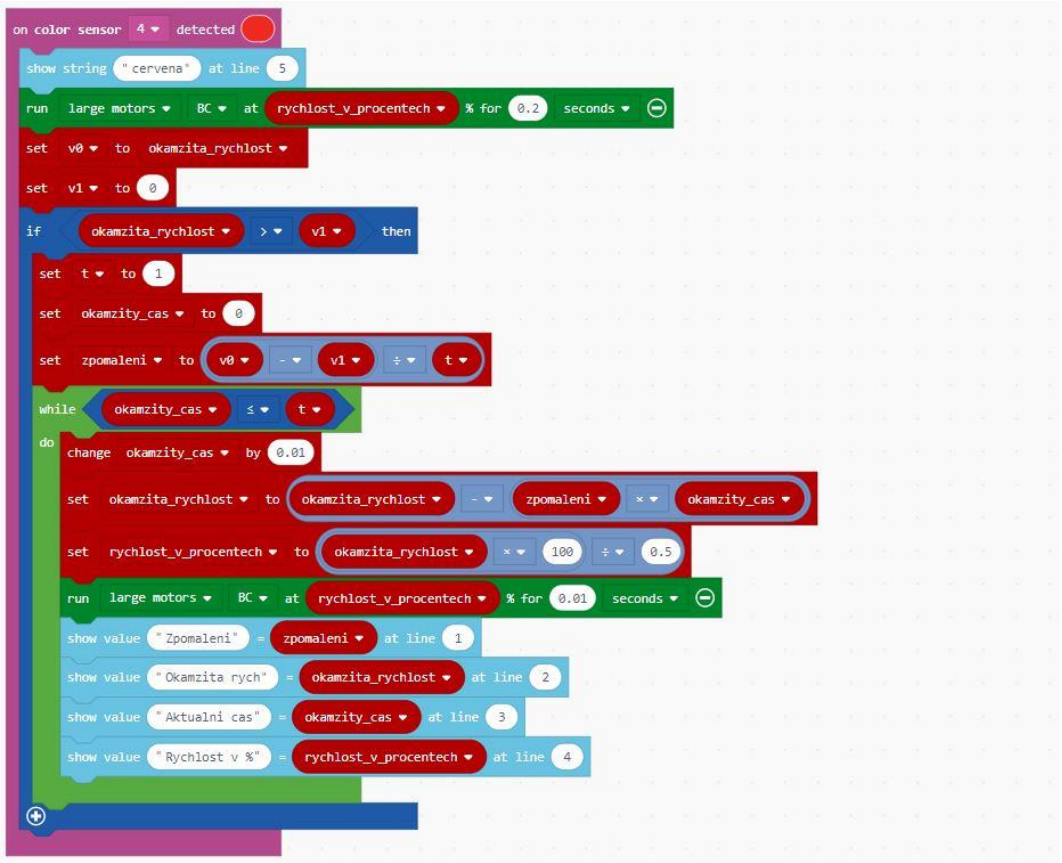
Detekce žluté barvy – zpomalení robota na 0,2 m/s:

Obrázek 23: Detekce žluté barvy – zpomalení robota na 0,2 m/s v blokovém editoru

```
sensors.color4.onColorDetected(ColorSensorColor.Yellow, function () {
    brick.showString("zluta", 5)
    motors.largeBC.run(rychlost_v_procentech, 0.2, MoveUnit.Seconds)
    v0 = okamzita_rychlost
    v1 = 0.2
    if (okamzita_rychlost > v1) {
        t = 1
        okamzity_cas = 0
        zpomaleni = (v0 - v1) / t
        while (okamzity_cas <= t) {
            okamzity_cas += 0.01
            okamzita_rychlost = okamzita_rychlost - zpomaleni * okamzity_cas
            rychlost_v_procentech = okamzita_rychlost * 100 / 0.5
            motors.largeBC.run(rychlost_v_procentech, 0.01, MoveUnit.Seconds)
            brick.showValue("Zpomaleni", zpomaleni, 1)
            brick.showValue("Okamzita rych", okamzita_rychlost, 2)
            brick.showValue("Aktualni cas", okamzity_cas, 3)
            brick.showValue("Rychlost v %", rychlost_v_procentech, 4)
        }
    }
    motors.largeBC.run(rychlost_v_procentech)
})
```

Obrázek 24: Kód v JavaScriptu pro detekce žluté barvy – zpomalení na 2 m/s

Detekce červené barvy – zastavení robota:



Obrázek 25: Detekce červené barvy – zastavení v blokovém editoru

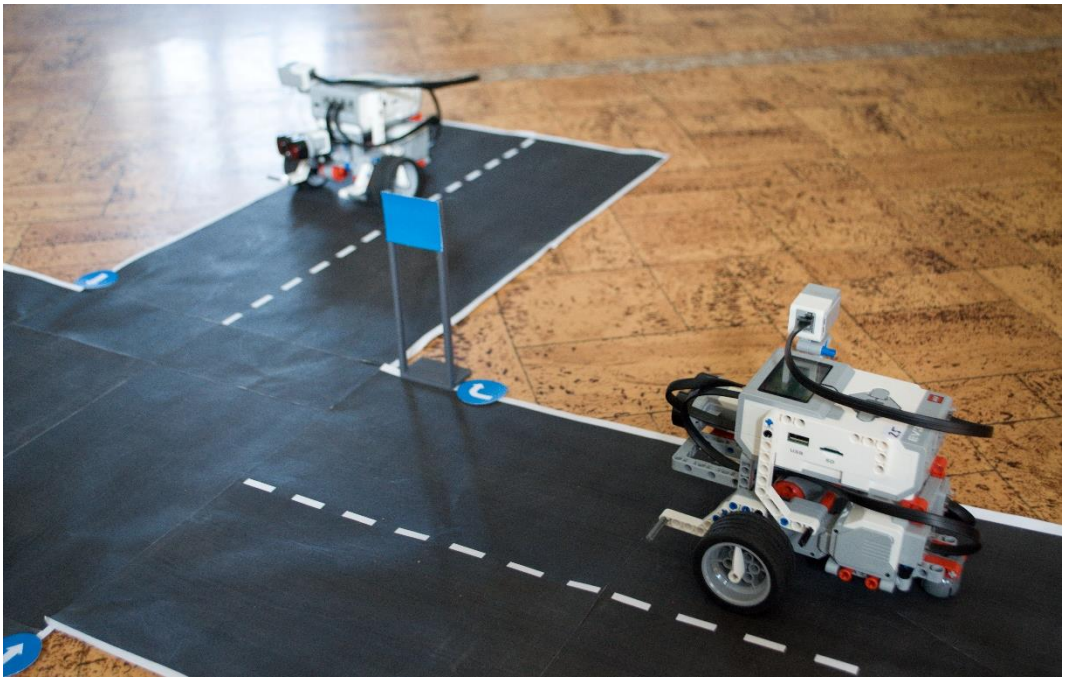
```
sensors.color4.onColorDetected(ColorSensorColor.Red, function () {
    brick.showString("cervena", 5)
    motors.largeBC.run(rychlost_v_procentech, 0.2, MoveUnit.Seconds)
    v0 = okamzita_rychlost
    v1 = 0
    if (okamzita_rychlost > v1) {
        t = 1
        okamzity_cas = 0
        zpomaleni = (v0 - v1) / t
        while (okamzity_cas <= t) {
            okamzity_cas += 0.01
            okamzita_rychlost = okamzita_rychlost - zpomaleni * okamzity_cas
            rychlost_v_procentech = okamzita_rychlost * 100 / 0.5
            motors.largeBC.run(rychlost_v_procentech, 0.01, MoveUnit.Seconds)
            brick.showValue("Zpomaleni", zpomaleni, 1)
            brick.showValue("Okamzita rych", okamzita_rychlost, 2)
            brick.showValue("Aktualni cas", okamzity_cas, 3)
            brick.showValue("Rychlost v %", rychlost_v_procentech, 4)
        }
    }
})
```

Obrázek 26: Kód v JavaScriptu pro detekci červené barvy – zastavení

5.4 Odbočení na křižovatce

Poslední úloha na detekci barvy je odbočení na křižovatce doprava po zaznamenání modré barvy. Robot musí zpomalit, otočit se o 90°, zrychlit a po dvou sekundách zastavit. Vše s využitím předchozích znalostí, tedy s rovnoměrným zrychlením a zpomalením.

Tento úkol měli žáci jako samostatnou práci na jednu vyučovací hodinu. Další hodinu museli na připravené dráze předvést výsledný pohyb robota.



Obrázek 27: Křižovatka s modrou značkou signalizující odbočení vpravo

5.4.1 Cíle

- využití znalostí z předchozích úloh – práce s proměnnými, práce se senzorem
- správná regulace chodu motorů pro dodržení úhlu 90°
- orientace v textovém kódu JavaScript, práce s proměnnými, podmínkou a cyklem

5.4.2 Postup

Žáci měli velmi rychle připravený první návrh pro otočení se robota, který pak ověřovali. Vstupní událost pro detekci barvy měli již vyzkoušenou, proto se jednalo především jen o jiný pohyb robota. Při zkoušení funkčnosti kódu se robot buď otáčel příliš málo, nebo naopak hodně. Některým skupinkám jsem doporučila, aby i motor (kolo), za kterým se robot otáčí, byl v chodu alespoň dvě rotace, aby se robot nedostal do smyku. Nastavení otáčení bylo pro žáky nejrychlejší programovat jej metodou pokus/omyl, co se týče hodnot, které do programu zadávali.

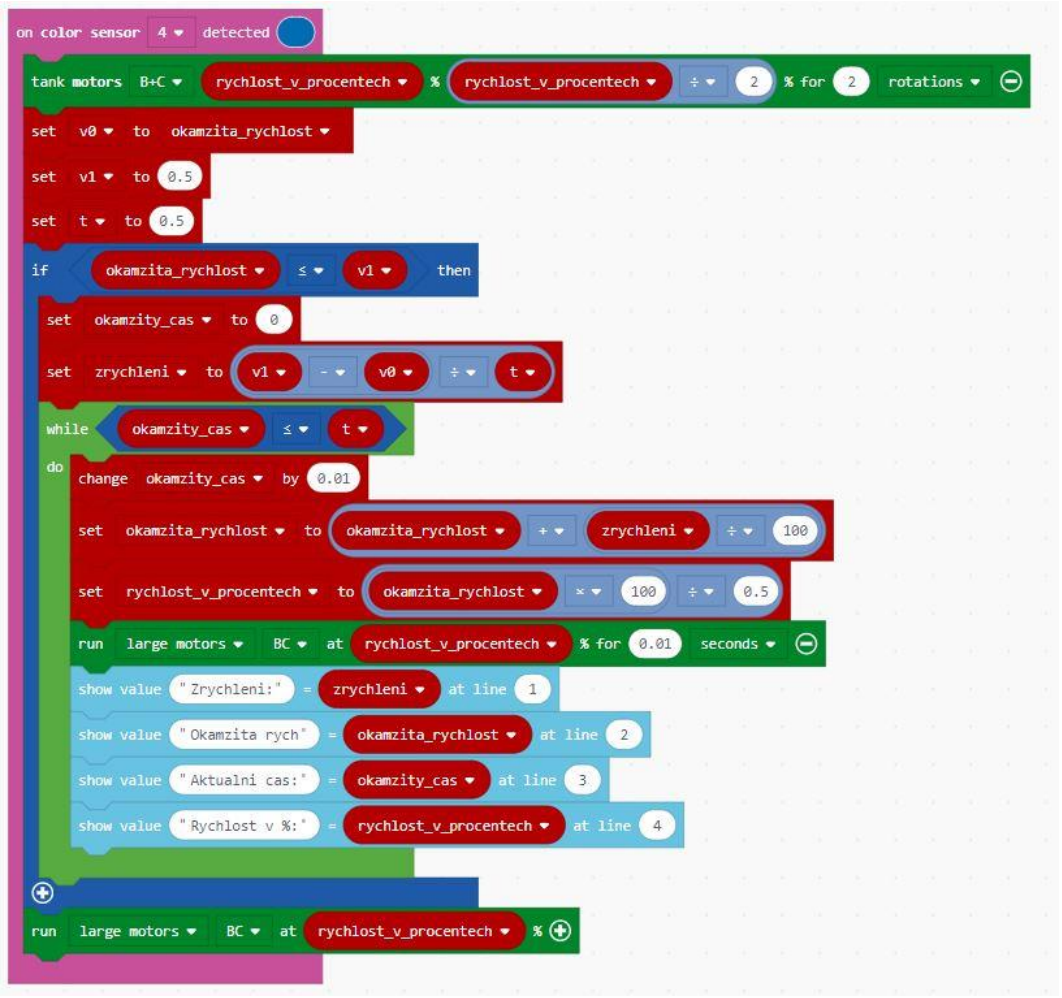
Obrázek 28: Kód pro odbočení na křižovatce

Pro zpomalování při detekci barvy a pro zrychlení po otočení se o 90° využili žáci kód z předešlé úlohy.

5.4.3 Problémy při řešení úlohy

Největším problémem při řešení úlohy byla špatná detekce barvy na stojanu. Někdy se stalo, že místo detekce modré barvy senzor zaznamenal zelenou barvu. Opět jsme zkusili změnit místo dráhy, abychom změnili světelné podmínky – přešli jsme z učebny na chodbu, měnili jsme tmavší místa za světlejší, přesto detekce byla chybná. Proto jsme nastavili stejný kód pro detekci modré barvy i pro vstupní událost zelené barvy, která v tomto úkolu nebyla jinak součástí – zrychlení probíhalo bez senzoru barev.

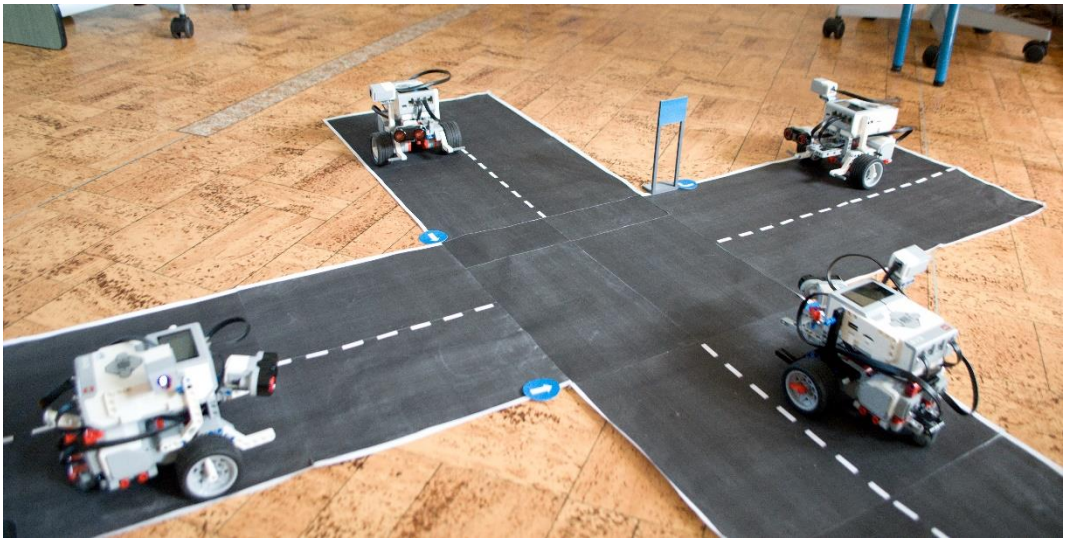
5.4.4 Řešení úlohy



Obrázek 29: Projetí křižovatky s následným zrychlením

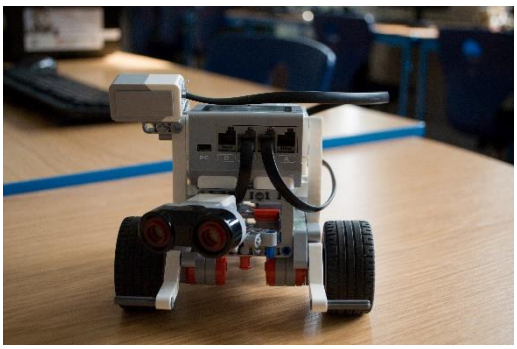
5.5 Dej přednost v jízdě

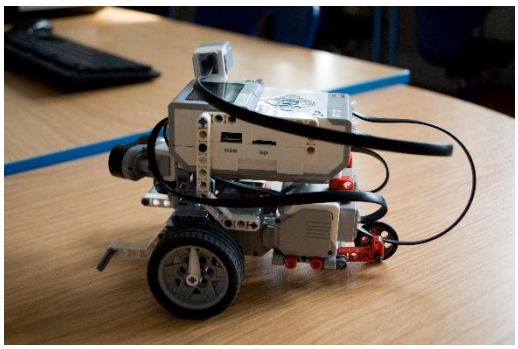
Pro závěrečnou úlohu je opět jako dráha připravena křižovatka, přes kterou tři roboti pokračují rovně, jeden robot odbočí vpravo (předchozí úloha – kapitola 5.4). Žáci si zde připomněli pravidla silničního provozu, co se týče složité křižovatky a dávání přednosti v jízdě.



Obrázek 30: Křižovatka s přikázanými směry jízdy

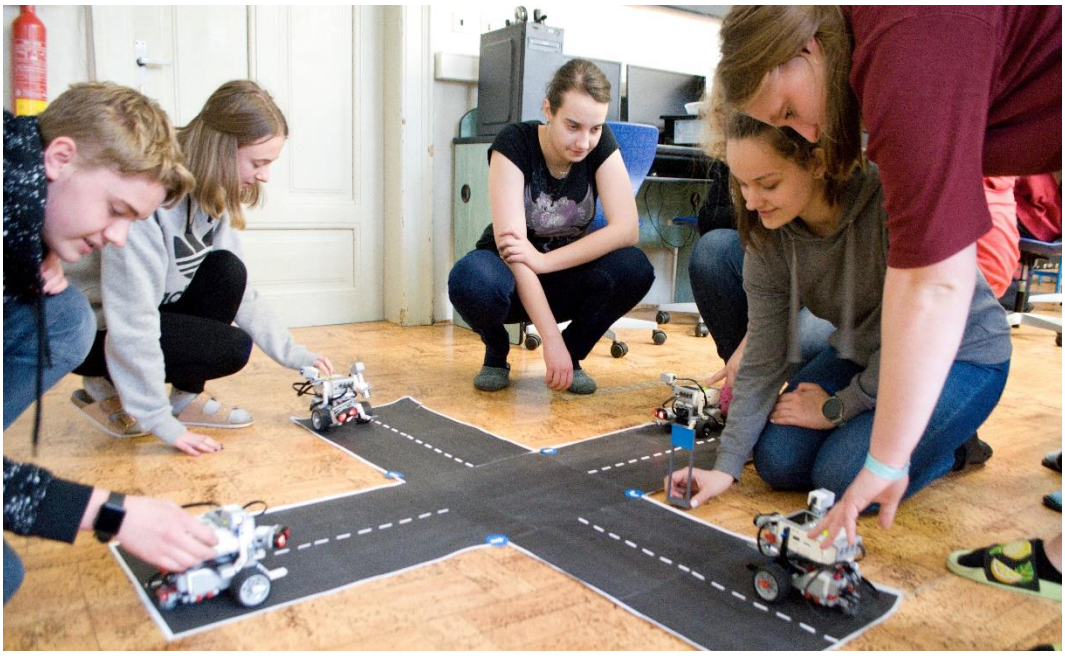
Úloha je zaměřena na detekci vzdálenosti jiného objektu, který se nachází mírně vpravo od robota. Žáci tedy museli připojit ke stavebnici ultrasonický senzor, díky němuž pak na základě naměřených hodnot mohli nastavit jeho chování. Úloha je koncipována tak, že robot jedoucí na neoznačenou křižovatku musí hlídat její pravou část, poněvadž platí pravidlo pravé ruky, tedy přednost v jízdě zprava.





Obrázek 31: Připevněný ultrasonický senzor detekující překážky mírně vpravo

Při startu se robot plynule rozjede na maximální rychlost (případně trochu nižší), pokud zaznamená překážku, která je blíž jak 50 cm, musí robot zpomalit (stejná rychlost jako při detekci žluté barvy z úlohy s barevnými značkami – kapitola 5.3). Přiblíží-li se robot k objektu/překážce na vzdálenost menší než 35 cm, musí plynule zastavit. Pokud překážka na pravé straně odjede, robot se musí opět rozjet do maximální rychlosti, případně rychlosti nastavené pro zpomalení, jestliže je ve vzdálenosti do 50 cm jiný objekt.



Obrázek 32: Žáci testující společně své řešení programu

5.5.1 Cíle

- nastavení detekci objektu ultrasonickým senzorem v aplikaci MakeCode
- aplikace kódu pro konkrétní barvu
- práce s podmínkami – jejich správné nastavení, vnoření do kódu
- vhodné vnoření kódu pro změnu rychlosti
- práce s proměnnými – deklarace, úprava, vnoření do kódu

- hledání a opravení chyb v kódu pomocí vyhodnocování zobrazujících se hodnot na simulátoru robota a při testovacích jízdách na připravené dráze
- orientace v textovém kódu JavaScript, práce s proměnnými, podmínkou a cyklem

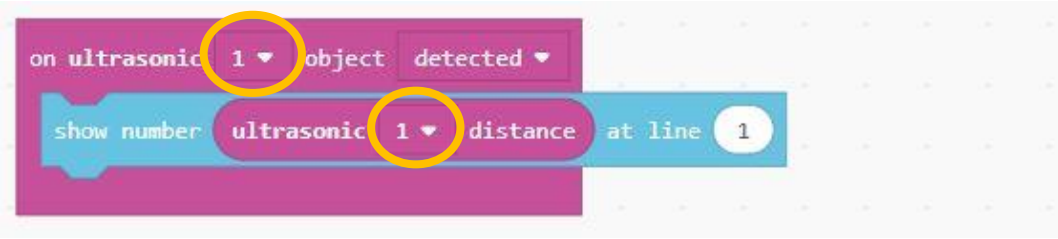
5.5.2 Postup

Úloha byla pro zpřehlednění rozdělena do dvou částí. V první části se žáci zabývali připojením senzoru a naprogramování signalizace překážek, v druhé části do programu vnořovali kód pro změnu rychlostí.

Detekce překážky

Aby byli žáci nuceni zjistit si o senzorech stavebnice EV3 více informací, museli sami přijít na to, jaký senzor pro detekci vzdálenosti vůči jinému objektu bude nejoptimálnější.

Po odsouhlasení, že se jim bude na vyřešení úlohy hodit ultrasonický senzor, měli za úkol ho pevně připevnit k robotu, aby se při jízdě nehýbal nebo dokonce nespadl. Dále měli připravit jednoduchý program, který po detekci objektu vypíše na displeji kostky danou vzdálenost, ve které se objekt od robota nachází. Stejně jako u senzoru barev i zde si museli uvědomit, že musí nastavit port, ke kterému ultrasonický senzor připojili.



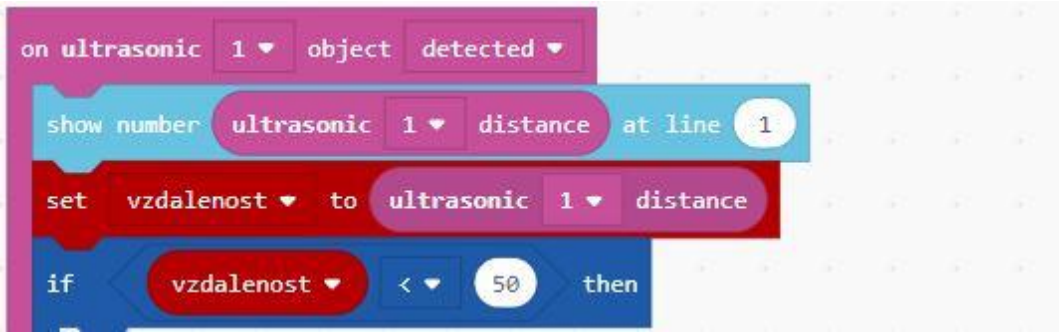
Obrázek 33: Kontrola hodnot ultrasonického senzoru na displeji kostky

Obrázek 34: Hodnota určující vzdálenost překážky od senzoru

Po vyzkoušení kódu na simulátoru měli žáci svůj kód ověřit i na svém robotu, aby si dokázali představit, jak přesná detekce vzdálenosti ve skutečnosti je a jaké je zpoždění při zobrazování hodnoty na kostce.

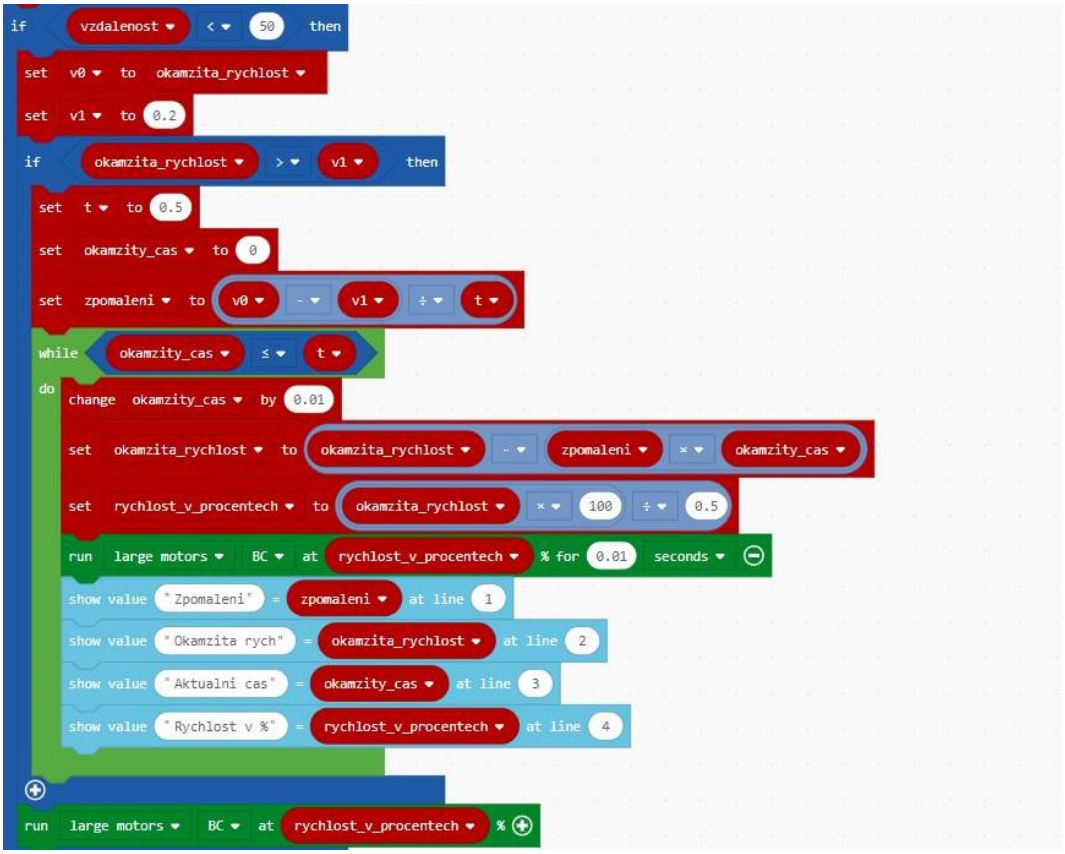
Přidání rovnoměrného zrychlení, zpomalení či zastavení

Rychlosti pohybu robota se mají měnit v závislosti na vzdálenosti objektu stojícího od něj mírně vpravo. Když jsem žákům vysvětlovala pravidla pro dané vzdálenosti, chtěla jsem vědět, pomocí jakých příkazů by omezení vzdáleností napsali do programu. Protože již s podmínkou pracovali, hned věděli, že v ní budou muset porovnávat vzdálenosti. Někteří si aktuálně naměřenou hodnotu nastavili do proměnné, jiní pracovali v podmínce rovnou s hodnotou senzoru.



Obrázek 35: Deklarace proměnné vzdalenost a podmínka pro první omezení rychlosti

Pokud je vzdálenost objektu/překážky menší než 50 cm, má robot zpomalit. Do těla této podmínky žáci proto vnořili kód pro zpomalení a odebrali v pohybu motorů čas, po který má robot jet po zpomalení nižší rychlostí. Zároveň pro rychlejší zpomalení zkrátili čas změny pohybu.



Obrázek 36: Vnořený kód pro zpomalení do těla podmínky pro detekci vzdálenosti překážky

Následně pracovali žáci opět v JavaScriptu, kde jsme si nejdříve společně připomněli správnou syntax nastavení podmínky a jejího těla. Pak samostatně ve skupinkách doplňovali kód pro zastavení, pokud je překážka vzdálena méně než 35 cm. Opět mohli čerpat z kódu z předešlých úloh, ale jen z těch, ve kterých byl použit JavaScriptový kód. Po vyřešení zápisu vyzkoušeli program na simulátoru. Pokud se měnila rychlost motorů dle zadání, kód testovali na připravené křižovatce. V případě, že by robot zastavil příliš brzo před křižovatkou nebo naopak v křižovatce, museli program upravit, aby výsledný pohyb robota splnil zadání. Pokud se robot pohyboval správně, řešili žáci již poslední úkol. Doplňovali kód detekce tak, že v případě, že

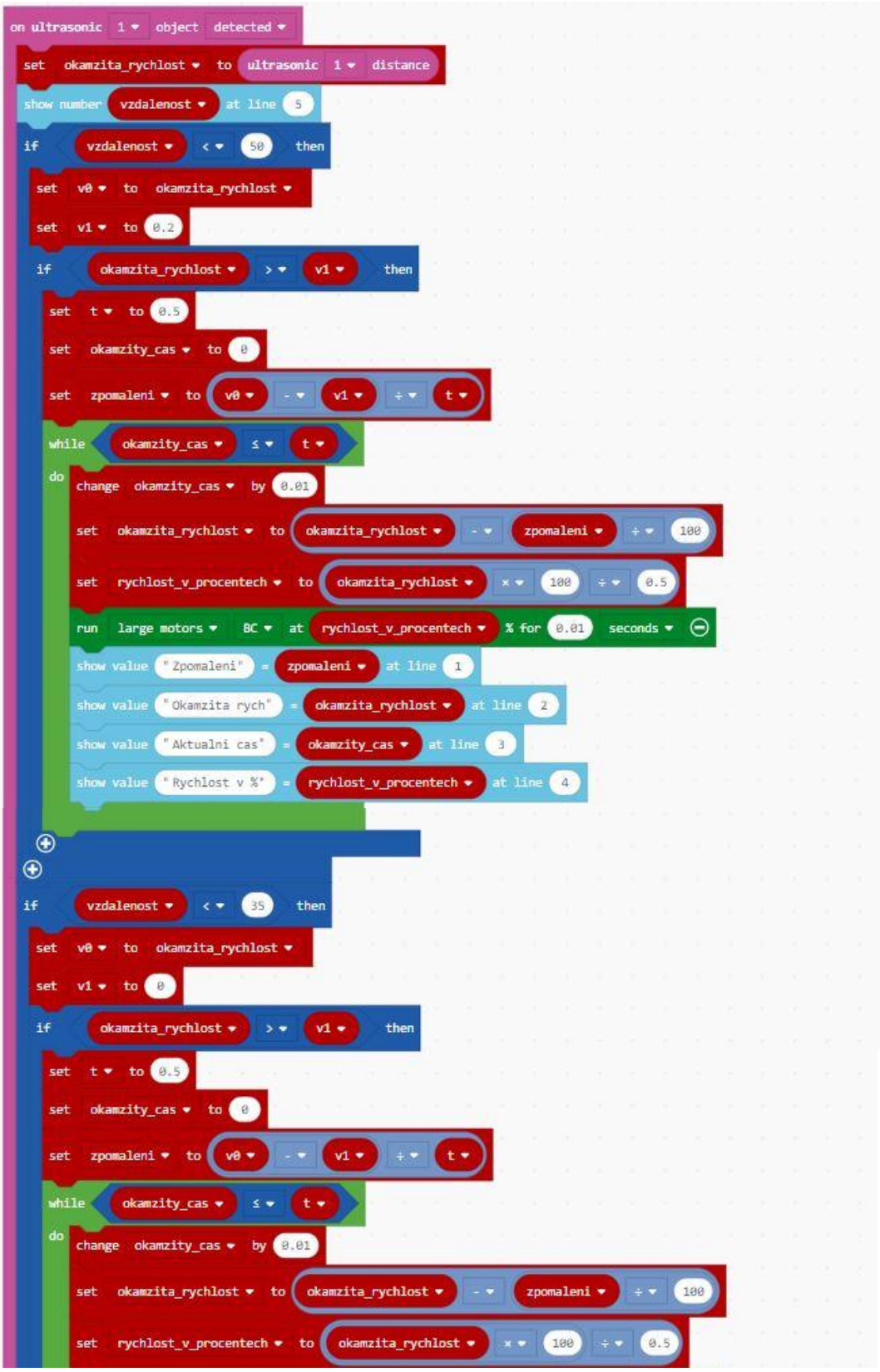
překážka „odjede“, tedy vzdálenost mezi překážkou bude větší než 50 cm, má se robot plynule rozjet a pokračovat dalších pěti sekund. Tento kód mohli opět převzít z druhé úlohy. Plynulý rozjezd robota při stálém zrychlení (kapitola 0). V případě, že robot od začátku nemá překážku po pravé straně, jede stále konstantní rychlostí po dobu šesti sekund.

5.5.3 Problémy při řešení úlohy

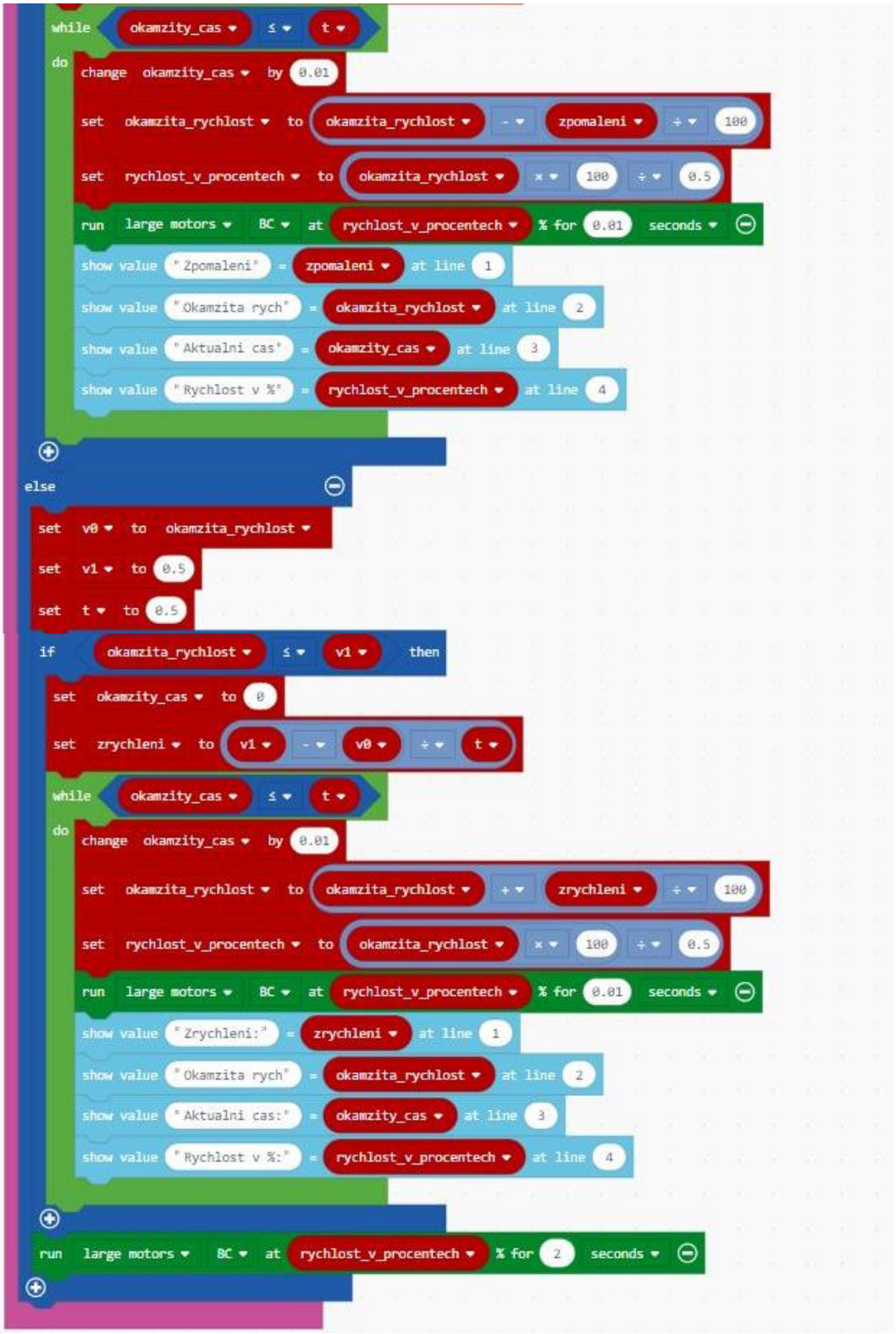
Hlavním problémem byla se špatná detekce překážky. Robot někdy zaznamenal pozdě jiný objekt a tím pádem došlo ke kolapsu na křižovatce. Protože program byl žáky zpracován správně, simulátor též neodhalil žádnou chybu, myslím si, že chybná situace na křižovatce byla způsobena větším množstvím příkazů v kódu.

Drobný problém nastal na začátku práce, protože si žáci neuvědomili, že se musí shodovat číslo portu robota v programu. Proto při první zkoušce, zda robot detekuje správnou vzdálenost, se na simulátoru vše ukazovalo správně, ale při použití vlastního robota se nezobrazovaly hodnoty.

5.5.4 Řešení úlohy



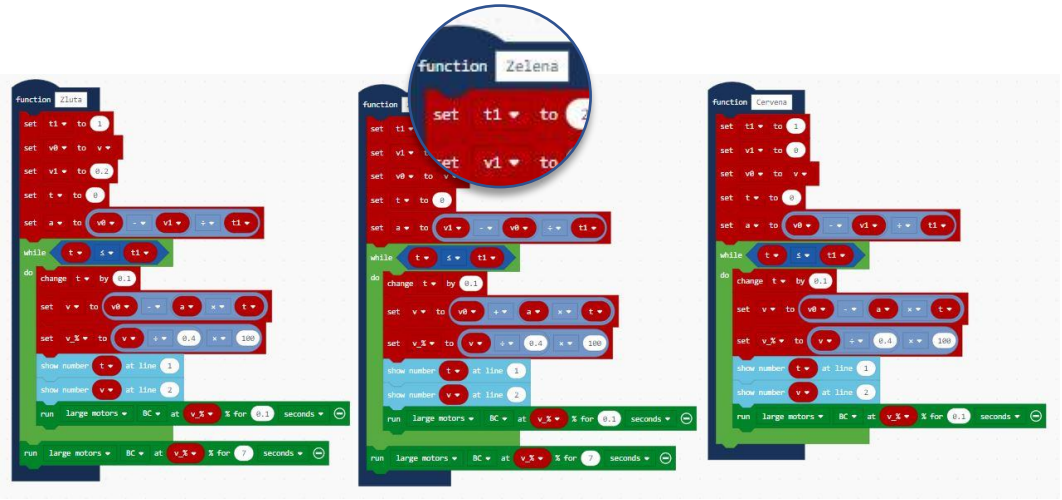
Obrázek 37: Řešení kódu na projetí křižovatky – 1. část



Obrázek 38: Řešení kódu na projetí křižovatky – 2. část

5.5.5 Pokročilé řešení úlohy

Jedna skupina žáků využila v prostředí MakeCode tzv. funkce, které mají vlastnosti podprogramů. Pro každou změnu rychlosti vytvořili funkce zvlášť, poté jen funkce volali v určitých částech kódu.



Obrázek 39: Využití funkci (podprogramů) v aplikaci MakeCode

O to jednodušší byla potom práce s ostatními úlohami zaměřenými na změny rychlosti.

Obrázek 40: Volání funkcí (podprogramů) při detekci překážky v křižovatce