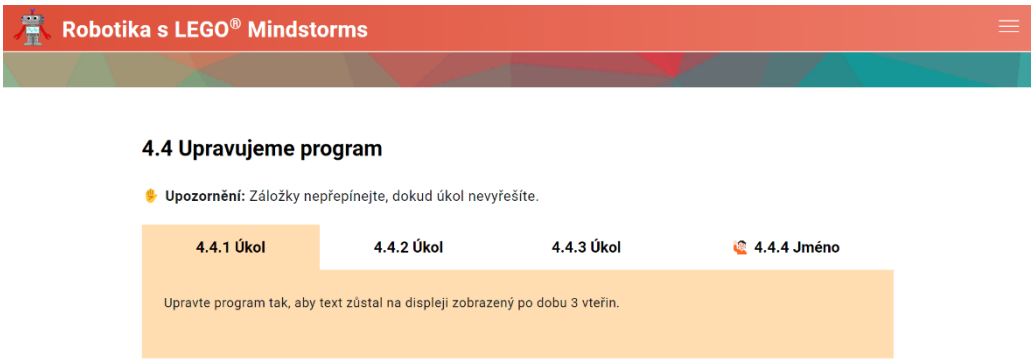
Úlohy pro robotickou stavebnici Lego MINDSTORMS EV3

Matěj Pelikán

7. Kapitoly učebnice a jejich zpracování

Některé z fotografií kódů mohou být vzhledem k rozsáhlosti programu špatně čitelné, proto všechny námi vytvořené kódy z jednotlivých kapitol byly nahrány do sdíleného Scratch studia, ve kterém se nachází 44 projektů a je dostupné na adrese: https://scratch.mit.edu/studios/33795681.

Učebnice se skládá z 11 kapitol. Každá kapitola má své podkapitoly, ve kterých se nacházejí jednotlivé úkoly jako tomu je na obrázku níže.



Obrázek 5 Snímek ze 4. kapitoly z učebnice

Zdroj: https://lego.zcu.cz/ucebnice/zvuk.html

7.1. Kapitola 1. Stavíme pojízdného robota

7.1.1. Cíl

Cílem první části kapitoly bylo samotné sestavení základního pojízdného robota. Toho bychom měli dosáhnout pomocí PDF manuálu, který byl součástí kapitoly. Dalším úkolem této kapitoly bylo propojení aplikace s kostkou pomocí USB kabelu.[14]

7.1.2. Prostředí originální aplikace

Po propojení počítače a robota se nám robot ukázal ihned v nabídce. Zde jsme se mohli rozhodnout, zda budeme chtít programy spouštět napřímo přes připojený USB kabel nebo až po jejich stažení do kostky.

7.1.3. Prostředí rozšíření Scratche

Propojení prostředí s robotem pomocí kabelu nebylo podporováno, z tohoto důvodu jsme byli nuceni použít rozhraní Bluetooth. Před spárováním robota a

rozšířeného prostředí Scratche jsme museli do počítače stáhnout aplikaci jménem Scratch Link, která propojení zajišťovala.

7.1.4. Problémy při řešení, postřehy a úpravy

Shodli jsme se na tom, že z důvodu nemožnosti propojení rozšířeného prostředí Scratche s robotem pomocí kabelu budeme nadále využívat možnosti propojení obou prostředí pouze pomocí rozhraní Bluetooth, abychom dosáhli stejných podmínek. Propojení USB kabelem se nám zdálo, i přes jeho délku, omezující, a proto jsme možnost spárování zařízení na dálku uvítali. Tímto krokem ovšem nechceme USB kabel nijak znehodnocovat, protože jsme si vědomi, že ne všechny školní počítačové učebny budou vybaveny počítači s technologií Bluetooth.

Obrázek 6 Složený robot podle návodu z originální krabice

7.1.5. Porovnání

Pokud pomineme nemožnost propojení rozšířeného prostředí Scratche a robota kabelem, tak bylo řešení této kapitoly zanedbatelně jednodušší pro uživatele aplikace. Při plnění úkolů ve Scratchi jsme nejdříve museli doinstalovat aplikaci Scratch Link, která se starala o připojení robota pomocí Bluetooth rozhraní.

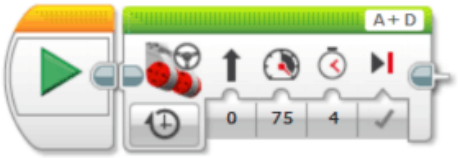
7.2. Kapitola 2. Oživení robota

7.2.1. Cíl

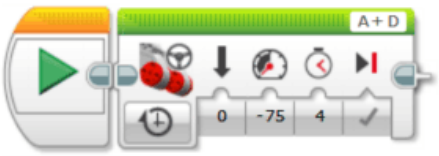
První úkol seznamuje programující s motory, porty na kostce a s jejich označením. Zbylé úkoly kapitoly odhalovaly, jak se robot ovládá a mění rychlosti při změně parametrů.[14]

7.2.2. Prostředí originální aplikace

Úkoly 2.2 a 2.4.x nám představovaly práci s blokem pro pohyb obou kol po dobu námi zvoleného časového úseku.



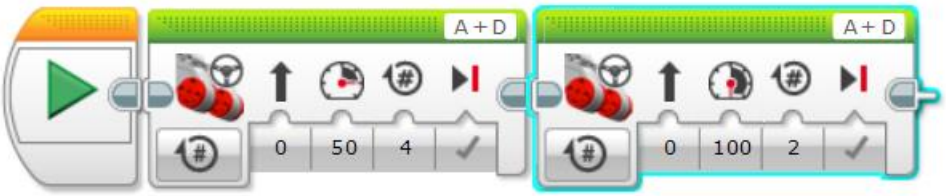
Obrázek 9 Úkoly 2.2 a 2.4.1 aplikace



Obrázek 7 Úkol 2.4.2 aplikace

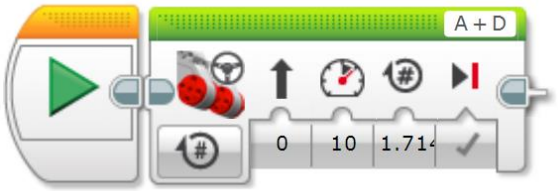
Obrázek 8 Úkol 2.4.3 aplikace

Úkol 2.5 ověřil naši zdatnost implementovat jízdu se změnou rychlosti.



Obrázek 10 Úkol 2.5 aplikace

V úkolu 2.6.5 nám šlo o vytvoření programu, pomocí kterého robot ujede vzdálenost 30 centimetrů. Náš postup pro vyřešení úkolu byl následující: změřit si, jakou vzdálenost robot ujede za jednu otočku jeho kol a následně výpočtem pomocí trojčlenky zjistit, kolik otoček bude potřeba na ujetí požadované vzdálenosti 30 centimetrů.



Obrázek 11 Úkoly 2.6.5 aplikace

Úkol 2.7 byl nepatrně náročnější na přípravu. Za pomoci papíru s úhly jsme několika pokusy vyzkoušeli, o kolik stupňů se přibližně robot otočí za jednu otočku kola. Pro finální výpočet počtu otoček pro pohyb vpravo o 90 stupňů jsme použili nám známou trojčlenku.

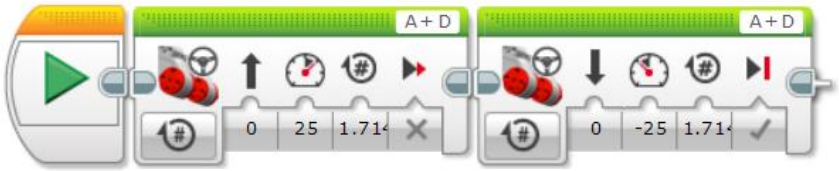
Obrázek 12 Úkol 2.7 aplikace

Obrázek 13 Měření úhlu za jednu otočku

Úkol 2.8.1 byl již z půlky hotov díky úkolu 2.6.5, kde jsme zjistili, kolik otoček kol je potřeba k uražení vzdálenosti 30 centimetrů. V tuto chvíli nám stačilo pouze vydělit počet otoček dvěma a použít blok čekání po dubu 3 sekund.

Obrázek 14 Úkol 2.8.1 aplikace

Pro úkol 2.8.2 jsme všechny potřebné údaje již znali. Stačilo jen zadat požadovaný počet otoček do bloků, jež pohybovaly robotem.



Obrázek 15 Úkol 2.8.2 aplikace

Úkol 2.8.3 byl též skoro celý vyřešený díky předchozím úkolům. Z úkolu 2.7 jsme převzali potřebný počet otoček pro otočení o 90 stupňů vpravo a vynásobili ho dvěma.

Obrázek 16 Úkol 2.8.3 aplikace

Jako finální úkol 2.9 měl robot ujet předem určenou dráhu, na které se střídala rychlost a počet otoček kol. Počet otoček pro otočení o 90 stupňů jsme již znali z předešlých úkolů, stačilo tedy pouze změnit směr zatočení v bloku pro pohyb robota dle potřeby.

Obrázek 17 Úkol 2.9 aplikace

7.2.3. Prostředí rozšíření Scratche

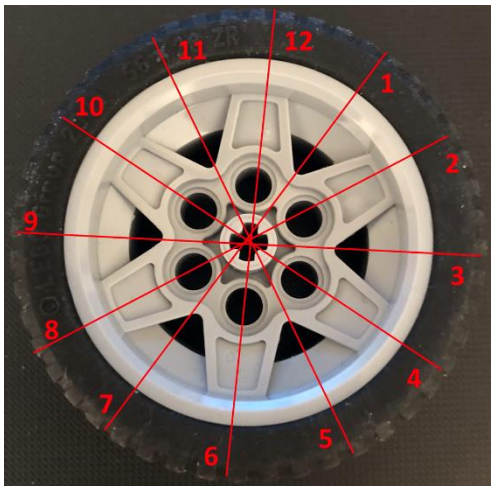
Řešení úkolů 2.2 a 2.4.x v tomto prostředí nebylo obtížnější, jen jsme museli myslet na to, že programujeme obě kola zvlášť a nemůžeme využít bloku pro pohyb celého robota jako tomu bylo v originální aplikaci.

Obrázek 18 Úkol 2.2 a 2.4.1 Scratch

Obrázek 19 Úkol 2.4.2 Scratch

Obrázek 20 Úkol 2.4.3 Scratch

Řešení úkolu 2.5 bylo náročnější na provedení. Rozšířené prostředí Scratche neposkytuje bloky s pohybem o určitý počet otoček. Tento blok jsme si museli vytvořit. Pomyslně jsme si rozdělili kolo na 12 dílů a zjišťovali, o kolik dílů se kolo otočí za 1 sekundu při 50% rychlosti. Pomocí trojčlenky jsme dopočítali počet dílů potřebných ke 4 otočkám. Obdobným pokusem jsme zjistili, kolik jich je potřeba na 2 otočky při 100% rychlosti.



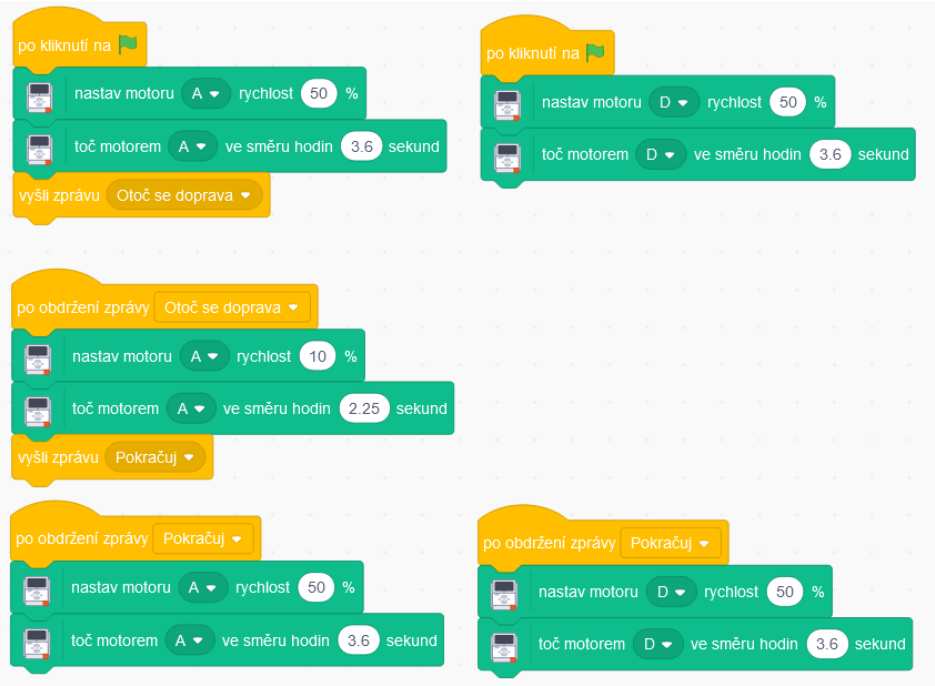
Obrázek 21 Rozdělení kola na 12 částí

Obrázek 22 Úkol 2.5 Scratch

Úkol 2.6.5 jsme řešili opět použitím trojčlenky. Změřili jsme uraženou vzdálenost robotem za dobu 1 sekundy a poté si dopočítali, kolik sekund je potřeba na ujetí vzdálenosti 30 centimetrů.

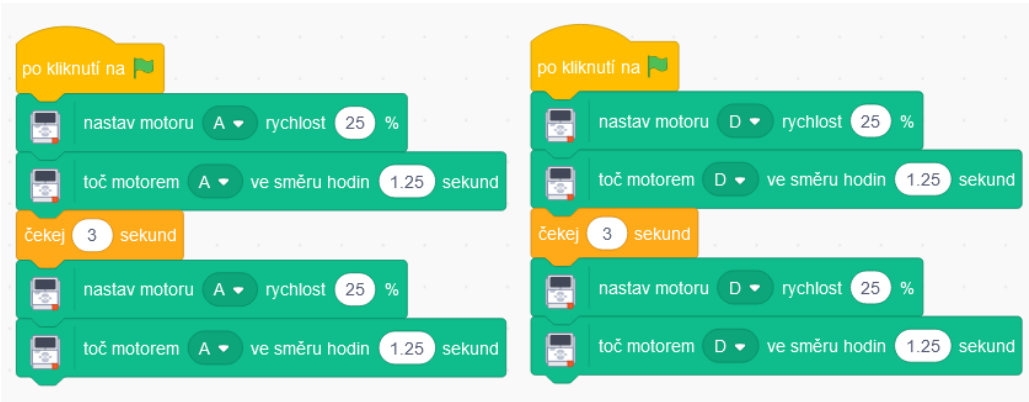
Obrázek 23 Úkol 2.6.5 Scratch

Úkol 2.7 jsme řešili podobně jako v aplikaci. Pomocí papíru s úhly jsme zjistili, kolik sekund bylo potřeba na otočení robota o 90 stupňů. Poté jsme pomocí trojčlenky vypočítali, jak dlouho musí robot jet, aby urazil vzdálenost 5 otoček stejně jako v úkolu 2.5.



Obrázek 24 Úkol 2.7 Scratch

Úkol 2.8.1 byl opět skoro celý splněný díky úkolu 2.6.5, kde jsme zjistili, kolik sekund je potřeba na překonání vzdálenosti 30 centimetrů. Počet sekund stačilo vydělit dvěma a zařadit mezi bloky blok s čekáním.

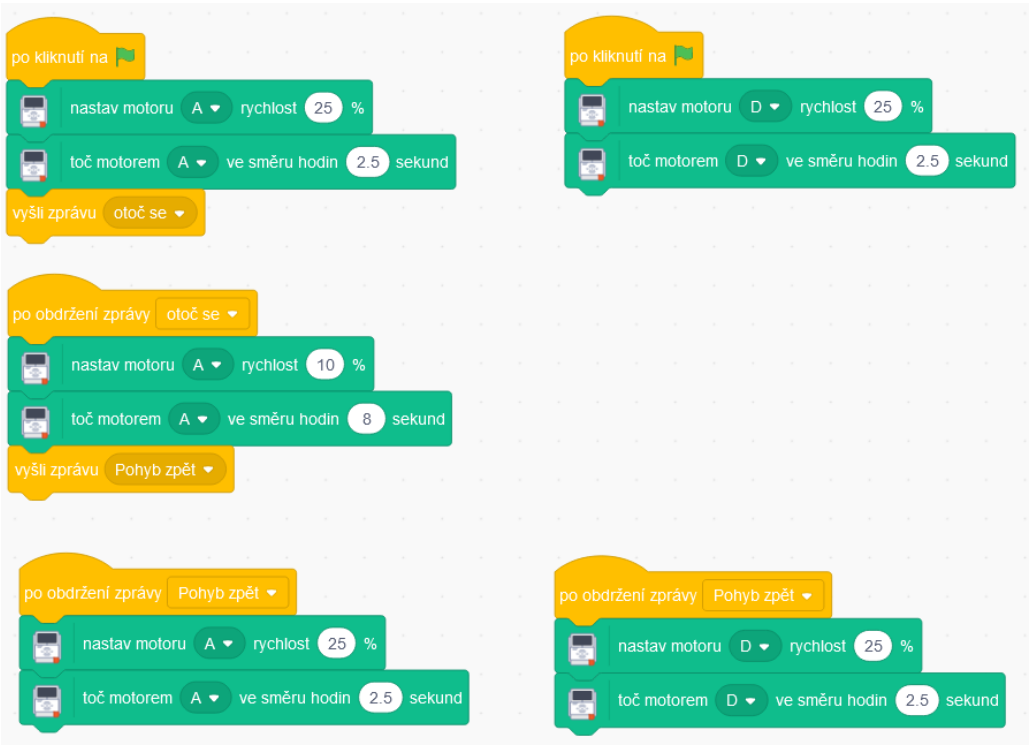


Obrázek 25 Úkol 2.8.1 Scratch

Do úkol 2.8.2 jsme opět využili poznatky z úkolu 2.6.5. Blok čekej jsme zařadili do kódu proto, aby sebou robot při změně pohybu vzad prudce netrhl.

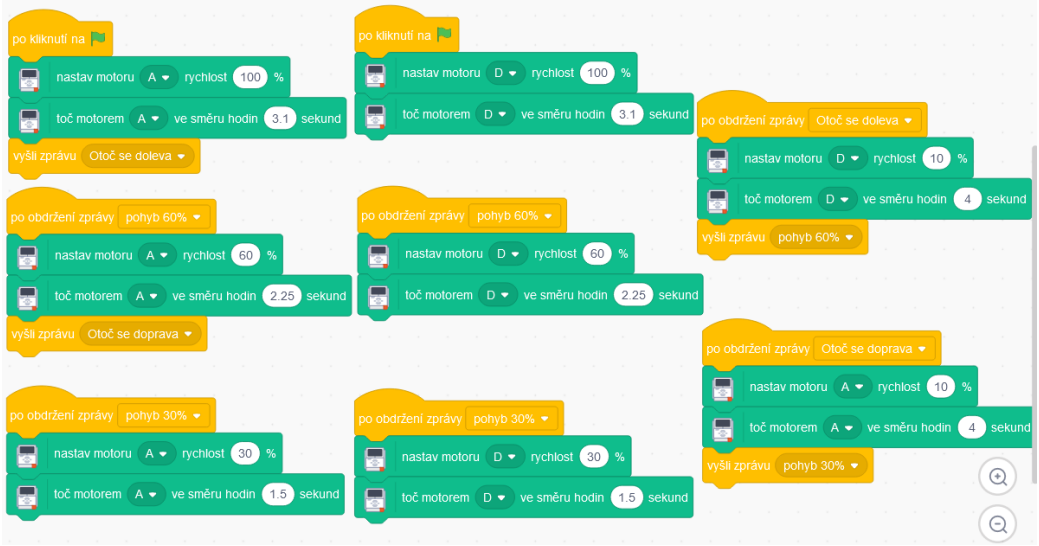
Obrázek 26 Úkol 2.8.2 Scratch

Úkol 2.8.3 byl kombinací úkolů 2.6.5 a 2.7. Čas potřebný pro pohyb vpřed o 30 centimetrů jsme již znali. Tento čas, potřebný k otočení robota o 90 stupňů, stačilo jen vynásobit dvěma, abychom provedli otočku o 180 stupňů.



Obrázek 27 Úkol 2.8.3 Scratch

Úkol 2.9 jsme nejdříve chtěli řešit trojčlenkou jako u předchozích úkolů, ale po několika testech se ujetá dráha neshodovala s řešením z aplikace. Museli jsme upravit čas jízdy robota tak, aby se ujetá dráha pro řešení ve Scratchi shodovala s dráhou z aplikace.



Obrázek 28 Úkol 2.9 Scratch

7.2.4. Problémy při řešení, poznatky a úpravy

Úpravy jsme museli dělat znatelné. Rozšířené prostředí Scratche nedisponuje bloky, které by ovládaly kola pomocí počtu otoček, ale pouze časem v provozu. Následkem tohoto jsme byli nuceni pracně zkoušet a přepočítávat, kolik sekund je rovno kolika otočkám.

Za další nedostatek rozšíření Scratche považujeme absenci bloku, který by ovládal obě kola naráz. Kvůli tomuto nedostatku jsme museli přistoupit k implementaci paralelního vlákna kódu pro každé kolo zvlášť.

7.2.5. Porovnání

Programování úkolů z této kapitoly bylo ve Scratchi z našeho pohledu obtížnější na provedení než práce s originální aplikací. Museli jsme provádět mnohá dodatečná měření, která předcházela samotnému programování, abychom byli schopni dosáhnout kýženého výsledku.

7.3. Kapitola 3. Robot ve městě

7.3.1. Cíl

Cílem kapitoly bylo naprogramování robota tak, aby se pohyboval po mapě města podle předem určených cest a pravidel. Programující by v této kapitole kombinovali bloky z minulé kapitoly a upravovali jejich atributy tak, aby určili přesnou dráhu robota na mapě.[14]

Kapitola nepřidávala žádné nové poznatky a sloužila spíše jako opakování. Z tohoto důvodu podobnosti s předchozí kapitolou jsme se rozhodli tuto kapitolu přeskočit.

7.4. Kapitola 4. Zvuk a displej

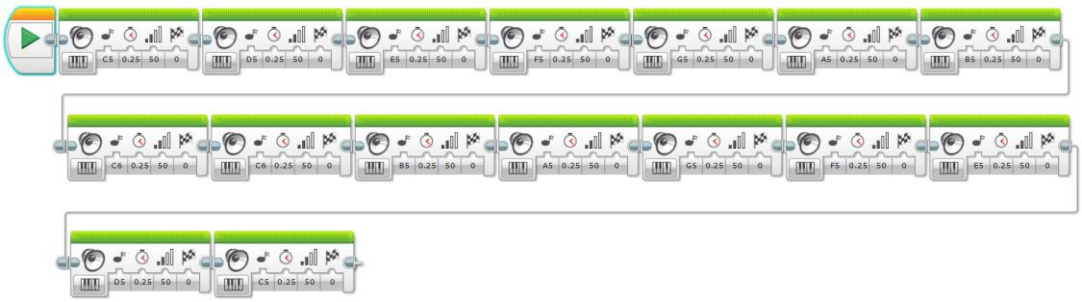
7.4.1. Cíl

Kapitola byla zaměřena na práci se zvukem a displejem kostky. Cílem bylo naučit se ovládat displej a reproduktory, jimiž je řídící kostka vybavena. Skrytým cílem této kapitoly bylo naučit programující přicházet na své i cizí chyby v programu.[14]

Vynechaly jsme úkoly 4.3.x a 4.4.x, kde se pracovalo s displejem kostky, protože v rozšířeném prostředí Scratche displej nelze programovat a nebylo by možné porovnávat implementované programy. Pracovat se zvukem lze pouze velice omezeně v rámci tónů.

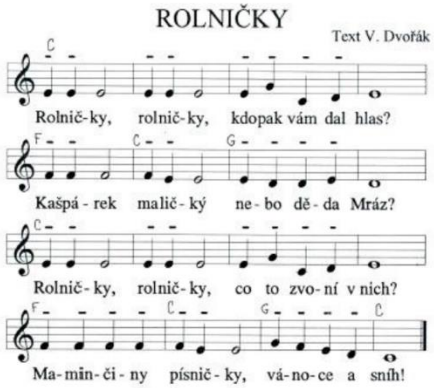
7.4.2. Prostředí originální aplikace

Úkol 4.5 nám poskytl kód, který byl záměrně implementován s chybami. My jsme v úkolu 4.6.1 chyby měli odhalit a opravit.



Obrázek 29 Úkolu 4.6.1 aplikace

Primární částí úkolu 4.6.4 bylo nalezení a opravení chyb v kódu. Sekundární část spočívala ve správném doplnění chybějících tónů. Chybějící tóny jsme nalezli na internetu.



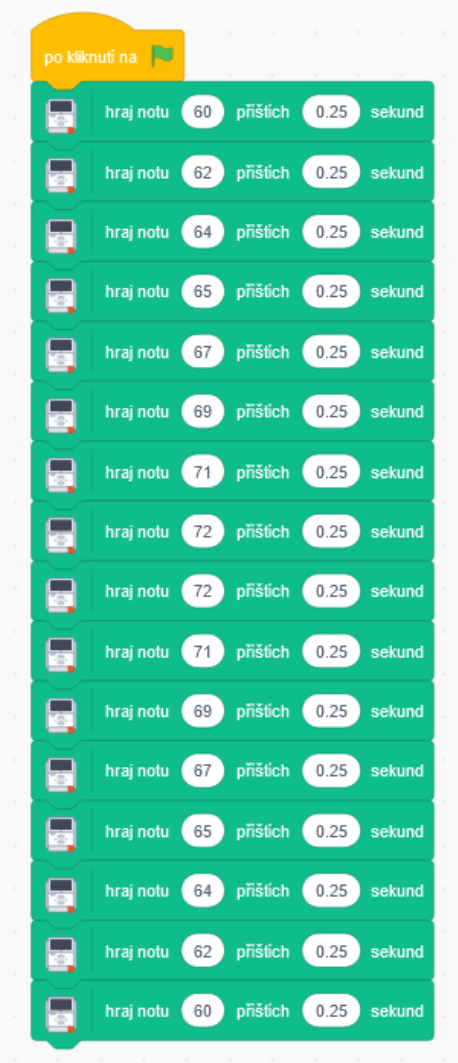
Obrázek 30 Písnička Rolničky s notami

Zdroj: https://cz.pinterest.com/pin/376543218842878225/

Obrázek 31 Úkol 4.6.4 aplikace

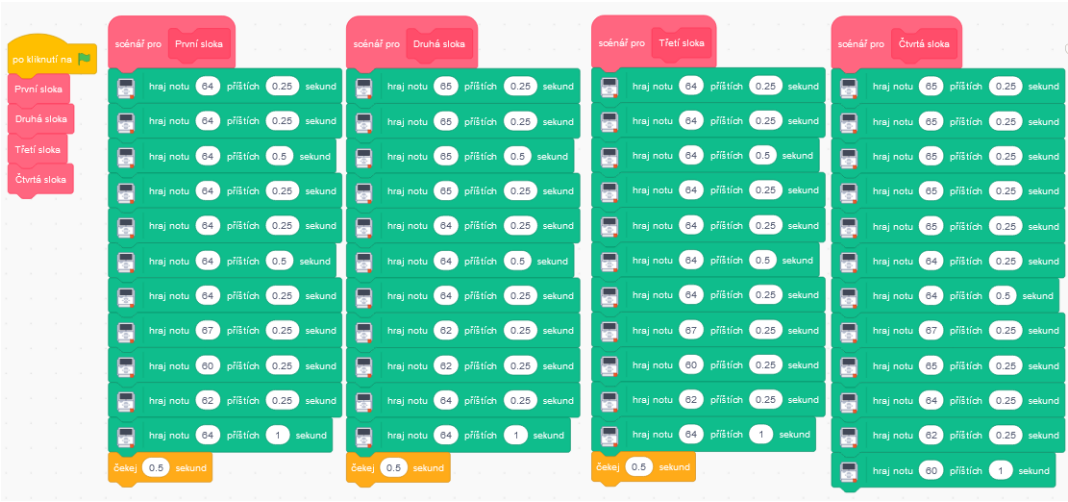
7.4.3. Prostředí rozšíření Scratche

Řešení úkolu 4.6.1 v rozšířeném prostředí Scratche bylo ztíženo tím, že jsme museli ještě před začátkem implementace kódu, poslechem odhadovat, které z daných not z rozšíření jsou nejvíce podobné notám z originální aplikace. Po 10 zvukových testech jsme přišli na to, že nota C5 z aplikace zněla přibližně stejně jako nota s číslem 60 z rozšíření, obdobně jsme přicházeli na ostatní tóny not.



Obrázek 32 Úkol 4.6.1 Scratch

Úkol 4.6.4 jsme rozdělili do slok pro lepší orientaci. Opět jsme používali náš sluch a přibližně srovnali tóny z aplikace s tóny z rozšíření.



Obrázek 33 Úkol 4.6.4 Scratch

7.4.4. Problémy při řešení, poznatky a úpravy

Hlavní překážkou byly rozdílné tóny, které jsme museli, alespoň přibližně, poslechem sjednotit. Domníváme se, že rozpoznávání tónů tímto způsobem by ve třídě mohlo způsobit zmatek a bylo by vhodné zvolit spolupráci celé třídy na tomto jednom problému. Zaregistrovali jsme, že po spuštění kódu programu z rozšířeného prostředí Scratche byla kvalita hraných tónů z kostky horší než u programů spuštěných z originální aplikace.

Jedním z možných řešení nemožnosti práce s displejem v rozšířeném prostředí by mohla být práce s oknem Scratche, kde se nachází postavička. V této kapitole by toto nahrazení postrádalo smysl, protože z úloh zaměřených na práci s displejem kostky by se staly úlohy na práci s oknem Scratche bez potřeby využití displeje robota.

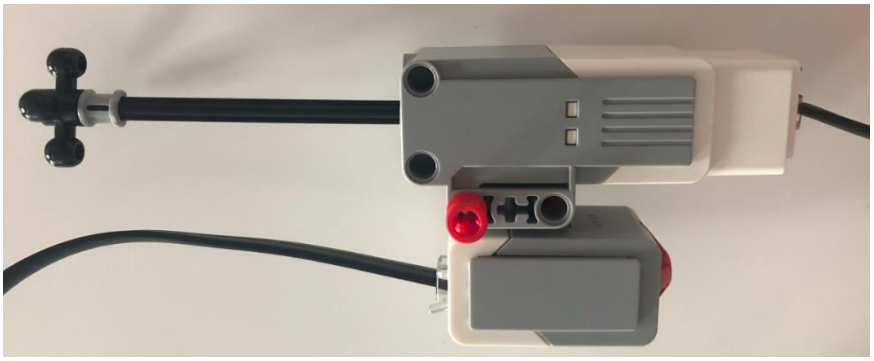
7.4.5. Porovnání

Pomineme-li chybějící bloky ovládající displej kostky a kvalitu hraných zvuků. Byla pro nás náročnost práce v obou prostředích ve výsledku stejná.

7.5. Kapitola 5. Mixér aneb pracujeme s motorem

7.5.1. Cíl

Cílem kapitoly bylo seznámit programující s problematikou podmínek a konečného i nekonečného opakování [14]. Kapitola nás měla seznámit s bloky, které jsou na tuto problematiku zaměřeny, na jednoduše vypadajícím mixéru.



Obrázek 34 Složený mixér

7.5.2. Prostředí originální aplikace

Úkol 5.2.4 měl nejspíše ověřit, zda jsme správně zapojili kabely od motoru a dotykového senzoru do kostky. Pro řešení jsme nevyužili žádné nové bloky.

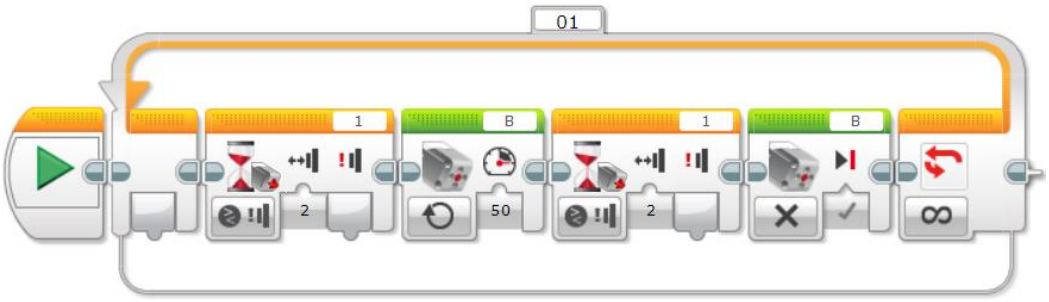
Obrázek 35 Úkol 5.2.4 aplikace

V úkolu 5.4.2 jsme využívali dotykový senzor. Úkol jsme mohli vyřešit použitím bloku Switch nebo bloku Wait. Obě možnosti vedly ke stejnému řešení. Pro ilustraci řešení jsme se rozhodli využít blok Wait.

Obrázek 36 Úkol 5.4.2 aplikace – blok Wait

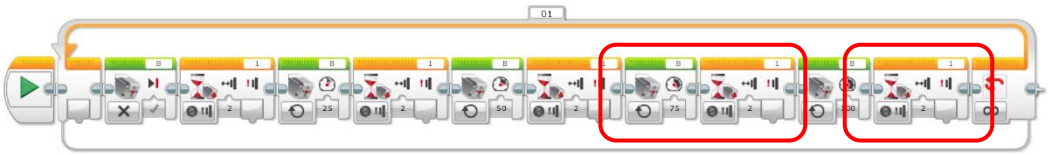
Úkol 5.4.3 byl rozšířením stávajícího programu o možnost mixér vypnout a znovu zapnout podobně jako u reálných mixérů. Použili jsme blok s nekonečným opakováním. Uvnitř nekonečného cyklu jsme pro inicializaci programu použili blok

čekající na stisknutí dotykového senzoru. Po jeho stisku se motor zapnul a běžel do doby, než druhý čekající blok zaregistroval stisknutí dotykového senzoru signalizující vypnutí pohybu.



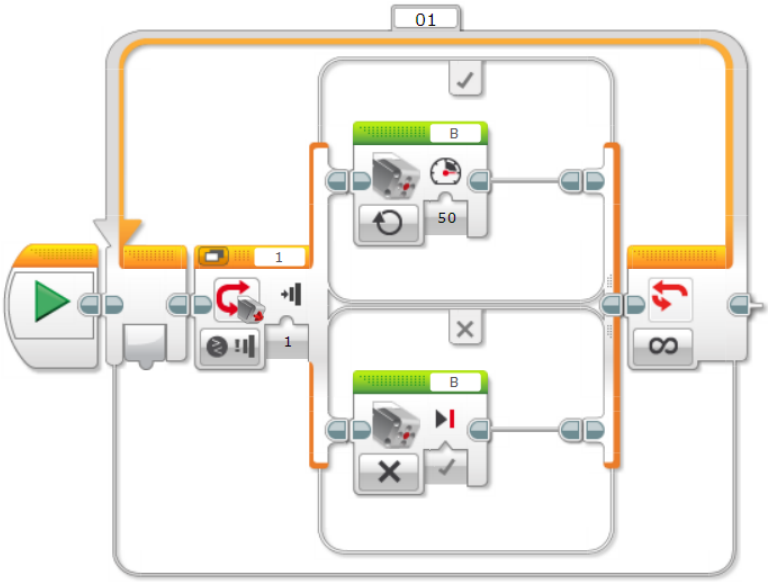
Obrázek 37 Úkol 5.4.3 aplikace

Úkol 5.5 byl letmým poohlédnutím do 4. kapitoly za úkolem 4.6.4, u kterého jsme opravovali chyby a doplňovali chybějící bloky programu. Ve stávajícím kódu jsme museli doplnit chybějící blok Wait na konec cyklícího bloku, aby se hned po stisku dotykového senzoru pro přepnutí motoru na sílu 100 program nevypnul. Další bloky, které jsme doplnili, byly bloky pohybu motoru se sílou 75 a blok Wait, který čekal na stisknutí dotykového senzoru.



Obrázek 38 Úkol 5.5 aplikace

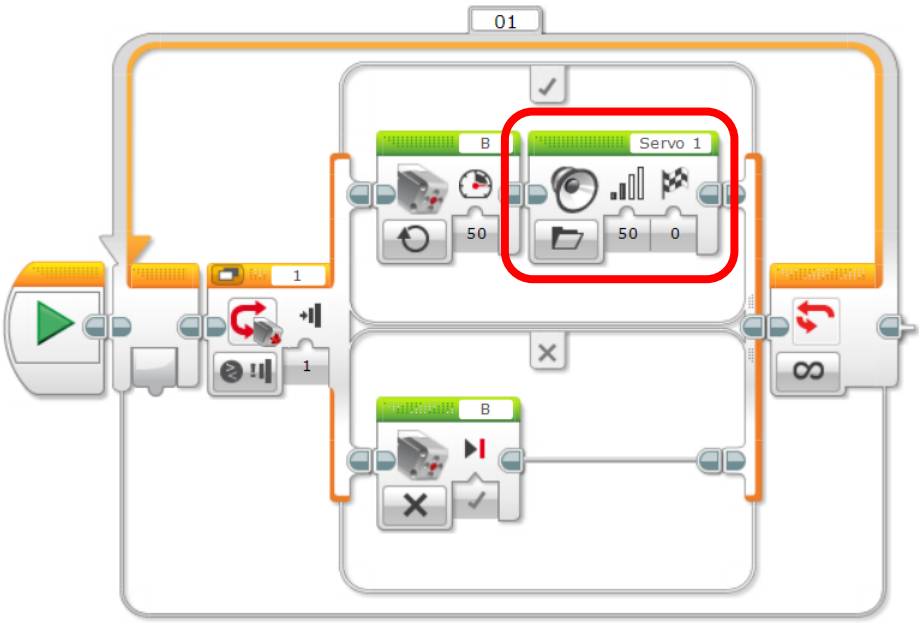
V úkolu 5.6.2 jsme využili blok nekonečného cyklu v kombinaci s blokem Switch. Blok s nekonečným cyklem obstarával stálé volání bloku Switch, který kontroloval podmínku, zda byl či nebyl stisknutý dotykový senzor.



Obrázek 39 Úkol 5.6.2 aplikace

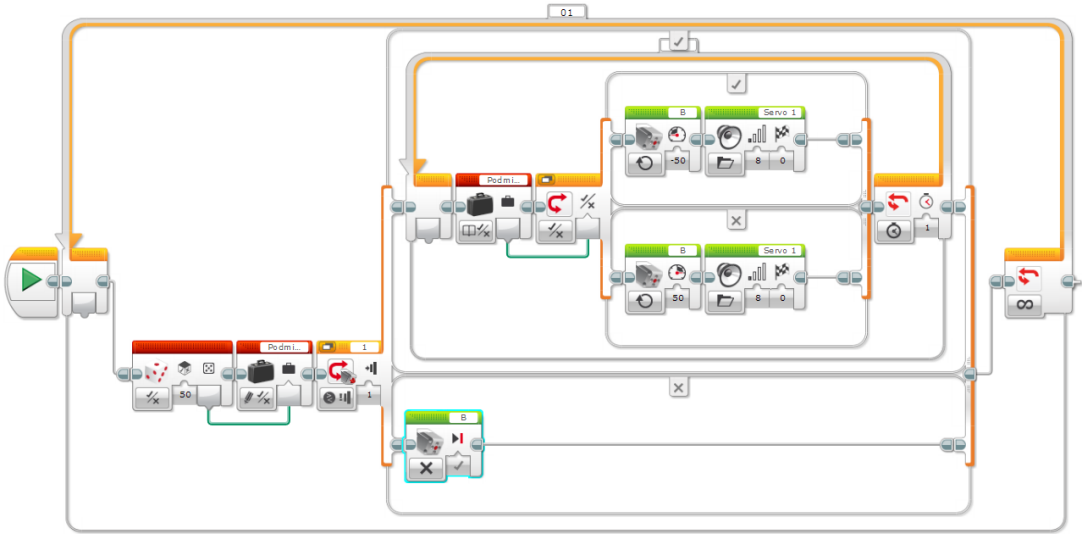
Úkol 5.6.3 jsme přeskočili z důvodu nemožnosti implementace programu v rozšířeném prostředí Scratche, které neumožňuje práci se světelným podsvícením kostky.

Úkol 5.6.4 byl rozšířením úkolu 5.6.2. Splnili jsme ho přidáním bloku se zvukem, který se při stisku dotykového senzoru zapnul.

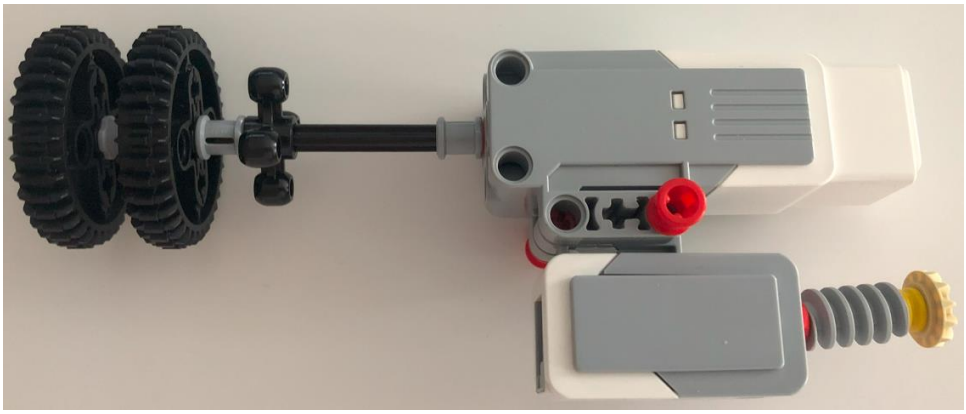


Obrázek 40 Úkol 5.6.4 aplikace

K vyřešení úkolu 5.7, který byl rozšířením úkolu 5.6.4, jsme použili bloky Random a Variable z karty Data operations. Blok Random zapisoval hodnotu do bloku Variable. Oproti řešení z úkolu 5.6.4 jsme navíc použili cyklus, který se opakoval po 1 sekundě a činil tak program plynulejším. Do tohoto jednosekundového cyklu jsme vložili blok Switch, který rozhodoval o rotaci motoru v kladném či záporném směru podle náhodně zvolené proměnné.



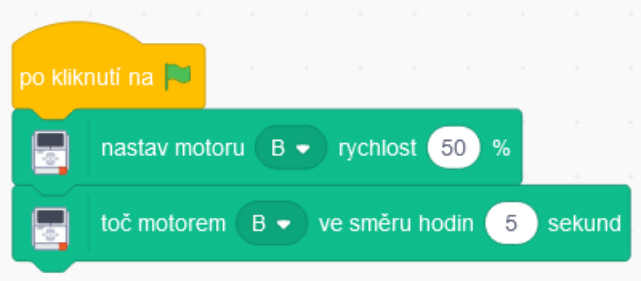
Obrázek 41 Úkol 5.7 aplikace



Obrázek 42 Vylepšený mixér – kvedlačka

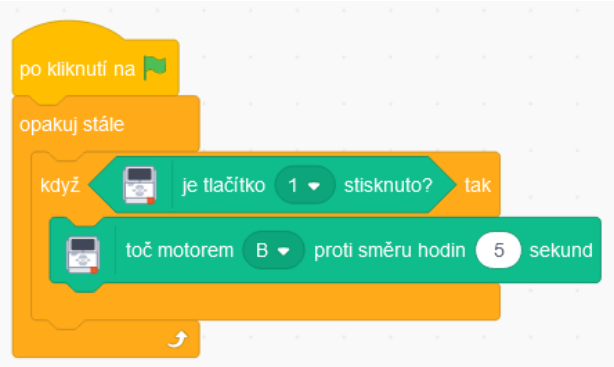
7.5.3. Prostředí rozšíření Scratch

Řešení úkolu 5.2.4 nebylo nijak extrémně rozdílné oproti řešení z prostředí aplikace. Tím, že jsme pracovali s jedním motorem, nebylo potřeba užití paralelního programování.



Obrázek 43 Úkol 5.2.4 Scratch

V úkolu 5.4.2 jsme využívali dotykový senzor. Oproti řešení z aplikace, jsme v rozšířeném prostředí Scratche využili podmínkový blok s volitelnou podmínkou, na co má blok čekat. Podmínkový blok ověřoval, zda byl dotykový senzor stisknut či ne.

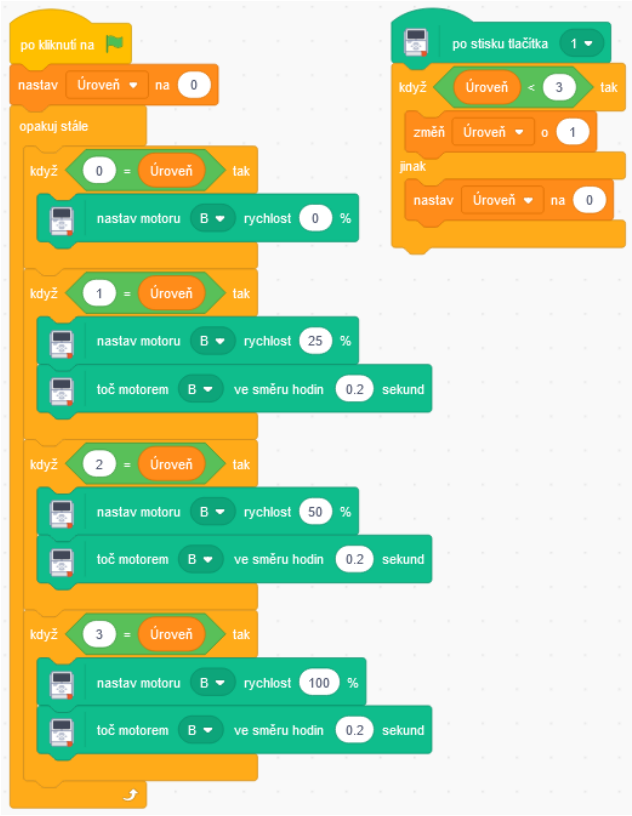


Obrázek 44 Úkol 5.4.2 Scratch

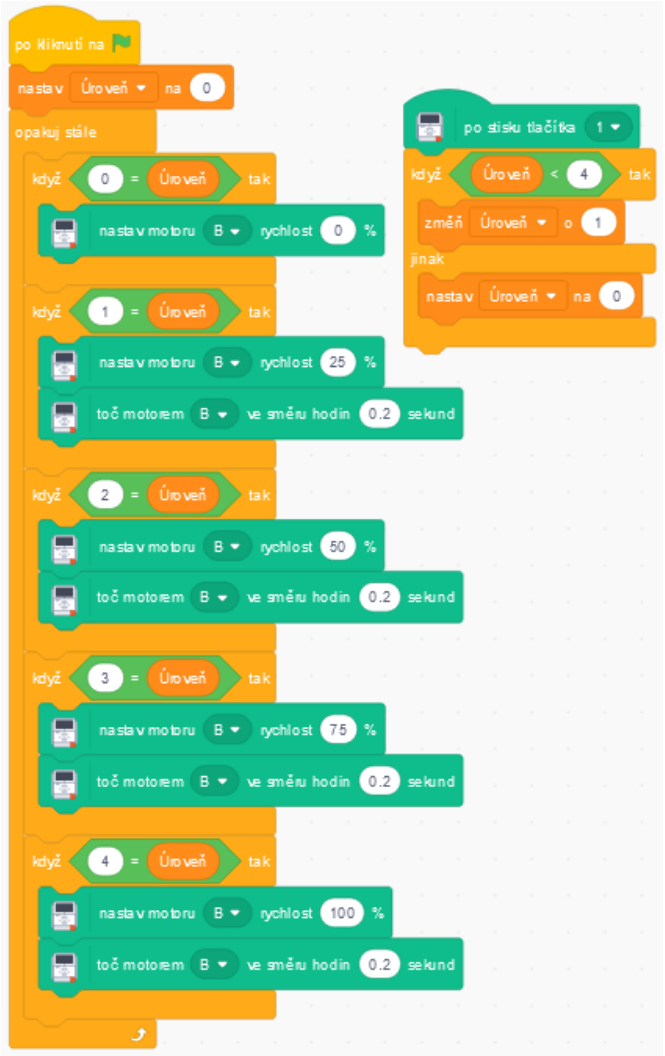
Úkol 5.4.3 potřeboval z naší strany trochu vynalézavosti. Využili jsme paralelního programování dvou vláken. Vlákno první kontrolovalo stisk dotykového senzoru a přepisovalo hodnotu proměnné mezitím, co druhé vlákno ovládalo pohyb mixéru. Motor byl zapnutý po dobu 0,2 sekundy, protože pouze takto dokázal rychle reagovat na vypnutí a zároveň sebou nejméně trhal.

Obrázek 46 Úkol 5.4.3 Scratch

Pro úkol 5.5 jsme vytvořili dvě podoby programu. První podoba programu byla před přidáním rychlosti 75 a druhá po jejím začlenění. Stejně jako u předchozího úkolu jsme využili paralelního programování pro kontrolu stisku dotykového senzoru a pro pohyb motoru. Použili jsme několik podmínkových bloků za sebou, abychom mohli snadně přepínat mezi úrovněmi rychlostí. Délku pohybu motoru jsme nechali na hodnotě 0,2 sekund.

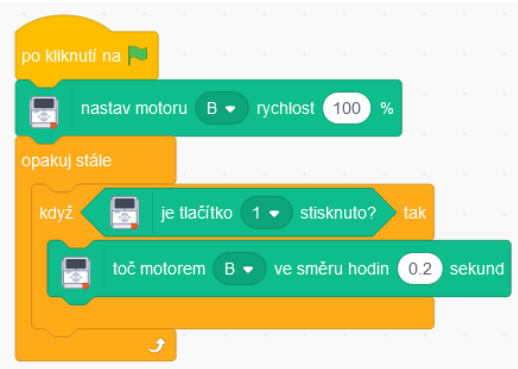


Obrázek 45 Úkol 5.5 před přidáním rychlosti 75 Scratch



Obrázek 48 Úkol 5.5 po přidáním rychlosti 75 Scratch

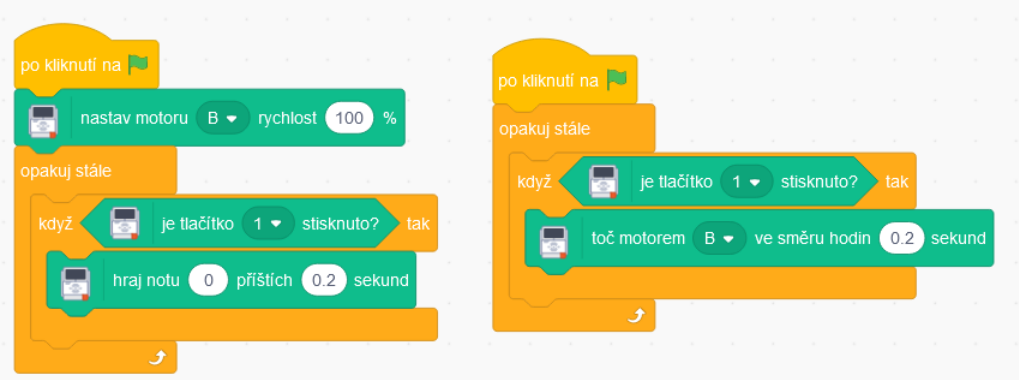
Řešení úkolu 5.6.2 v rozšíření Scratche bylo značně jednodušší než řešení v aplikaci. Za použití nekonečného cyklu a podmínkového bloku, který testoval stisk dotykového senzoru, jsme vyřešili celý úkol. Ze stejného důvodu jako u úkolu 5.4.3 jsme využili pohyb motoru po dobu 0,2 sekundy.



Obrázek 47 Úkol 5.6.2 Scratch

Úkol 5.6.3 jsme přeskočili z již zmíněného důvodu nemožnosti programovat podsvícení na kostce.

Úkol 5.6.4 jsme vyřešili použitím dvou vláken. V prvním i druhém vláknu běžela část kódu, který kontroloval, zda byl dotykový senzor stisknut. Vnitřní části vláken se lišily podle úkolu, který měly vykonávat, buď pohyb nebo hraní tónu.



Obrázek 49 Úkol 5.6.4 Scratch

K vyřešení úkolu 5.7 jsme použili proměnnou, do které jsme vložili náhodnou hodnotu 0 nebo 1. Tato čísla měla reprezentovat pravdu či nepravdu pro podmínkový blok, jež určoval směr rotace motoru. Pokud bychom do řešení chtěli obsáhnout i vydávání zvuku při stisku dotykového senzoru, museli bychom použít dalšího vlákna, které by testovalo, zda byl dotykový senzor stisknut jako v předešlém úkolu.

Obrázek 50 Úkol 5.7 Scratch

7.5.4. Problémy při řešení, poznatky a úpravy

Problémy při řešení úkolů z této kapitoly nebyly náročné, jako ty z kapitoly 2, kde jsme museli dopočítávat otočky a úhly. Potom, co jsme zapojili do programu několik paralelně spuštěných vláken, se všechny úkoly daly naprogramovat v rozšířeném prostředí Scratche bez větších obtíží.

Námi zaregistrovaný rozdíl v řešení z aplikace oproti řešení z rozšířeného prostředí Scratche byl v možnostech používat různé zvuky. Scratch je omezen pouze na tóny o určité délce na rozdíl od aplikace, která má k dispozici celou knihovnu zvuků.

Posledním postřehem při řešení úloh bylo zjištění, že i když jsme měli zvuky v kostce nastaveny na maximální hodnotu hlasitosti, tak i přesto byly tóny programů spouštěných z rozšíření Scratche tišší než totožné tóny a zvuky spouštěné z aplikace.

7.5.5. Porovnání

Řešení v aplikaci nevyžadovaly použití paralelního programování a samotná aplikace disponovala větším množstvím zvuků. Vytvořené programy v aplikaci se postupem kapitolou stávaly obsáhlejší a pro někoho by se mohly zdát komplexní a nepřehledné oproti řešením ze Scratche. U této kapitoly bychom dali přednost plnění úkolů v rozšíření Scratche.

7.6. Kapitola 6. Závora na parkovišti

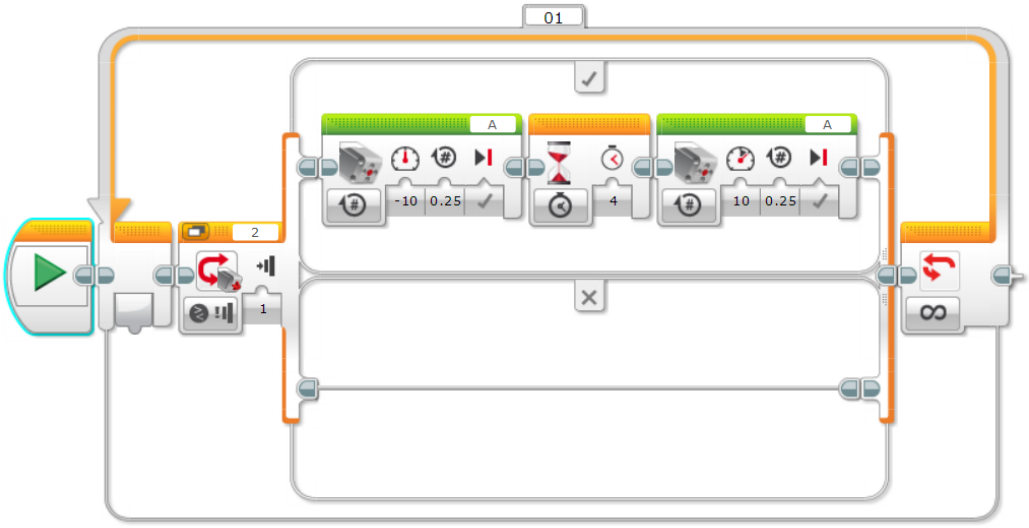
7.6.1. Cíl

Prvním z cílů kapitoly bylo osvěžení si doposud získaných znalostí z předchozích kapitol. Dalším z cílů bylo správné pochopení toho, že se zadané úkoly dají řešit rozdělením na dílčí části, které je následně jednodušší programovat. [14]

Obrázek 51 Složená závora podle návodu v online učebnici

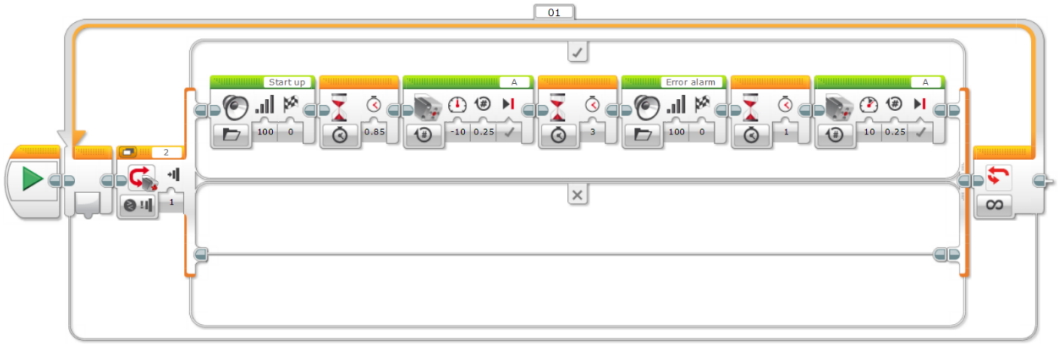
7.6.2. Prostředí originální aplikace

V úkolu 6.3.4 jsme využili dotykový senzor, abychom mohli ovládat otevírání a zavírání brány. Brána se měla po stisku dotykového senzoru otevřít na dobu 4 sekund a po uplynutí času zase zavřít. Využili jsme blok pro nekonečné cyklení a dovnitř tohoto bloku vložili blok Switch, který kontroloval, zda byl dotykový senzor stisknut.



Obrázek 52 Úkol 6.3.4 aplikace

Úkol 6.4.1 byl rozšířením předešlého úkolu 6.3.4. Měli jsme za úkol vylepšit bránu o zvukovou signalizaci před otevřením i zavřením. Abychom dodrželi časovou dotaci 4 sekund na průjezd pod bránou, snížili jsme čas otevřené brány na 3 sekundy, ale zároveň přidali čekání 1 sekundu po zvukové signalizaci před zavřením.

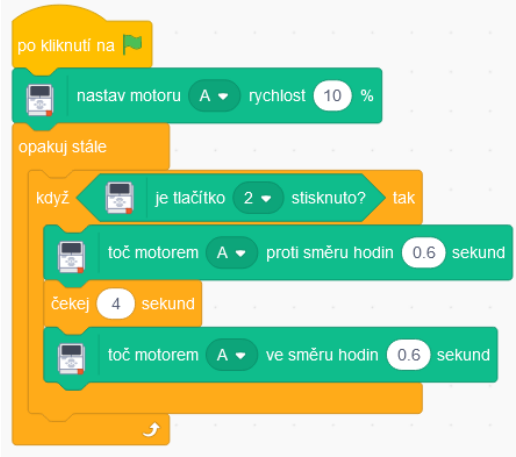


Obrázek 53 Úkol 6.4.1 aplikace

Úkol 6.4.2 jsme vynechali, protože v rozšířeném prostředí Scratche nelze programovat podsvícení kostky, tím pádem nemělo cenu programovat úkol ani v aplikaci.

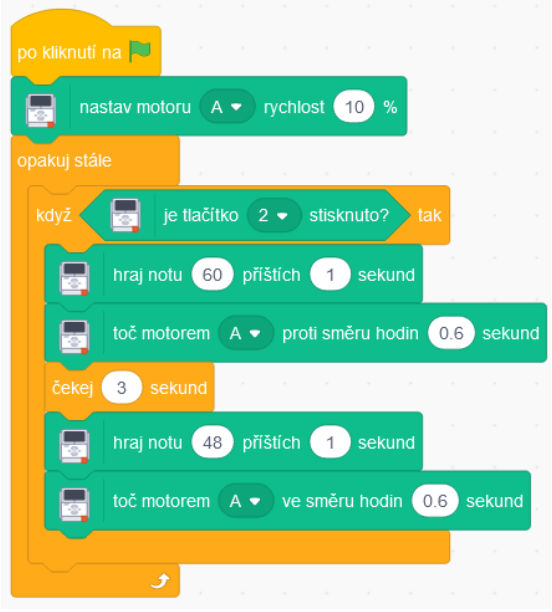
7.6.3. Prostředí rozšíření Scratch

Úkol 6.3.4 jsme začínali blokem pro nekonečný cyklus, do kterého jsme umístili podmínkový blok, jež testoval stisk dotykového senzoru. Na čas potřebný k otevření a zavření závory jsme přišli několika testy.



Obrázek 54 Úkol 6.3.4 Scratch

Úkol 6.4.1 rozšiřoval úkol předchozí o zvukovou signalizaci před otevřením a zavřením brány. Opět jsme nechtěli zasahovat do časové dotace otevřené brány, a proto jsme upravili program tak, aby celkový čas zůstal 4 sekundy.



Obrázek 55 Úkol 6.4.1 Scratch

Úkol 6.4.2 jsme přeskočili, protože vyžadoval práci s podsvícením kostky, které v rozšířeném prostředí Scratche nelze programovat.

7.6.4. Problémy při řešení, úpravy a poznatky

Před samotným programováním jsme museli několika málo pokusy zjistit, jak dlouho motor při své 10% síle musí rotovat, aby se závora dostala do svislé polohy. Tyto pokusy bychom označili jako poznatky než problémy.

7.6.5. Porovnání

U této kapitoly nebylo potřeba používat složitějších struktur bloků k úspěšnému splnění úkolů. Obě prostředí poskytovala uspokojivé řešení. Drobným a jediným rozdílem v rozšířeném prostředí Scratche byla nutnost vyzkoušet, kolik sekund je potřeba k otevření nebo zavření závory.

7.7. Kapitola 7. Automatická závora

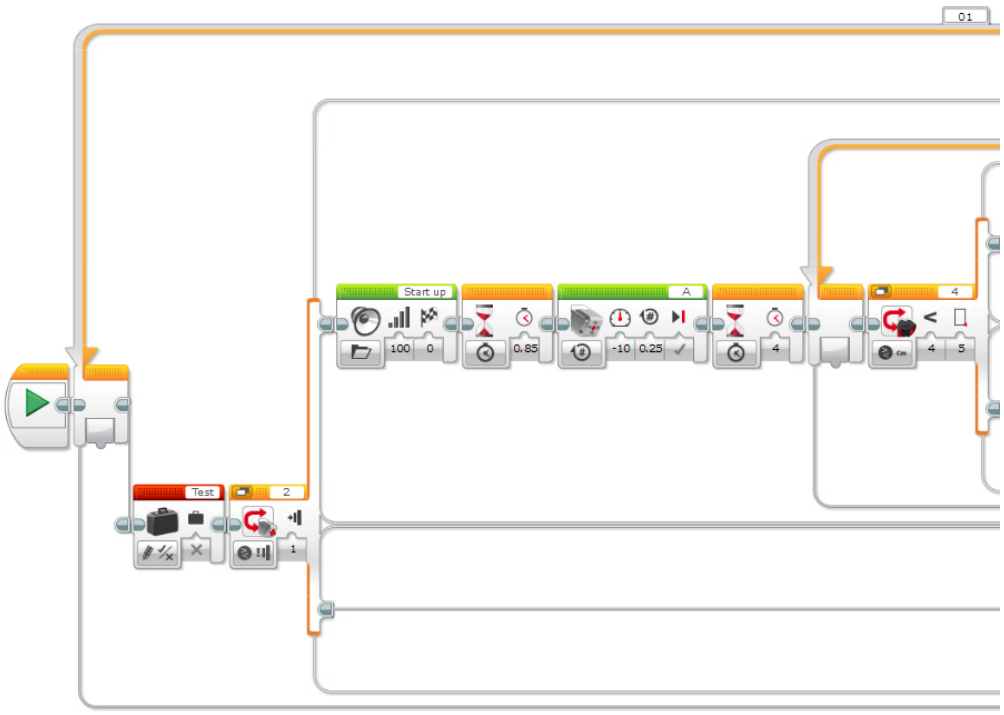
7.7.1. Cíl

Cílem této rozšiřující kapitoly bylo plně automatizovat chod závory za pomoci dvou nových senzorů na určení barvy a vzdálenosti od překážky. [14]

7.7.2. Prostředí originální aplikace

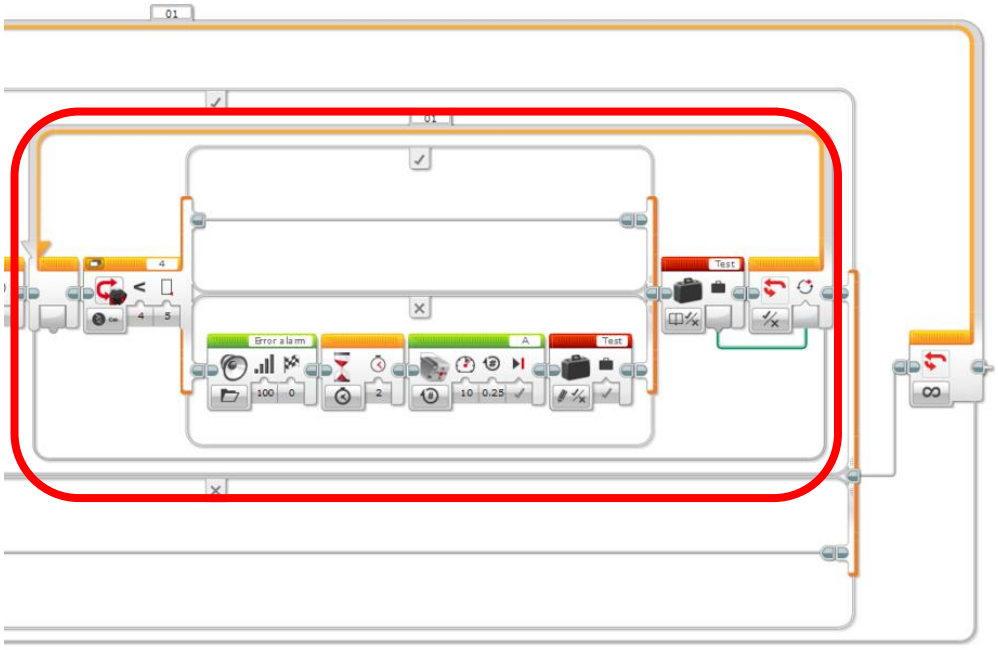
Úkol 7.4.1 měl rozšířit program vytvořený v minulé kapitole zapojením senzoru indikace vzdálenosti. Senzor hlídal prostor pod závorou a nedovoloval ji sklopit do doby, než z prostoru pod ní překážka nebyla odklizena. Obrázek kódu jsme rozdělili na 2 části pro lepší čitelnost.

Do 1. části kódu jsme vložili na začátek cyklu proměnnou se jménem Test, která je po začátku vnějšího nekonečného cyklu nastavena na hodnotu nepravda. Tato proměnná se starala o přerušení vnitřního cyklu obstarávajícího uzavření brány.

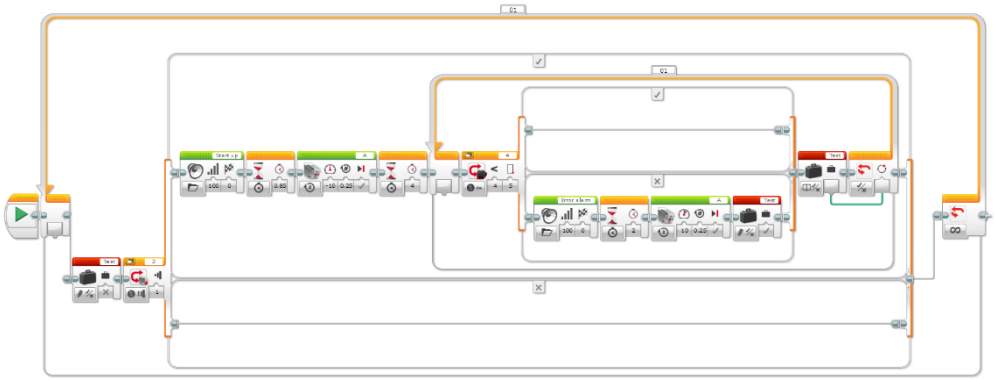


Obrázek 56 Úkol 7.4.1 část 1. aplikace

Ve 2. části kódu je vidět zvýrazněný cyklus v němž je blok Switch. Pokud byla překážka ve vzdálenosti větší než námi zvolená hranice, tak se závora sklopila a proměnná Test se přepsala na pravdu. Tato změna zapříčinila ukončení cyklu kontroly překážky pod závorou a závora se následně zavřela.

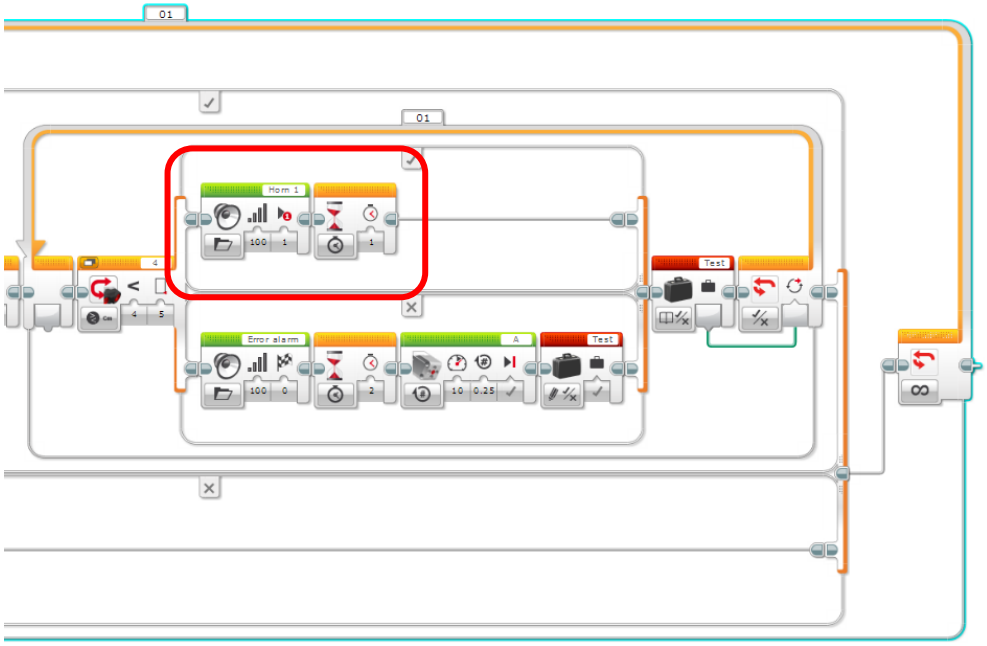


Obrázek 57 Úkol 7.4.1 část 2. aplikace



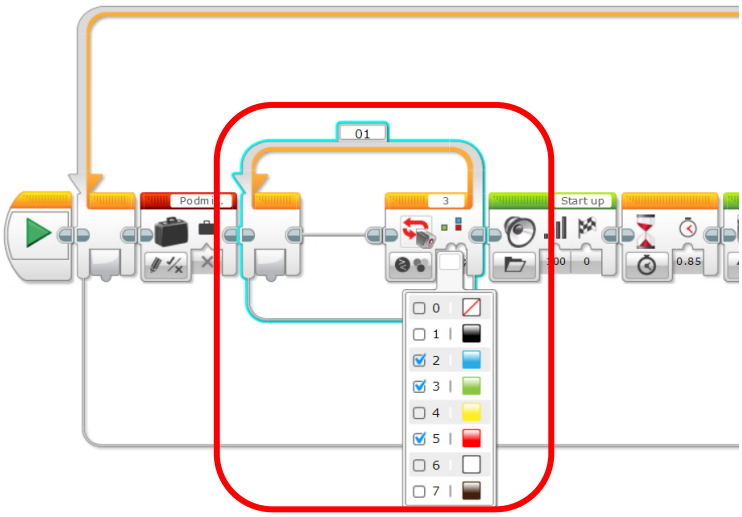
Obrázek 58 Úkol 7.4.1 celý kód aplikace

Úkol 7.5 rozšiřoval úkol předešlý o zvukovou signalizaci, která zazněla při detekci překážky pod závorou po uplynutí 4 sekund. Změna bloků proběhla pouze v bloku Switch. Přidali jsme zvukovou signalizaci a čekání, aby zvuk nebyl nepřetržitý. Do řešení jsme nezahrnuli bloky ovládající světlo kostky, protože je nebylo možné naprogramovat v rozšířeném prostředí Scratche.



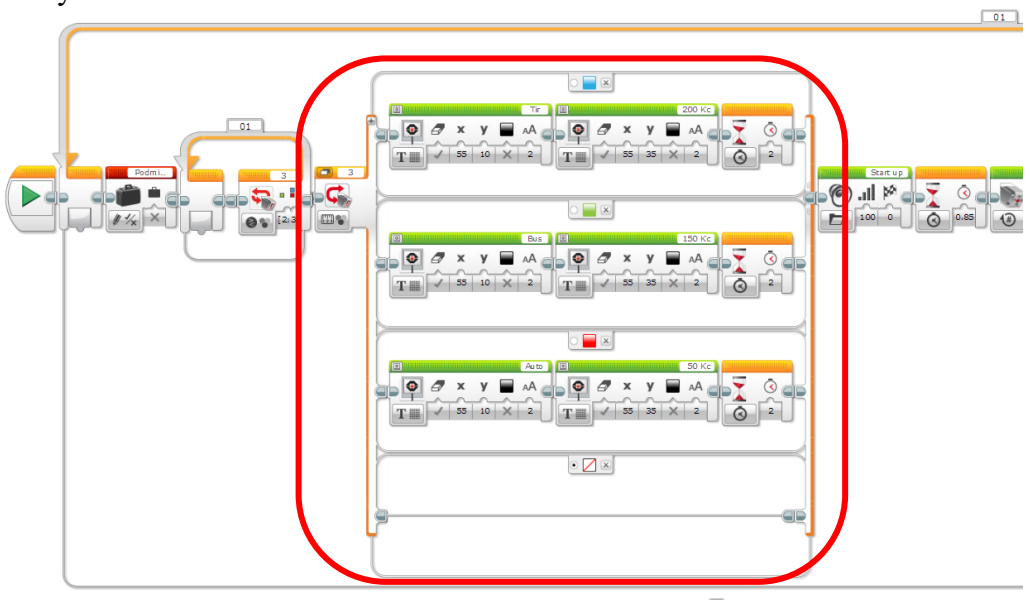
Obrázek 59 Úkol 7.5 aplikace

Úkol 7.9.1 mýtná brána. Úkol byl prostý, pomocí senzoru rozpoznávání barev poznat červenou, modrou a zelenou barvu. Vyjmenovaným barvám bylo dovoleno otevřít bránu. Kód jsme modifikovali přidáním bloku cyklu, který se opakoval do doby, než byla přiložena správná barva, a odebráním části kódu, jež po stisku dotykového senzoru otevíral bránu.



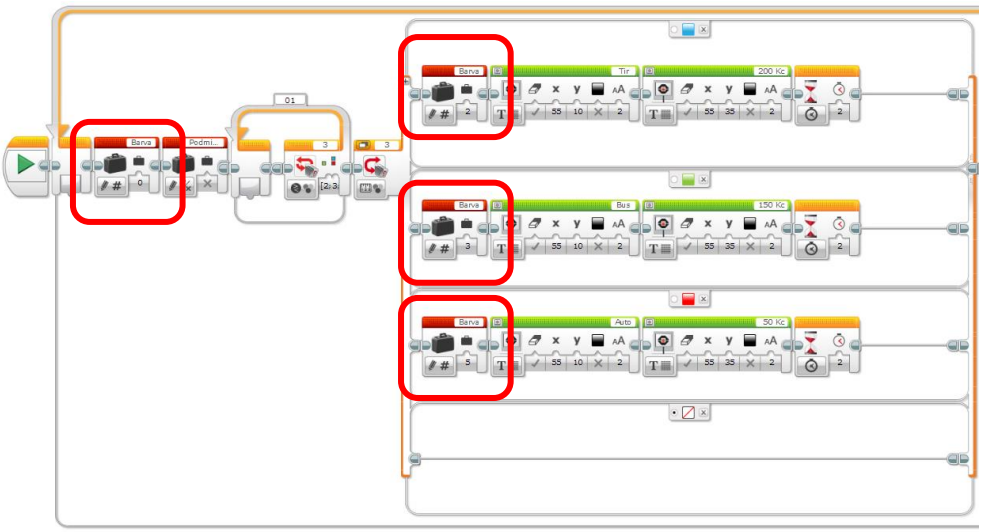
Obrázek 60 Úkol 7.9.1 aplikace

Řešení úkolu 7.9.2 mělo za úkol vypsat na displej kostky, po přiložení barevné karty k senzoru, o jaký typ dopravního prostředku se jednalo a kolik by zaplatil. Toto jsme vyřešili přidáním bloku Switch s bloky pro výpis za blok cyklu kontroly barvy.

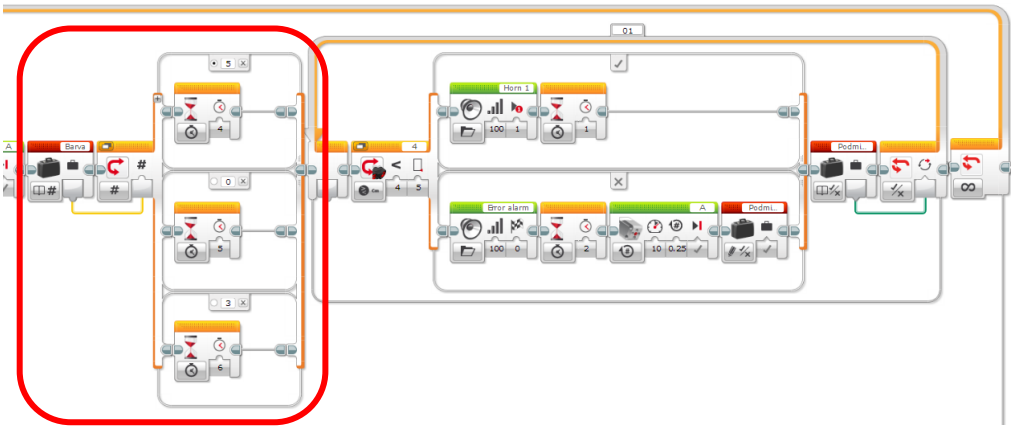


Obrázek 61 Úkol 7.9.2 aplikace

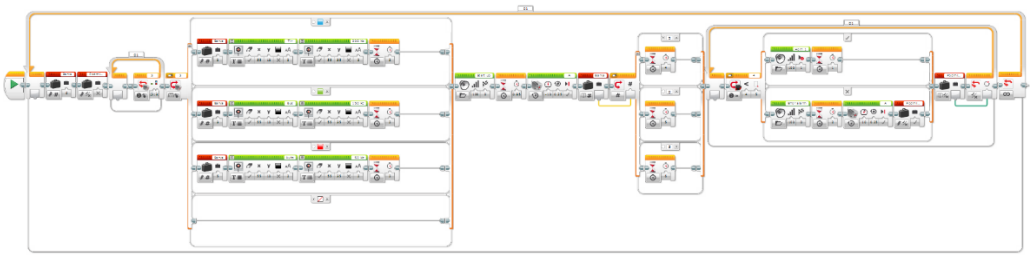
Úkol 7.9.3 měl za cíl prodloužit dobu, po kterou brána byla otevřena pro určený typ dopravního prostředku. Úkol jsme řešili přidáním proměnné Barva numerického typu, kterou jsme začátkem vnějšího cyklu nastavili na hodnotu 0 a v bloku Switch měnili její hodnotu podle přiložené karty. Díky tomuto stačilo přidat blok Switch, který rozhodl podle čísla barvy, po jakou dobu byla brána otevřena.



Obrázek 62 Úkol 7.9.3 část s proměnnými. aplikace



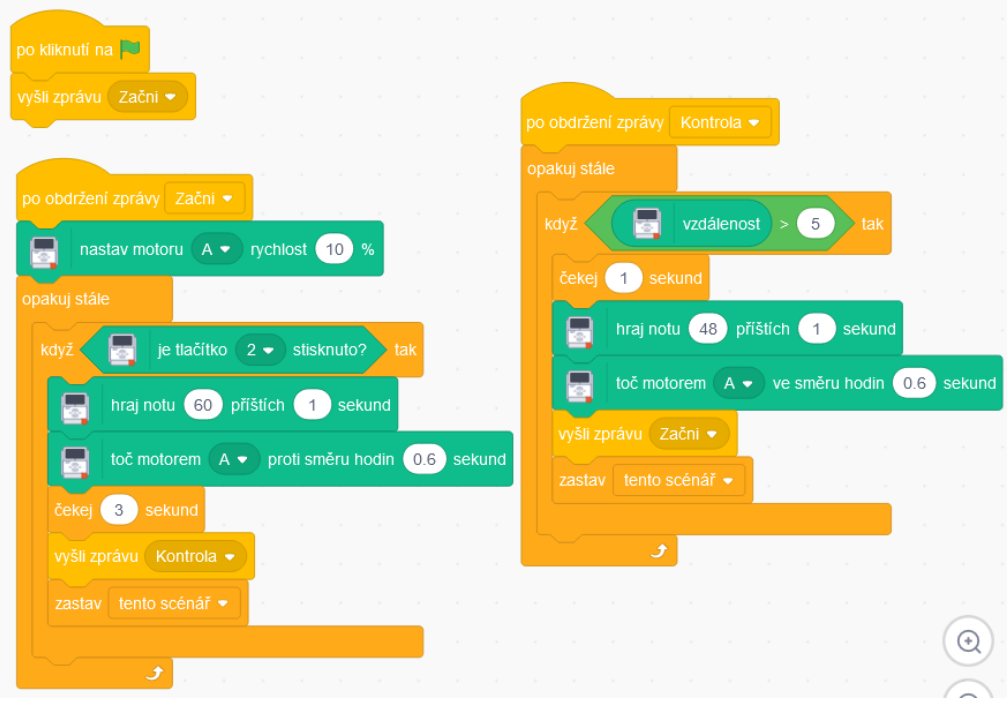
Obrázek 63 Úkol 7.9.3 část s blokem Switch aplikace



Obrázek 64 Úkol 7.9.3 celý kód aplikace

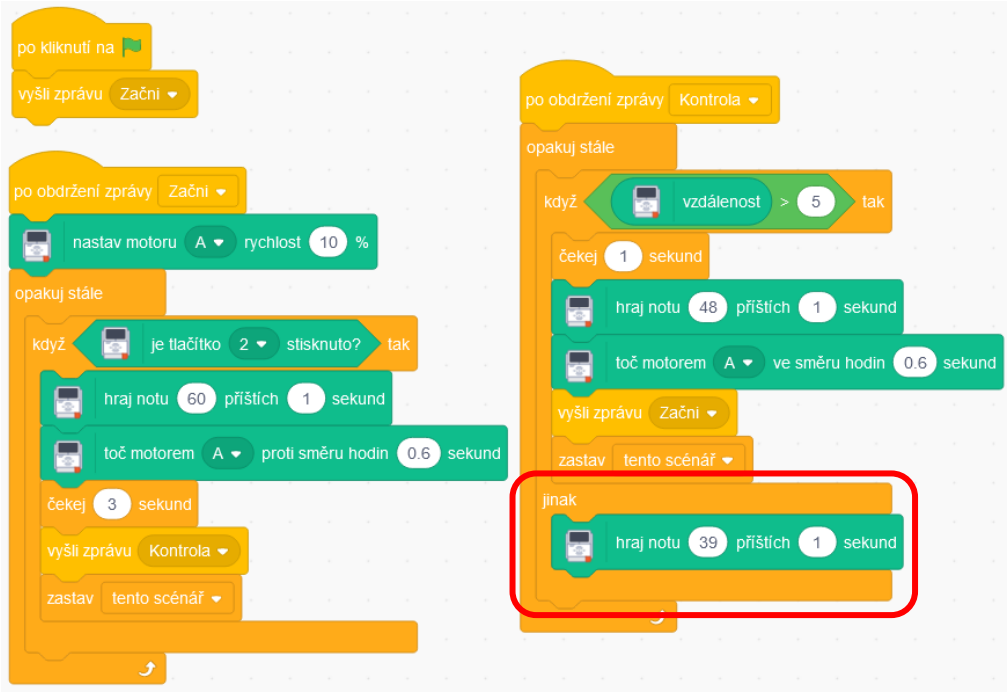
7.7.3. Prostředí rozšíření Scratch

Úkol 7.4.1, který vylepšoval kód z minulé kapitoly, jsme vyřešili přidáním několika bloků pro vyslání a obdržení zprávy. Docílili jsme tím toho, že se úseky kódu mohly vzájemně volat. Pravá část kódu kontrolovala vzdálenost od překážky nekonečným cyklem. Pokud část kódu pod blokem Kontrola nenašla překážku pod závorou, vyslala zprávu k bloku Začni a program byl připraven na nový stisk dotykového senzoru. Důležitým komponentem byly také bloky pro zastavení tohoto scénáře, kdybychom tyto bloky do řešení nezařadili, mohlo se stát, že by se program spustil znovu při nechtěném stisku dotykového senzoru a narušil by tím tak celý jeho chod.



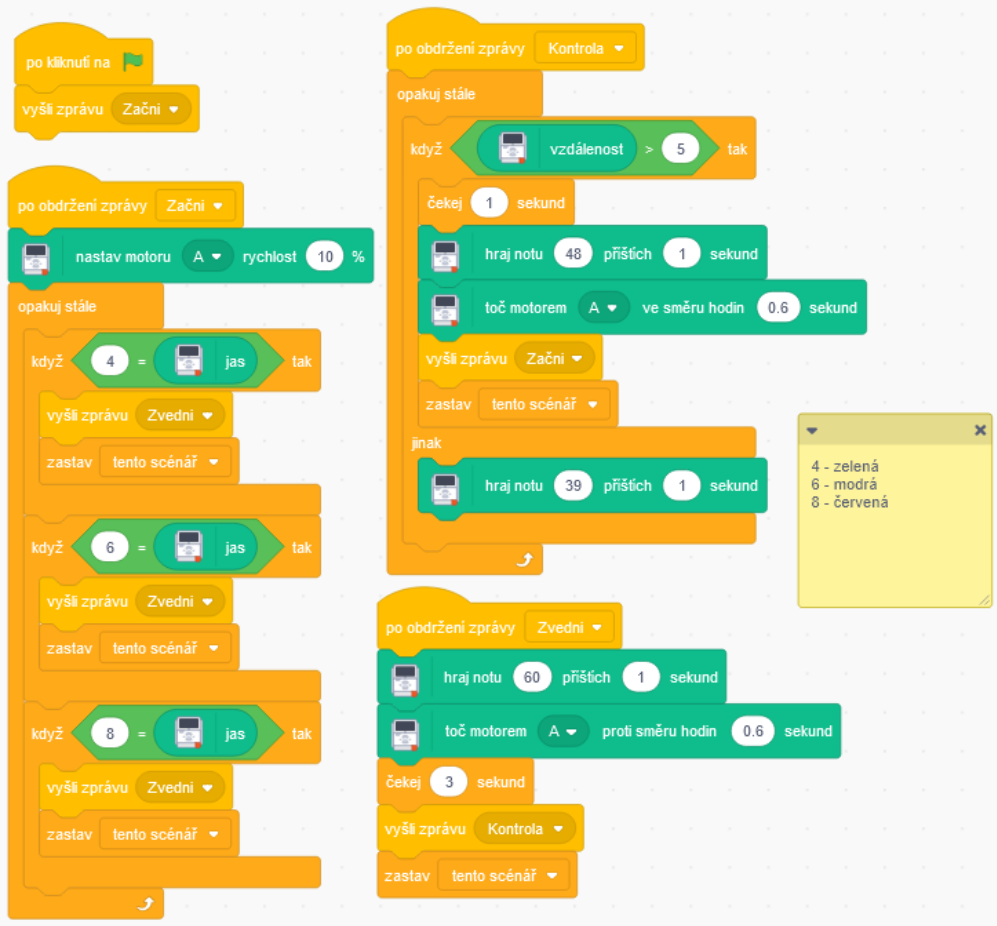
Obrázek 65 Úkol 7.4.1 Scratch

Úkol 7.5 rozšířil předešlý úkol o zvukovou signalizaci pro překážku pod bránou, jež se zavírala. Stačilo přidat blok hrající tón a vyměnit blok Když za blok Když - Jinak.



Obrázek 66 Úkol 7.5 Scratch

Úkol 7.9.1 byl složitější na provedení, protože rozšířené prostředí Scratche nám neumožnilo práci s barvami, ale pouze s jasem. Po několika desítkách testů, kde jsme měnili vzdálenost karty od senzoru a světelné podmínky, jsme zjistili, že ideální vzdáleností pro snímání karty bylo ~ 5 milimetrů od senzoru a jemný stín. Samotným zkoušením ideálních podmínek jsme strávili více času, než vyžadovalo samotné programování. Využili jsme dalšího bloku se zprávou, aby se nám bloky kódu pro zdvih závory zbytečně neopakovaly. Implementaci světelného podsvícení kostky jsme přeskočili, protože v rozšířeném prostředí ho nebylo možné naprogramovat.

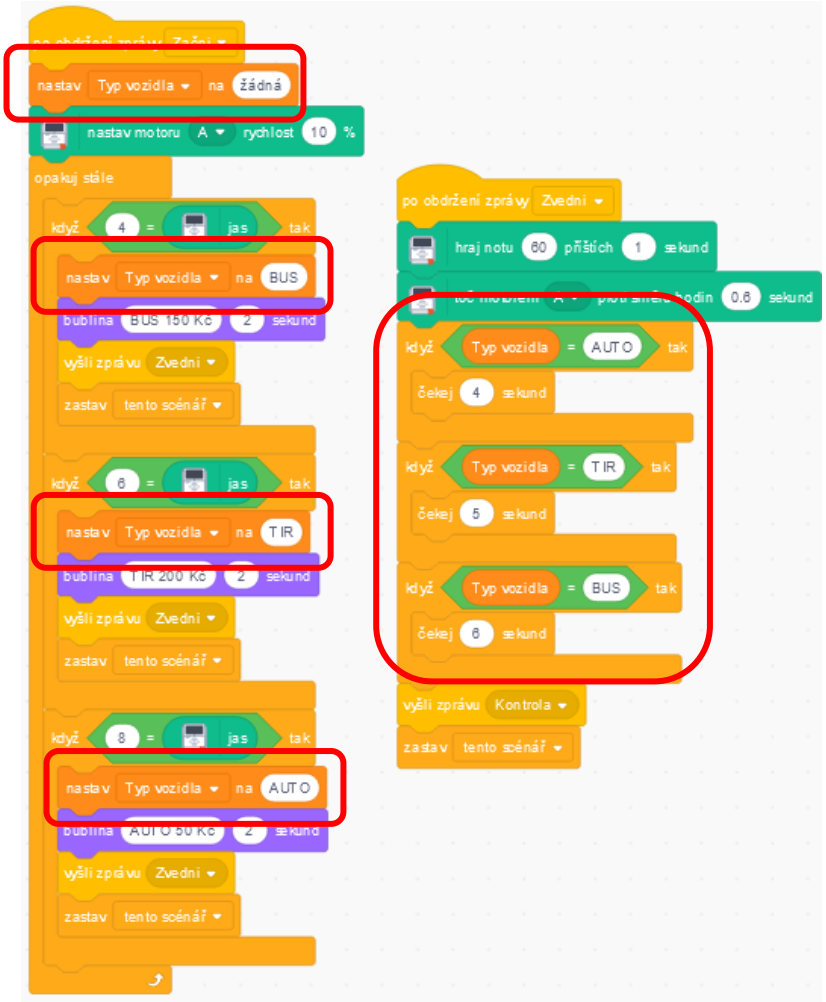


Obrázek 67 Úkol 7.9.1 Scratch

Aby bylo možné úkol 7.9.2 splnit, nahradili jsme vypisování na displej kostky výpisem do okna Scratche pomocí bloku bublina s příslušným textem. Zbylou část bloků kódu jsme neměnili.

Obrázek 68 Část úkolu 7.9.2 Scratch

Úkol 7.9.3 jsme řešili přidáním proměnné jménem Typ vozidla, kterou jsme použili v úseku kódu se zprávou Zvedni a 3 bloků Když pro rozlišení časového úseku zvednuté závory. Zbytek kódu z úkolu 7.9.2 zůstal stejný.



Obrázek 69 Část úkolu 7.9.3 Scratch

7.7.4. Problémy při řešení, poznatky a úpravy

Doposud největším problémem při programování v rozšířeném prostředí Scratche byla nemožnost rozpoznání barvy pomocí senzoru. Tuto překážku jsme odstranili použitím hodnoty jasu, který symbolizoval hodnotu odráženého světla z karty před senzorem.

Dalším problém byly podmínky, za jakých senzor hodlal spolupracovat. Správně pracoval pouze na vyvýšené ploše vzdálené 0,8 metru od země a ~ 1,5 metru od okna v jemném dopoledním stínu, který jsme vytvářeli kartonovou deskou v okně.

7.7.5. Porovnání

Pro práci v této kapitole jednoznačně doporučujeme aplikaci. Aplikace má citlivější vnímání senzorů a nevyžaduje extrémně specifické podmínky pro jejich správné fungování.

Rozšíření Scratche má výhodu ve volání částí kódu. Kód se tak pro nás zdál díky těmto možnostem přehlednější než programy implementované v aplikaci. Tento plusový bod ovšem nedokáže převážit mínusy rozšíření jako celku.

7.8. Kapitola 8. Adaptivní tempomat – detekce překážky

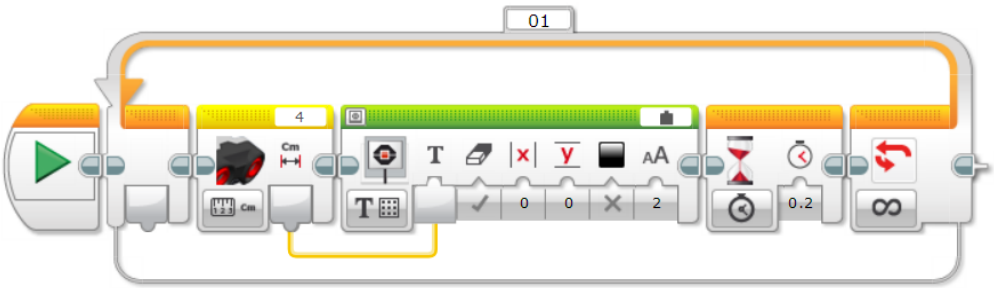
7.8.1. Cíl

Cílem kapitoly bylo ukázat, že prostředí, ve kterém se robot pohybuje nemusí být jen statické, ale může se v čase měnit. V této kapitole byl plně využit senzor zjišťující vzdálenost od předmětů v kombinaci s motory robota [14].

Obrázek 70 Robot se senzorem vzdálenosti

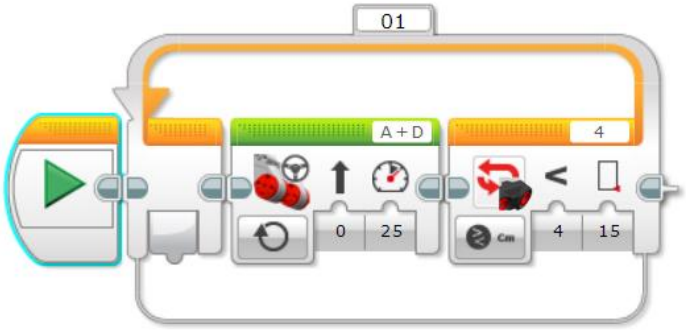
7.8.2. Prostředí originální aplikace

Cílem úkolu 8.1.1 bylo naprogramovaní metru, jež senzor používal k určení vzdálenosti. Program opakovaně měřil vzdálenost a zobrazoval ji na displeji kostky.



Obrázek 71 Úkol 8.1.1 aplikace

Úkol 8.2.1 byl úvodním úkolem do nastávající problematiky této kapitoly. Robot se pohyboval směrem vpřed do doby, než byl vzdálený 15 centimetrů od překážky. K vyřešení nám stačilo zkombinovat pohyb robota s informacemi o vzdálenosti od překážky ze senzoru.

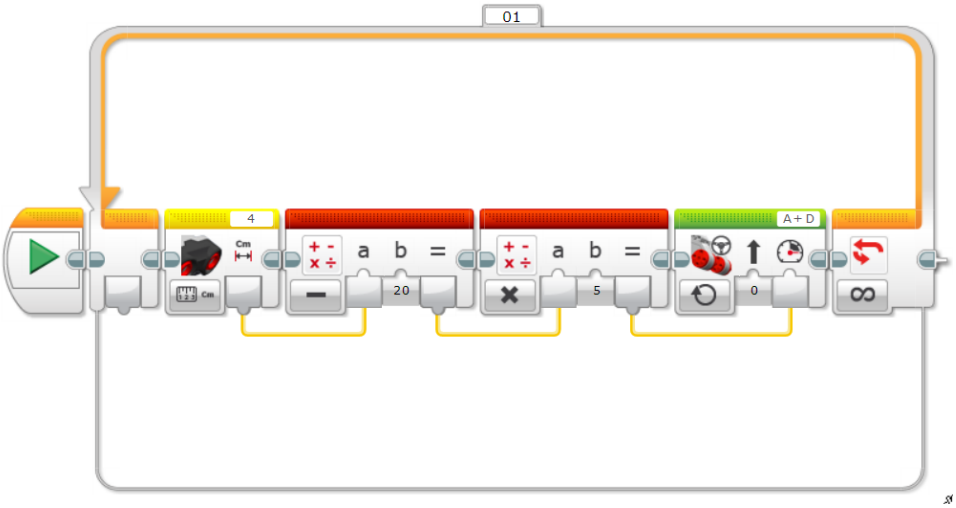


Obrázek 72 Úkol 8.2.1 aplikace

Řešením úkolu 8.3.1 měl být program upravující rychlost a směr pohybu robota podle překážky před ním. Robot se zastavil 20 centimetrů před překážkou. Pokud se vzdálila, dojel ji. Byla-li blíže než 20 centimetrů, zacouval. Pro určení směru a rychlosti jsme použili jednoduchý výpočet.

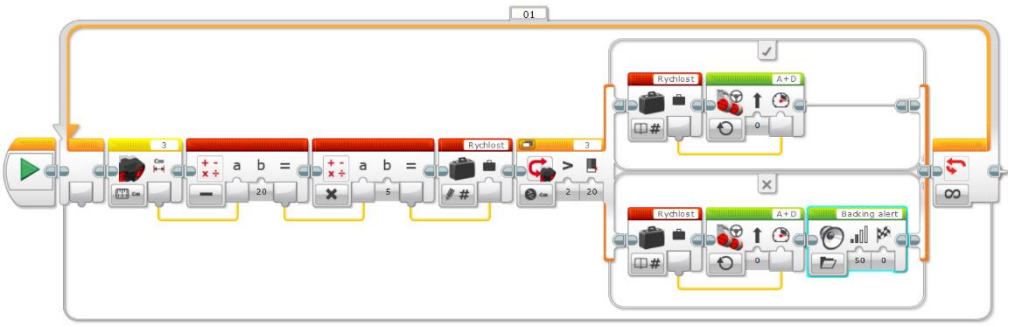
$$rychlost\ robota = (\ vzdálenost\ překážky - 20)\ \times 5$$

Rovnice 1 Výpočet rychlosti robota



Obrázek 73 Úkol 8.3.1 aplikace

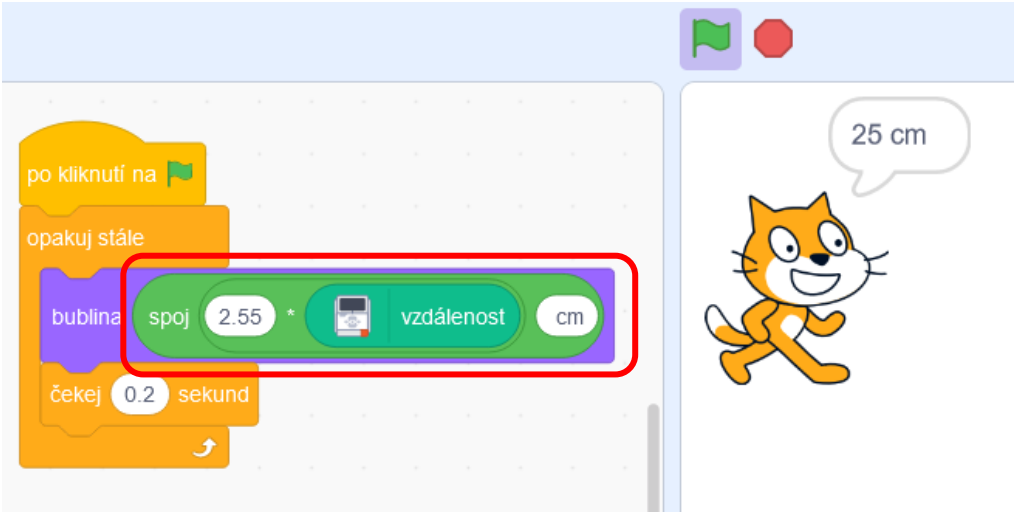
Úkol 8.5 rozšiřoval stávající program o varovný zvuk při couvání. Naším řešením bylo přidání numerické proměnné Rychlost a bloku Switch, který rozlišoval vzdálenosti od překážky. Pokud tato vzdálenost byla menší než 20 centimetrů, tak robot couval a vydával zvuk.



Obrázek 74 Úkol 8.5 aplikace

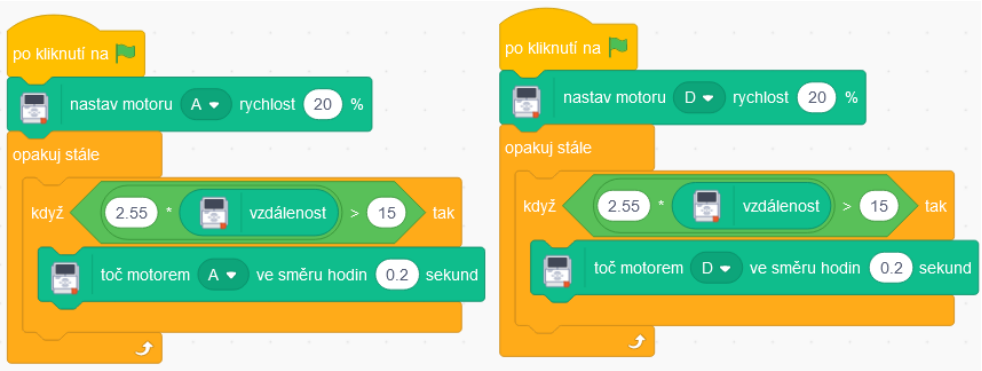
7.8.3. Prostředí rozšíření Scratch

Úkol 8.1.1 byl odlišný rozsahem senzoru, místo udávání vzdálenosti od předmětu v příslušných jednotkách délky fungoval na stupnici od 0 do 100 procent. Místo výpisu na displej kostky jsme zvolili výpis vzdálenosti do okna Scratche. Senzorem vracená procenta jsme převáděli na centimetry pomocí mezivýpočtu (označen červeně).



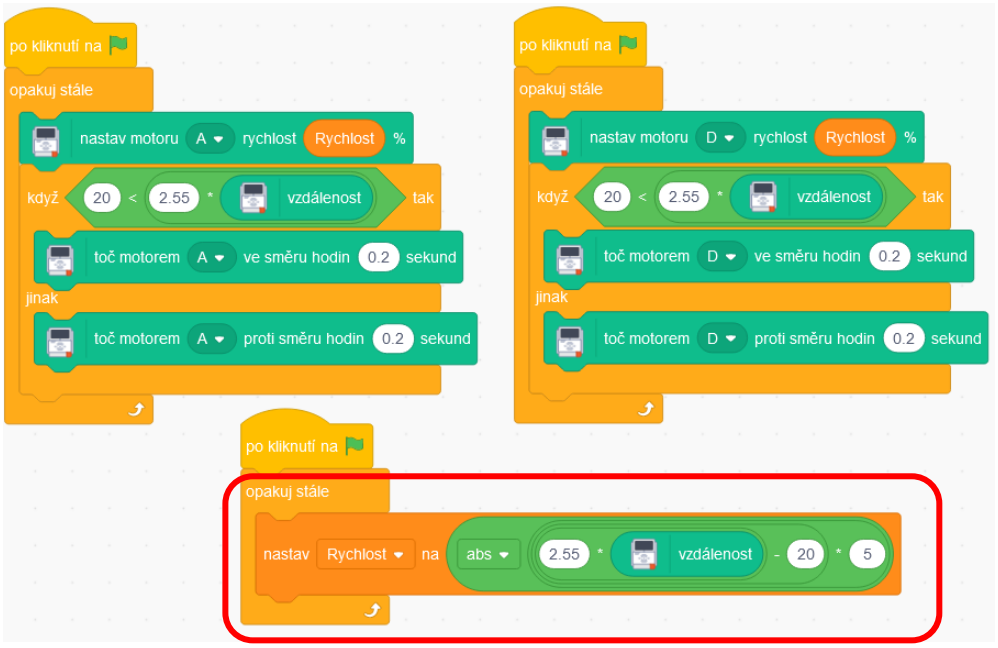
Obrázek 75 Úkol 8.1.1 Scratch

Úkol 8.2.1 jsme nebyli schopni naprogramovat tak, aby robot vykonával plynulý pohyb, jako tomu bylo u řešení z aplikace. Příčinou trhavého pohybu robota byla nemožnost naprogramovat chod obou motorů jinak než po dobu sekund. Opět jsme použili paralelní vlákna kódu.



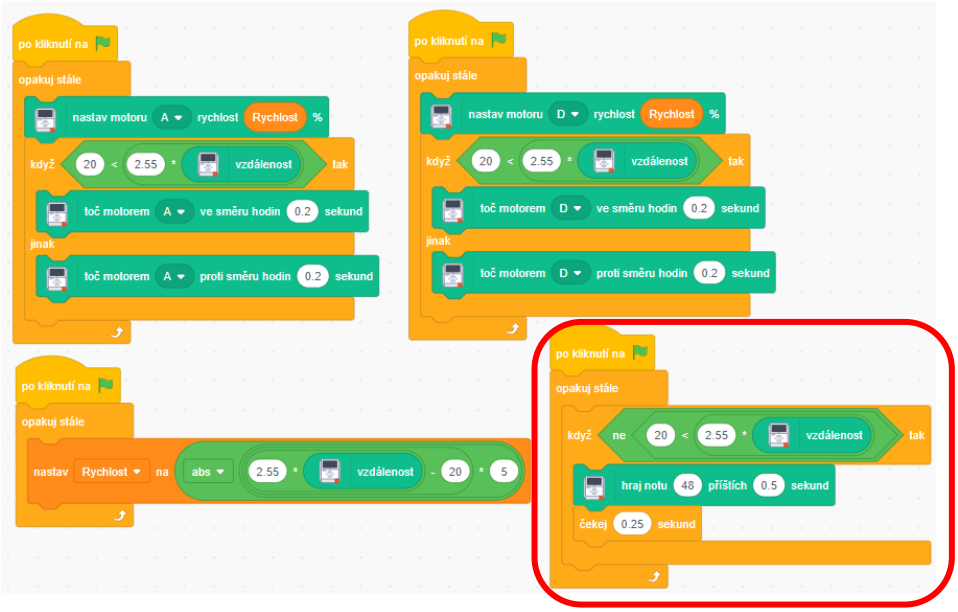
Obrázek 76 Úkol 8.2.1 Scratch

V úkolu 8.3.1 jsme řešili rychlost motorů pomocí výpočtu vzdálenosti mezi překážkou a robotem, jak tomu bylo u řešení z aplikace. Jediným rozdílem byla nutnost vkládat výsledek do absolutní hodnoty (označeno červeně), protože motory v rozšířeném prostředí Scratche neuměly pracovat se zápornou hodnotou rychlosti. Pro pohyb dopředu a couvání jsme museli využít podmínkových bloků Když – Jinak.



Obrázek 77 Úkol 8.3.1 Scratch

Řešení úkolu 8.5 se od minulého lišilo pouze přidaným úsekem, který při couvání robota vydával tón.



Obrázek 78 Úkol 8.5 Scratch

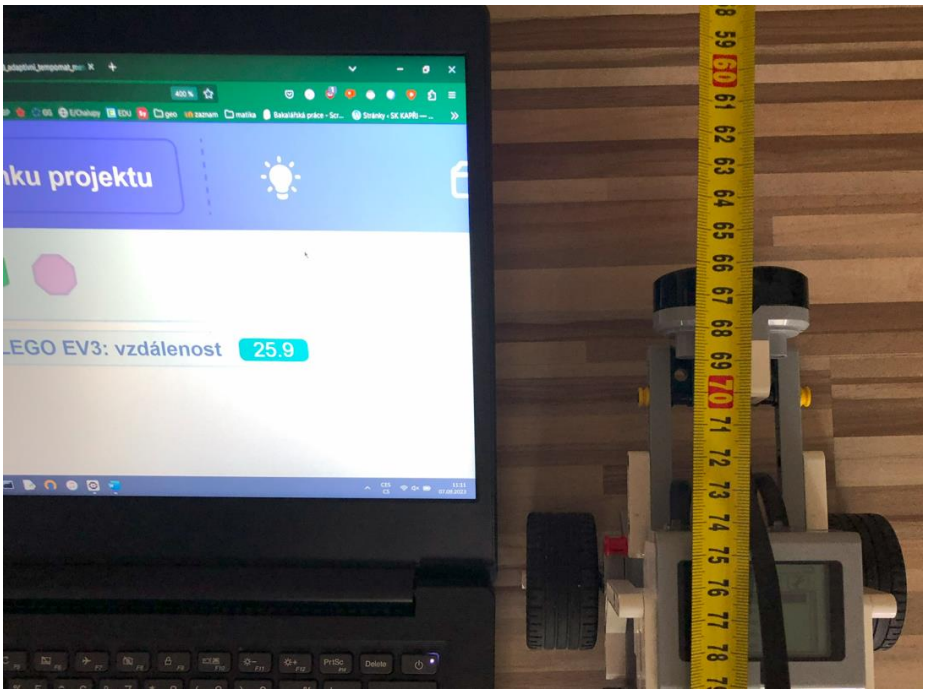
7.8.4. Problémy při řešení, poznatky a úpravy

Do kolonky úpravy bychom zahrnuli použití okna ve Scratchi na místo displeje kostky.

Jako problém bychom označili měření vzdálenosti. Aplikace měla rozsah od 0 do 255 centimetrů na rozdíl od rozšíření ve Scratchi, kde rozmezí bylo mezi 0 a 100 procenty. Pro zjištění reálné vzdálenosti jsme museli při implementaci kódu ve Scratchi naměřenou hodnotu v procentech převádět násobením na centimetry.

$$Reálná\ vzdálenost = \frac{255\ cm}{100} \times (naměrená\ vzdálenost\ v\ procentech)$$

Rovnice 2 Výpočet reálné vzdálenosti robota od objektu



Obrázek 79 Měření v procentech vs reálná délka

7.8.5. Porovnání

Pro snazší průchod kapitolou bychom doporučili aplikaci. Hodnoty ze senzoru v aplikaci nejsou uváděny v procentech, je možný výpis na displej, motory mohou fungovat bez časového omezení a jejich pohyb není trhaný.

7.9. Kapitola 9. Inteligentní pojízdný robot

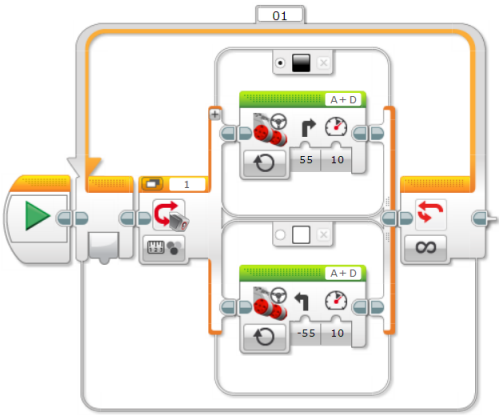
7.9.1. Cíl

Cílem kapitoly bylo ukázat programujícím využití barevného senzoru za účelem pohybu robota po čáře, tak jako tomu bývá v továrních halách nebo skladech.[14]

Obrázek 80 Upravený robot s barevným senzorem

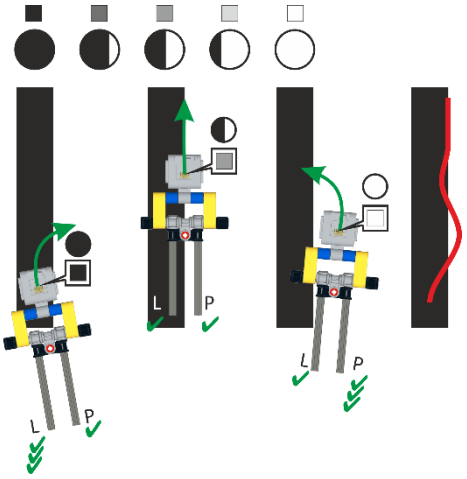
7.9.2. Prostředí rozšíření Scratch

V úkolu 9.4 bylo našim cílem doplnit rozpracovaný kód tak, aby se robot pohyboval po čáře kmitavým pohybem a neopustil ji. Tento úkol jsme vyřešili doplněním bloků pohybu.

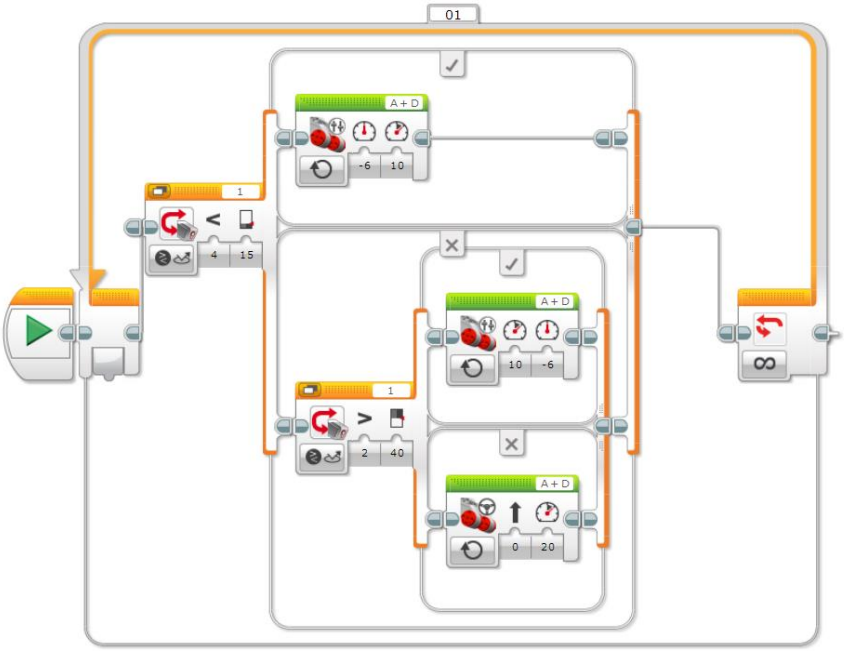


Obrázek 81 Úkol 9.4 aplikace

Úkol 9.6 měl zlepšit pohyb po čáře. Minulé řešení se dalo vylepšit přidáním nových mezí s rychlostmi, které se lépe přizpůsobovaly křivkám čáry. Pro ilustraci jsme přidali 3. mez. Nově přidaná 3. mez umožňovala robotovi pohybujícímu se na okraji čáry použít vetší rychlost pohybu a rotovat oběma koly stejně rychle. Pro maximalizaci potenciálu bychom přidali do programu další meze, které jsou vidět v horní části Obrázku 83.



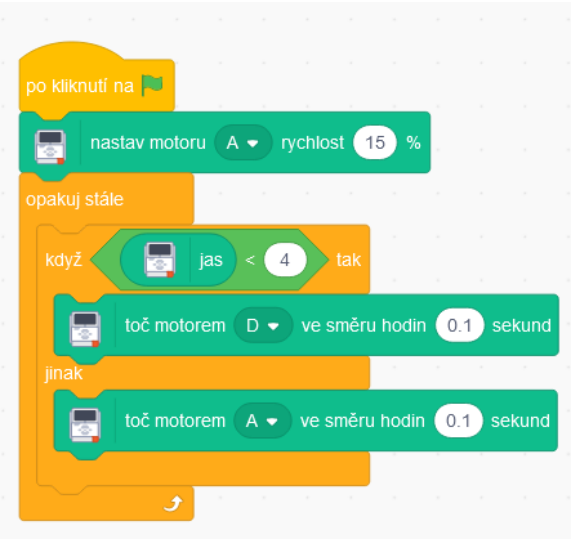
Obrázek 83 Zobrazení mezí a pohyb robota
Zdroj: https://lego.zcu.cz/ucebnice/cara.html



Obrázek 82 Úkol 9.6 aplikace

7.9.3. Prostředí rozšíření Scratch

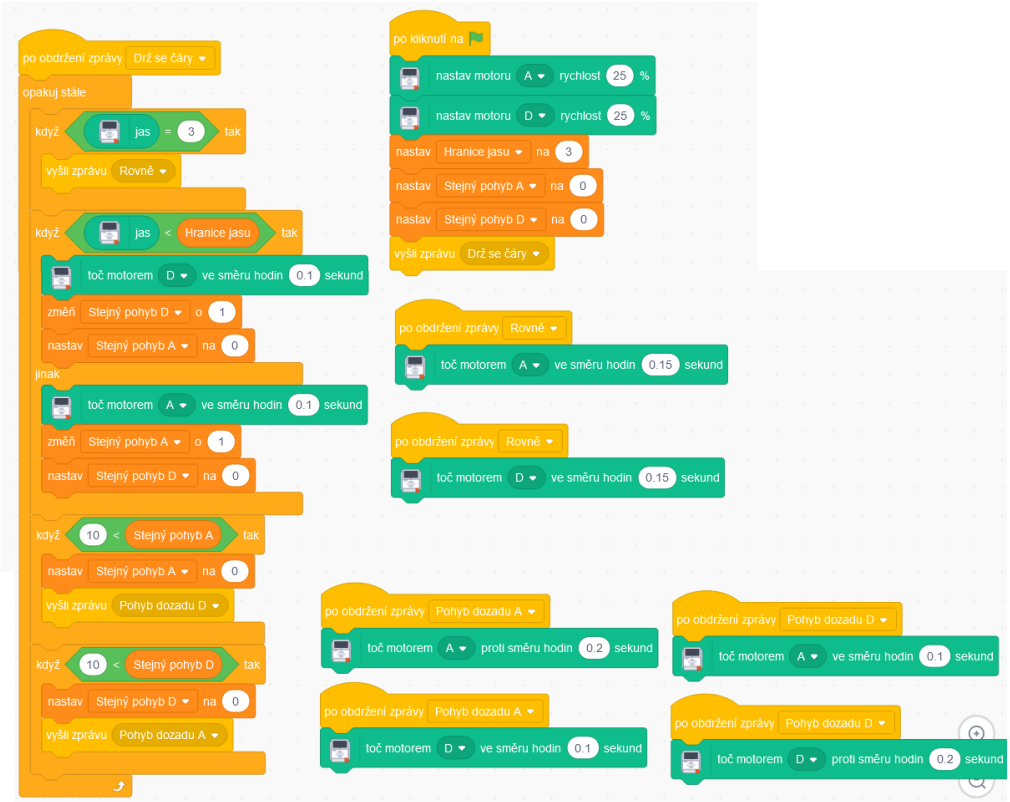
Úkol 9.4 byl kvůli (ne)citlivosti senzoru v rozšíření Scratche tím nejtěžším na splnění. Senzor fungoval jen za určitých světelných podmínek, jak už jsme popisovali v kapitole 7.



Obrázek 84 Úkol 9.4 Scratch

Úkol 9.6 rozšiřoval a vylepšoval řešení předešlého úkolu přidáním dalších mezí. Kvůli žalostně malé citlivosti senzoru jsme se nepokoušeli programovat více mezí než jen jednu pro pohyb na okraji čáry.

Nyní vysvětlení našeho myšlenkového postupu při řešení úkolu. Jelikož jsme byli omezeni pouze na 2 typy pohybových bloků, byli jsme nuceni improvizovat pohybem po dobu desetin sekund. Tento pohyb byl trhaný a nepřesný. Pro jeho korekci jsme zapojili do kódu několik nových proměnných, které při 10násobném opakování stejného pohybu zapříčinily znatelnější a méně trhané otočení robotem stejným směrem.



Obrázek 85 Úkol 9.6 Scratch

7.9.4. Problémy pří řešení, postřehy a úpravy

Světelný senzor měl mít rozmezí od 0 do 100 jednotek jasu. Námi maximální získaná hodnota v rozšířeném prostředí Scratche dosahovala pouze do výše 56 jednotek po zasvícení svítilnou přímo na senzor. Tento pokus jsme pro jistotu opakovali i v aplikaci. Již na 25 % síly svítilny senzor hlásil 100 jednotek jasu.

Nepříjemným problémem byl trhavý pohyb, který každé řešení v rozšířeném prostředí Scratche doprovázel a značně ho ztěžoval.

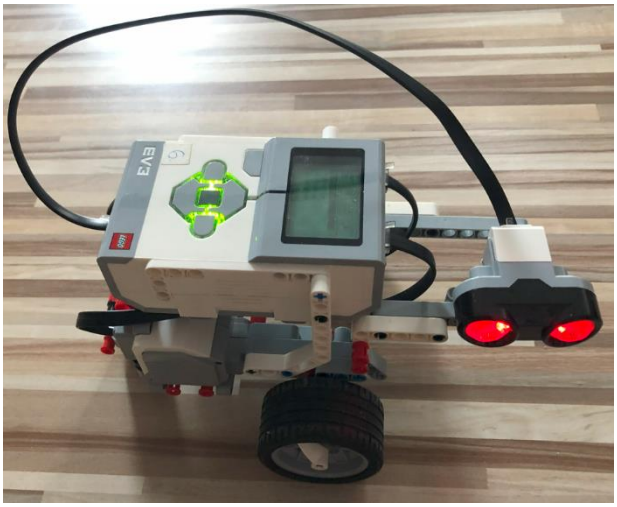
7.9.5. Porovnání

Kapitolu je možné naprogramovat v obou prostředích. To ovšem neznamená, že jsou si prostředí rovna. Aplikace má znatelně lepší citlivost senzoru a poskytuje možnost jemnějšího a přesnějšího pohybu robotem.

7.10. Kapitola 10. Parkovací asistent

7.10.1. Cíl

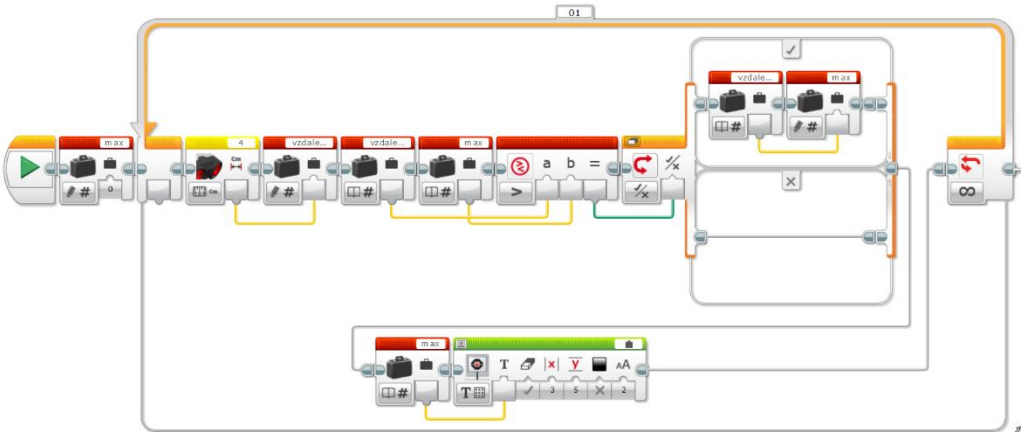
Cílem kapitoly bylo procvičení si již známých proměnných, senzorů a bloků, které ovládají pohyb robota. Programující si opět museli umět úkoly rozdělit na podúkoly, které budou popořadě plnit.[14]



Obrázek 86 Robot se senzorem vzdálenosti

7.10.2. Prostředí originální aplikace

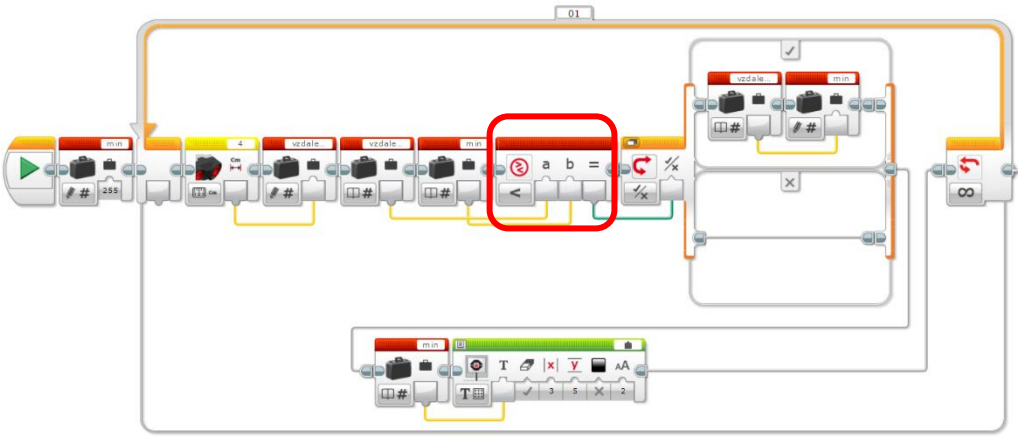
Úkol 10.1.4 byl vyřešen od autorů učebnice. Rozhodli jsme se ho sem zařadit, protože budeme kód předělávat do rozšíření Scratche.



Obrázek 87 Úkol 10.1.4 aplikace

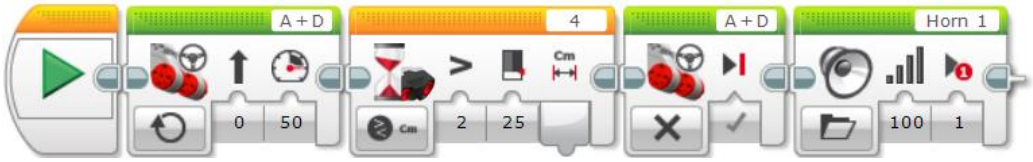
Zdroj: https://lego.zcu.cz/ucebnice/assets/files/F_10_parkovaci_asistent_metodika_D1.pdf

V úkolu 10.1.5 jsme změnili název proměnné Max na Min. Začátkem programu jsme proměnnou nastavili na hodnotu maximální vzdálenosti a otočili znaménko nerovnosti.



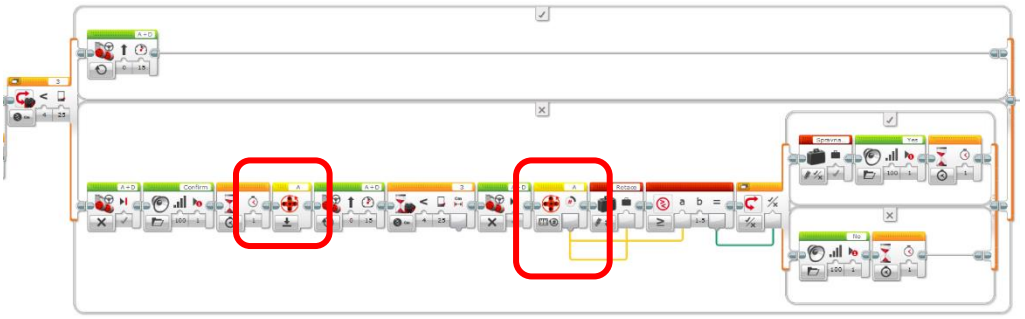
Obrázek 88 Úkol 10.1.5 aplikace

Úkol 10.3.1. Robot se měl pohybovat rovně do doby, než detekoval volné místo, následně se měl zastavit a vydat zvuk.

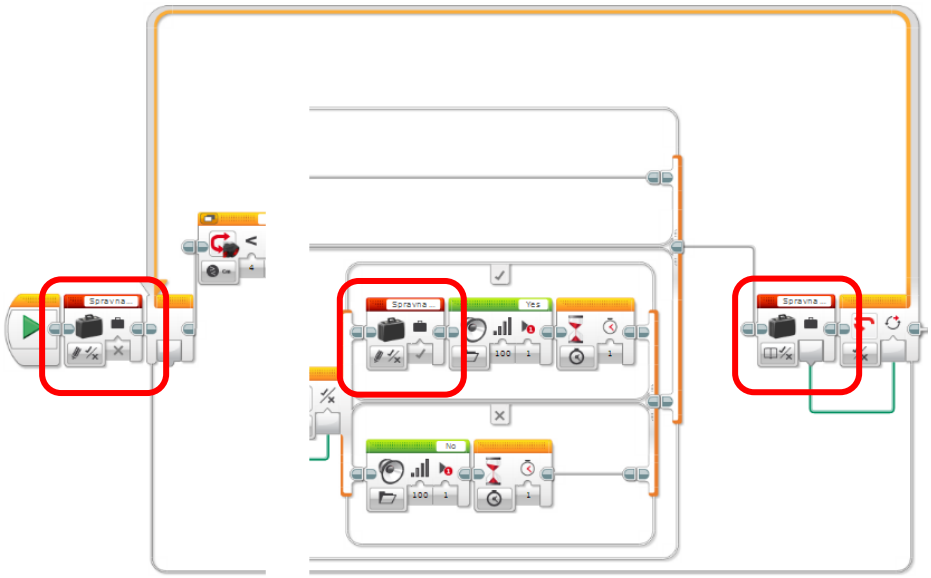


Obrázek 89 Úkol 10.3.1 aplikace

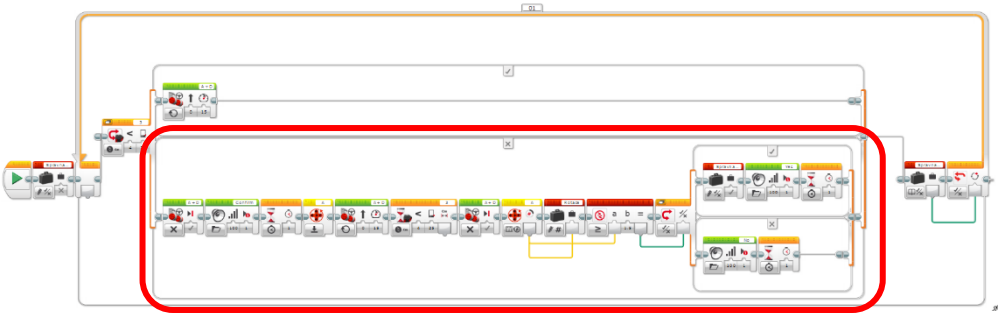
Úkol 10.5.1 rozšiřoval program z úkolu 10.3.1 tím, že po zaznění signálu volného místa měl robot sám změřit, zda se do volného prostoru vejde. Změřili jsme délku robota, zkusili s ním ujet tuto vzdálenost v hodnotě otoček kol a dané poznatky zakomponovali do programu. Pokud robot narazil na parkovací místo, do kterého se vejde, přestal s měřením a zastavil se. Kdyby při měření usoudil, že se na místo nevejde, pokračoval by v hledaní dalšího volného místa.



Obrázek 90 Část úkolu 10.5.1 uvnitř vnějším bloku Switch



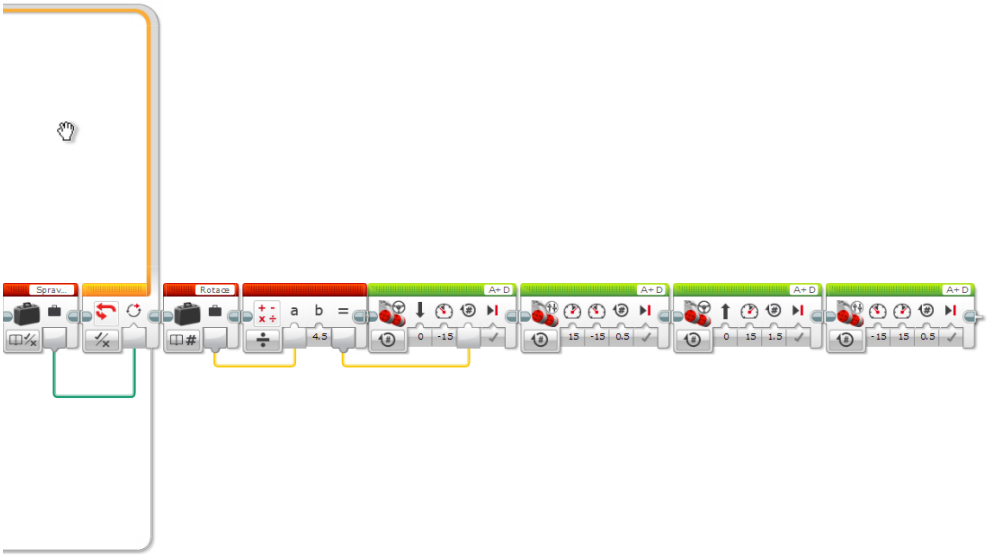
Obrázek 91 Detail podmínky, která ukončuje chod programu úkolu 10.5.1



Obrázek 92 Celý kód úkolu 10.5.1 aplikace

Úkol 10.6.1 nemá svou vlastní fotodokumentaci, protože při řešení úkolu 10.5.1 jsme na tento možný problém brali ohled.

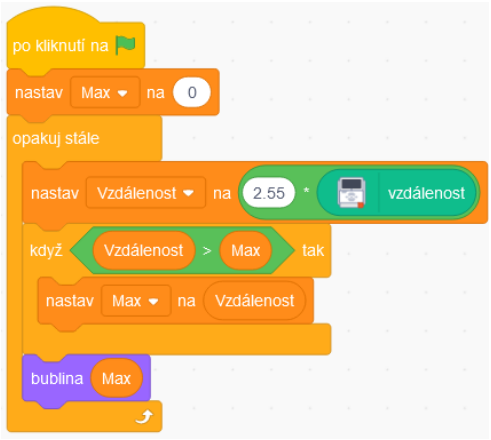
Úkol 10.7.1 jsme splnili přidáním 6 bloků za stávající kód. Neměli jsme potřebu dělat komplexnější kód parkování, protože samotný proces je totožný pro každé místo, kam by robot parkoval.



Obrázek 93 Úkol 10.7.1

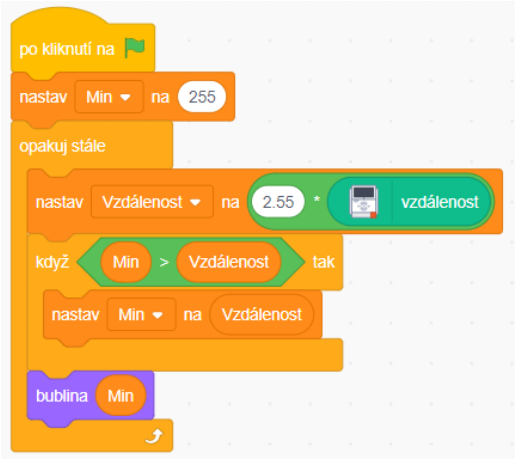
7.10.3. Prostředí rozšíření Scratch

Úkol 10.1.4. bylo možné předělat i do prostředí rozšíření Scratche. Nutnost násobení vzdálenosti číslem 2,55 jsme popsali v kapitole 9.



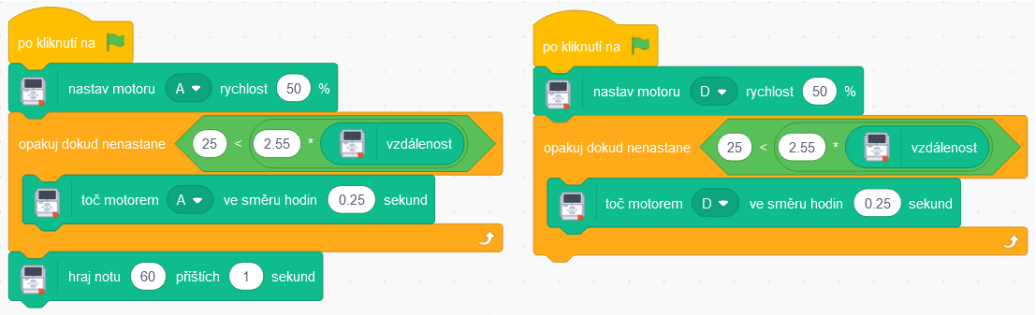
Obrázek 94 Úkol 10.1.4 Scratch

V úkolu 10.1.5 byl námi uplatněn stejný postup pro implementaci jako v aplikaci. Přejmenovali jsme proměnnou Max na Min a obrátili znaménko nerovnosti.



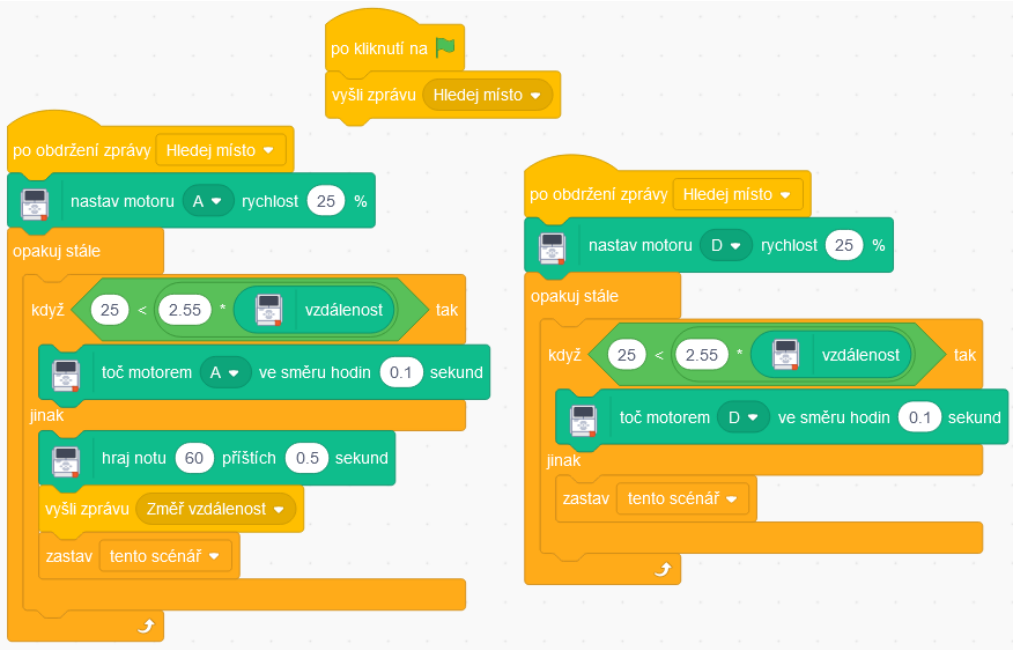
Obrázek 95 Úkol 10.1.5 Scratch

K vyřešení úkolu 10.3.1 jsme použili dvou vláken pro pohyb kol. Použili jsme dobu pohybu 0,25 sekund, protože cílem úkolu bylo pouze konec překážky detekovat a zastavit se.

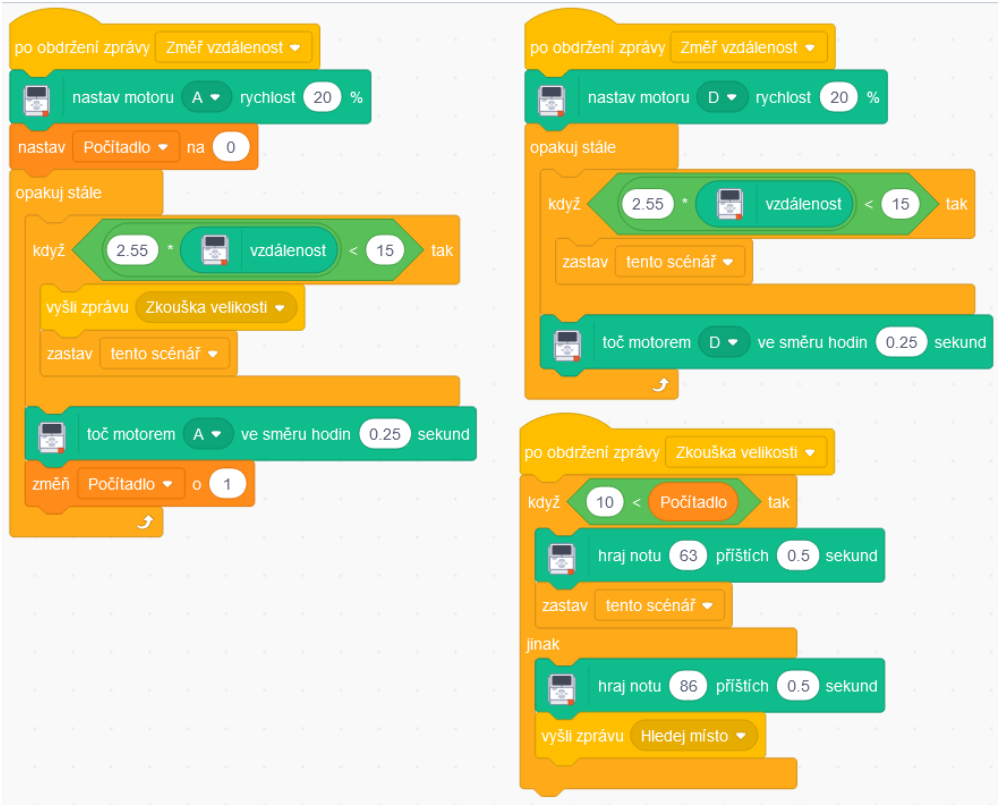


Obrázek 96 Úkol 10.3.1 Scratch

Úkol 10.5.1 navazoval na úkol 10.3.1. Délku robota jsme již znali z řešení z aplikace, stačilo tedy zjisti kolikrát musí robot vykonat pohyb vpřed, aby se na parkovací místo vešel. Velikost robota se rovnala 10 pohybům vpřed trvajícím 0,25 sekundy. Počet vykonaných pohybů jsme ukládali do proměnné Počítadlo.



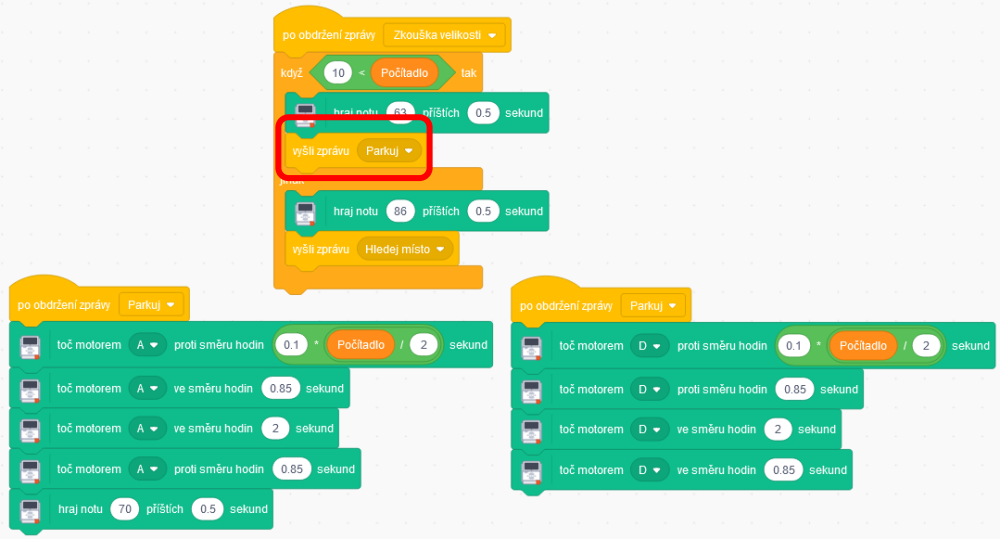
Obrázek 97 Část 1. úkolu 10.5.1 Scratch



Obrázek 98 Část 2. úkolu 10.5.1 Scratch

Úkol 10.6.1 nemá fotodokumentaci, protože jsme na tento možný problém mysleli při implementaci úkolu 10.5.1.

V úkolu 10.7.1 jsme zachovali všechny bloky z úkolu 10.5.1 a přidali k nim kód, díky kterému robot zacouvá, otočí se, vjede na volné místo a srovná se s ostatními vozidly. Menší úpravu obdržel úsek s názvem Zkouška velikosti, kde jsme přidali blok vysílající zprávu Parkuj.



Obrázek 99 Úkol 10.7.1 Scratch

7.10.4. Problémy při řešení, postřehy a úpravy

Opět jsme postrádali blok pohybující oběma koly zároveň či jiné bloky pohybu něž po dobu sekund.

7.10.5. Porovnání

Kdyby rozšíření ve Scratchi umožňovalo jiný pohyb motoru než po dobu sekund, neváhali bychom a dali rozhodně přednost tomuto prostředí nad prostředím aplikace z důvodu možnosti volání částí kódu bez komplikovaného cyklení. I přes tento nedostatek nám řešení úkolů v rozšířeném prostředí přijde jako to elegantnější.

7.11. Kapitola 11. Hra „Kdo má lepší postřeh“

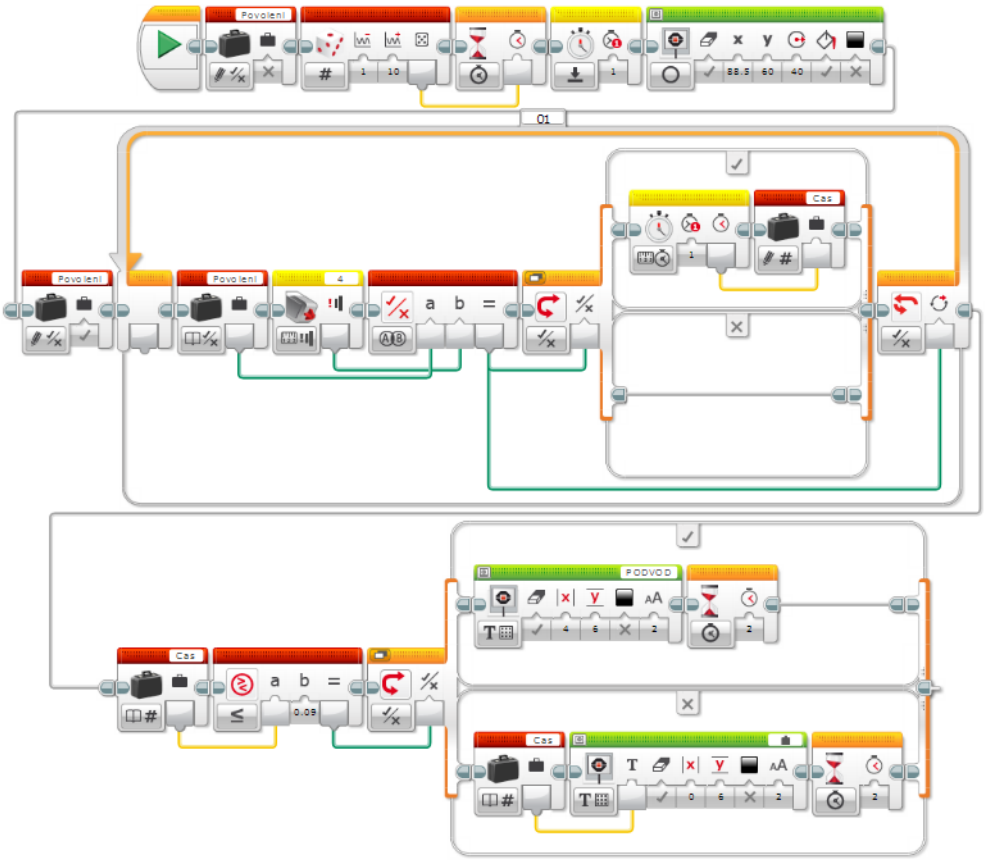
7.11.1. Cíl

Cílem kapitoly byla práce s proměnnými, dotykovým senzorem, displejem a stopkami. Programující měli maximálně využít získané znalosti, protože se v této kapitole kombinují podmínkové bloky, bloky cyklů, proměnné a jejich využití při porovnávání nebo uchovávání získaných dat.[14]

Obrázek 100 Robot s dotykovými senzory

7.11.2. Prostředí originální aplikace

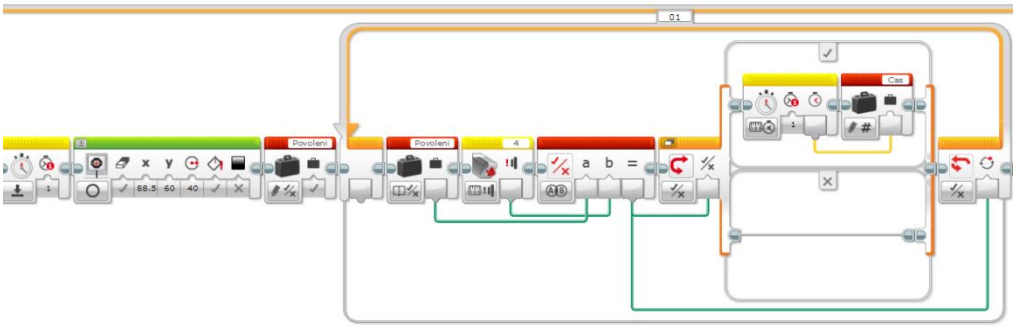
Cílem úkolu 11.2 bylo naprogramování základní postřehové hry pro jednoho hráče. Použili jsme několik proměnných. První proměnná Cas ukládala hodnotu z bloku Timer, kde byl uložen reakční čas hráče. Druhá proměnná Povoleni se v kombinaci s blokem Switch starala o umožnění stisku dotykového senzoru. Na konci kódu jsme umístili úsek kontrolující zakázané mačkání dotykového senzoru bezprostředně po spuštění hry.



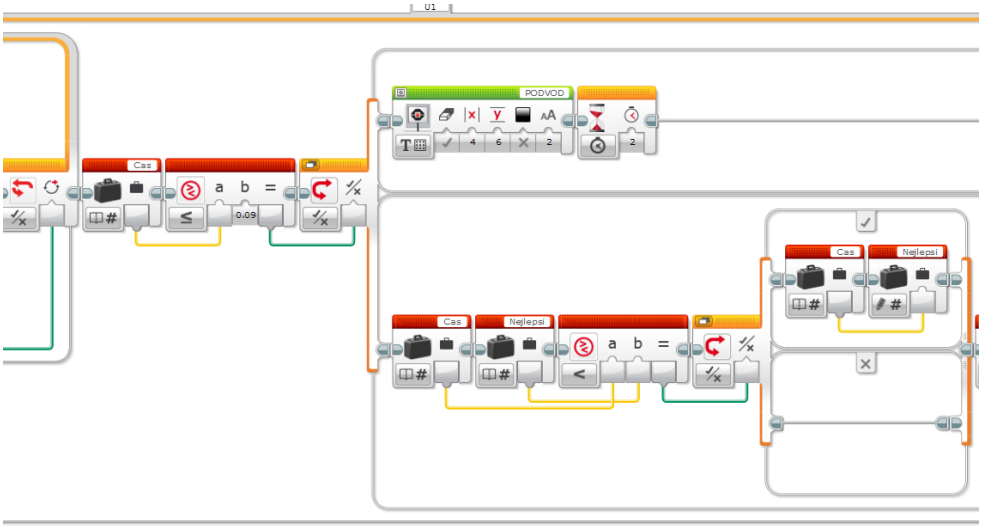
Obrázek 101 Úkol 11.2 aplikace

Úkol 11.3 byl zaměřen na práci s proměnnými. Hra měla po skončení ukázat výsledný čas hráče, čas nejlepšího hráče a ideálně se znovu spustit nebo se nechat po dohrání přerušit. Museli jsme rozšířit program o proměnnou Nejlepsi, která ukládala nejlepší výsledek ze všech hráčů, kteří hru po spuštění hráli. Další úpravou bylo přidání cyklu, který opakoval hru a ukončil se při stisku korespondujícího dotykového senzoru na konci kola.

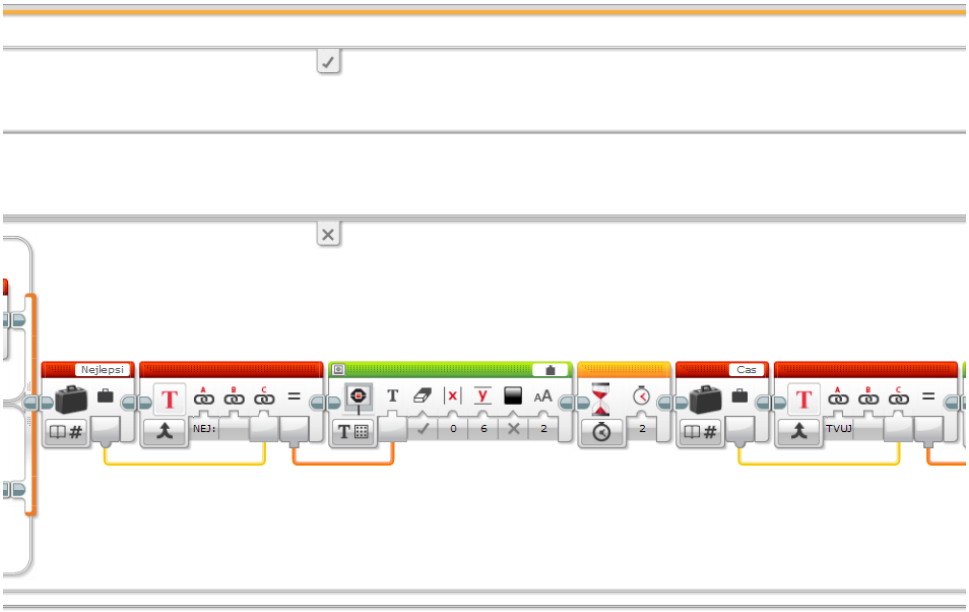
Obrázek 102 Úkol 11.3 část 1. aplikace



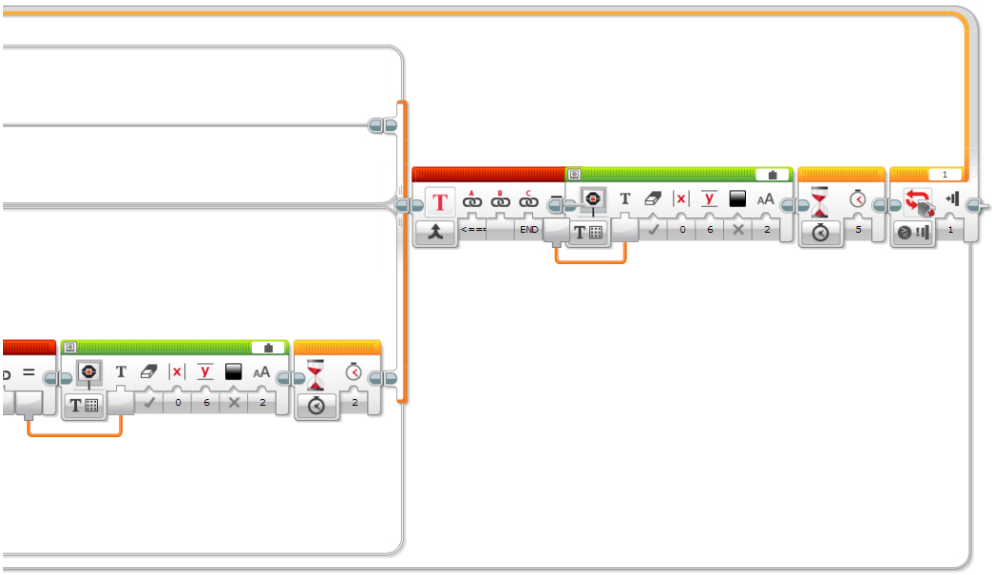
Obrázek 103 Úkol 11.3 část 2. aplikace



Obrázek 104 Úkol 11.3 část 3 aplikace.



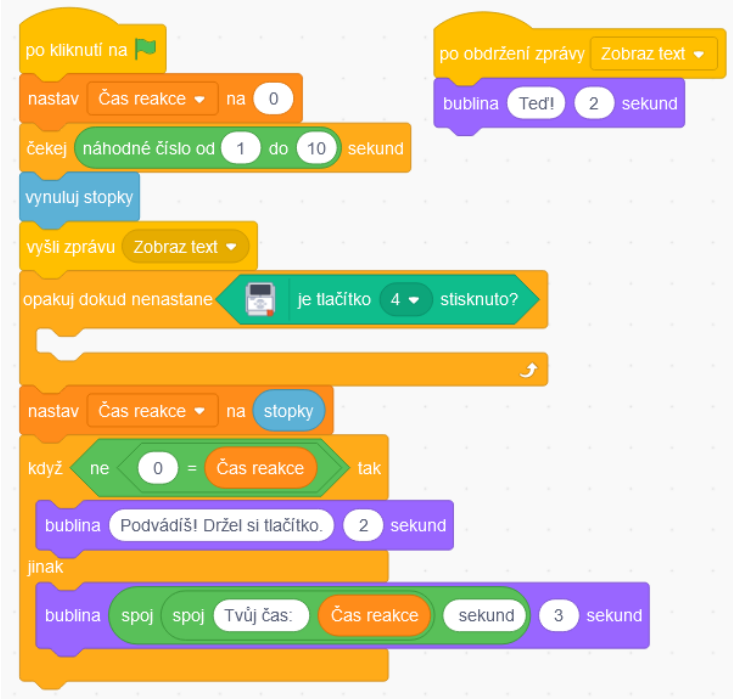
Obrázek 105 Úkol 11.3 část 4. aplikace



Obrázek 106 Úkol 11.3 část 5. aplikace

7.11.3. Prostředí rozšíření Scratch

Úkol 11.2 jsme mohli řešit za pomoci bloků, které posílají a přijímají zprávy. K vyřešení úkolu jsme přistupovali stejně jako k tomu z aplikace. Používali jsme proměnné a bloky s podmínkou.



Obrázek 107 Úkol 11.2 Scratch

Úkol 11.3 vylepšuje program o možnost zaznamenávat nejlepší výsledek a opětovné spuštění, které jsme propojili se stiskem dotykového senzoru 1.

Obrázek 108 Úkol 11.3 Scratch

7.11.4. Problémy při řešení, postřehy a úpravy

Problémem byla mohutnost kódu v aplikaci, registrovali jsme trhaný pohyb ukazatele myši, dlouhou prodlevu mezi tažením a umístěním bloků. Problém přetrvával i po změně notebooku za mnohem výkonnější počítač.

V rozšíření Scratche jsme místo displeje používali výpis do okna s postavičkou.

7.11.5. Porovnání

Pro tuto kapitolu bychom zvolili prostředí rozšíření Scratche. Kód byl pro nás přehlednější, bylo možné využívat volání bloků a neregistrovali jsme trhaný pohyb při pohybu v kódu i s jeho větší mohutností.