STAVEBNICE LEGO MINDSTORMS EDUCATION EV3, MATERIÁLY KU ŠKOLENIAM

Pavel Petrovič, FMFI UK, Bratislava, v spolupráci so spoločnosťou Eduxe Slovensko, s.r.o.

Úvod

Robotická stavebnica LEGO MINDSTORMS Education EV3 je učebnou pomôckou, ktorá slúži ako prostriedok pre žiakov na objavovanie poznatkov a zbieranie skúseností o mechanike, dynamike, konštrukcii, snímačoch, elektrických motoroch, procesoch riadenia systémov, ale predovšetkým programovania. Práca so stavebnicou rozvíja nielen manuálne zručnosti a jemnú motoriku, pozitívne ovplyvňuje priestorovú predstavivosť žiakov, kombinačné schopnosti, tímové a komunikačné schopnosti, logické myslenie, schopnosť plánovania, abstraktného i analytického uvažovania.

Riadiaca jednotka EV3

Ide už o tretiu generáciu stavebnice v rade LEGO MINDSTORMS, z ktorej je možné konštruovať autonómne roboty. Prvou generáciou boli RCX, na trh prišli v roku 1998, druhou NXT (rok 2006) a treťou sú EV3 (rok 2013). Ako názov napovedá (EV3 – od slova evolution – postupná zmena), ide o vylepšenú verziu rovnakého konceptu. Preto používatelia, ktorí pracovali s predchádzajúcimi verziami stavebníc môžu začať používať EV3 plynule a väčšina čŕt systému im bude prirodzená a dobre známa. Ešte pred verziou RCX predávalo LEGO už dlhšiu dobu stavebnice zo série LEGO Dacta Control Lab, ktoré rovnako obsahovali motory, senzory a bolo v nich možné vytvárať automatizované alebo robotické modely. Control Lab však neobsahoval žiadnu autonómnu riadiacu jednotku, program bežal priamo na PC (resp. Apple). Tento systém LEGO Education vyvinulo v spolupráci so Seymourom Papertom – otcom didaktického štýlu konštrukcionizmus, ktorý vychádzajúc z myšlienok konštruktivizmu (budovania, konštruovania poznatkov na základe skúseností) pridáva získavanie skúseností priamo konštruovaním fyzických artefaktov (napr. robotov), prácou v skupine (sociálny aspekt). Dôležitým je aj zdieľanie výsledkov detí so svojimi rovesníkmi.

Jadrom stavebnice je riadiaca jednotka – kocka EV3, okolo ktorej sa vybuduje robot, alebo nejaké iné automatizované zariadenie. EV3 je elektrický komponent, takže potrebuje zdroj elektrickej energie, tým je buď 6 článkov AA (bežné tuškové batérie, môžeme použiť aj nabíjateľné), alebo 7.4 V baterkový nabíjateľný modul s kapacitou 2050 mAh. Baterkový modul je súčasťou stavebnice vo verzii Education, nie je súčasťou stavebnice predávanej na hračkárskom trhu. Nabíja sa napájacím adaptérom, ktorý je potrebné zakúpiť zvlášť (z dôvodu, že pri dobrej organizácii nabíjaní batérií nie je potrebné mať ku každej baterke aj adaptér).

EV3 je počítač (na ktorom beží operačný systém Linux, ale to používateľ vôbec nepotrebuje tušiť). Má štyri vstupné porty označené 1, 2, 3, 4 – tam sa pripájajú snímače (iným slovom senzory) a štyri výstupné prípojky označené A, B, C, D – tam sa pripájajú motory. Používateľ s počítačom komunikuje pomocou 6 tlačidiel na vrchu riadiacej jednotky, najmä na výber položiek a pohyb v menu, ktoré sa zobrazuje na grafickom monochromatickom displeji s rozlíšením 178 x 128 bodov.

Programy pre riadiacu jednotku sa pripravujú buď priamo na nej cez menu, alebo na bežnom počítači (PC, Apple, tablet) a do riadiacej jednotky sa prenesú cez mini-USB kábel, alebo cez bezdrôtové spojenie BlueTooth, prípadne Wifi – ak do jednotky pripojíme USB wifi sieťovú kartu, ktorú EV3 rozpozná.

EV3 si pamätá všetky programy, ktoré sme do nej nahrali aj po vypnutí: sú uložené v pamäti typ flash, veľkosti 16 MB. Ukladajú sa tam aj obrázky, zvuky, záznamy dát, alebo iné súbory. EV3 má vlastný (linuxový) súborový systém, súbory môžeme pomerne jednoducho kopírovať medzi EV3 a PC (robí sa to priamo z menu softvéru Tools – Memory Browser). Program, ktorý práve beží, alebo údaje v jeho premenných (ako i celý bežiaci systém) sú uložené v pamäti RAM veľkosti 64 MB.

Spôsob práce s robotickou stavebnicou

Práca s celou stavebnicou je navrhnutá tak, aby používateľ nepotreboval žiadne technické znalosti, softvér obsahuje množstvo príkladov a ukážok a programovací jazyk je grafický, prirodzene zrozumiteľný a nastavený na spôsob konštrukcionistickeho bádania, v ktorom dieťa samé objavuje základné princípy programovania.

Ani učiteľ vopred nepotrebuje žiadne špeciálne znalosti, konštrukcionizmus predpokladá, že učiteľ nie je formálna autorita, ale rovnocenný člen tímu. S ostatnými členmi tímu sa delí o svoje ľudské skúsenosti a iný vyzretejší pohľad na svet. Tím sa spoločne učí objavovaním, riešením výziev a motivujúcou prácou pripomínajúcou hru.

Jedna stavebnica je vhodná pre skupinu dvoch žiakov – pri stavbe jeden vyberá súčiastky a radí, druhý skladá, neskôr sa vymenia. Pri programovaní sa navzájom radia, spolu rozmýšľajú a dohadujú sa na spôsobe riešenia. Pri práci cez bezdrôtové spojenie môže jeden obsluhovať robota a druhý PC, kde upravuje program podľa pokynov “operátora v teréne”.

Okrem práce so skupinou žiakov je možné stavebnicu využiť ako ukážkový model, ktorý prinesie učiteľ a demonštruje na ňom nejaký princíp (napríklad na matematike alebo fyzike).

Do tretice, učiteľ môže skupine aktívnych žiakov zadať na dlhšie obdobie (napr. 3 týždne) tému, ktorú skupina spracuje vo forme menšieho projektu a výsledok názorne a interaktívne odprezentuje pred ostatnými žiakmi.

Účelom tohto školenia je na praktických ukážkach informovať učiteľa o rozsahu možností stavebnice a previesť ho cez príklady robotov a programov, na ktorých pochopí jednotlivé princípy, ktoré EV3 vie efektívne demonštrovať, aby získal nejaký náskok pred žiakmi pri spoločnom objavovaní, ktoré je ale základom ku konštrukcionistickému úspechu.

K stavebniciam je priložená brožúrka s návodom na základný model robota (Robot Educator), ku ktorému sa jednoducho pripájajú všetky druhy senzorov. Návod je jeho stavbu je nielen v priloženej brožúre, ale je aj súčasťou softvéru. S modelom Robot Educator budeme pracovať počas celého školenia. Preto sa tieto materiály sústredia najmä na programovanie, keďže tam sa skrýva najviac zaujímavých konceptov.

V softvéri sú návody na viacero zaujímavých modelov, ktoré sa dajú poskladať zo základnej stavebnice a sada zložitejších modelov, ku ktorým je potrebná aj doplnková sada.

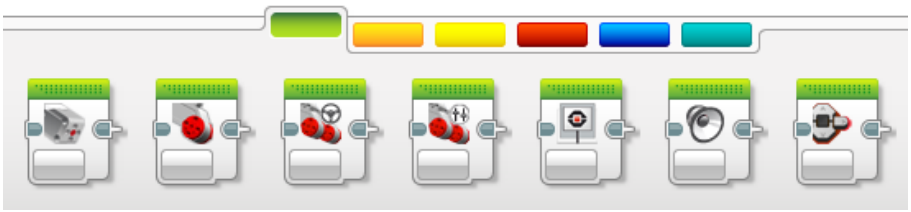
Okrem toho je množstvo modelov k dispozícii priamo na webových stránkach LEGO MINDSTORMS (legomindstorms.com). Vlastné kvalitné stavebné návody sa dajú vytvárať napr. pomocou softvéru LEGO Digital Designer, alebo LDraw.

Štruktúra dokumentu

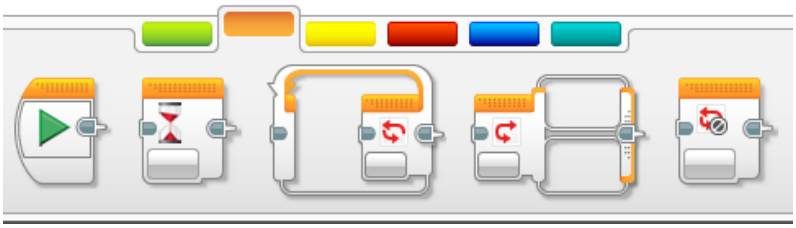
Nasledujúca časť dokumentu obsahuje najskôr prehľad paliet programovacích príkazov, potom prehľad príkazov v jednotlivých paletách (tieto je možné používať aj ako samostatné pomôcky) a následne sú všetky príkazy podrobne vysvetlené s príležitostnými príkladmi. Ide teda o pokus o referenčnú príručku k programovaciemu jazyku EV3 v slovenčine. Na záver je zaradených 27 úloh v poradí postupného objavovania jednotlivých konceptov programovacieho jazyka – zadania i riešenia. Potešíme sa spätnej väzbe, opravám prípadných chýb, alebo návrhom na vylepšenia, napíšte na pavel.petrovic@gmail.com.

SKUPINY PRÍKAZOV

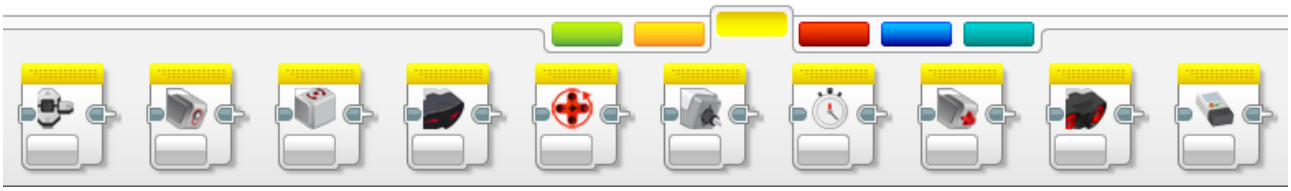
1. **Výstupy (motory, displej, zvuky, podsvietenie tlačidiel)**



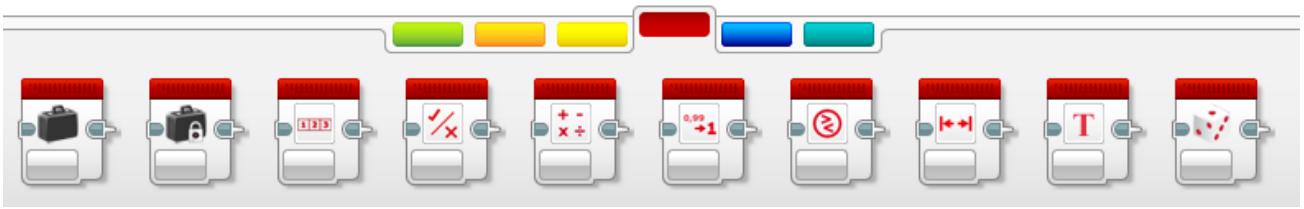
2. **Štruktúra – čakanie na udalosť, cyklus, vetvenie programu, prerušenie cyklu**



3. **Vstupy – tlačidlá, všetky typy senzorov, otáčky motorov, časovač, diaľkový IR ovládač**



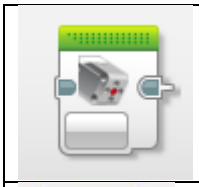
4. **Práca s údajmi – premenné, polia, logické, aritmetické a textové operácie, porovnávanie, náhodné čísla**



5. **Pokročilé – súbory, meranie, Bluetooth, priame riadenie výstupov a vstupov, stop programu, poznámka**

6. **Používateľom vytvorené**

1. Výstupy (motory, displej, zvuky, podsvietenie tlačidiel

	**Pohyb stredným motorom (medium motor)**
	**Pohyb veľkým motorom (large motor)**
	**Pohyb dvoma veľkými motormi (volant)**
	**Pohyb dvoma veľkými motormi (tank)**
	**Výstup na displej**
	**Zvukový výstup**
	**Podsvietenie tlačidiel**

2. Štruktúra – čakanie na udalosť, cyklus, vetvenie programu, prerušenie cyklu

	**Začiatok programu**
	**Čakanie na udalosť (senzory, čas, …)**
	**Opakovanie (cyklus)**

	**Vetvenie programu (podmienka)**
	**Prerušenie opakovania cyklu**

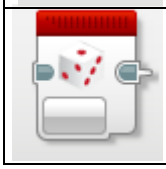
3. Vstupy – tlačidlá, všetky typy senzorov, otáčky motorov, časovač, diaľkový IR ovládač

POZOR!	**Tieto príkazy nič nerobia, na nič nečakajú, iba merajú hodnoty na vstupoch!**
	**Zistenie stavu tlačidiel na riadiacej jednotke EV3**
	**Farebný (svetelný) senzor**
	**Gyroskopický senzor**
	**Infračervený senzor (iba hračkárska stavebnica)**
	**Otáčkový senzor zabudovaný v motoroch**
	**Tepelný senzor (extra doplnok)**
	**Časovač**
	**Dotykový senzor**
	**Ultrazvukový senzor na meranie vzdialenosti**

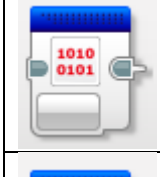
	**Čítanie z tehličky na meranie energie (samostatná súprava)**
	**Zvukový senzor zo stavebnice NXT**

4. Práca s údajmi – premenné, polia, logické, aritmetické a textové operácie, porovnávanie, náhodné čísla

	**Kufrík – premenná: zápis alebo čítanie z premennej**
	**Kufrík – konštanta (ak nejakú hodnotu potrebujeme viackrát v programe, uložíme ju sem)**
	**Práca s poľom (premenná typu pole schová v jednej premennej viacero údajov naraz)**
	**Logické operácie**
	**Aritmetické operácie**
	**Zaokrúhľovanie čísel**
	**Porovnávanie čísel**
	**Test na interval (číslo je vnútri alebo vonku intervalu OD – DO)**

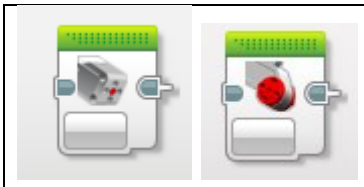
	**Textové operácie (lepenie viacerých textov dokopy)**
	**Generovanie náhodných čísel a logických hodnôt**

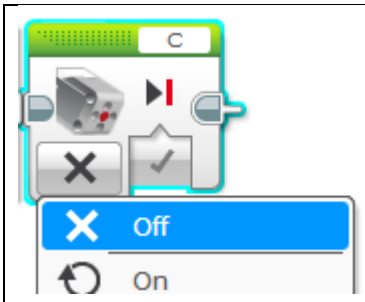
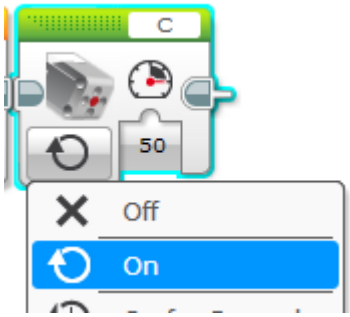
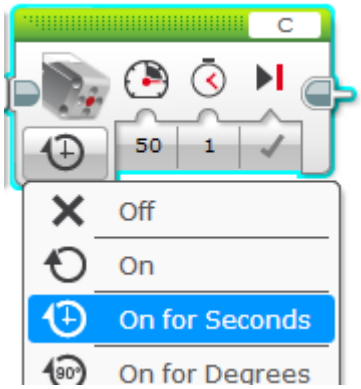
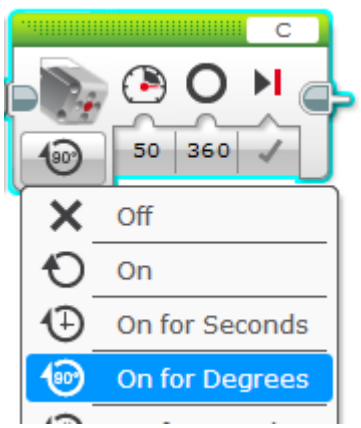
5. Pokročilé – súbory, meranie, Bluetooth, priame riadenie výstupov a vstupov, stop programu, poznámka

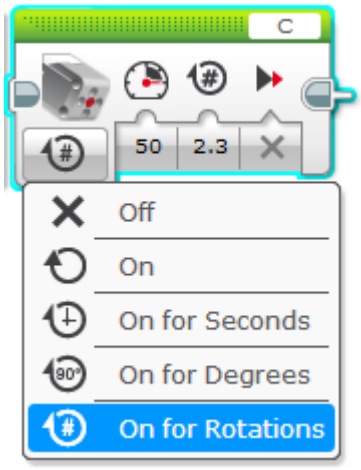
	**Práca so súbormi**
	**Záznam dát (meranie)**
	**Posielanie správ cez Bluetooth**
	**Nastavovanie spojenia Bluetooth priamo z programu (dá sa aj ručne priamo na kocke)**
	**Nastavenie časového intervalu automatického vypínania**
	**Priama (surová, nespracovaná) hodnota zo senzora**
	**Priamy (neregulovaný) výstup na motor**
	**Obrátenie otáčania neregulovaného motora**

	**Zastavenie celého programu na určitom mieste uprostred programu**
	**Komentár (komentáre sa dajú vytvárať aj pomocou funkcie “Comment” s rovnakou ikonou v toolbare)**

1. Výstupy (motory, displej, zvuky, podsvietenie tlačidiel

	**Pohyb stredným a veľkým motorom (medium motor, large motor)** *Nastavenia sú pre oba príkazy rovnaké – len ikonka príkazu sa líši*

	Vypni medium motor Vpravo hore nastaviť port A,B,C,D Vpravo dole nastaviť, či sa má aktivovať brzda (zaškrtnuté, BREAK), alebo nie (nezaškrtnuté – COAST)
	Zapni medium motor Odteraz sa bude medium motor na danom porte pohybovať zadanou rýchlosťou, ale program pokračuje ďalej! Motor sa bude touto rýchlosťou pohybovať až dovtedy, kým mu zasa nejakým pohybovým príkazom rýchlosť nezmeníme. Vpravo hore nastaviť port A,B,C,D Nastaviť rýchlosť pohybu
	Pohybuj medium motorom zadaný počet sekúnd Medium motor na zadanom porte sa bude pohybovať zadanou rýchlosťou stanovenú dobu v sekundách. Program nepokračuje ďalej, kým sa pohyb nedovykonáva! Potom sa motor zabrzdí, ak posledné nastavenie (BREAK) je zaškrtnuté, alebo zastane iba zotrvačnosťou, ak nie je (COAST). Záporná rýchlosť = spätný chod. Vpravo hore nastaviť port A,B,C,D Nastaviť rýchlosť (-100 – 0 – 100 % maximálnej rýchlosti) Nastaviť čas v sekundách Nastaviť brzdu na konci áno/nie
	Pohybuj medium motorom zadaný počet stupňov Medium motor na zadanom porte sa zadanou rýchlosťou otočí o zadaný počet stupňov. Program nepokračuje ďalej, kým sa pohyb nedovykonáva! V každom prípade motor na konci zastane, ale ak je aj posledné nastavenie (BREAK) zaškrtnuté, tak ostane zabrzdený, inak ostane uvoľnený (COAST). Záporná rýchlosť = spätný chod. Vpravo hore nastaviť port A,B,C,D Nastaviť rýchlosť (-100 – 0 – 100 % maximálnej rýchlosti) Nastaviť stupne, o ktoré sa má motor (pozor: nie robot!) otočiť Nastaviť brzdu na konci áno/nie

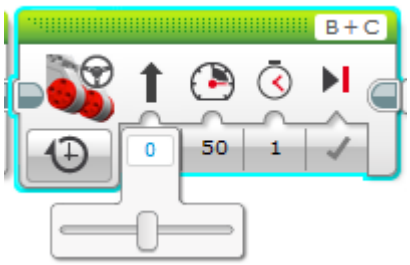


Pohybuj medium motorom zadaný počet otáčok (napr. 2.3 otáčky)

Medium motor na zadanom porte sa zadanou rýchlosťou otočí o zadaný počet otáčok (desatinné čisla zadať s bodkou). Program nepokračuje ďalej, kým sa pohyb nedovykonáva! V každom prípade motor na konci zastane, ale ak je aj posledné nastavenie (BREAK) zaškrtnuté, tak ostane zabrzdený, inak ostane uvoľnený (COAST). Záporná rýchlosť = spätný chod.

Vpravo hore nastaviť port A,B,C,D
Nastaviť rýchlosť (-100 – 0 – 100 % maximálnej rýchlosti)
Nastaviť otáčky, o ktoré sa má motor (pozor nie robot!) otočiť
Nastaviť brzdu na konci áno/nie

Pohyb dvoma veľkými motormi (volant)

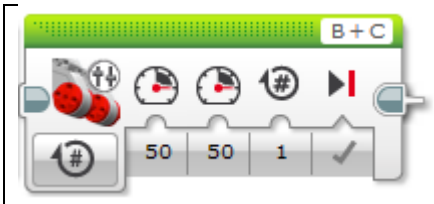


Nastavenia pre tento blok sú rovnaké ako nastavenia pre pohyb jedným motorom (pozri vyššie), jedine pribúda nastavenie smeru pohybu. Pohyb motorov bude regulovaný pomocou otáčkových senzorov podľa zadaného smeru:

-100: otáčanie na mieste vľavo (oba motory sa točia rovnakou rýchlosťou ale opačným smerom)
medzi -100 a 0: pohyb po kruhovom oblúku vľavo, hodnota určí prudkosť zákruty
0: pohyb priamo vpred
medzi 0 a 100: pohyb po kruhovom oblúku vpravo, hodnota určí prudkosť zákruty
+100: otáčanie na mieste vpravo

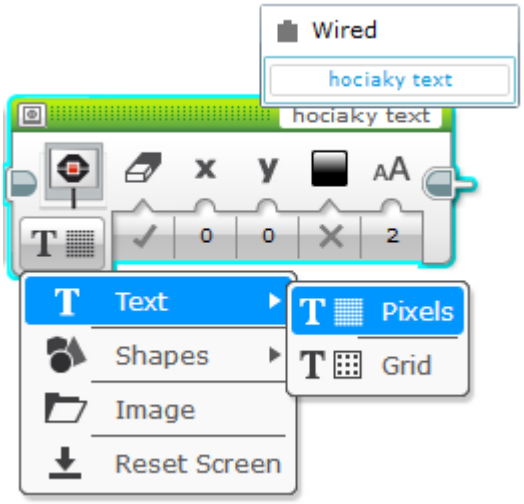
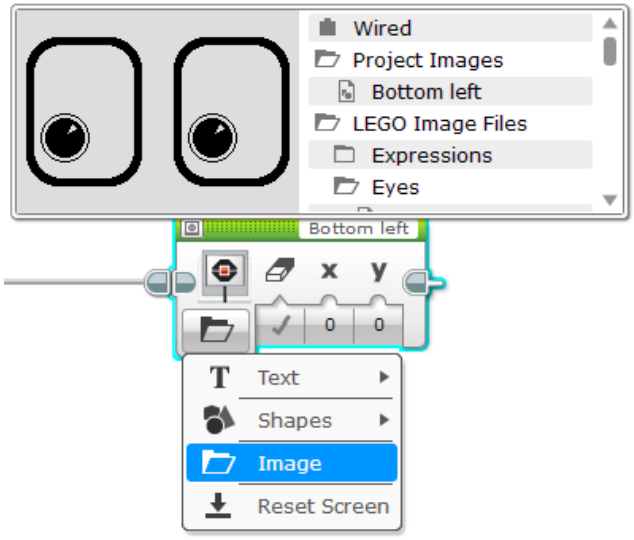
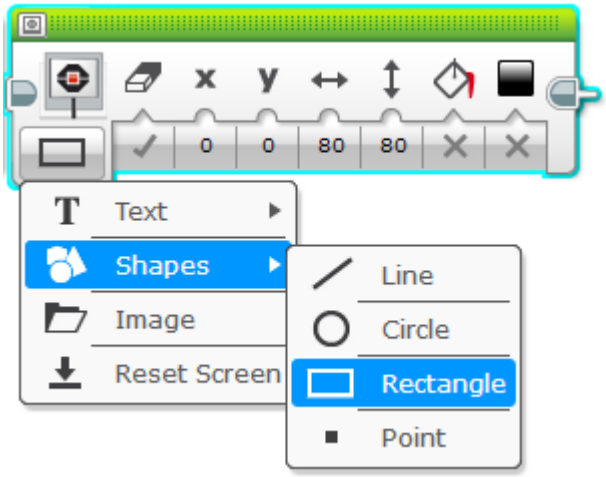
Vpravo hore nastaviť dva porty spomedzi A,B,C,D
Nastaviť smer otáčania, rýchlosť (a prípadne dĺžku trvania, resp. stupne, otáčky) a či sa má aktivovať brzda (zaškrtnuté – BREAK), alebo nie (nezaškrtnuté – COAST)

Pohyb dvoma veľkými motormi (tank)

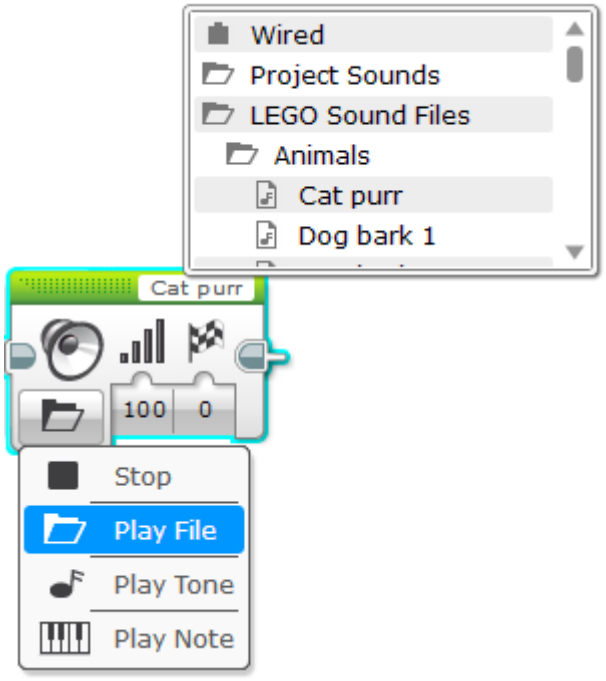
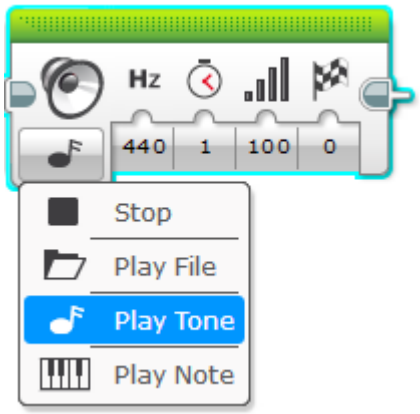
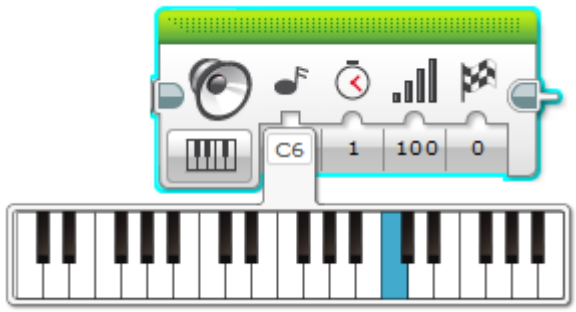


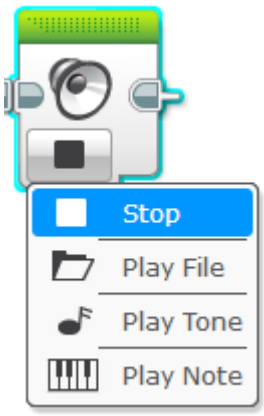
Nastavenia pre tento blok sú rovnaké ako nastavenia pre pohyb jedným motorom (pozri vyššie), pribúda nastavenie smeru pohybu aj pre druhý motor.
Vpravo hore nastaviť dva porty spomedzi A,B,C,D
Nastaviť rýchlosti pre oba motory (a prípadne dĺžku trvania, resp. stupne, otáčky) a či sa má aktivovať brzda (zaškrtnuté, BREAK), alebo nie (nezaškrtnuté – COAST)

	**Výstup na displej**

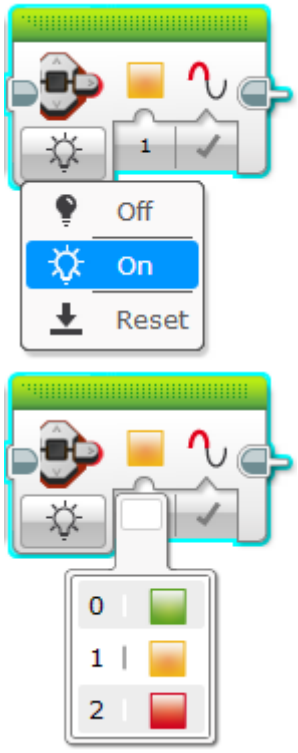
	Zobrazenie textu na displeji Vyberáme si, či súradnice na displeji zadáme v bodoch (*Pixels*) alebo v riadkoch/stĺpcoch znakov (*Grid*), vpravo hore zadávame text, ktorý sa má vypísať, prípadne nastavíme voľbu “Wired” – vtedy pribudne ešte jeden vstup, kam dátovým drôtikom môžeme do bloku poslať hodnotu, ktorá sa má vypísať. Guma – má sa pred vypísaním textu celý displej zmazať? (áno/nie) X, Y – súradnice na displeji, kam text vypísať Farba – vypisovať čiernou/bielou Font – veľkosť písma
	Zobrazenie obrázku na displeji Vpravo hore vyberáme obrázok, ktorý sa má nakresliť, prípadne nastavíme voľbu “Wired” – vtedy pribudne ešte jeden vstup, kam dátovým drôtikom môžeme do bloku poslať meno súboru obrázka, ktorý sa má vykresliť. Guma – má sa pred vykreslením obrázku celý displej zmazať? (áno/nie) X, Y – súradnice na displeji, kam obrázok vykresliť
	Vykreslenie geometrického útvaru Vyberáme si útvar: *Line* – úsečka, *Circle* – kružnica, *Rectangle* – obdĺžnik, *Point* – bod Guma – má sa pred vykreslením útvaru celý displej zmazať? (áno/nie) X, Y – súradnice na displeji, kam sa má útvar nakresliť (podľa typu útvaru sa nastavenia menia – napr. obdĺžnik má šírku, výšku) Výplň – má sa útvar vyfarbiť? (áno/nie) Farba výplne – biela/čierna

	**Zvukový výstup**

	Zahrať zvolený zvukový sample. Vpravo hore si vyberieme zvuk, alebo zadáme "Wired" – v tom prípade pribudne ešte jedna prípojka, do cez ktorú drôtikom privedieme meno súboru so zvukovým samplom. Voľba *Tools – Sound Editor* umožňuje vytvoriť vlastnú nahrávku (napríklad krátke hovorené slovo), ktorá pribudne do zoznamu projektových zvukov. Nastavíme hlasitosť 0-100 a či má program čakať, kým zvuk skončí (0), alebo či má má zvuk hrať v pozadí (1), alebo sa opakovať (2).
	Zahrať zvuk Nastavíme frekvenciu zvuku v Hz, dobu trvania zvuku v sekundách, hlasitosť v percentách maximálnej hlasitosti a či program má čakať, kým zvuk dohrá (0), alebo hrať zvuk v pozadí (1), alebo sa opakovať (2). Frekvencia zvuku nemusí zodpovedať žiadnemu tónu (hudobnej note).
	Zahrať tón Zahrá zvolený tón po zadanú dobu trvania zadanou hlasitosťou. Nastavíme, či zvuk má hrať v pozadí (1), či má program čakať, kým doznie (0), alebo či sa má opakovať (2).

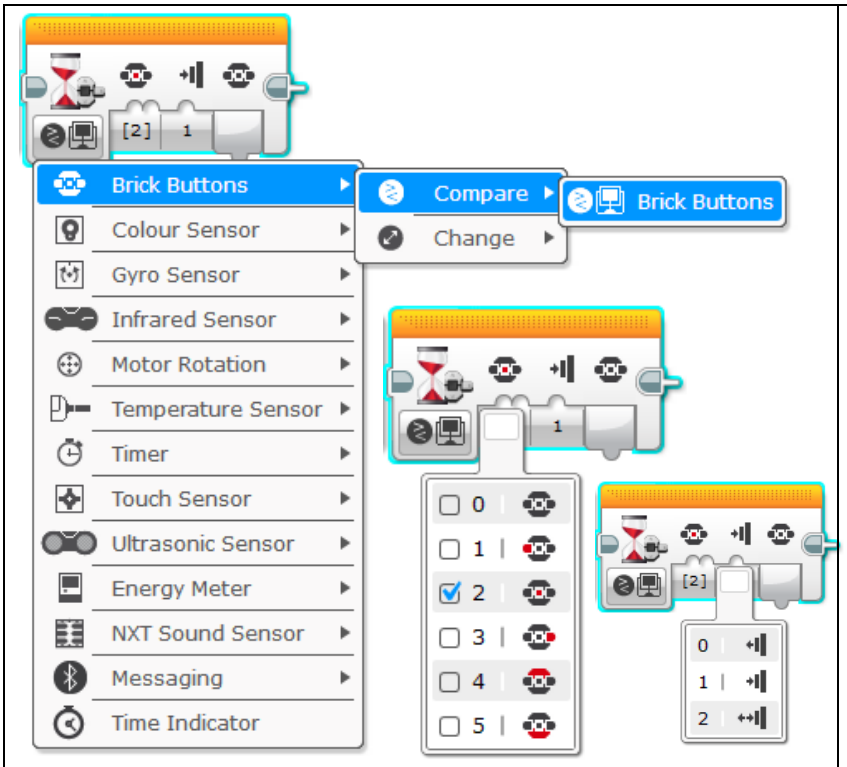
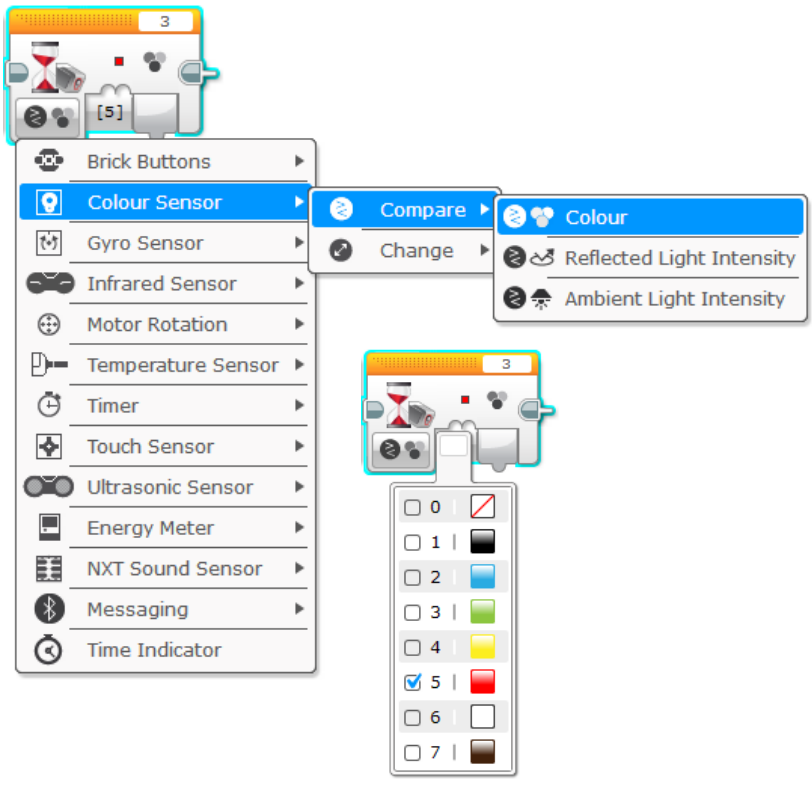
	Vypnúť zvuk Ak práve v pozadí hrá nejaký zvuk alebo tón, tento príkaz ho vypne.

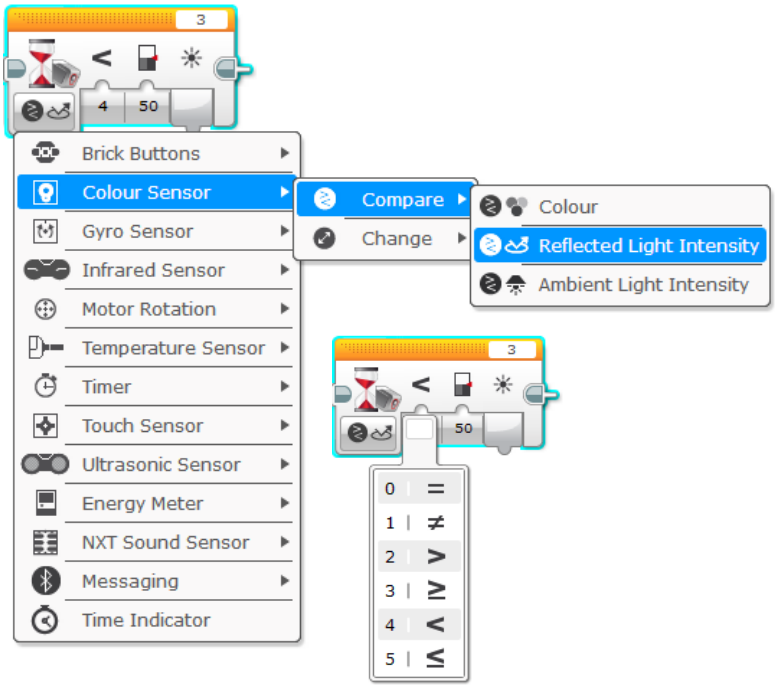
	**Podsvietenie tlačidiel**

	Nastaví farebné podsvietenie tlačidiel Vybrať akciu – zapnúť (*On*), alebo vypnúť (*Off*), resp. *Reset* – bežné nastavenie (pri nízkom napätí baterky kocka signalizuje žltým podsvietením) Vybrať farbu – zelenú, žltú, alebo červenú. Zaškrtávatko pod vlnkou vyberá blikajúce podsvietenie.

	**Začiatok programu:** Týmto blokom označujeme miesta, kde má začať vykonávanie programu.

	**Čakanie na udalosť (senzory, čas, …)**

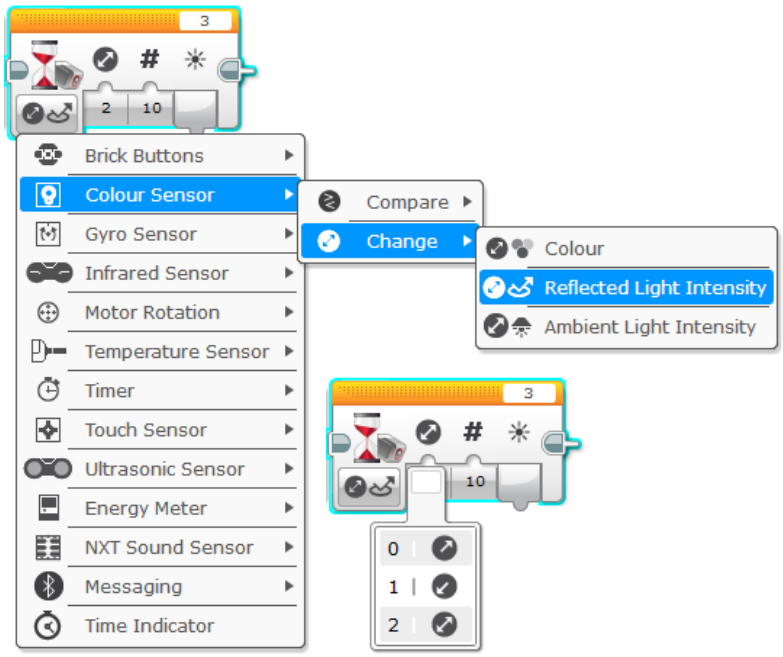
	Čakanie na tlačidlá na riadiacej jednotke EV3 *Compare* – testuje, či sú tlačidlá v požadovanom stave: 0 = pustené, 1 = stlačené, 2 = bolo stlačené a potom pustené *Change* – testuje, či došlo k akejkoľvek zmene Zaškrtnúť všetky tlačidlá na ktorých stav chceme reagovať (červeným je označené príslušné tlačidlo) Označiť stav, na ktorý reagujeme.
	Čakanie na farbu Program na tomto príkaze zastane, až dovtedy, kým senzor na zadanom porte (vpravo hore) nezosníma jednu z požadovaných farieb. V zozname farieb môžeme zaškrtnúť jednu alebo viacero farieb. Výstupnú prípojku môžeme využiť na odoslanie zosnímanej farby do iného bloku dátovodom. V režime *Change* bude program čakať dovtedy, kým sa farba, ktorú senzor sníma akokoľvek nezmení.

<table>
<tr>
<td> 
</td>
<td>Čakanie na intenzitu odrazeného svetla

Program čaká na tomto príkaze, kým farebný senzor v režime merania intenzity odrazeného svetla nezosníma hodnotu, ktorá spĺňa určenú podmienku.

Nastavíme vpravo hore port, kde je senzor pripojený, znamienko porovnania a hodnotu, s ktorou sa má výstup zo senzora porovnať. napr. < 50 znamená, že kým bude hodnota senzora vyššia alebo rovná ako 50, program bude na tomto príkaze čakať.

Režim Ambient Light Intensity funguje analogicky – meria rozptýlené a nie odrazené svetlo, čiže senzor nebude svietiť červeným svetlom, iba slabým modrým.

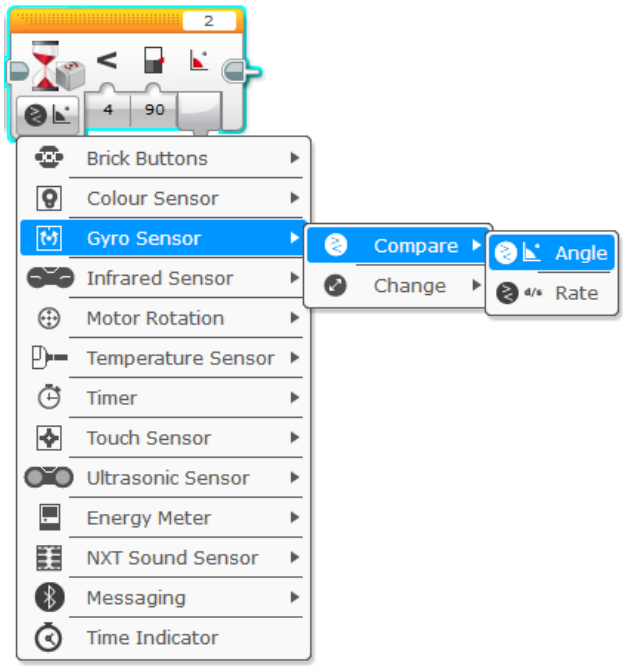
Výstupná prípojka obsahuje nameranú intenzitu odrazeného svetla.</td>
</tr>
<tr>
<td> 
</td>
<td>Čakanie na zmenu intenzity odrazeného svetla

Na rozdiel od predchádzajúceho príkazu (Compare), v tomto prípade (Change) program čaká, kým sa intenzita odrazeného svetla nezmení o zadanú hodnotu a to buď vzrastie (0), klesne (1), alebo sa zmení ľubovoľným smerom - vzrastie, alebo klesne (2). Program čaká dovtedy, kým sa hodnota senzora nezmení aspoň o zadanú hodnotu.

Nastaviť číslo portu vpravo hore, smer zmeny, veľkosť zmeny.

Hodnota senzora sa dá vytiahnuť z poslednej, výstupnej prípojky.

Režim rozptýleného svetla funguje analogicky.</td>
</tr>
</table>

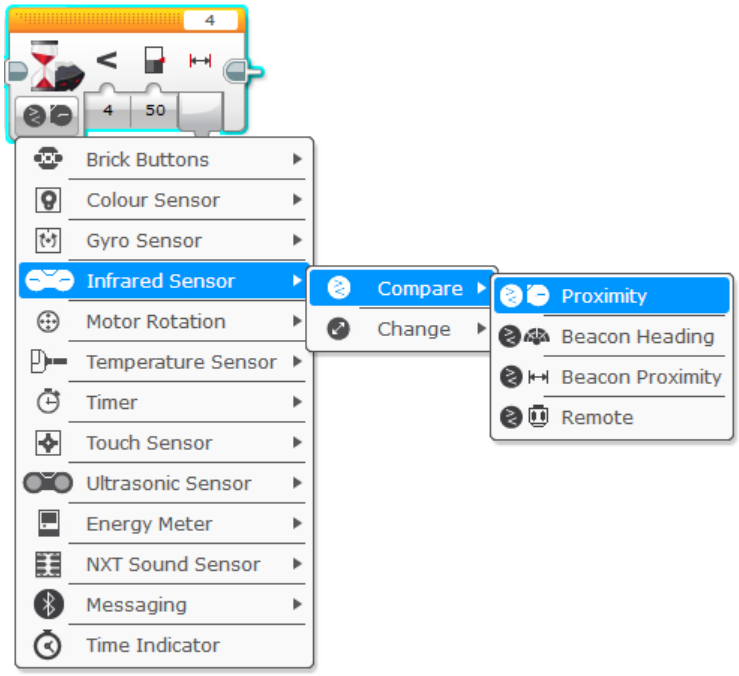
<table>
<tr>
<td> 
</td>
<td>Čakanie na otočenie gyrosenzora

Program čaká, kým sa gyroskop neotočí do polohy, v ktorom jeho hodnota bude spĺňať stanovenú podmienku.

Compare:

Angle – porovnáva sa uhol natočenia gyroskopu
Rate – porovnáva sa rýchlosť zmeny natočenia gyroskopu (v stupňoch za sekundu)

Change:

Porovnáva sa veľkosť zmeny hodnoty senzora, nie hodnota senzora samotná.</td>
</tr>
<tr>
<td> 
</td>
<td>Čakanie na vzdialenosť nameranú infračerveným senzorom

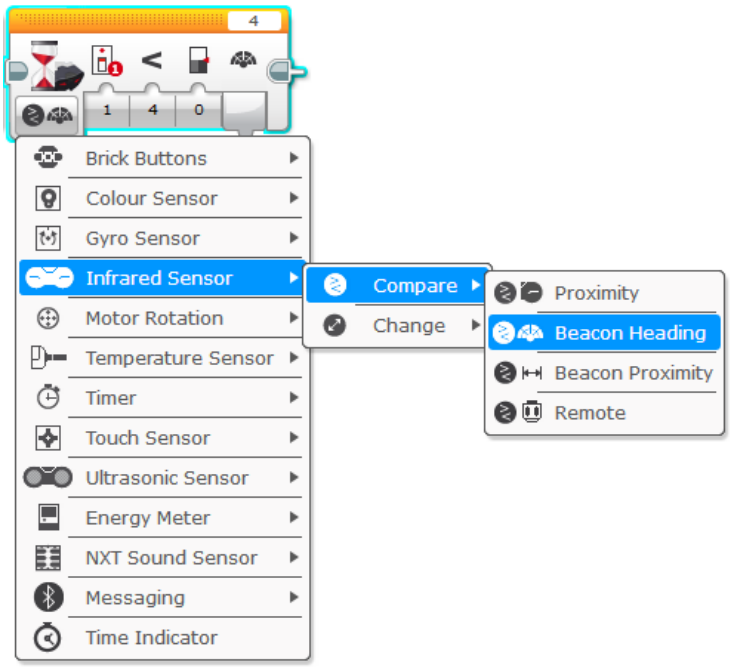
Infračervený senzor sa nachádza v hračkárskej a nenachádza sa v školskej verzii stavebnice.

Program čaká, kým vzdialenosť k prekážke nebude spĺňať stanovenú podmienku.

Nastaviť port, kde je senzor pripojený, podmienku a hraničnú hodnotu vzdialenosti (pozri Čakanie na intenzitu odrazeného svetla).

Change – vzdialenosť k prekážke sa zmenila o zadanú hodnotu (pozri Čakanie na zmenu intenzity odrazeného svetla)

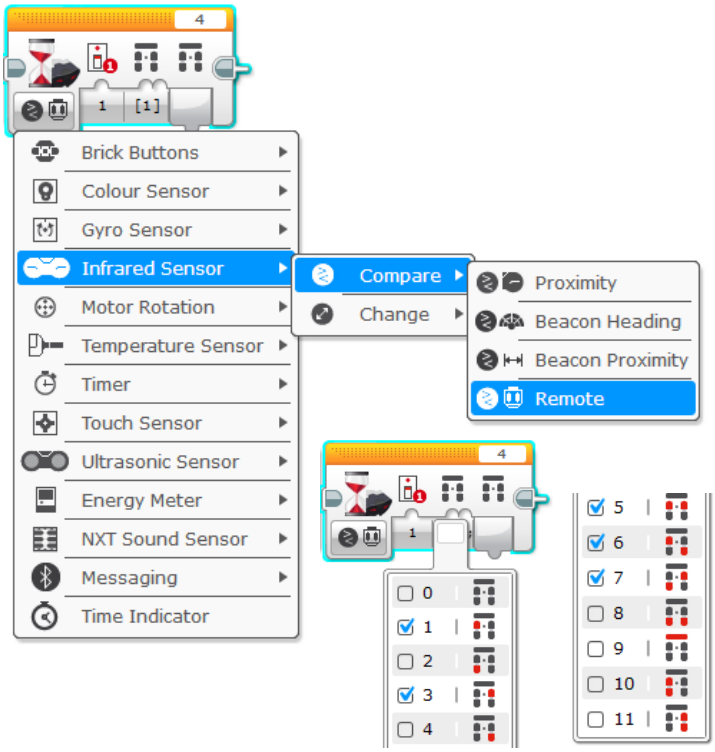
Hodnota senzora v okamihu splnenia podmienky je k dispozícii v poslednej výstupnej prípojke.</td>
</tr>
</table>

<table>
<tr>
<td> 
</td>
<td>Čakanie na smer (B.Heading) / vzdialenosť (B.Proximity) k vysielaču smerového lúča

Program čaká, kým lúč zo smerového vysielača (IR diaľkové ovládanie v režime beacon) nebude infračerveným senzorom prijatý zo smeru/vzdialenosti, ktorý/á spĺňa stanovenú podmienku.

Nastaviť port, kde je senzor pripojený, vysielací kanál, ktorý má byť rovnaký ako na vysielači (1-4), porovnávacie znamienko a hraničnú hodnotu vzdialenosti (pozri Čakanie na intenzitu odrazeného svetla).

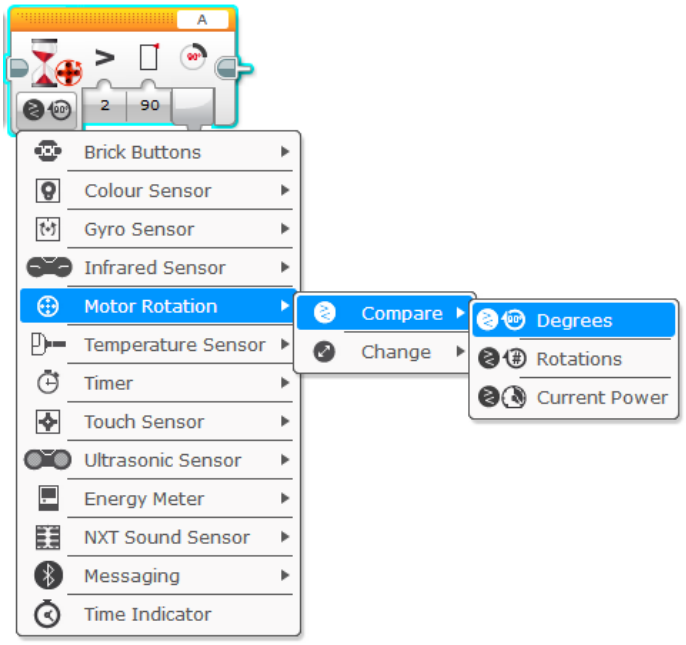
Change – smer/vzdialenosť k vysielaču sa zmenil o zadanú hodnotu (pozri Čakanie na zmenu intenzity odrazeného svetla).

Hodnota smeru/vzdialenosti k vysielaču v okamihu splnenia podmienky je k dispozícii v poslednej výstupnej prípojke.</td>
</tr>
<tr>
<td> 
</td>
<td>Čakanie na stlačenie klávesy alebo kombinácie kláves na IR vysielači

Program čaká, kým IR senzor neprijme zadaný signál, vyslaný na základe stlačenia klávesy alebo kombinácie kláves z IR vysielača.

Nastaviť port, kde je pripojený IR senzor, kanál, ktorý je na IR vysielači navolený, a všetky kombinácie stlačení kláves, pričom ľubovoľná z nich ukončí čakanie programu na tomto príkaze.

Z poslednej prípojky môžeme vytiahnuť kód klávesy alebo kombinácie, ktorá bola stlačená.</td>
</tr>
</table>



Čakanie na otočenie motora

Program čaká, kým sa motor neotočí o zadaný počet stupňov (*Degrees*), alebo otáčok (*Rotations*), 1 otáčka = 360 stupňov.

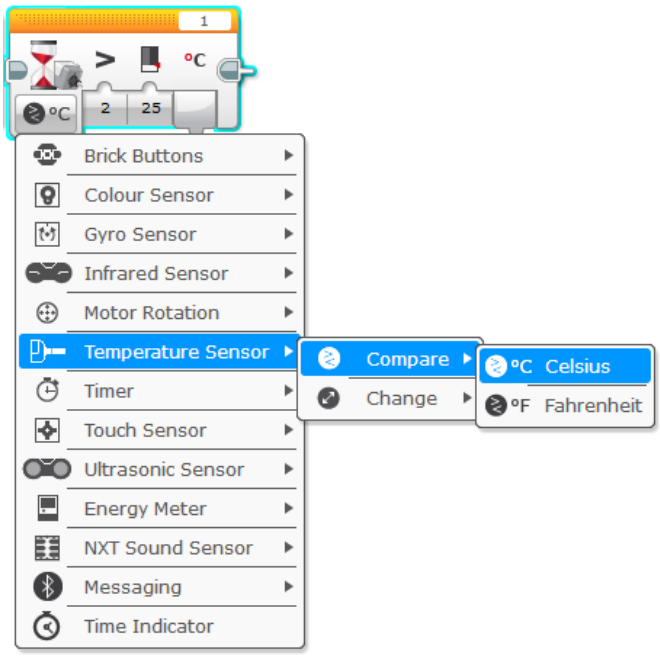
Nastaviť port, na ktorom je pripojený motor, ktorého otáčky chceme snímať, porovnávacie znamienko a hraničnú hodnotu v stupňoch alebo otáčkach (pozri *Čakanie na intenzitu odrazeného svetla*).

Otáčanie v jednom smere vedie k zvyšovaniu hodnoty otáčkového senzora a v opačnom smere k jeho znižovaniu (aj do záporných čísel). Nezabudnite pred použitím tohto bloku použiť žltý blok, ktorým otáčkový senzor na rovnakom porte vynulujete (reset).

Z poslednej prípojky môžeme vytiahnuť hodnotu otáčkového senzora, keď bola podmienka splnená.

Voľba *Current Power* porovnáva aktuálnu rýchlosť otáčania motora (v jednotkách ako sú použité v zelenom pohybovom prikaze).

Voľba *Change* reaguje na zmenu hodnoty.



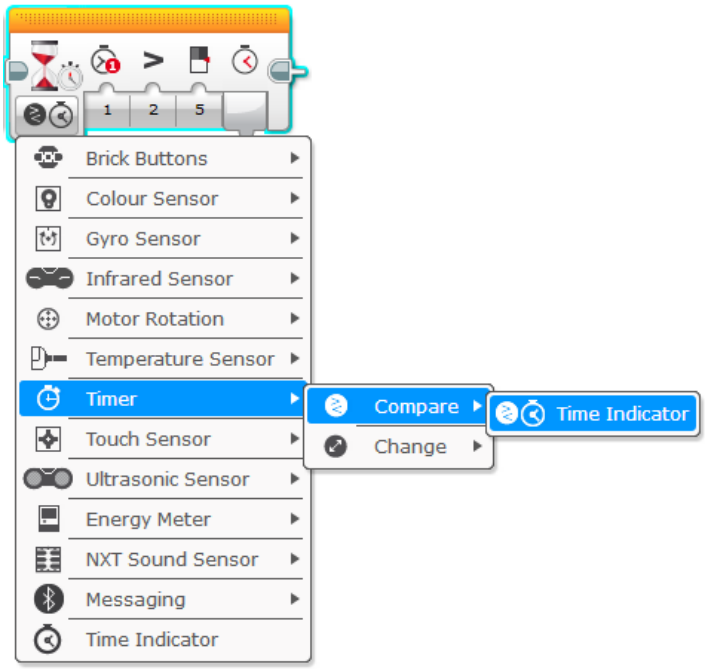
Čakanie na teplotu

Program čaká, kým teplota nameraná NXT/EV3 teplotným senzorom (9749) nebude spĺňať zadanú podmienku.

Nastaviť port, kde je senzor pripojený, podmienku a hraničnú hodnotu teploty (pozri *Čakanie na intenzitu odrazeného svetla*).

Change – teplota sa zmenila o zadanú hodnotu (pozri *Čakanie na zmenu intenzity odrazeného svetla*)

Hodnota senzora v okamihu splnenia podmienky je k dispozícii v poslednej výstupnej prípojke.



Čakanie na časovač (timer)

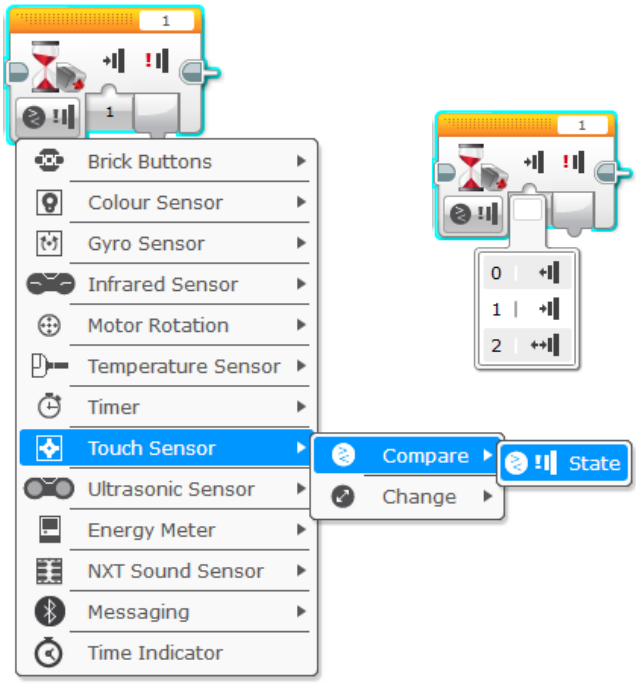
Časovač je počítadlo, ktoré sa automaticky zvyšuje o 0.001 každú milisekundu. Nuluje sa žltým blokom časovača s voľbou reset. EV3 má osem nezávislých časovačov.

Program čaká, kým zvolený časovač nebude spĺňať nastavenú podmienku.

Nastaviť číslo časovača 1-8, operátor porovnania a hraničnú hodnotu v sekundách (pozri *Čakanie na intenzitu odrazeného svetla*).

Change – počítadlo časovača sa zmenilo zadaným smerom o zadanú hodnotu.

Hodnota počítadla časovača v okamihu splnenia podmienky sa dá vytiahnuť z poslednej prípojky.

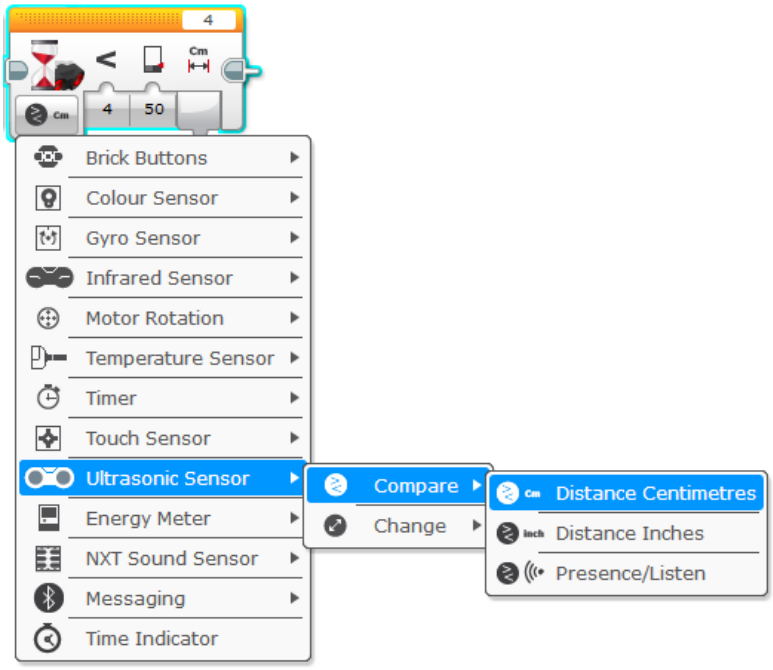
<table>
<tr>
<td> 
</td>
<td>Čakanie na dotykový senzor

Program čaká, kým stav na dotykovom senzore nebude podľa nastavenia v príkaze:

0 – senzor nie je stlačený
1 – senzor je stlačený
2 – senzor bol stlačený a potom pustený

V prípade režimu Change program čaká na ľubovoľnú zmenu stlačenia senzora.

Stav stlačenia senzora v okamihu splnenia podmienky je možné vytiahnuť z poslednej výstupnej prípojky.

Nastaviť číslo portu, kde je senzor pripojený.</td>
</tr>
<tr>
<td> 
</td>
<td>Čakanie na ultrazvukový senzor

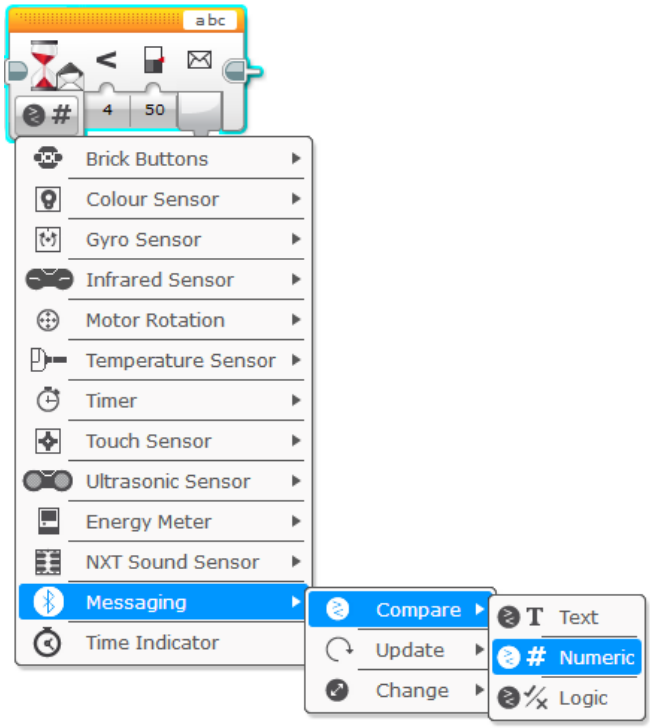
Program čaká, kým vzdialenosť k prekážke nameraná ultrazvukovým senzorom v cm alebo v palcoch (Inches) nebude spĺňať zadanú podmienku.

Nastaviť port, kde je senzor pripojený, podmienku a hraničnú hodnotu vzdialenosti (pozri Čakanie na intenzitu odrazeného svetla).

Change – vzdialenosť sa zmenila o zadanú hodnotu (pozri Čakanie na zmenu intenzity odrazeného svetla)

Nameraná vzdialenosť v okamihu splnenia podmienky je k dispozícii v poslednej výstupnej prípojke.

Režim Presence/Listen prepne senzor do pasívneho režimu (nevysiela signál) a program na príkaze čaká, kým senzor nezachytí signál z iného senzora (napr. vysielaného z iného robota).</td>
</tr>
</table>



Čakanie na prijatie správy zo spojenia BlueTooth

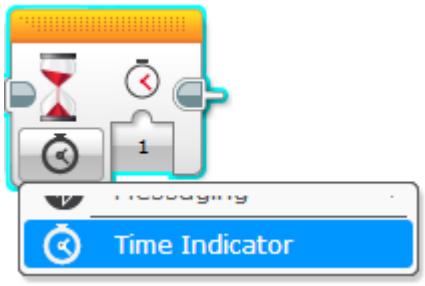
Program čaká, kým nebude prijatá správa, ktorá by spĺňala zadanú podmienku.

V pravom hornom rohu treba nastaviť názov správy, ktorý musí byť zhodný v príkaze, ktorý správu odosiela.

Cez spojenie je možné posielať správy všetkých troch dátových typov, treba vybrať zhodný typ ako v príkaze, ktorý správu posiela.

Po splnení podmienky program pokračuje a prijatá správa sa dá vytiahnuť z poslednej výstupnej prípojky.

Režim *Change* – reaguje na zmenu hodnoty prijatej správy, režim Update na každú novú prijatú správu.



Čakanie stanovený počet sekúnd

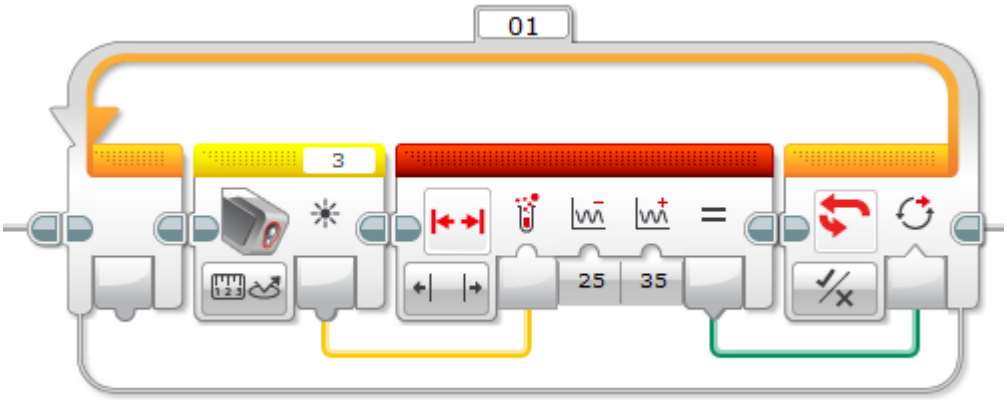
Základné nastavenie čakacieho bloku: program počká, kým neuplynie zadaný počet sekúnd.

Opakovanie (cyklus)

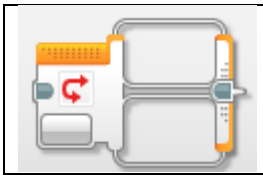
Všetky príkazy, ktoré sú uvedené v cykle, sa opakujú dovtedy, kým podmienka uvedená na konci cyklu nezačne platiť.

Podmienka sa testuje iba na konci cyklu. Je možné použiť rovnaké podmienky ako v prípade čakacieho bloku. Okrem toho je možné zadať počet opakovaní priamo –

ako je zobrazené na obrázku – 10 opakovaní – prípadne tento počet priviesť dátovodom. Špeciálna hodnota je nekonečno (*unlimited*), vtedy cyklus beží až dovtedy, kým sa nevykoná príkaz prerušenia cyklu s rovnakým menom cyklu ako je uvedené na vrchu cyklu (na obrázku "cyk1"). Na ľavej hranici cyklu sa z výstupnej prípojky dá vytiahnuť poradové číslo opakovania cyklu. V príklade na obrázku sa táto hodnota zobrazuje na displeji. V prípade komplikovanejšej podmienky možno podmienku ukončenia cyklu nastaviť na "*Logic*" a do vstupnej prípojky na konci cyklu priviesť logickú hodnotu (pravdivý (*True*) / nepravdivý (*False*)) – cyklus skončí, keď káblikom pritečie hodnota pravdivý:

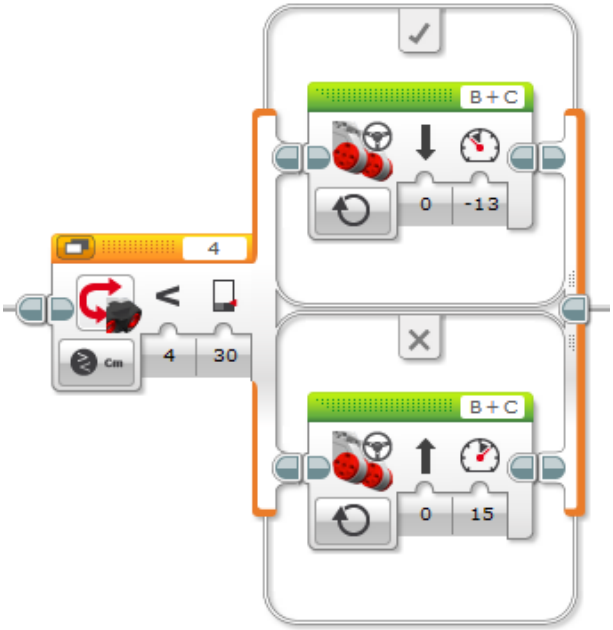


Cyklus na obrázku sa bude opakovať, kým množstvo odrazeného svetla dopadajúceho do farebného senzora na porte 3 bude v rozmedzí 25 – 35. Keď sa táto veličina dostane mimo uvedený interval, opakovanie cyklu skončí.



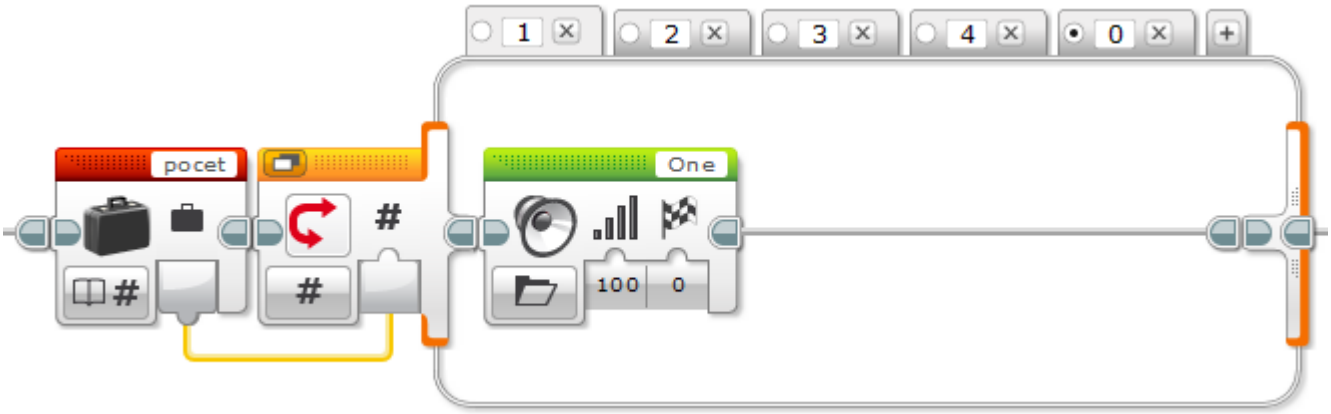
Vetvenie programu (podmienka)

Vetvenie umožňuje zadať skupinu príkazov, ktoré sa majú vykonať len za určitých okolností – ak je splnená nejaká zadaná podmienka. Zároveň možno zadať príkazy, ktoré sa majú vykonať, ak podmienka splnená nie je. Podmienky, ktoré možno zadať, sú rovnaké ako v prípade čakacieho príkazu a cyklu.



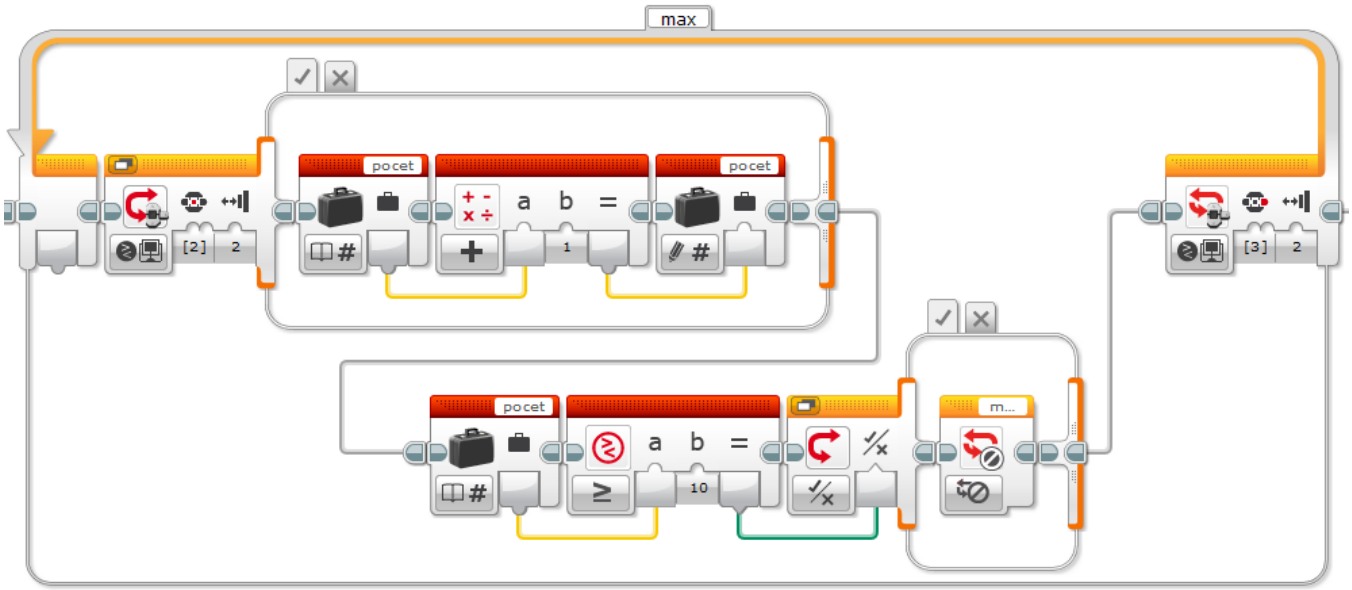
Na obrázku je príklad podmienky, ktorá preskúma hodnotu prečítanú z ultrazvukového senzora na porte 4 – nameranú vzdialenosť k prekážke. Ak je prekážka v blízkosti (menej ako 30 cm), robot sa začne pohybovať vzad, ak je dostatočne ďaleko (aspoň 30 cm), robot sa začne pohybovať vpred.

Okrem podmienok vymenovaných v čakacom príkaze vyššie môžeme zvoliť tri ďalšie typy podmienok – text, logic, numeric, ktoré dovoľujú do podmienky doručiť cez dátovod nejakú textovú, logickú, alebo číselnú hodnotu. Príkazy v danej vetve podmienky sa vykonajú, len ak je táto hodnota rovná požadovanej hodnote určenej v podmienke. Takýmto spôsobom môže podmienka rozvetviť program aj na viac ako dve vetvy – pre každú samostatnú hodnotu je možné pridať ďalšiu vetvu – podmienku je však ikonkou v ľavom hornom rohu kvôli prehľadnosti vhodné prepnúť na "tabbed view" – záložkový mód:



Príklad na obrázku podľa hodnoty v premennej *pocet* vykoná jednu z piatich rôznych skupín príkazov – schovaných v záložkách 1 - 4 a 0. Zobrazená je práve záložka 1, v ktorej robot vysloví číslicu "One". Ak je hodnota v premennej iná ako 0-4, tak vykoná príkazy v záložke 0, lebo je (čiernym puntíkom) označená ako "*default case*" – prípad, pre všetky ostatné situácie.

	**Prerušenie opakovania cyklu**



Príkaz preruší vykonávanie určeného cyklu. V pravom hornom rohu treba zadať označenie cyklu, ktorého vykonávanie sa má prerušiť – program pokračuje príkazom, ktorý nasleduje za koncom určeného cyklu – nezávisle od toho, či podmienka zadaná na konci cyklu platí alebo nie.

Príklad na obrázku je cyklus, v ktorom sa pri každom stlačení a pustení stredného tlačidla na riadiacej jednotke zvýši hodnota premennej pocet o 1. Podmienka ukončenia cyklu je stlačenie a pustenie šípky vpravo na riadiacej jednotke. Ak však hodnota premennej pocet vzrastie aspoň na hodnotu 10, opakovanie cyklu sa preruší aj v prípade, že šípka vpravo nebola stlačená.

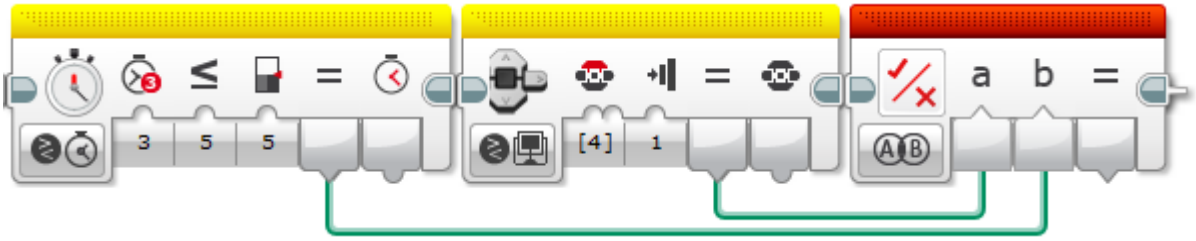
6. Vstupy – tlačidlá, všetky typy senzorov, otáčky motorov, časovač, diaľkový IR ovládač

POZOR! Tieto príkazy nič nerobia, na nič nečakajú, iba merajú okamžité hodnoty na vstupoch!

	**Zistenie stavu tlačidiel na riadiacej jednotke EV3**

V okamihu, keď sa program dostane na tento príkaz, zosníma sa stav tlačidiel. Buď ich stav dá k dispozícii na výstupnej prípojke (režim *Measure*) – ako ukazuje príklad – kód stlačeného tlačidla sa tu uloží do premennej *stlaceny*:

alebo nameranú hodnotu rovno vyhodnotí podľa zadanej podmienky:

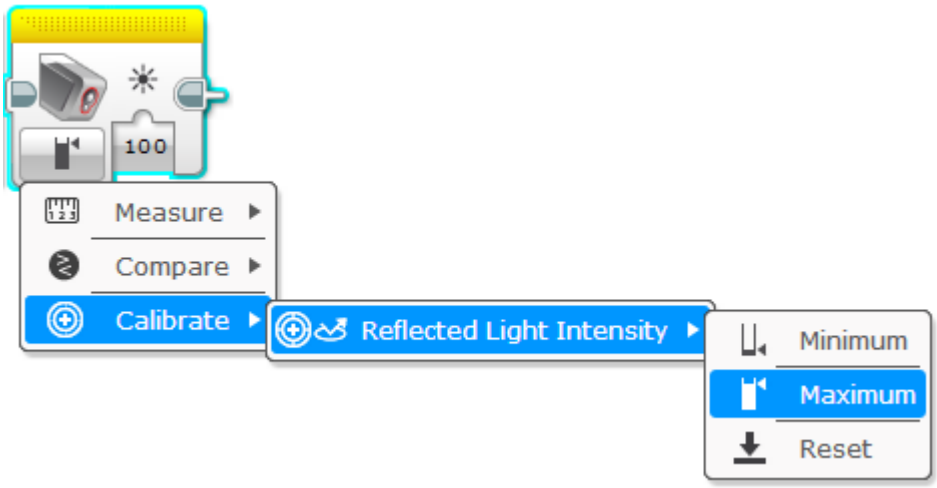


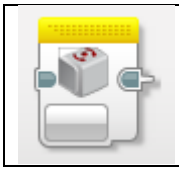
v príklade na obrázku sa zistí, či je stlačené tlačidlo 4 (šípka hore), výsledok vstupuje do logickej operácie AND, ktorá zároveň požaduje, aby hodnota počítadla časovača 3 bola nanajvýš 5 sekúnd.

	**Farebný (svetelný) senzor**

V okamihu, keď sa program dostane na tento príkaz, zosníma sa hodnota na farebnom senzore vo zvolenom režime: buď farba (reprezentovaná číslom 0-7), alebo intenzita odrazeného svetla (0 – 100 %), alebo intenzita rozptýleného svetla (0 – 100 %). V režime *Measure* je nameraná hodnota k dispozícii vo výstupnej prípojke, v režime *Compare* je hodnota porovnaná podľa zadanej podmienky a v prvej výstupnej prípojke je pravdivostná hodnota vyhodnotenia podmienky a v druhej samotná nameraná hodnota. Analogické príklady použitia sú na začiatku tejto časti o "žltých" blokoch.

V režime *Calibrate* sa upravuje merací rozsah senzora pre režim intenzity odrazeného svetla: intenzita pôvodne zodpovedajúca hodnote *minimum* (a slabšia) bude po nakalibrovaní hlásená ako hodnota 0 a intenzita zodpovedajúca hodnote *maximum* (a silnejšia) bude po nakalibrovaní hlásená ako hodnota 100. Ostatné hodnoty v rozsahu *minimum – maximum* budú úmerne preškálované do rozsahu 0 - 100. Voľba *Reset* kalibráciu zruší a senzor zasa pracuje ako predtým. Pozor - kalibrácia má vplyv na všetky pripojené farebné senzory!

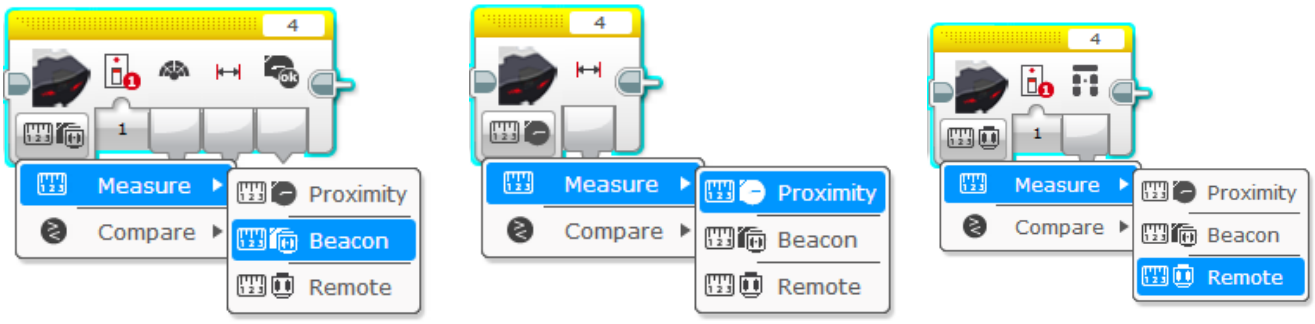


	**Gyroskopický senzor**

V okamihu, keď sa program dostane na tento príkaz, zosníma sa hodnota na gyroskopickom senzore. V režime *Angle* senzor meria uhol, o ktorý sa senzor otočil od posledného resetu. V režime *Rate* senzor meria aktuálnu uhlovú rýchlosť otáčania senzora (stupne za sekundu).

Rovnako ako v prípade predchádzajúcich senzorov máme na výber režim *Measure* – iba meranie aktuálnej hodnoty a režim *Compare* – porovnanie nameranej hodnoty podľa zadanej podmienky.

	**Infračervený senzor (iba hračkárska stavebnica)**



Infračervený senzor je zaujímavý tým, že dokáže nielen zmerať vzdialenosť k prekážke, ale aj zamerať smer lúča vysielaného z diaľkového IR ovládača i jeho približnú vzdialenosť a dekódovať riadiace príkazy diaľkového IR ovládača.

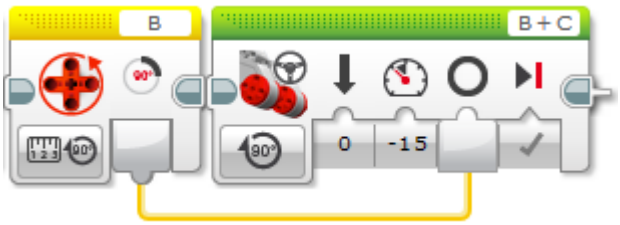
V režime *Measure* si teda vyberáme z troch rôznych druhov merania, pričom pre *Beacon* a *Remote* je potrebné nastaviť kanál zvolený na IR vysielači. V okamihu, keď program príde na tento príkaz, vykoná meranie a výsledok je k dispozícii v jednej z výstupných prípojok.

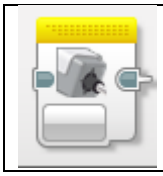
V režime *Compare* okrem druhu merania zadáme aj podmienku, ktorá sa hneď vyhodnotí a výsledok splnenia podmienky je dostupný ako logická hodnota. Podmienka sa zadáva analogicky ako v prípade ostatných senzorov.

	**Otáčkový senzor zabudovaný v motoroch**

Vo všetkých EV3 aj NXT motoroch sú zabudované otáčkové senzory, ktoré umožňujú regulovaný pohyb zadanou rýchlosťou a hlavne pohyb na požadované vzdialenosti nezávisle od aktuálneho stavu batérie (za rovnaký čas prejde robot s nabitou batériou aj dvojnásobnú vzdialenosť ako robot s batériou takmer vybitou). Hodnota otáčkového senzora môže byť v danom okamihu zosnímaná a ďalej spracovávaná alebo využitá.

V okamihu, keď program dobehne na tento príkaz zosníma hodnotu otáčkového senzora motora na zadanom porte. Počet otáčok (*Rotations*), resp. počet stupňov (*Degrees*) od posledného vynulovania senzora (*Reset*) môžeme vytiahnuť z výstupnej prípojky. Príklad ukazuje využitie hodnoty stupňov v pohybovom bloku:



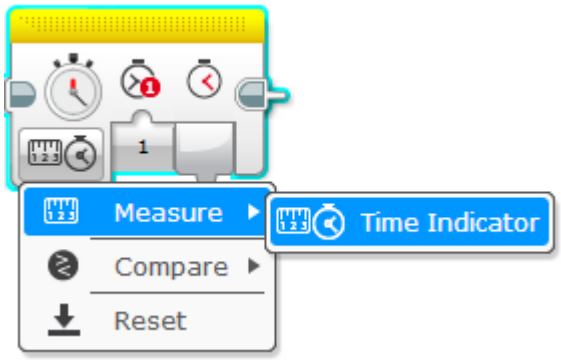
	**Tepelný senzor (extra doplnok)**

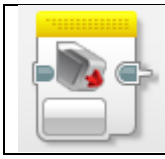
Tepelný senzor nie je súčasťou žiadnej základnej stavebnice, ale dá sa zaobstarať samostatne (9749). V okamihu, keď program dobehne na tento príkaz, zmeria aktuálnu teplotu v stupňoch Celzia alebo vo Farenheitoch a nameranú hodnotu dá k dispozícii cez výstupnú prípojku.

Režim *Compare* umožňuje zadať podmienku, ktorá sa hneď vyhodnotí, analogicky ako pri ostatných senzoroch.

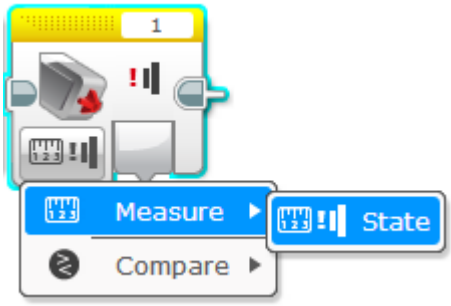
	**Časovač**

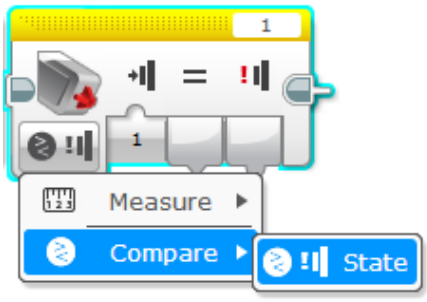
Žltý časovačový blok prečíta (*Measure*) alebo porovná so zadanou hodnotou (*Compare*) aktuálnu hodnotu počítadla zadaného časovača (1-8) v okamihu, keď program dobehne na tento príkaz a výsledok v sekundách poskytne formou výstupnej prípojky. Možnosť *Reset* vynuluje zvolený časovač 1-8. Časovače bežia (odpočítavajú čas) neustále, netreba ich štartovať.



	**Dotykový senzor**

Dotykový senzor môže byť iba v dvoch polohách: stlačený a pustený (nestlačený). Softvér však vie zareagovať aj na postupnosť úkonov – ak senzor bol najskôr stlačený a potom neskôr pustený. Nastavíme číslo portu, kde je senzor pripojený a v režime *Measure* na výstupnej prípojke získame hodnotu 0 – ak je pustený a 1 – ak je stlačený. V režime *Compare* môžeme stanoviť typ udalosti, ktorý chceme kontrolovať (0, 1, alebo 2 – bol stlačený a pustený) a na výstupných prípojkách máme k dispozícii aj logický výsledok porovnania aj aktuálne nameranú hodnotu. Dotykový senzor je vhodné použiť buď na riadenie – signalizáciu vstupu od používateľa, najmä v prípade, že tlačidlá na riadiacej jednotke sú nedostupné, alebo ako nárazník – do krížového otvoru sa jednoducho vsadí oska s namontovanou nárazníkovou plochou.

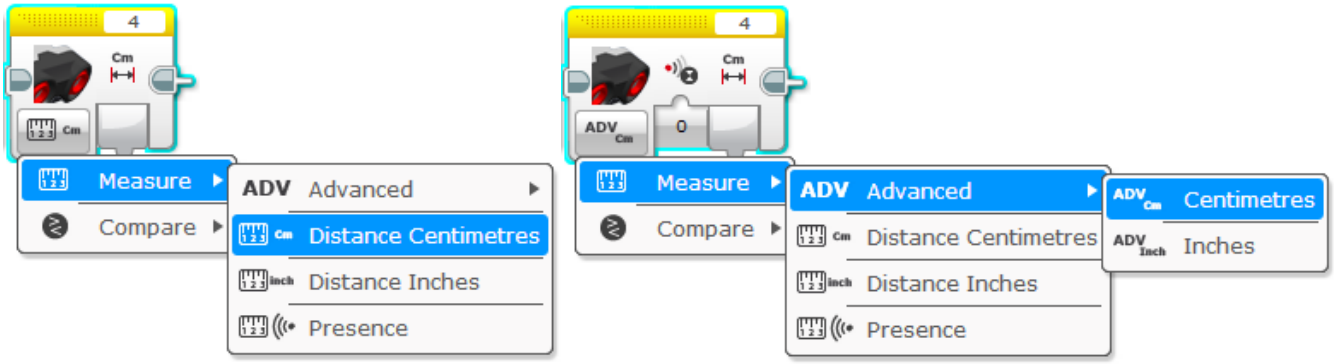




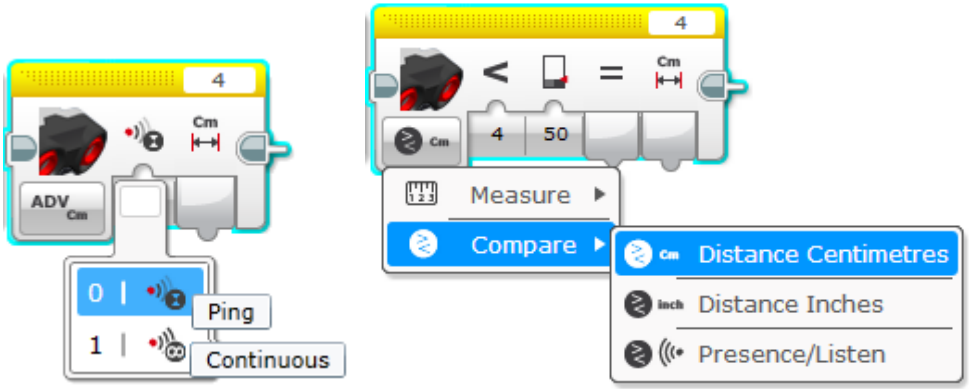
	**Ultrazvukový senzor na meranie vzdialenosti**

Ultrazvukový senzor vysiela zvukové signály na vysokej (pre človeka nepočuteľnej) frekvencii a meria čas, za ktorý sa zvuk po odrazení od prekážky vráti naspäť do senzora. Z nameraného času a známej rýchlosti šírenia zvuku vypočíta vzdialenosť k najbližšej prekážke v cm alebo v palcoch (*inch*). Signál sa v priestore šíri vpred

v oblasti rozbiehajúceho sa kužela. Plochy, ktoré sú voči senzoru natočené šikmo, nemusí zamerať spoľahlivo. Vpravo hore nastavíme port, na ktorom je senzor pripojený.



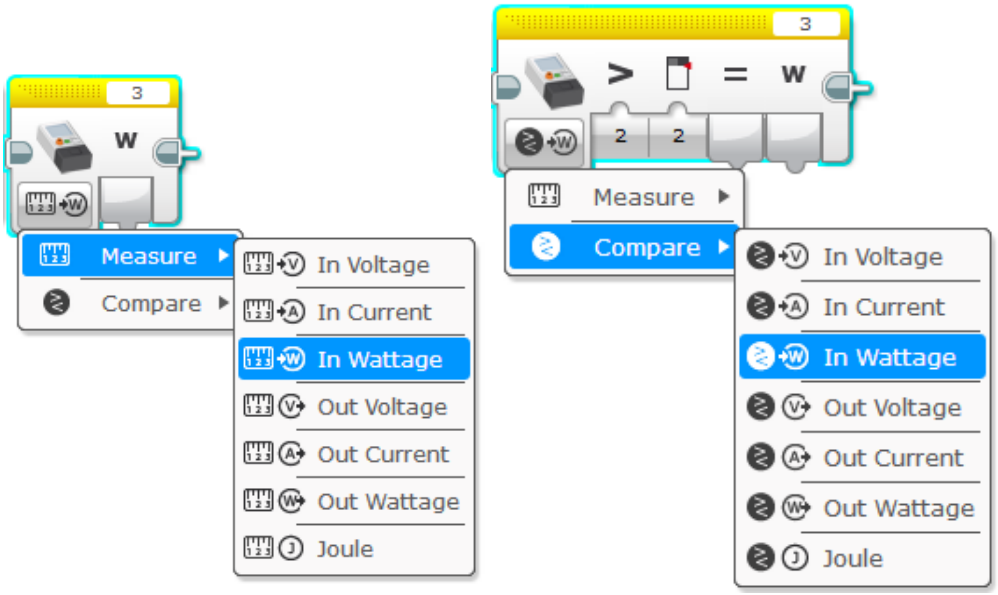
V režime *Measure – Distance Centimeters* je nameraná vzdialenosť k dispozícii vo výstupnej prípojke. V režime *Presence* senzor nevysiela žiaden signál, ale na výstupnej prípojke nastaví hodnotu pravdivý, ak zaznamená nejaký signál pochádzajúci z iného ultrazvukového senzora. Takýmto spôsobom sa môžu dva roboty navzájom zamerať. Režim *Advanced* umožňuje prepnúť bežne používaný sústavný signál na režim vysielania samostatných pulzov.



V režime *Compare* môžeme hneď otestovať, či je splnená určená podmienka a výsledok porovnania vytiahnuť z výstupnej prípojky označenej rovnítkom. Aj v tomto prípade je v inej prípojke k dispozícii samotná nameraná hodnota.

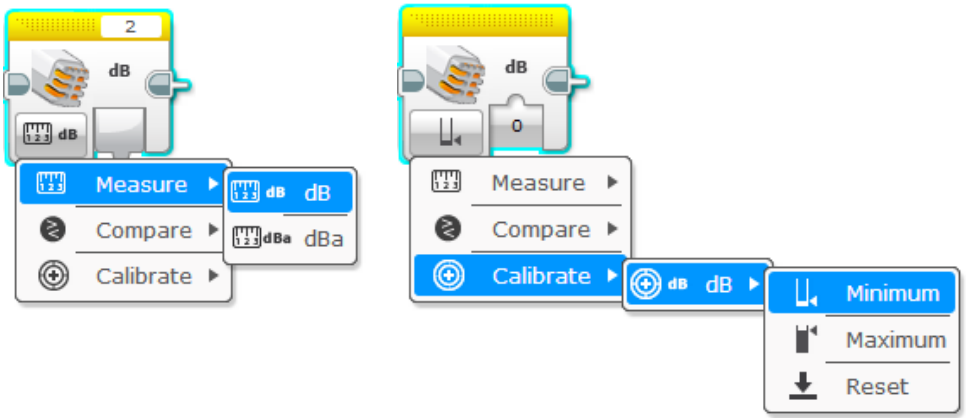
Čítanie z tehličky na meranie energie (samostatná súprava)

Energy meter je špeciálna meracia jednotka dodávaná so stavebnicou obnoviteľných zdrojov energie. Je vhodná na stavbu modelov pre rôzne fyzikálne pokusy. Dokáže zhromažďovať alebo vydávať zhromaždenú energiu. V režime *Measure* môžeme merať vstupné (dodávané) napätie, prúd a výkon a v prípade zapojenia nejakého spotrebiča (typicky motorček), môžeme tieto tri veličiny merať aj na výstupe. Množstvo zhromaždenej energie je k dispozícii v jednotkách Joule. V režime *Compare* hodnotu zmeranú v okamihu, keď program vykoná tento príkaz môžeme zároveň porovnať podľa nastavenej podmienky. Tento blok je potrebné downloadnuť zo stránky www.lego.com/en-us/mindstorms/downloads a doinštalovať cez menu Tools – Block Import Wizzard.



Zvukový senzor zo stavebnice NXT

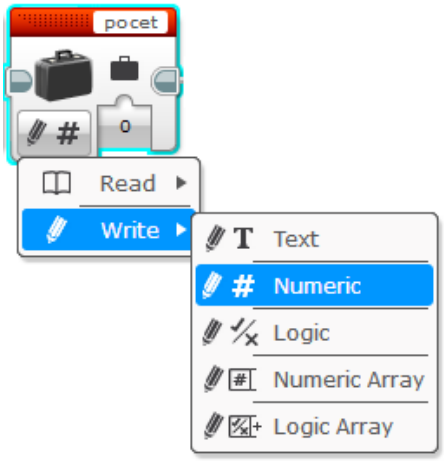
V predchádzajúcej generácii stavebníc (LEGO MINDSTORMS NXT) je dodávaný zvukový senzor, ktorý dokáže merať okamžitú intenzitu hluku v decibeloch. Pomocou neho robot môže reagovať na tlesknutie, alebo dupnutie, alebo vykonať nejakú činnosť v okamihu, keď sa zvýši hluk, nastane nejaká hlučná udalosť, alebo naopak, keď v prostredí niečo stíchne. Ako väčšina príkazov v tejto kategórii, aj tento dovoľuje buď merať (*Measure*) aktuálny hluk, alebo jeho hodnotu hneď aj porovnať podľa zadanej podmienky (*Compare*). Výsledok je k dispozícii v dvoch verziách: dB (decibel) alebo dBa – čo sú decibely preškálované podľa citlivosti ľudského (detského) ucha. Pre rozličné výšky tónov má ľudské ucho rozličnú citlivosť. V pravom hornom rohu ako vždy nastavujeme číslo portu, kde je senzor pripojený. Režim *Calibrate* umožňuje prestaviť rozsah a citlivosť merania určením krajných hodnôt *minimum*, *maximum*, pričom interval hlasitosti *minimum* – *maximum* sa v ďalších meraniach preškáluje tak, že hlasitosti v tomto intervale budú senzorom oznamované ako hodnoty v intervale 0 – 100.

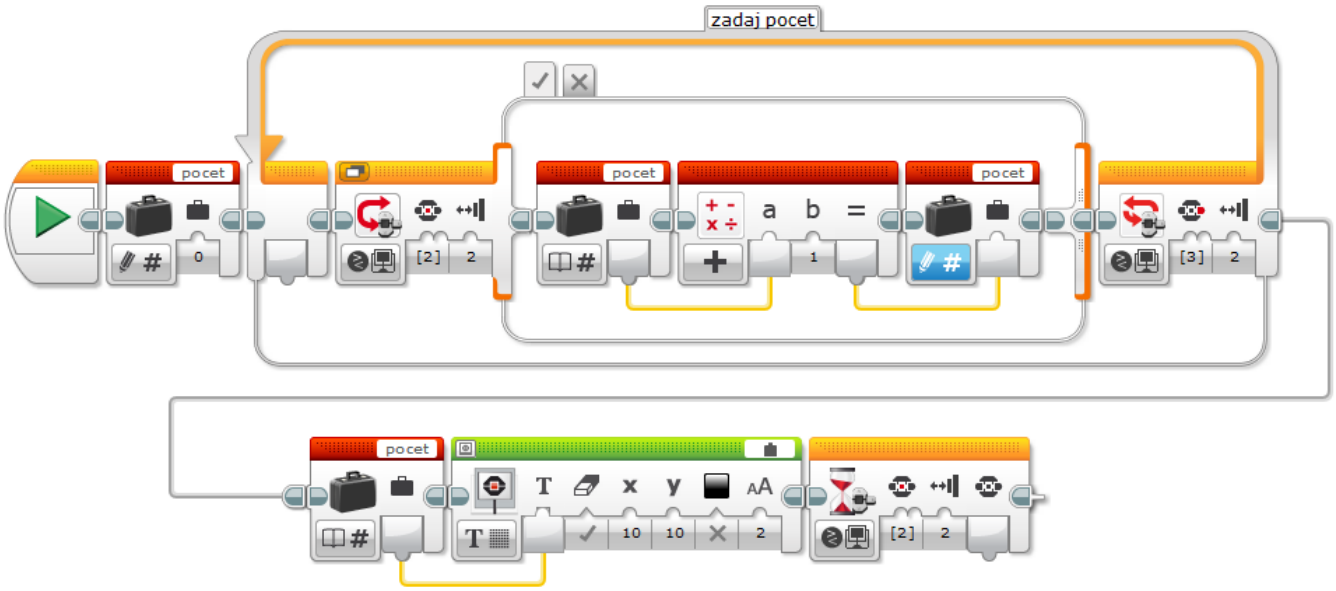


7. Práca s údajmi – premenné, polia, logické, aritmetické a textové operácie, porovnávanie, náhodné čísla

	**Kufrík – premenná: zápis alebo čítanie z premennej**

V prípadoch, keď si program potrebuje uložiť nejaký údaj pre neskoršie použitie (napríklad informáciu o tom, či už robot naložil, alebo nenaložil náklad), môžeme použiť premennú. Premenné si môžeme predstaviť ako kufríky, do ktorých sa dajú odkladať nejaké hodnoty. Premenné ako také v programe nikde nevidíme, ale pristupujeme k nim pomocou príkazu kufrík. V okamihu, keď sa príkaz kufrík vykoná, tak sa vtedy do premennej nejaká hodnota – ktorá je zadaná vo vstupnej prípojke uloží (zapíše, *Write*), alebo sa z nej vyberie (*Read*) – a je k dispozícii vo výstupnej prípojke. Každá premenná musí mať pevne stanovený typ, ktorý sa nedá zmeniť: buď obsahuje texty, čísla (*Numeric*), alebo logické hodnoty (*Logic*: pravdivý – *True*, alebo nepravdivý – *False*). Okrem týchto troch základných troch typov existujú zoznamy, resp. polia (*Array*) – ale len pre číselné a logické typy položiek. V jednej premennej typu zoznam môžeme uchovať dve alebo viacero hodnôt. Názov premennej vyplníme v pravom hornom rohu príkazu. Snažíme sa používať krátke, ale zmysluplné názvy premenných. Nasledujúci príklad počíta stlačenia stredného klávesu na riadiacej jednotke a po stlačení šípky vpravo vypíše celkový počet stlačení na displej:

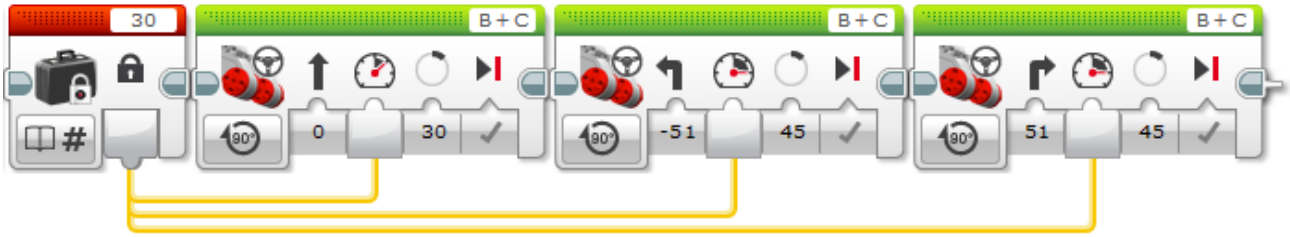




	**Kufrík – konštanta (ak nejakú hodnotu potrebujeme viackrát v programe, uložíme ju sem)**

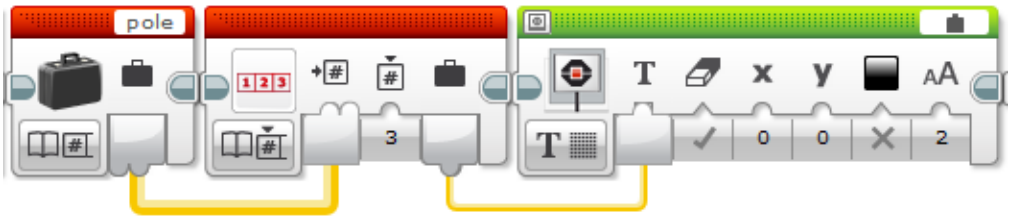
Počas ladenia programov môže byť praktické, keď hodnoty, ktoré používame na viacerých miestach v programe, zapíšeme iba na jednom mieste. Túto hodnotu na všetky potrebné miesta potom dovedieme dátovým drôtikom. Vyberieme typ hodnoty, v pravom hornom rohu vyplníme hodnotu. Ako konštantu možno

uviesť hodnoty základných typov (*Text*, *Numeric*, *Logic*) ale i polia. Napríklad môžeme ladiť rýchlosť motorov vo všetkých pohybových blokoch upravovaním jedinej konštanty:

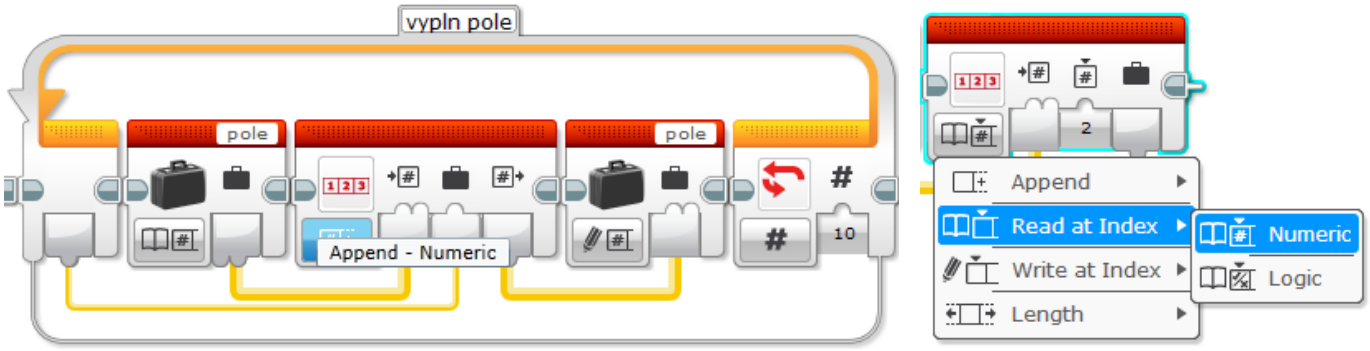


	**Práca s poľom (premenná typu pole schová v jednej premennej viacero údajov naraz)**

Premenná alebo konštanta môže mať typ pole (zoznam). V tom prípade ale príkazom kufrík nevieme priamo pristupovať do príslušnej položky poľa, ale iba pole ako celok z kufríka vybrať (prečítať) alebo uložiť (zapísať). Ak chceme v poli robiť úpravy – pridávať alebo odstraňovať položky, meniť ich hodnoty, alebo uložené hodnoty použiť, musíme použiť príkaz na prácu s poľom. Program na nasledujúcom obrázku zobrazí položku na 3. pozícii z poľa na displeji – najskôr sa celé pole prepošle z kufríka do príkazu na prácu s poľom, ktorý je v režime *Read at Index – Numeric*:



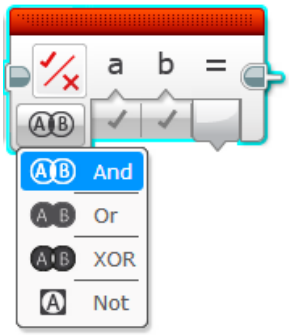
V druhom prípade sa pole postupne naplní hodnotami 0..9. Tu vidieť, že aby sme pole zmenili, musíme ho najskôr vybrať z kufríka, potom urobiť úpravu pomocou príkazu na prácu s poľom a napokon zmenené pole zasa odložiť do kufríka. V tomto prípade používame operáciu *Append – Numeric*, ktorá pridá číselnú hodnotu na koniec poľa:



Okrem použitých operácií *Append* a *Read at Index* príkaz obsahuje režimy *Write at Index* – zapíše hodnotu na zadanú pozíciu v poli a *Length* – zistí aktuálnu dĺžku poľa.

Logické operácie

Ak potrebujeme vyjadriť zložitejšiu podmienku, ktorá sa skladá z dvoch alebo viacerých porovnaní (napríklad: ak vypršal stanovený čas, alebo robot senzorom zaznamenal čiernu čiaru, má zastaviť), môžeme ich vyjadriť pomocou logických operácií: do bloku vstupujú dve logické hodnoty a vystupuje jedna výsledná logická hodnota:

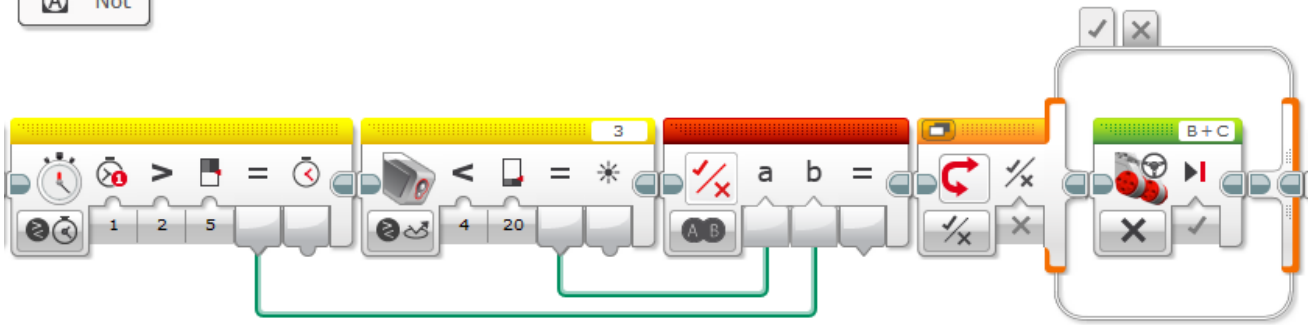


AND – podmienka je splnená, iba ak sú oba vstupy splnené

OR – podmienka je splnená, iba ak aspoň jeden vstup je splnený

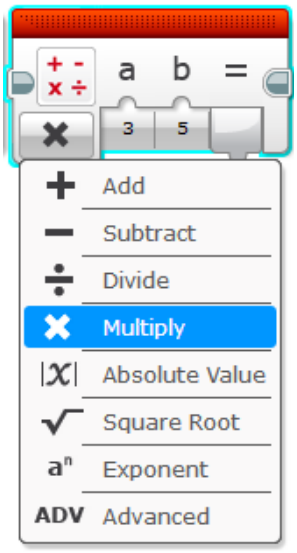
XOR – podmienka je splnená, iba ak práve jeden zo vstupov je splnený

Výnimkou je operácia *NOT*, ktorá má iba jeden vstup, podmienka je splnená iba vtedy, keď vstup nie je splnený**.** Obrázok zobrazuje podmienku opísanú v zátvorke vyššie.



Aritmetické operácie

Tento veľmi silný príkaz dovoľuje do programu zaradiť výpočty. Programátor má k dispozícii 7 základných aritmetických operácií:



S dvoma vstupmi:

Add – sčítanie a + b

Subtract – odčítanie a - b

Divide – delenie a / b

Multiply – násobenie a * b

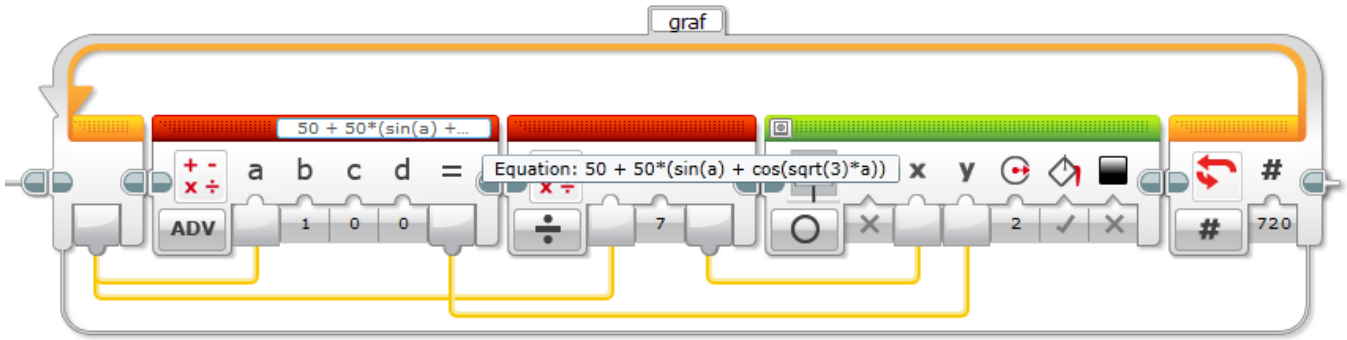
Exponent – umocnenie a ^ n

S jedným vstupom:

Absolute Value – absolútna hodnota (rovnaké číslo bez znamienka) |a|

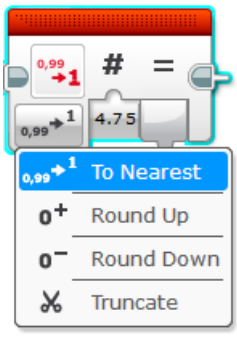
Square Root – druhá odmocnina sqrt(a)

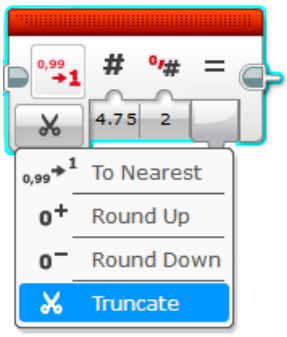
Posledný režim – *Advanced* – dovoľuje zadať zložitejší vzorec obsahujúci štyri premenné A, B, C, D a ľubovoľné matematické operátory a funkcie: +, -, *, /, % (zvyšok po delení), ^ (umocnenie), horná i dolná celá časť, zaokrúhlenie, absolútna hodnota, logaritmus so základom 10, prirodzený logaritmus, sin, cos, tan, asin, acos, atan, odmocnina. Nemusíme využiť všetky vstupné premenné.



Zaokrúhľovanie čísel

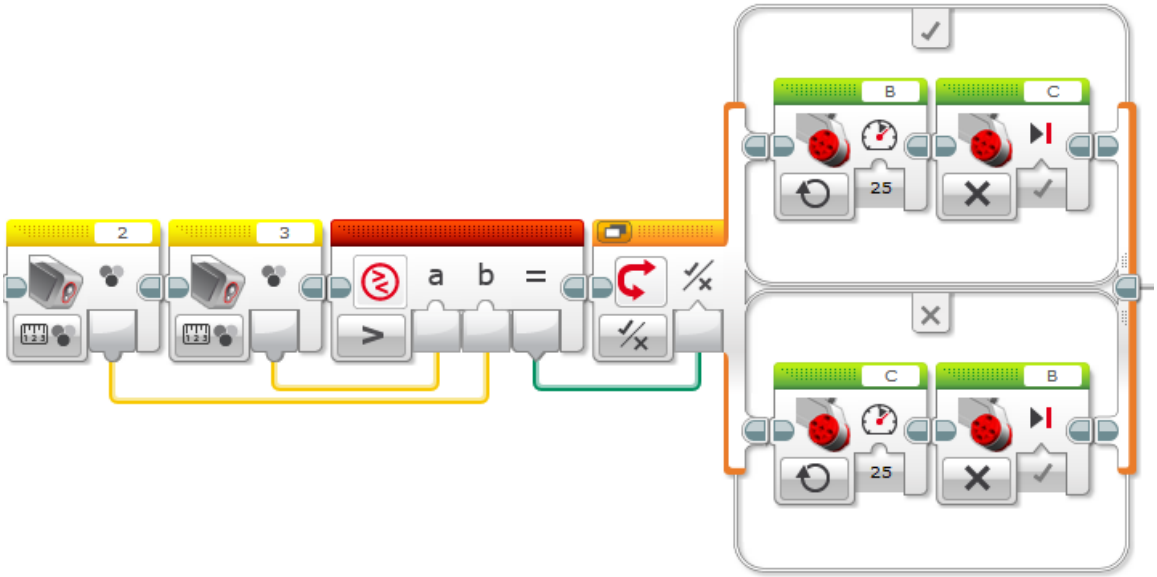
Tento príkaz s jednou vstupnou a jednou výstupnou prípojkou zaokrúhli desatinné číslo na celé číslo. Rozlišuje štyri rôzne spôsoby zaokrúhlenia: *To Nearest* – zaokrúhlenie na najbližšie celé číslo (napr. 2.3 => 2, 4.7 => 5), *Round Up* – horná celá časť (napr. 1.2 => 2, 2.5 => 3, -3.6 => -3), *Round Down* – dolná celá časť (napr. 4.2 => 4, 5.7 => 5, -4.2 => -5), výnimkou je štvrtý režim *Truncate* – odrezanie niektorých desatinných číslic, ktorý má dve vstupné prípojky – druhá prípojka udáva počet desatinných číslic, ktoré sa majú zachovať (napr. 2.3; 0 => 2, 5.943;2 =>5.94, -4.61271; 3 => -4.612).





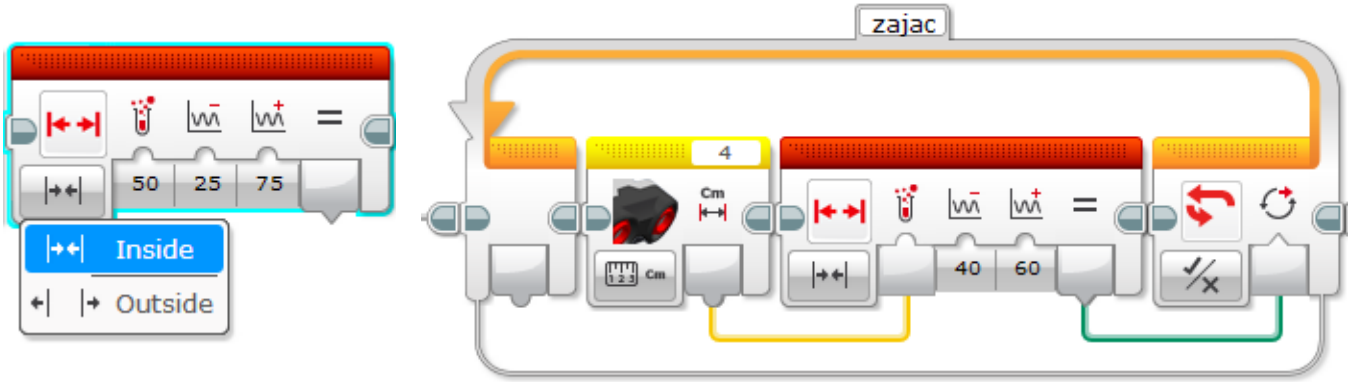
Porovnávanie čísel

Príkaz porovná dve rôzne číselné hodnoty a výsledok vo forme logickej hodnoty pripraví na výstupnej prípojke. Napríklad môžeme porovnať hodnoty dvoch farebných/svetelných senzorov a podľa toho vykonať pohyb rôznymi smermi:



	**Test na interval (číslo je vnútri alebo vonku intervalu OD – DO)**

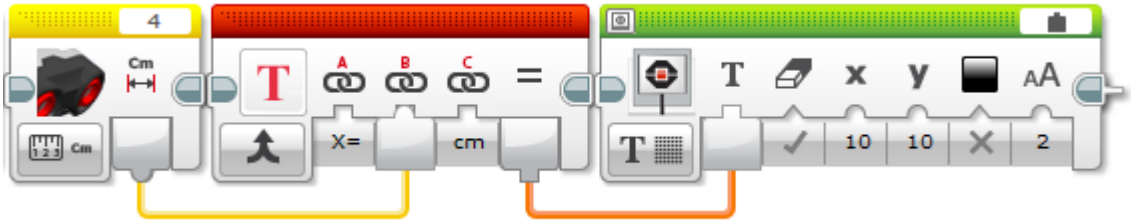
Zložitejšiu podmienku, ktorá otestuje, či je nejaká hodnota vnútri (*Inside*) alebo mimo rozsah (*Outside*) nejakého intervalu, by sme mohli naprogramovať pomocou dvoch porovnávacích blokov a logickej operácie AND, tento príkaz nám ten kúsok programu zjednoduší a zostruční:



V príklade na obrázku vpravo bude cyklus čakať, kým zajac, ktorého chce robot nakŕmiť, nepríde do intervalu 40-60 cm od senzora.

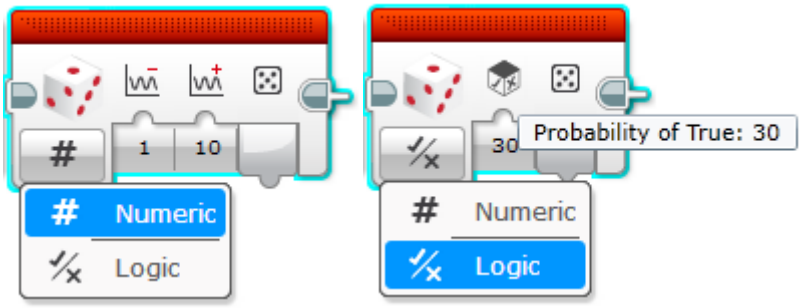
	**Textové operácie (lepenie viacerých textov dokopy)**

Tento praktický blok zjednodušuje prácu s textami, pričom dovoľuje spojiť tri texty do jedného súvislého textu. Hodí sa to napríklad pri vypisovaní výsledkov na displeji, alebo ich ukladanie do súboru:



	**Generovanie náhodných čísel a logických hodnôt**

Pomocou tohto užitočného príkazu môžeme robota naprogramovať tak, aby sa správal náhodne. V režime *Numeric* nastavíme číselný interval, z ktorého sa vygeneruje celé číslo dostupné na výstupnej prípojke. V režime *Logic* môžeme upraviť pravdepodobnosť P, že sa na výstupnej prípojke vygeneruje logický signál pravdivý (*True*). S pravdepodobnosťou 1-P sa vygeneruje signál nepravdivý (*False*).



8. Pokročilé – súbory, meranie, Bluetooth, priame riadenie výstupov a vstupov, stop programu, poznámka

	**Práca so súbormi**

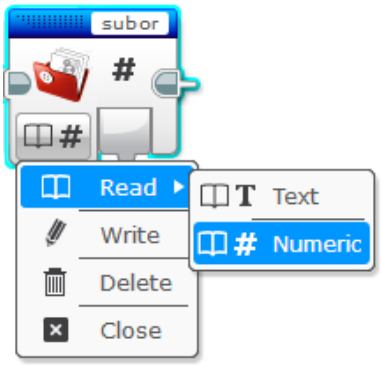
Riadiaca jednotka EV3 beží interne na systéme Linux (ktorý je natoľko zoštíhlený, aby vošiel do 64 MB RAM). Súčasťou Linuxu je aj súborový systém na 16 MB internom pamäťovom médiu typu flash. Všetky programy, zvuky, obrázky i automatické záznamy dát sa na tomto médiu ukladajú ako samostatné súbory. Programy, ktoré vytvára používateľ, môžu na tomto disku vytvárať súbory nové a zapisovať do nich, alebo otvárať dátové súbory, ktoré tam už sú vytvorené, alebo nakopírované cez dialóg dostupný z menu *Tools* – *Memory Browser*. Názvy súborov v príkazoch uvádzame bez prípony, systém im automaticky pridáva príponu ".txt". Príkaz na prácu so súbormi dovoľuje:

Delete – vymazať už existujúci súbor – to je vhodné vtedy, keď program nejaký súbor vytvára a spúšťame ho opakovane: pri každom novom spustení chceme súbor najskôr vymazať, inak by sa nové údaje pridávali na koniec už existujúceho súboru.

Read – čítať údaje z nejakého už existujúceho súboru – buď vo forme *Text* – vtedy sa načíta celý riadok textu, alebo vo forme *Number* – vtedy sa načíta ďalšia číselná hodnota z textového súboru

Write – zapísať údaje, čiže pridať na koniec nejakého súboru – každý pridaný text alebo číslo vytvorí v súbore nový riadok

Close – zatvoriť súbor, do ktorého sa zapisovalo, aby sa údaje správne uložili – na konci každej postupnosti zápisov by sa súbor mal zatvoriť.

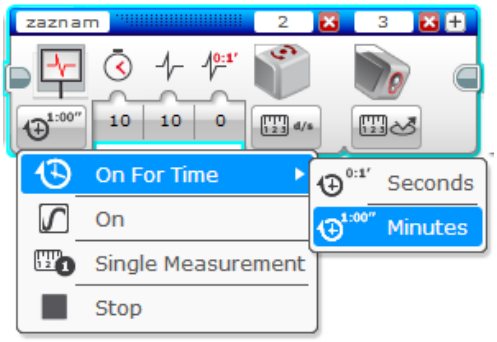


Vo všetkých blokoch v pravom hornom rohu uvedieme meno súboru (bez prípony .txt), prípadne zvolíme voľbu Wired – a potom meno súboru do príkazu doručíme cez vstupnú prípojku.

Príslušné bloky na čítanie a zapisovanie poskytujú zodpovedajúce výstupné, resp. vstupné prípojky.
Takto vytvorený súbor môžeme preniesť na PC a údaje ľahko zobraziť v tabuľke alebo grafe v tabuľkovom editore.

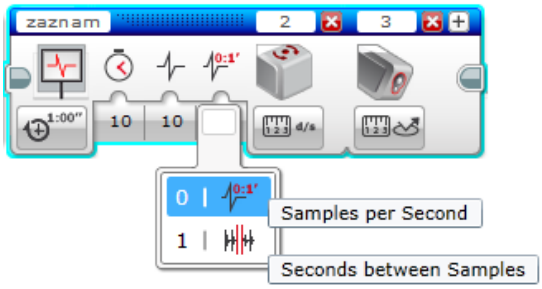
Záznam dát (meranie)

Jedno z podstatných vylepšení riadiacej jednotky EV3 oproti predchádzajúcej verzii NXT podľa spoločnosti LEGO Education je, že jednotka EV3 je tzv. “true datalogger”. To znamená, že sa dá používať na automatický záznam rozsiahlych dát zo senzorov. V prípade pripojenia 32 GB SD karty môže EV3 zaznamenávať dáta napr. ako meteorologická stanica aj niekoľko rokov. Dáta je možné zaznamenávať v programe priamo pomocou príkazu na prácu so súbormi opísanom vyššie, alebo naštartovať automatický zápis hodnôt zvolených senzorov vo zvolenej frekvencii, ktorý sa môže vykonávať v pozadí bežiaceho programu.



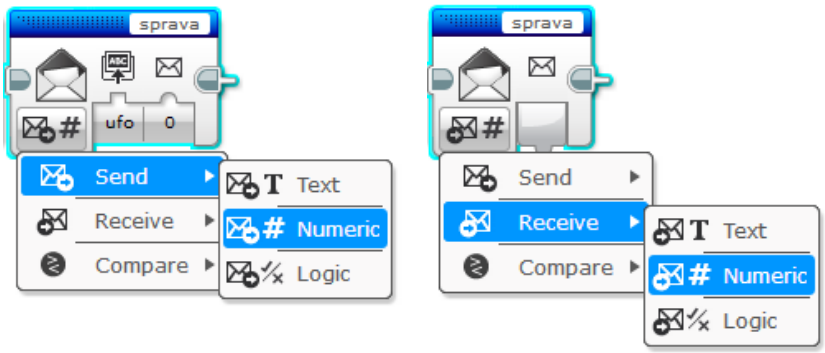
V ľavom hornom rohu zadáme názov záznamu, vyberieme režim: začať zaznamenávať (*On*), prestať zaznamenávať (*Off*), odštartovať zaznamenávanie na určitý čas (*On For Time* – v sekundách alebo v minutách), alebo *Single Measurement* – pridať na koniec záznamu jedno meranie.

Pomocou tlačidla ‘+’ (resp. pomocou ‘x’) v pravom hornom rohu môžeme popridávať (resp. zasa odobrať) senzory, ktoré sa majú zaznamenávať a nastaviť vstupné porty, na ktorých sú pripojené. V prípade záznamu na čas určíme v prvej prípojke počet sekúnd alebo minút a v druhej prípojke nastavíme počet záznamov za sekundu, resp. počet sekúnd, ktoré majú uplynúť medzi dvoma záznamami – podľa toho, ktorý režim v tretej prípojke zvolíme:



Posielanie správ cez Bluetooth

Tento príkaz používame na posielanie (*Send*) a prípadne na prímanie (*Receive*) správ z druhej riadiacej jednotky, ktorá je pripojená cez BlueTooth. Alternatívny spôsob prijatia správy je cez podmienkový príkaz, prípadne cyklus. Režim *Compare* umožňuje prijatú hodnotu porovnať podľa určenej podmienky. Dôležité je nastaviť rovnaký nadpis správy (v pravom hornom rohu) aj na odosielajúcej aj na prímajúcej jednotke.



	**Nastavovanie spojenia Bluetooth priamo z programu (dá sa aj ručne priamo na kocke)**

Komunikácia s druhou riadiacou jednotkou EV3 cez spojenie BT je možná iba potom, čo sa jedna riadiaca jednotka (iniciujúca spojenie) pripojí na druhú. Bežne sa to robí priamo z menu na riadiacej jednotke *Settings* – *BlueTooth* - *Connections* – Search a potom potvrdit pripojenie a rovnaky kód na oboch jednotkách. Ak chceme pripojenie na inú riadiacu jednotku cez BT urobiť priamo z programu, môžeme použiť verziu "*Initiate*" tohto príkazu a do prípojky zadať meno pripájanej jednotky.

Na odpojenie použijeme analogický príkaz *Close* resp. *Clear*.

Pre ušetrenie energie je BT možné z programu príkazom *Off* vypnúť a príkazom *On* zasa zapnúť.

	**Nastavenie časového intervalu automatického vypínania**

V nastaveniach EV3 je možné stanoviť čas v minútach, po ktorých sa riadiaca jednotka automaticky vypne (bežne po 30 minútach). Ak bežiaci program narazí na tento príkaz, počítadlo merajúce čas do vypnutia sa vynuluje. Týmto spôsobom môžeme predísť tomu, aby sa riadiaca jednotka vypla (pokiaľ má stále dosť energie z batérií, resp. napájacieho adaptéra). Výstupom príkazu je čas zostávajúci do vypnutia v sekundách.

	**Priama (surová, nespracovaná) hodnota zo senzora**

Hodnoty zo senzorov sú preškálovávané podľa typu zvoleného senzora. V prípade, že máme nejaký vlastný alebo iný neštandardný senzor, alebo chceme použiť hodnotu zo senzora bez toho, aby bol preškálovaný, tak tento príkaz v okamihu, keď ho program vykoná, prečíta nespracovanú hodnotu zo senzora v rozsahu 0-1023 a dá ju k dispozícii do výstupnej prípojky.

	**Priamy (neregulovaný) výstup na motor**

Všetky zelené pohybové bloky využívajú otáčkové senzory na to, aby regulovali požadovanú rýchlosť motora. Regulovanie samotného napätia nestačí na udržanie požadovanej rýchlosti, pretože motor môže byť rôzne zaťažený (rôzna hmotnosť robota, veľkosť kolies, naklonená rovina smerom nahor/nadol). V niektorých prípadoch (napríklad rozličné fyzikálne experimenty) ale potrebujeme na výstupnom porte nastaviť konkrétnu silu (PWM signálom regulované napätie) – treba si však uvedomiť, že potom bude výkon/rýchlosť závisieť od aktuálneho stavu nabitia batérie.

Na tomto bloku nastavujeme iba výstupný port a percento výstupného napätia. Tento príkaz sa vykoná okamžite – na motor (resp. výstupný port) sa nastaví nová hodnota a program pokračuje ďalším príkazom.

	**Obrátenie otáčania neregulovaného motora**

Tento príkaz umožňuje zmeniť smer otáčania motora, ktorý sme roztočili príkazom na priamy neregulovaný pohyb motora. Motor sa bude otáčať naďalej rovnakou rýchlosťou, ale v opačnom smere. Príkaz má vplyv na všetky príkazy ovládajúce motor – zelené aj modré. Spôsobí, že od teraz sa budú všetky pohyby na danom motore vykonávať opačným smerom, ako je určené v ľubovoľnom pohybovom príkaze. Príkaz má jeden parameter, ktorý udáva, či motor má byť, alebo nemá byť otočený (*True* – motor bude odteraz otočený, *False* – motor bude fungovať normálne). Dvojnásobné otočenie motora nespôsobí jeho prepnutie do normálneho smeru. Typické využitie tohto príkazu je v situácii, keď sú motory v robotovi namontované tak, že pohyb robota vpred spôsobujú záporné rychlosti otáčania motora. Motoru na začiatku programu týmto príkazom jednoducho zmeníme smer a potom v celom programe používame kladné rýchlosti pre pohyb robota vpred.

	**Zastavenie celého programu na určitom mieste uprostred programu**

V prípade, že na nejakom mieste – napr. v nejakej vetve podmienky potrebujeme za určitých okolností zastaviť pokračovanie programu, použijeme tento príkaz.

	**Komentár**

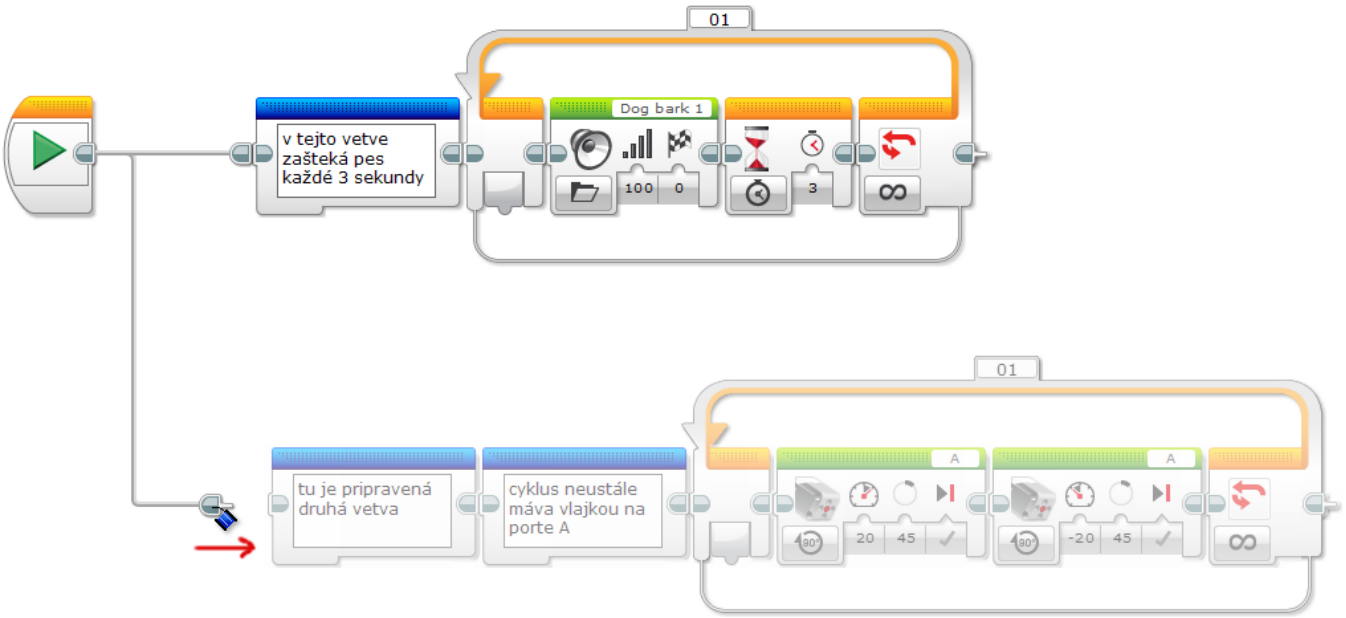
Pomocou tohto veľmi užitočného príkazu môžeme do programu na ľubovoľnom mieste zaradiť akýkoľvek textový komentár. Komentáre sa dajú vytvárať aj pomocou funkcie "*Comment*" s rovnakou ikonou ako označuje tento príkaz v toolbare. Výhodnou príkazu príkazu komentár je, že sa posúva v programe počas editovania programu, zatiaľ čo plávajúci komentár vytvorený cez toolbar zostáva na mieste. Preto plávajúce komentáre pridávame až do hotových programov.

Paralelné vykonávanie príkazov

Programovací jazyk pre EV3 je výborným jazykom na zoznámenie sa so všetkými základnými princípmi programovania:

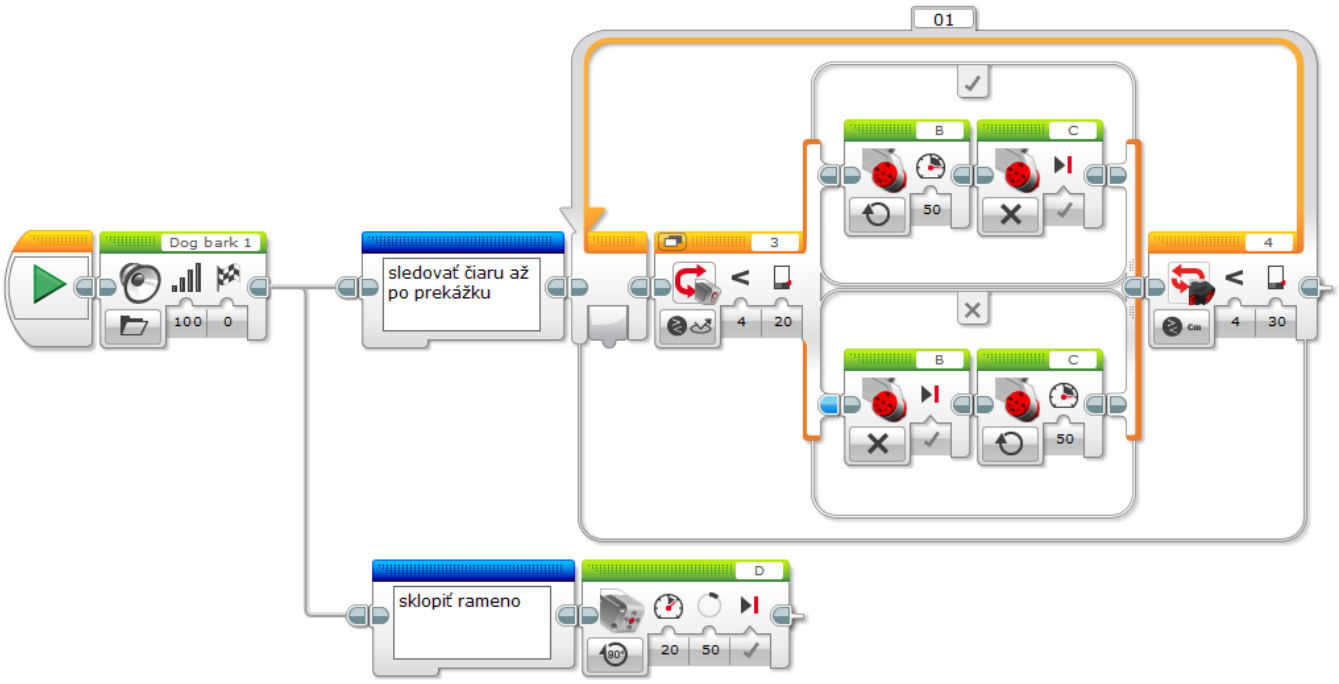
- Skladanie programov z postupností príkazov (programový tok)
- Základné riadiace štruktúry: podmienky, cykly
- Premenné a práca s údajmi medzi jednotlivými príkazmi (dátový tok)
- Práca s poliami
- Práca so súbormi

Okrem toho je v ňom aj názorne a zrozumiteľne realizovaný princíp paralelného programovania – programátor môže vytvoriť dve alebo viac paralelne vykonávaných vetiev programu, ktoré sa vykonávajú súčasne. Novú paralelnú vetvu vytvoríme duplicitným vytiahnutím prípojky programového toku a pripojením na druhú – paralelnú vetvu programu. Najskôr si pripravíme druhú vetvu s aspoň jedným príkazom a z predchádzajúceho bloku – za ktorým nastáva rozdvojenie – vytiahneme prepojku programového toku a zapojíme do prvého príkazu novej vetvy:



Až v okamihu, keď sa druhá vetva naplno zafarbí, je do programu pripojená. Rozdvojenie môže nastať za ľubovoľným príkazom, nielen ako je zobrazené na obrázku – na začiatku za štartovným blokom. Ak niektorá paralelná vetva skončí, program vo všetkých ostatných vetvách normálne pokračuje ďalej. Jednotlivé vetvy sa môžu synchronizovať pomocou premenných – všetky premenné sú viditeľné a upravovateľné vo všetkých vetvách. Typické použitie paralelizmov je na súčasné nezávislé riadenie rozličných motorov. Prvá vetva sa môže starať napríklad o sledovanie čiary pomocou motorov B, C a druhá vetva môže priebežne sklápať rameno, ktorým robot naloží náklad.

Riadenie tých istých motorov z rozličných vetiev programu naraz sa neodporúča – výsledky nie sú predvídateľné.



Riešené úlohy

Prehľad všetkých úloh:

1. stvorec - pohyb robota do štvorca
2. osmicka - pohyb robota v tvare osmičky
3. prekazka - robot sa rozbehne a pred prekážkou zastane
4. prieskumnik - robot náhodne preskúmava miestnosť bez narážania do prekážok
5. ciara - sledovanie čiary pomocou čakacích blokov
6. ciara pip - zaznamenávanie čiar zvukovým signálom, cez ktoré robot prechádza
7. sleduj_s_cakanim - jednoduché sledovanie čiary pomocou čakacích blokov - pohyb vždy len 1 motorom
8. sleduj_ciaru2 - jednoduché sledovanie čiary pomocou podmienky, pred prekážkou zastane
9. ustup - pohyb po čiare, na prekážkach robot čaká a vydáva zvuk
10. zobraz_otacky - príklad na zobrazovanie hodnoty zo senzora na displeji
11. jednadruha - netradičný spôsob sledovania čiary
12. tam a spat - meranie vzdialenosti pomocou otáčkových senzorov
13. meranie vzdialenosti - meranie a výpočet - pohyb k prekážke
14. chod za svetlom - robot sa pomocou dvoch farebných senzorov pohybuje za zdrojom svetla
15. sleduj_ciaru - sledovanie čiary pomocou dvoch senzorov - príklad bloku s argumentom
16. tri_ultrazvuky - robot sa pohybuje za najbližšou prekážkou - porovnávacie a logické bloky
17. zigzag_stavy - sledovanie čiary spôsobom zig-zag a využitie stavového automatu
18. ciara pip aj na konci - počítanie čiar (využitie premenných)
19. turista - príbeh o turistovi, ktorý ide na stanicu, má zabočiť na zadanej odbočke
20. menu - príklad ako vytvoriť vlastné štartovacie menu na riadiacej jednotke
21. flasa - fyzikálny experiment o priemernej rýchlosti priamočiareho pohybu
22. dve_veci_naraz - robot sa pohybuje a súčasne máva vlajkou (paralelné procesy)
23. pipaj_ked_blika - príklad na využitie paralelných procesov - rozpoznáva blikajúce a stále svetlo
24. zaznam - príklad zápisu hodnôt (ukadá sa pohyb motorom)
    zopakovanie - prehrá zaznamenaný pohyb (príklad čítania zo súboru)
25. zaznam_stvorec - príklad využitia datalogu
26. zaznam_ultra - príklad na záznam hodnôt z dvoch senzorov do súboru
27. btovladac, - diaľkové ovládanie cez BT - ovládač (riadiaca jednotka do ruky)
    btriadeny - diaľkové ovládanie cez BT - prímač (riadený robot)

Zadania úloh

1. **Štvorec**: Využitím zelených pohybových blokov naprogramujte robota, aby sa pohyboval čo najpresnejšie do štvorca. Rozhodnite sa, či sa v rohoch bude robot otáčať na mieste, alebo okolo vnútorného kolieska, prípadne si vyskúšajte obe možnosti. Riesenie: stvorec.

2. **Osmička:** Využitím zelených pohybových blokov naprogramujte robota, aby vykonával opakovaný pohyb v tvare osmičky. Najskôr si rozmyslite, z koľkých pohybov a akého tvaru sa celkový osmičkový pohyb bude skladať. Riešenie: osmicka.

3. **Prekážka:** Naprogramujte robota, aby sa rozbehol v pred a na prekážke sa zastavil. Riešenie: prekazka.

4. **Prieskumník:** Robot má vykonávať nasledujúci opakovaný pohyb: rozbehne sa a pohybuje sa vpred, až kým nepríde k prekážke. Tam sa zacúvaním otočí do iného smeru a pokračuje zasa priamo vpred. Riešenie: prieskumnik.

5. **Čiara:** Naprogramujte robota, aby sledoval čiernu čiaru pomocou jedného farebného/svetelného senzora. Čiara sa môže zatáčať vľavo i vpravo. Využite cyklus a čakacie bloky so svetelným senzorom v režime merania odrazeného svetla, na pohyb použite zelený pohybový blok pre dva motory. Riešenie: ciara.

6. **Pípanie na čiarach:** Naprogramujte robota, aby sa pohyboval vpred až k prekážke. Počas pohybu zapíska na každej čiare, cez ktorú prejde. Riešenie: ciara pip.

7. **Čiara s jedným motorom:** Naprogramujte robota, aby sledoval čiernu čiaru pomocou jedného farebného/svetelného senzora. Čiara sa môže zatáčať vľavo i vpravo. Využite cyklus a čakacie bloky so svetelným senzorom v režime merania odrazeného svetla, na pohyb použite zelený pohybový blok pre jeden motor – robot sa má pohybovať naraz vždy len jedným motorom. Riešenie: ciara_s_cakanim.

8. **Čiara s podmienkou:** Naprogramujte robota, aby sledoval čiernu čiaru, pomocou jedného farebného/svetelného senzora. Čiara sa môže zatáčať vľavo i vpravo. Využite cyklus a podmienkový blok so svetelným senzorom v režime merania odrazeného svetla. Robot pred prekážkou prestane sledovať čiaru a zastane. Riešenie: sleduj_ciaru_2.

9. **Opatrné sledovanie čiary:** Naprogramujte robota, aby sledoval čiernu čiaru pomocou jedného farebného/svetelného senzora. Čiara sa môže zatáčať vľavo i vpravo. Keď sa na čiare objaví prekážka (zaznamenaná ultrazvukovým senzorom), robot zastane a vydá zvukový signál. Po odstránení prekážky pokračuje v sledovaní čiary. Riešenie: ustup.

10. **Zobrazenie otáčok:** Vytvorte program, ktorý neustále na displeji zobrazuje hodnotu otáčkového senzora. Riešenie: zobraz_otacky.

11. **Čiara skusmo:** Naprogramujte robota, aby sledoval čiernu čiaru pomocou jedného svetelného senzora, ktorá sa môže zatáčať vľavo i vpravo. Keď je na čiare, má sa pohybovať priamo vpred, keď vyjde z čiary von, skúsi ju nájsť na jednej strane, ak tam nie je, skúsi na druhej strane. Riešenie: jednadruha.

12. **Tam a naspäť:** Naprogramujte robota, aby sa pohyboval vpred až ku stene a odtiaľ aby zacúval naspäť tak, aby sa vrátil na pôvodné miesto. Využite otáčkový senzor. Riešenie: tam a spat.

13. **K prekážke:** Naprogramujte robota, aby odmeral vzdialenosť ultrazvukovým senzorom k prekážke. Na základe nameranej vzdialenosti a polomeru kolieska vypočíta počet otáčok o ktoré sa má posunúť vpred. Približne 10 cm pred prekážkou zastane. Riešenie: meranie vzdialenosti.

14. **Choď za svetlom:** Naprogramujte robota, aby pomocou dvoch farebných senzorov otočených smerom vpred a nastavených v režime merania rozptýlenia svetla jazdil za svetlom – neustále sa pohybuje vpred a zatáča vľavo alebo vpravo podľa toho, z ktorej strany svieti svetlo silnejšie. Riešenie: chod za svetlom.

15. **Sleduj čiaru:** Naprogramujte sledovanie čiary pomocou dvoch svetelných senzorov, vytvorte blok, ktorý ako argument dostane rýchlosť, ktorou sa robot pri sledovaní čiary má pohybovať. sleduj_ciaru.

16. **Sleduj človeka:** Vytvorte program pre robota s tromi ultrazvukovými senzormi, ktoré sa dívajú vpred pod rozličným uhlom tak, aby sa robot pohyboval vždy za najbližšou prekážkou. Využite porovnávacie a logické bloky. Riešenie: tri_ultrazvuky.

17. **Čiara zig-zag:** Naprogramujte sledovanie čiary pomocou jedného senzora tak, že robot bude počas sledovania čiary striedavo prechádzať celkom na jednu a potom na druhú stranu čiary. Využite premennú, v ktorej si bude pamätať jeden zo štyroch stavov: vchádzam do čiary sprava, vychadzam do čiary smerom vľavo, vchádzam do čiary zľava, vychádzam z čiary smerom vpravo. Riešenie: zigzag_stavy.

18. **Počítanie čiar:** Upravte program pípanie na čiarach tak, aby robot na konci pred prekážkou zastal a znovu zapískal presne toľkokrát, koľko čiar od štartu prekrižoval. Využité premennú. Riesenie: ciara pip aj na konci

19. **Turista ide na stanicu:** Turista je na dlhej ulici (na začiatku dlhej čiernej čiary), z ktorej kolmo vychádza viacero odbočiek vpravo. Jedna z nich vedie na železničnú stanicu, ale turista nevie, ktorá to je. Preto dostane pomoc od miestneho obyvateľa: po odštartovaní stlačí jeho dotykový senzor toľkokrát, koľkou odbočkou má robot zatočiť vpravo. Potom sa robot vydá sledovať rovnú dlhú čiaru a počíta odbočky vpravo. Na tej správnej zatočí a sleduje ju, až kým nepríde na stanicu (k prekážke), tam zastane. Riešenie: turista.

20. **Vlastné menu:** Niekedy potrebujeme šikovne štartovať rozličné programy z riadiacej jednotky. Namiesto komplikovaného výberu programov cez štandardné menu EV3 ich môžeme všetky spojiť do jedného programu a naprogramovať si vlastné menu, kde si šípkami na riadiacej jednotke vyberáme, ktorý program chceme spustiť a stredným tlačidlom ho odštartujeme. Naprogramujte si vlastné univerzálne menu na štartovanie zopár jednoduchých programov. Riešenie: menu.

21. **Priemerná rýchlosť:** Automobil sa pohybuje rovnomerným pohybom rýchlosťou 40 km/h na trase 10 km, potom zrýchli na 60 km/h a pohybuje sa tak ďalších 10 km. Aká bola jeho priemerná rýchlosť? Na túto otázku môžeme nájsť odpoveď experimentálne: vždy, keď robot zbadá pomocou ultrazvukového senzora fľašu napravo od jeho dráhy, bude vedieť, že prechádza do ďalšieho úseku a má zmeniť rýchlosť pohybu. Časovačom meria uplynutý čas a z neho pomocou matematických výpočtov vypočítava celkovú priemernú rýchlosť. Riešenie: flasa.

22. **Súčasné akcie:** Naprogramujte robota, ktorý sa pomocou gyroskopického senzora pohybuje do štvorca a súčasne tretím motorom neustále máva vlajkou. Využite paralelné procesy. Riešenie: dve_veci_naraz.

23. **Blikanie:** Naprogramujte robota, ktorý pomocou svetelného senzora bude vedieť rozlíšiť medzi sústavne svietiacim a blikajúcim svetlom. V prípade, že zaznamená blikajúce svetlo, bude vydávať zvukový signál. Využite paralelné procesy. Riešenie: pipaj_ked_blika.

24. **Zaznam pohybu:** Naprogramujte program, ktorý zaznamenáva pohyb motora do súboru. V druhom programe bude čítať zaznamenaný pohyb zo súboru a otáčať kolesom podľa zaznamenaných hodnôt. Riešenie: zaznam, zopakovanie.

25. **Datalog:** Využite záznam do datalogu, aby ste si mohli zobraziť postupne sa meniace hodnoty otáčkových senzorov pri pohybe robota do štvorca (vonkajší motor prejde väčšiu dráhu a na grafe sa to pekne zobrazí). Riesenie: zaznam_stvorec.

26. **Záznam dvoch senzorov:** Napíšte program, ktorý do súboru bude zaznamenávať dve hodnoty: vzdialenosť nameranú ultrazvukovým senzorom a hodnotu otáčkového senzora. Použite blok na spájanie textu, aby bola dvojica hodnôt vždy uložená do toho istého riadku. Pokúste sa výsledok zobraziť v grafe pomocou nejakého tabuľkového editora (Excel a pod.). Riešenie: zaznam_ultra.

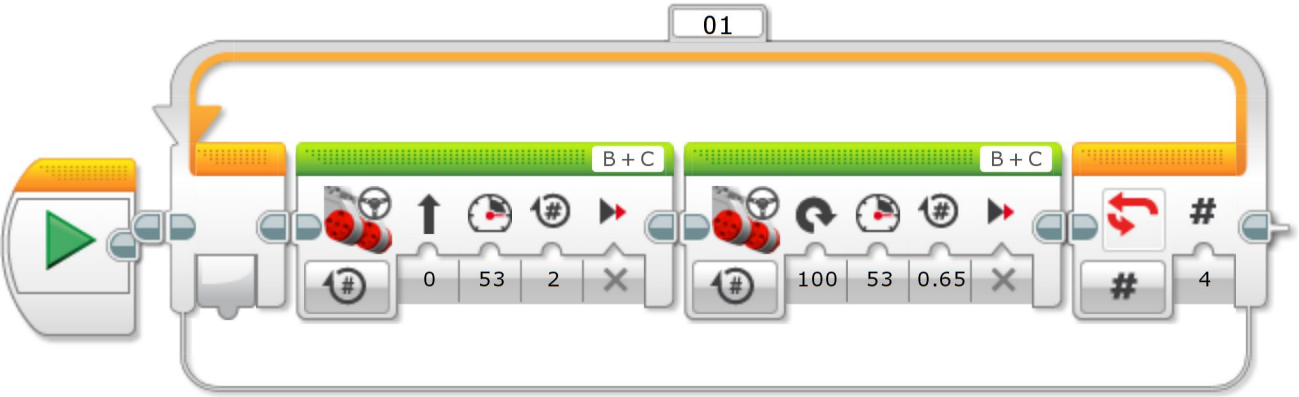
27. **Ďiaľkové riadenie cez BlueTooth**: Naprogramujte jednu riadiacu jednotku, aby vysielala do druhej riadiacej jednotky signály cez spojenie BlueTooth o tom aké tlačidlá používateľ stĺača. Druhá riadiaca jednotka bude robot, ktorý sa podľa prijatých správ bude pohybovať – vpred, vzad, vľavo, vpravo, zastaviť. Riešenie: btovladac,
btriadeny.

Riešenia úloh

V jednotlivých riešeniach sme umiestnili komentáre, ktoré by čitateľovi mohli pomôcť lepšie porozumieť programu a využitiu jednotlivých príkazov.

poznámka: ^^^^^

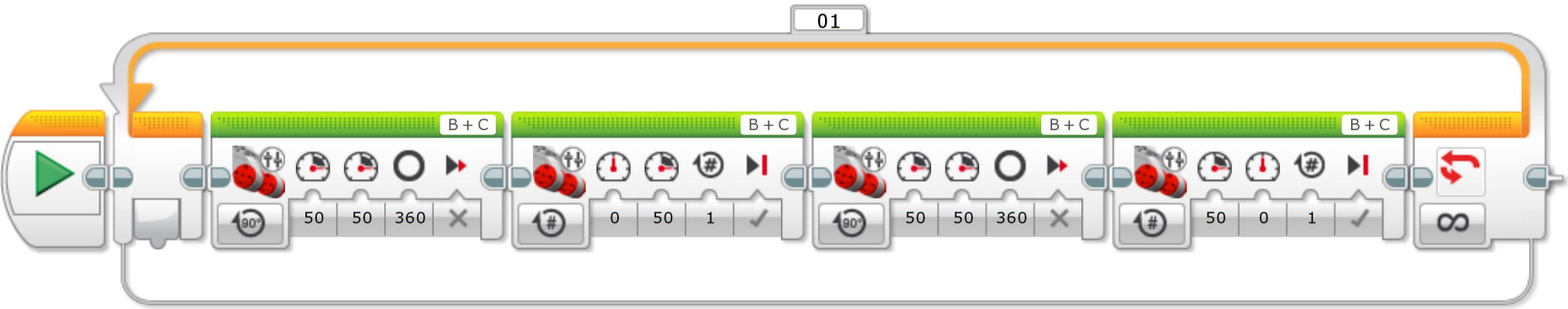
všimnite si, že v projekte je viacero programov, ako vôjde na hornú lištu - sú dostupné cez tlačidlo s ikonou dokumentu s malou šípkou nadol...



úloha: naprogramujte jednoduchý pohyb robota do štvorca

robot 4-krát zopakuje priamy pohyb vpred a otáčanie na mieste približne o 90 stupňov.

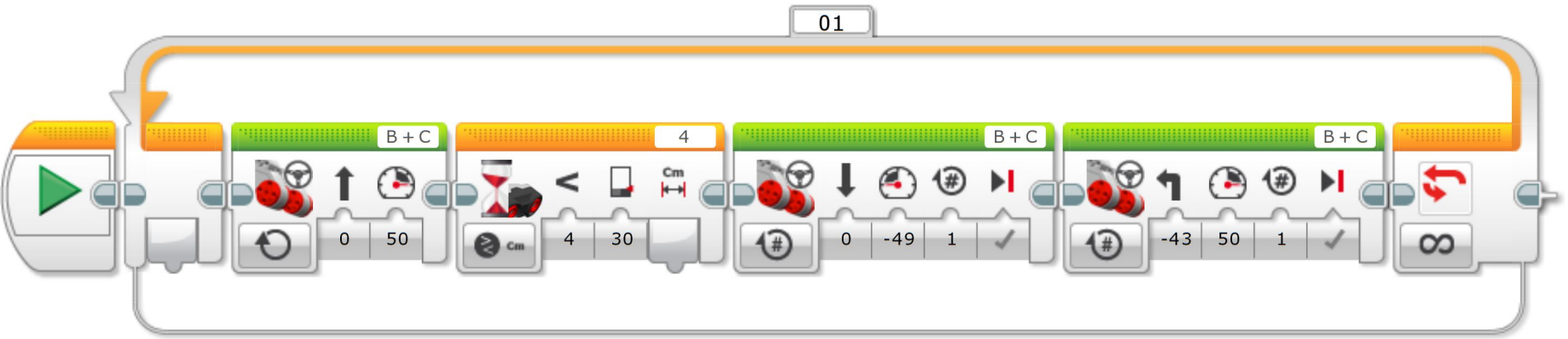
Či 0.65 otáčky výjde na 90 stupňov závisí od veľkosti koliesok a vzdialenosti medzi ľavým a pravým kolieskom.



úloha: naprogramujte pohyb robota do osmičky

v tomto jednoduchom riešení sa robot pohybuje striedavo vpred a striedavo sa otáča len jedným (raz ľavým a raz pravým) kolesom, čím vytvára oblúky osmičky.

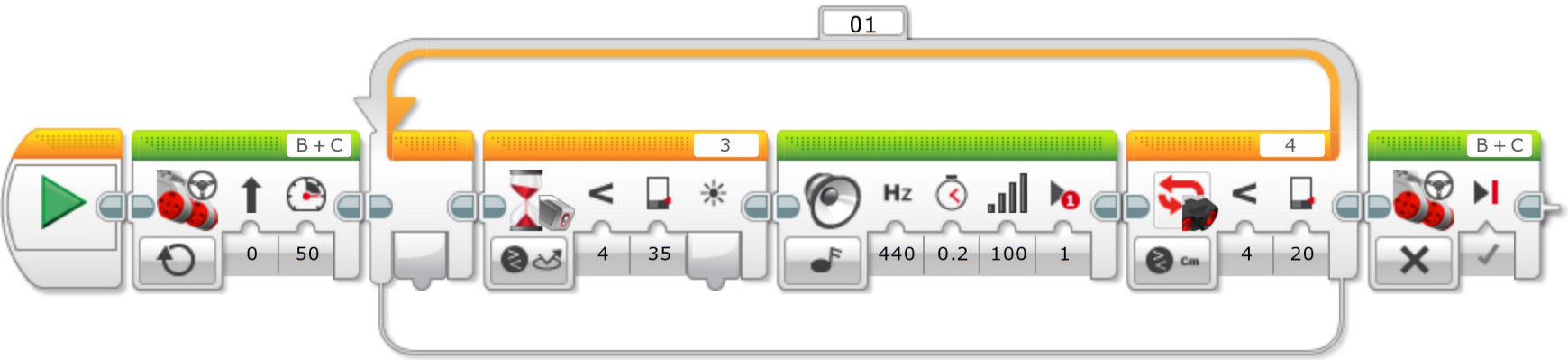
úloha: program sa pohybuje vpred a pred prekážkou zastane



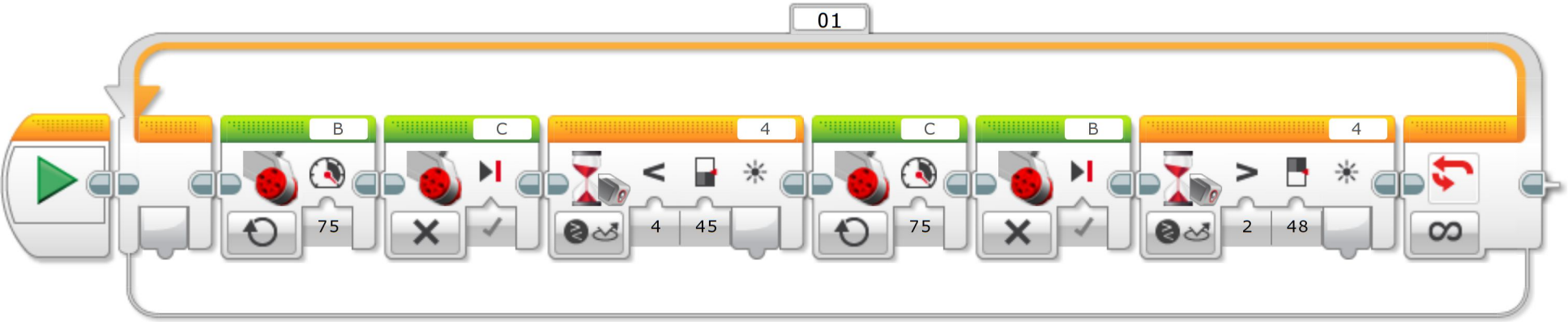
úloha: robot sa pohybuje vpred až kým nepríde k prekážke, tam si kúsok cúvne, otočí sa a pokračuje pohybom vpred.

poznámka: spôsob otočenia pred prekážkou by bol vhodnejší cúvaním s otočením - stačí na to použiť jeden blok.

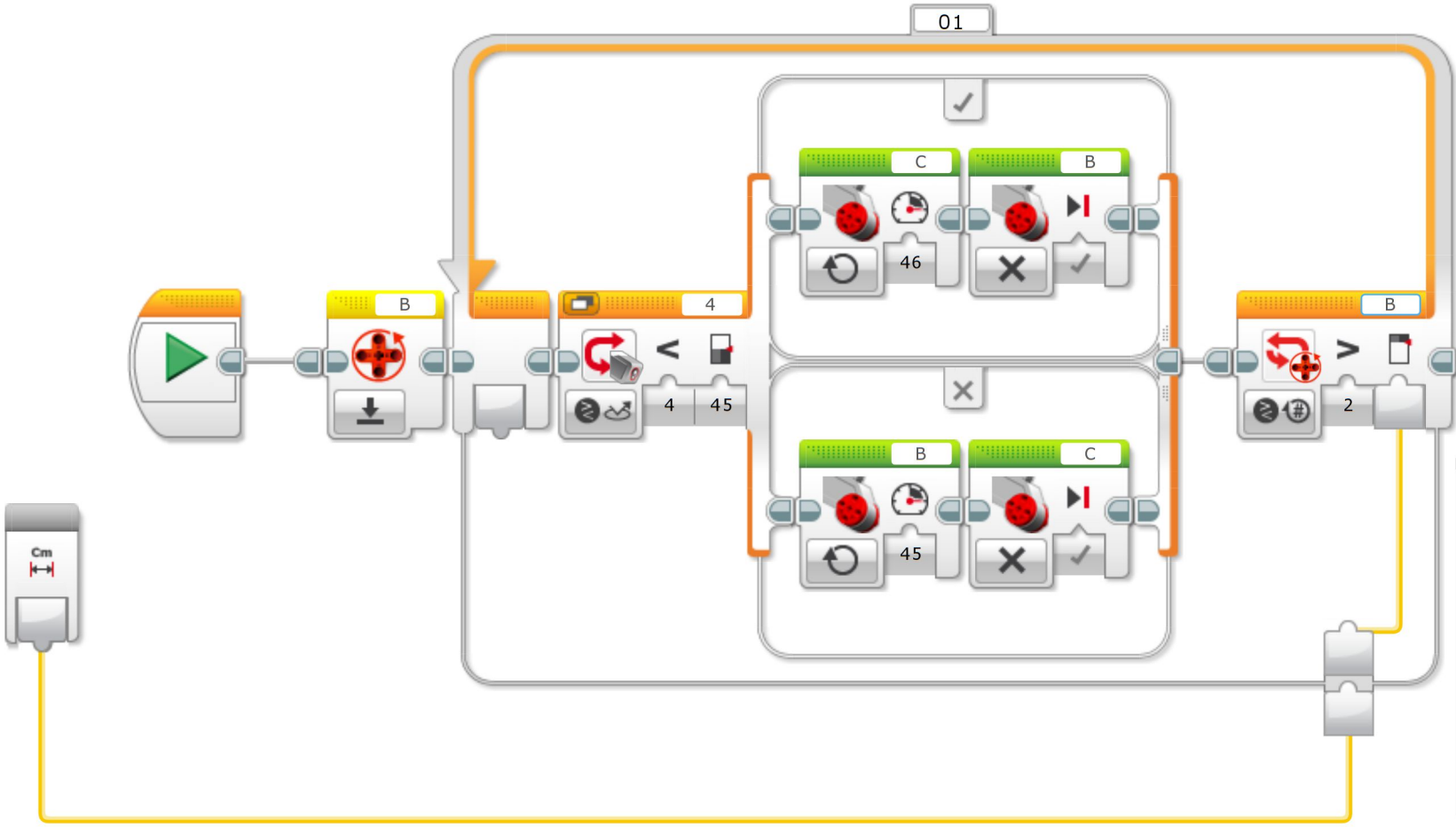
jednoduchý pohyb po čiare pomocou čakacích blokov



Úloha: robot sa rozbehne vpred, díva sa svetelným senzorom nadol a vždy keď prejde cez čiernu čiaru, zapíska. Pri stene zastane.



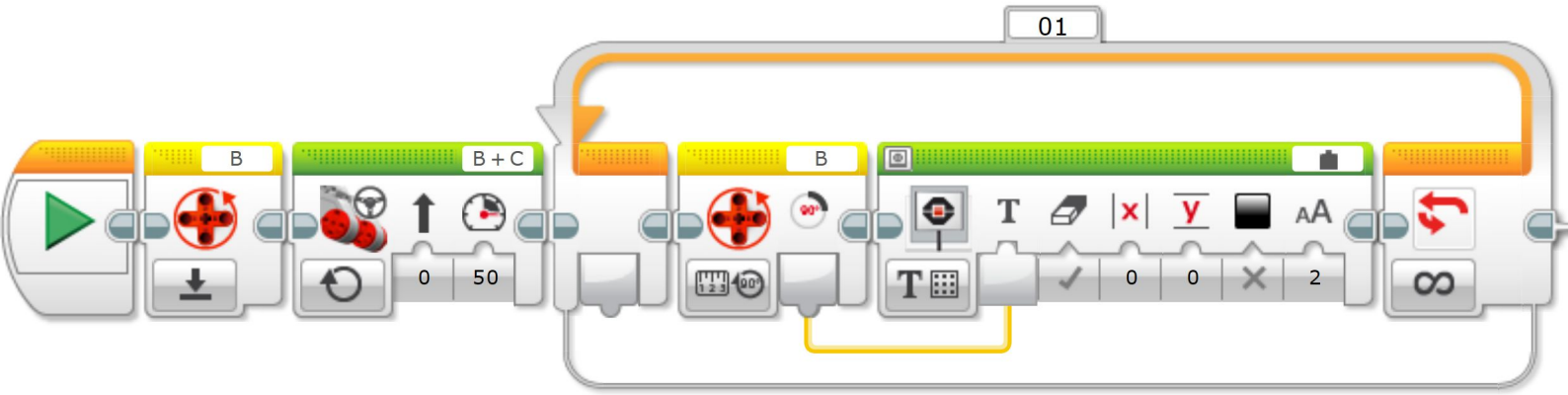
Toto je príklad jednoduchého sledovania čiary s jedným senzorom pripojeným na porte 4 pomocou čakacích blokov.
Vždy sa pohybuje len jeden motor (jedno koliesko) a druhý musíme vypnúť.
Má nevýhodu, že počas vykonávania čakacieho bloku nevieme testovať žiadnu inú podmienku, ale kým ide robot po čiare, tak to až tak nevadí, lebo čakania netrvajú veľmi dlho, keďže robot stále vychádza von z čiary a vzápätí sa na ňu vráti späť.



Sledovanie čiary pomocou podmienky - je to lepšie ako pomocou čakacieho bloku, lebo takto môžeme v cykle kontrolovať aj iné podmienky - program nikdy nečaká - nezastaví na nejakom bloku.
Na druhej strane bude frekvencia čítania zo senzorov o niečo málo menšia ako v prípade použitia čakacích blokov, prejaví sa to až pri vyšších rýchlostiach pohybu robota.

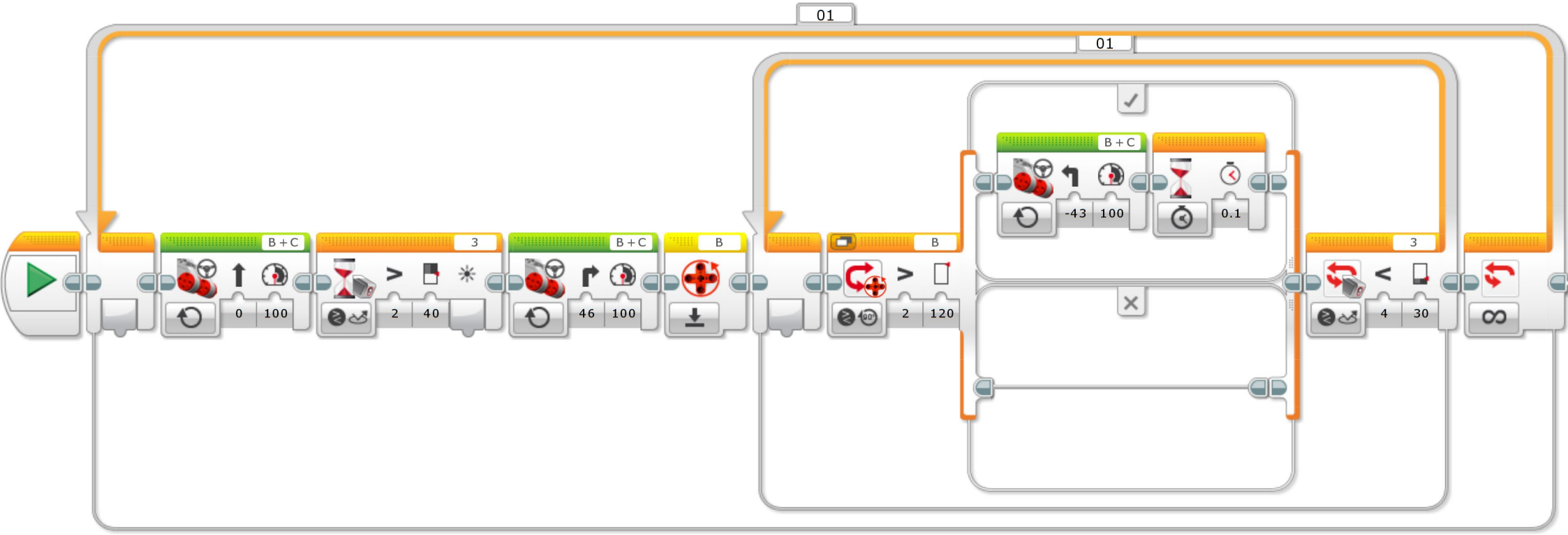
Toto je blok s parametrom a teda ho nie je vhodné používať ako samostatný program, príklad použitia je z programu priklad_bloku.

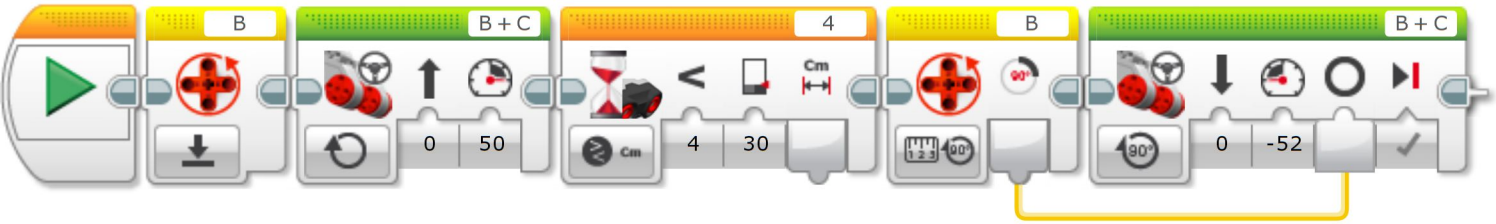
Robot sa pohybuje po čiare a keď pritom stretne prekážku, zastane a zakričí na ňu, aby uhla. Ak uhne, tak pokračuje v sledovaní čiary ďalej.



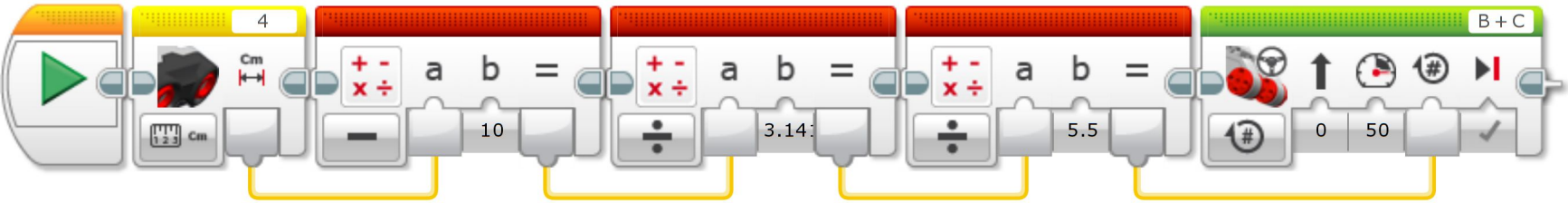
jednoduchá ukážka dátového toku (žltý drôtik) - kde sa hodnota otáčkového senzora z motora B (ktorý sa na začiatku rozbehne) prenáša do zobrazovacieho bloku a ukazuje sa na displeji
(v pravom hornom rohu sme najskôr museli prepnúť na "wired")

ukážka srandovného pohybu po čiare, kde na čiare robot ide priamo vpred a potom, keď z čiary vybehne skúsi ju nájsť najskôr do jednej strany - ak sa mu to nepodarí (zistí to pomocou otáčkového senzora), tak hľadá pokračovanie čiary na druhej strane



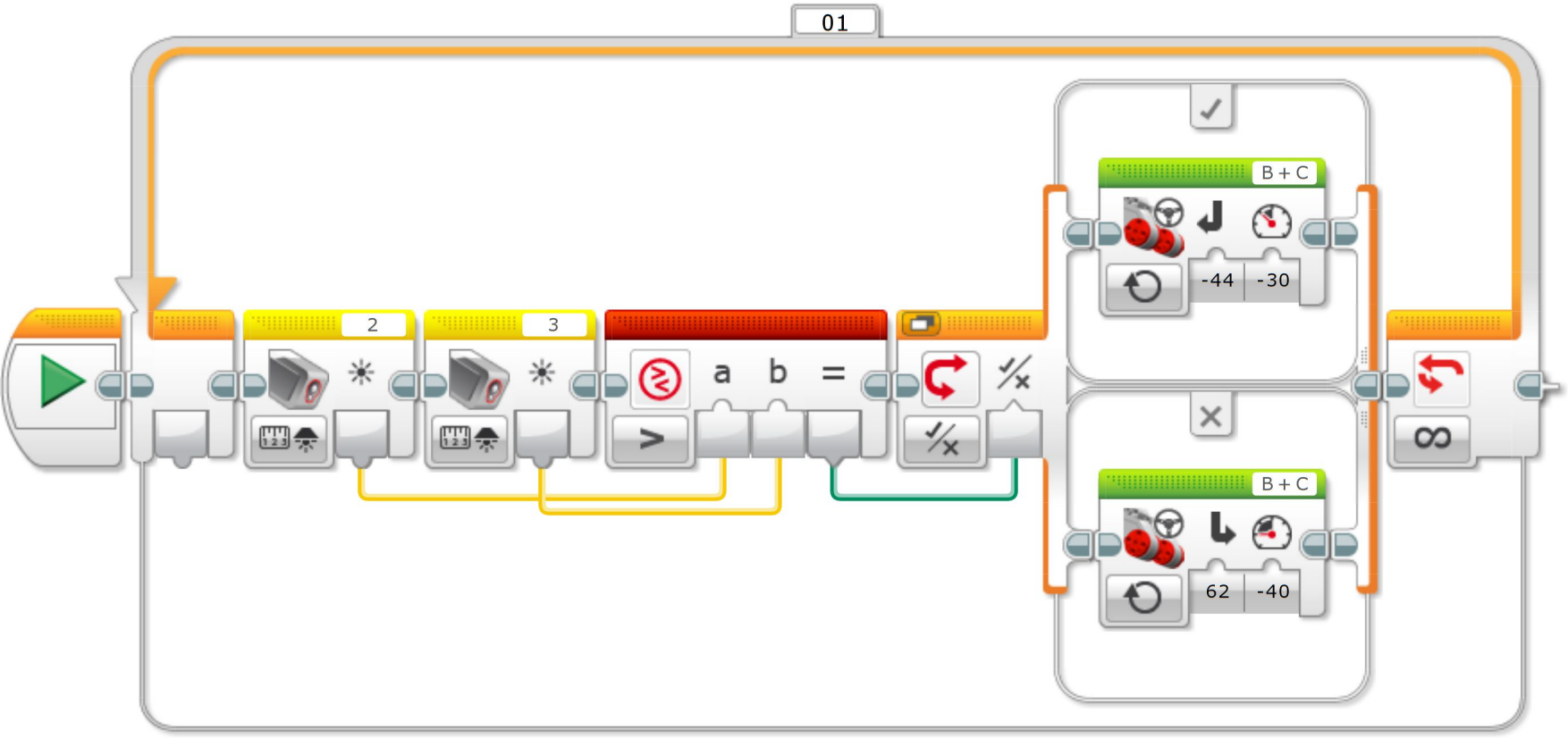


Úloha: robot sa pohybuje vpred, kým nepríde k prekážke. Tam sa otočí a vráti sa na pôvodné miesto, odkiaľ vyštartoval.

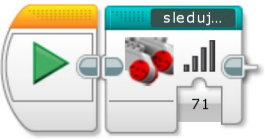


príklad využitia výpočtov: robot si zmeria vzdialenosť v prekážke a z nej vypočíta počet otáčok, ktorými musí otočiť svojimi kolesami (s priemerom 5.5 cm), aby sa k prekážke dostal. Zastane 10 cm pred prekážkou.

Robot má namontované dva farebné/svetelné senzory, ktoré fungujú v režime rozptýleného svetla - "dívajú" sa vpred, jeden vľavo, druhý vpravo. Program porovnáva ich hodnoty tak, aby sa robot pohyboval za svetlom.

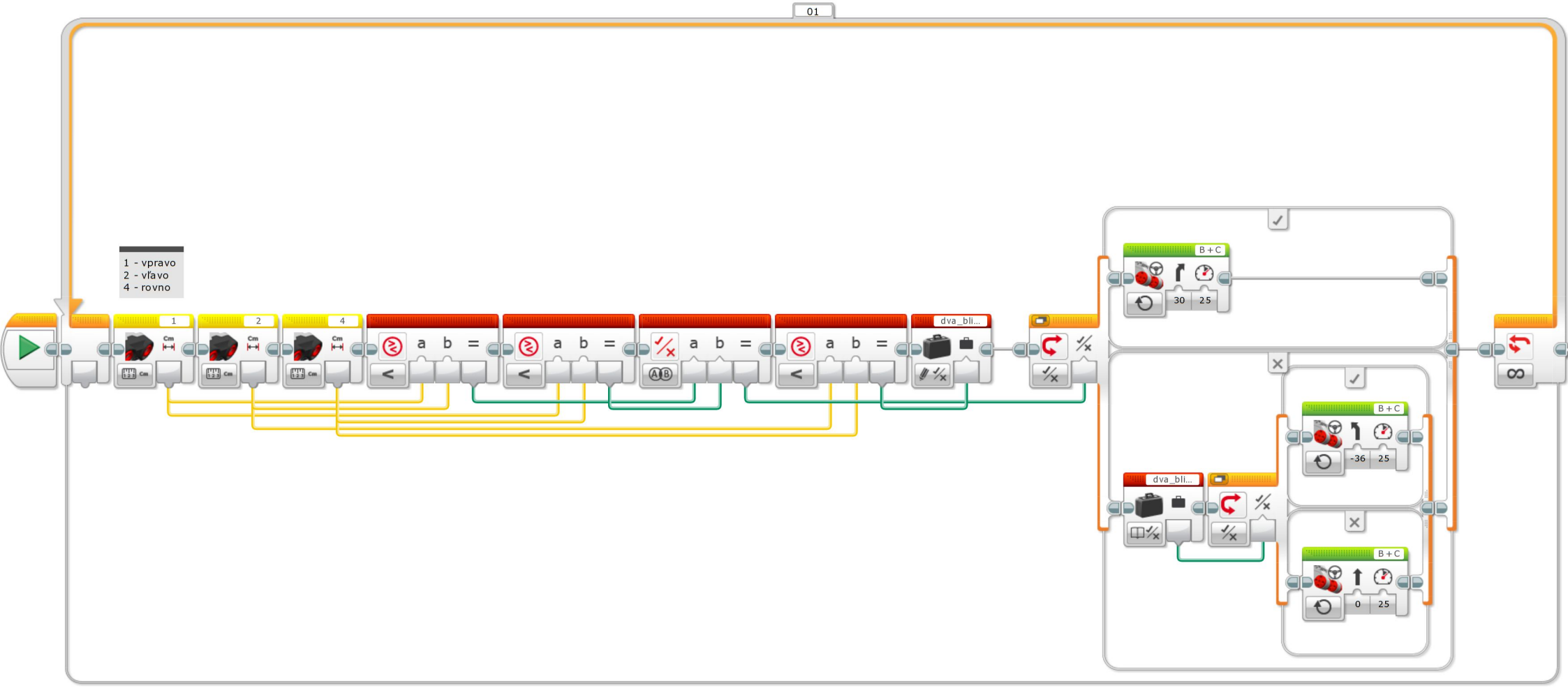


príklad využitia bloku na sledovanie čiary, ktorý má jeden parameter - rýchlosť pohybu robota počas sledovania čiary.

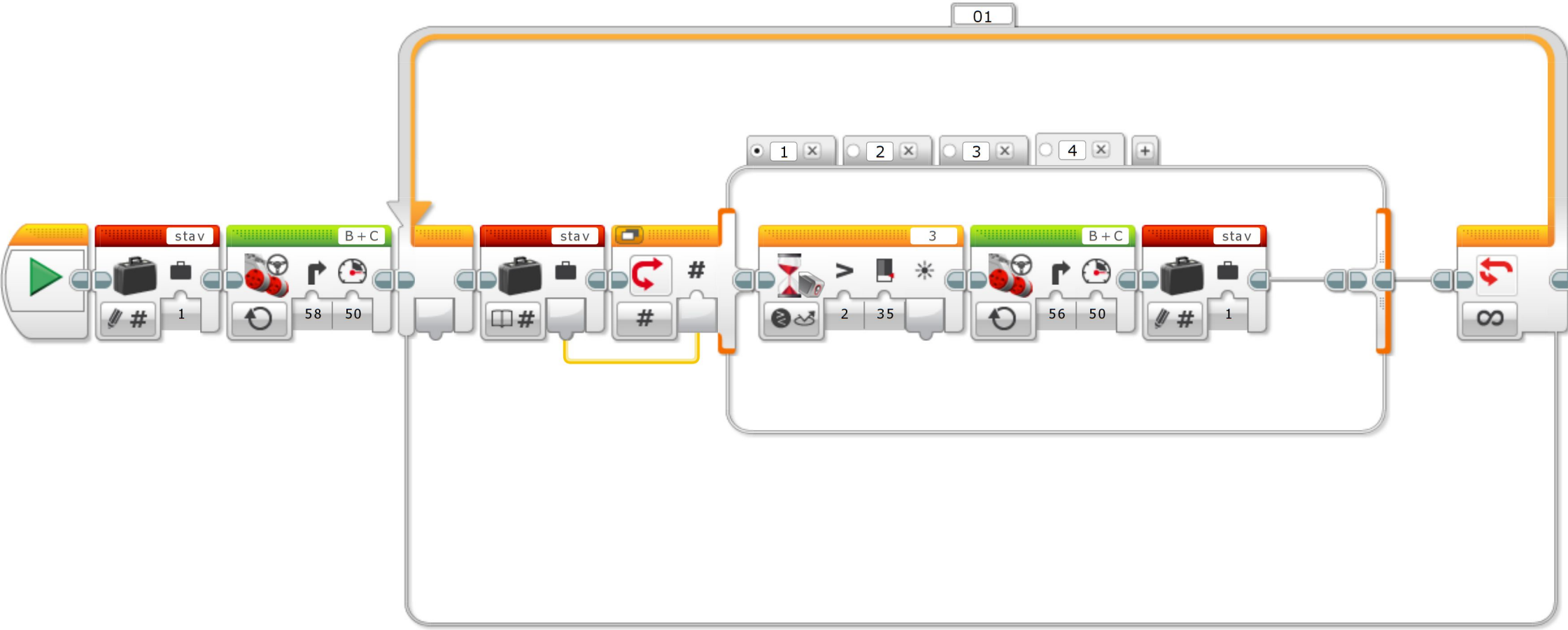


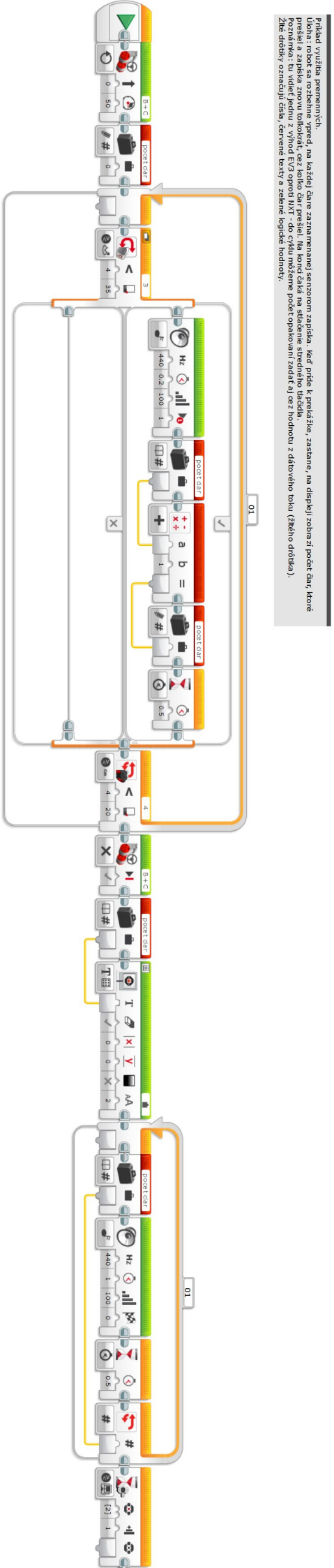
Blok na jednoduché sledovanie čiary pomocou dvoch senzorov na portoch 3 a 4. Parameter udáva rýchlosť pohybu robota pri sledovaní čiary. Blok si túto rýchlosť interne uloží do premennej A. Keď robot príde k prekážke, blok končí a program pokračuje ďalej...

Robot má namontované tri ultrazvukové senzory - jeden sa "díva" vľavo, druhý vpravo a tretí priamo dopredu. Program porovná ich hodnoty a otáča robotom za najbližšou prekážkou, alebo sa pohybuje vpred, ak je priamo pred ním. Výsledkom je robot, ktorý nasleduje človeka podľa toho, kam pred ním ustupuje.

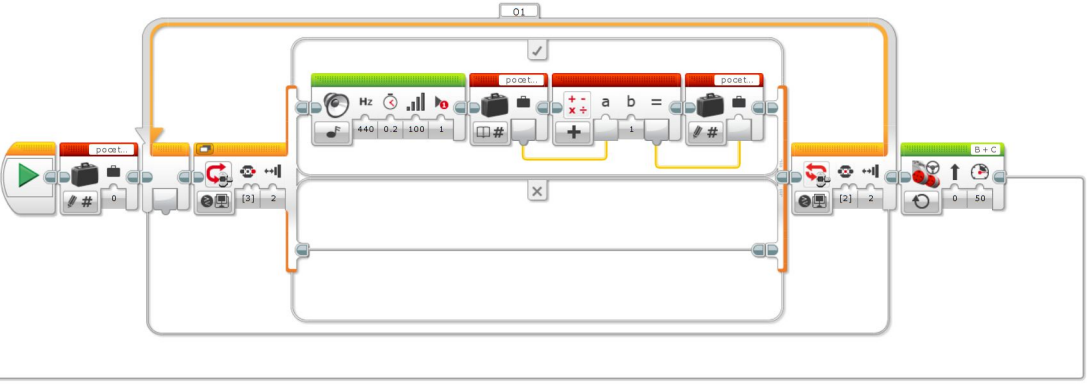


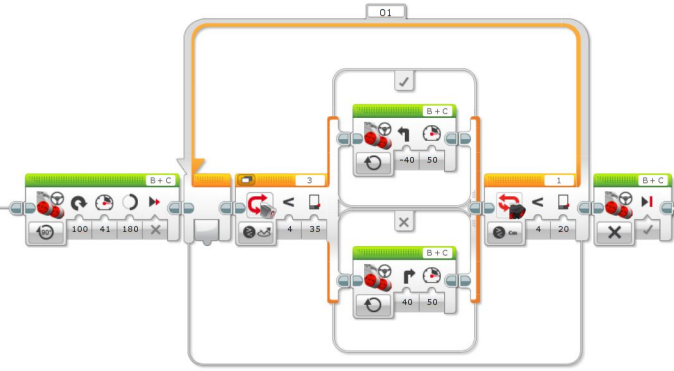
Príklad využitia jednoduchého stavového automatu: sledovanie čiary pomocou jedného senzora v režime cik-cak, t.j. robot nesleduje "hranu" ako pri bežnom sledovaní čiary, ale prechádza z jednej strany na druhú. Stav je uložený v premennej stav a máme 4 stavy: 1. sme naľavo od čiary a čakáme, kým do nej nevôjdeme, 2. sme v čiare a snažíme sa vyjsť von na pravú stranu, 3. sme napravo od čiary a snažíme sa do nej vojsť z pravej strany, 4. sme v čiare a snažíme sa z nej vyjsť na ľavej strane. Stavy sa cyklicky striedajú 1 -> 2 -> 3 -> 4 -> 1 ...

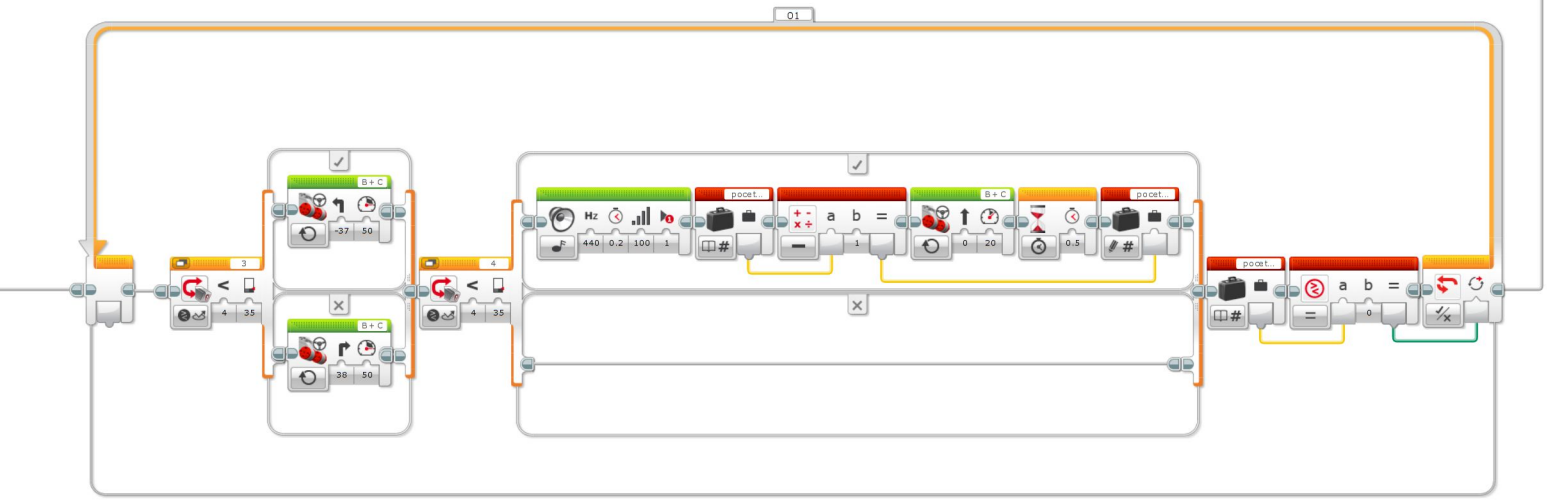


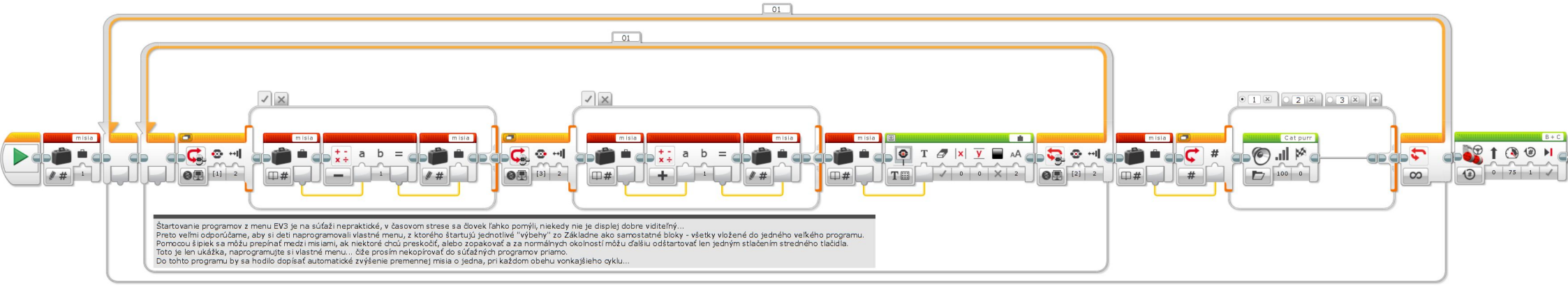

Príklad využitia premenných.
Úloha: robot sa rozbehne vpred, na každej čiare zaznamenanej senzorom zapíska. Keď príde k prekážke, zastane, na displeji zobrazí počet čiar, ktoré prešiel a zapíska znovu toľkokrát, cez koľko čiar prešiel. Na konci čaká na stlačenie stredného tlačidla.
Poznámka: tu vidieť jednu z výhod EV3 oproti NXT - do cyklu môžeme počet opakovaní zadať aj cez hodnotu z dátového toku (žltého drôtika).
Žlté drôtiky označujú čísla, červené texty a zelené logické hodnoty.

Úloha: robot sa nachádza v cudzom meste a chce sa dostať na železničnú stanicu. Na začiatku sa opýtal miestneho okoloidúceho ako sa tam dostane... Okoloidúci stlačí šípku vpravo na EV3 kocke toľkokrát (=P), na koľkej odbočke má turista odbočiť vpravo. Zadanie počtu okoloidúci potvrdí stlačením stredného tlačidla. Robot sa pohybuje po čiare pomocou jedného senzora a druhým, ktorý je umiestnený o niečo viac vpravo počíta odbočky. Do P-tej odbočky potom zatočí a opäť sleduje čiaru, až kým nepríde k budove stanice, ktorú zaznamená senzorom na meranie vzdialenosti.
V programe využívame porovnávací blok, ktorý zistí, či už robot dosiahol správnu odbočku - výstupom je zelená (logická = áno/nie) hodnota, ktorá vstupuje do ukončenia cyklu ako podmienka ukončenia cyklu.

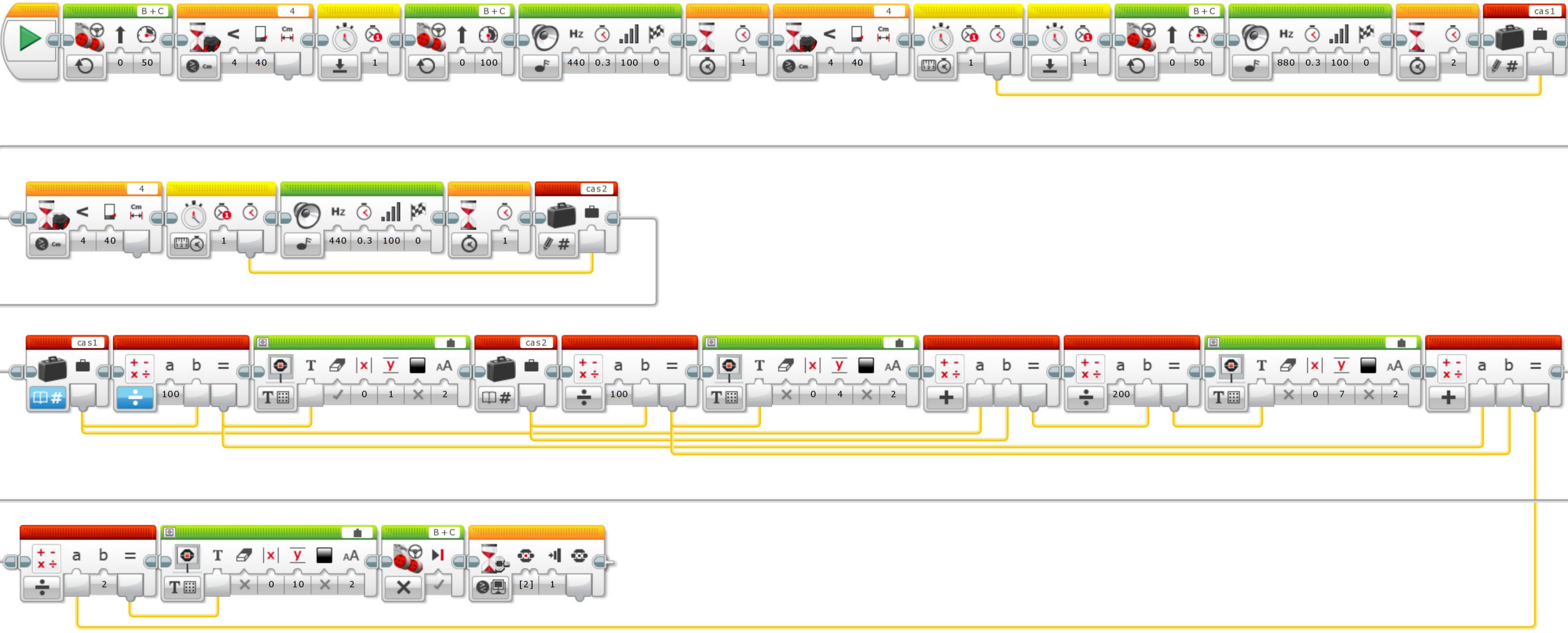




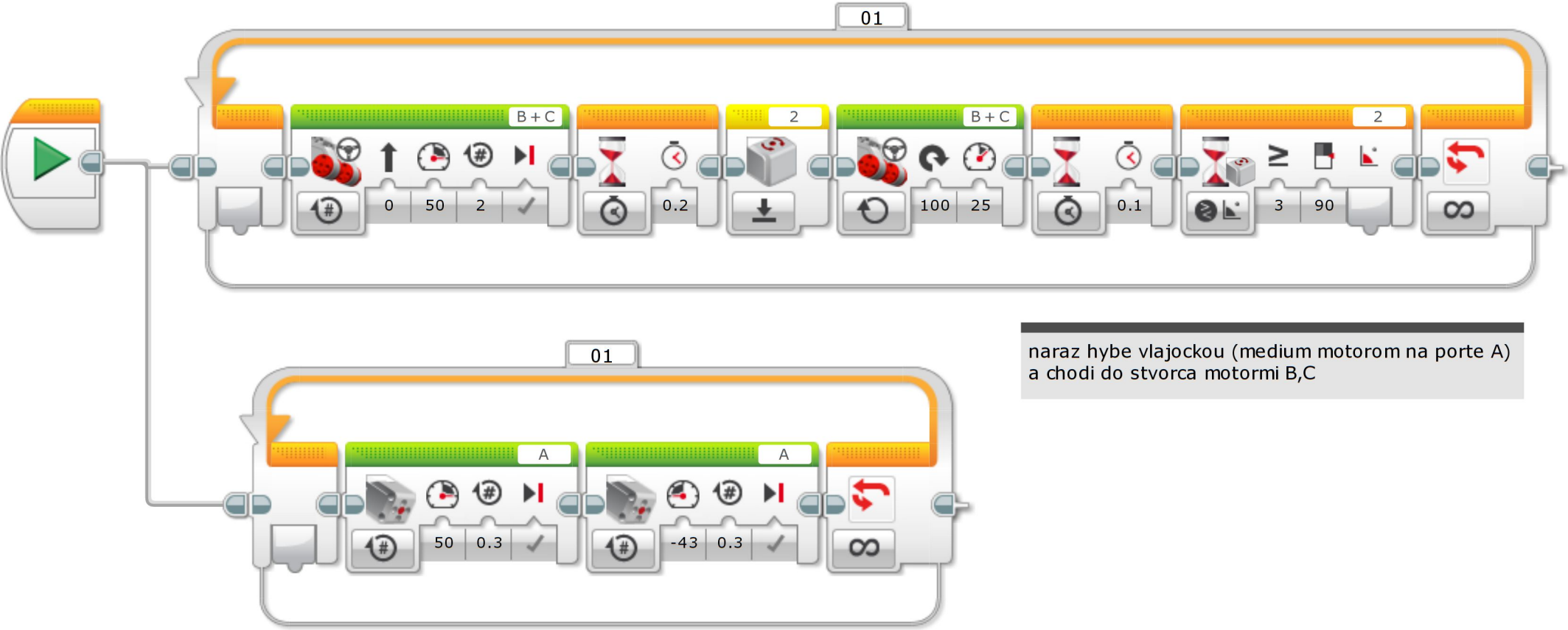



01
01
misia
misia
misia
misia
misia
misia
misia
misia
misia
Cat purr
Štartovanie programov z menu EV3 je na súťaži nepraktické, v časovom strese sa človek ľahko pomýli, niekedy nie je displej dobre viditeľný...
Preto veľmi odporúčame, aby si deti naprogramovali vlastné menu, z ktorého štartujú jednotlivé "výbehy" zo Základne ako samostatné bloky - všetky vložené do jedného veľkého programu.
Pomocou šípiek sa môžu prepínať medzi misiami, ak niektoré chcú preskočiť, alebo zopakovať a za normálnych okolností môžu ďalšiu odštartovať len jedným stlačením stredného tlačidla.
Toto je len ukážka, naprogramujte si vlastné menu... čiže prosím nekopírovať do súťažných programov priamo.
Do tohto programu by sa hodilo dopísať automatické zvýšenie premennej misia o jedna, pri každom obehu vonkajšieho cyklu...

Úloha: Predstavme si automobil, ktorý ide rýchlosťou 40 km/h úsek 1 km a potom rýchlosťou 60 km/h ďalší úsek dĺžky 1 km. Aká bude priemerná rýchlosť? Niekto by povedal 50 km/h, avšak v skutočnosti... a o tom je táto úloha.
Robot má namontovaný ultrazvukový senzor, ktorým sa "díva" vpravo. Popri jeho dráhe na pravej strane sú tri flaše, ktoré vytvárajú úseky rovnakej dĺžky. Medzi prvou a druhou fľašou sa pohybuje rýchlosťou v1, medzi druhou a treťou fľašou rýchlosťou v2. Napokon na displej vypíše nameranú rýchlosť v1, v2 a vypočítanú celkovú priemernú rýchlosť a pre porovnanie aj rýchlosť (v1 + v2) / 2, ktorá nie je priemernou rýchlosťou.



príklad dvoch paralelne vykonávaných výpočtových vlákien (procesov)



naraz hybe vlajockou (medium motorom na porte A) a chodi do stvorca motormi B,C

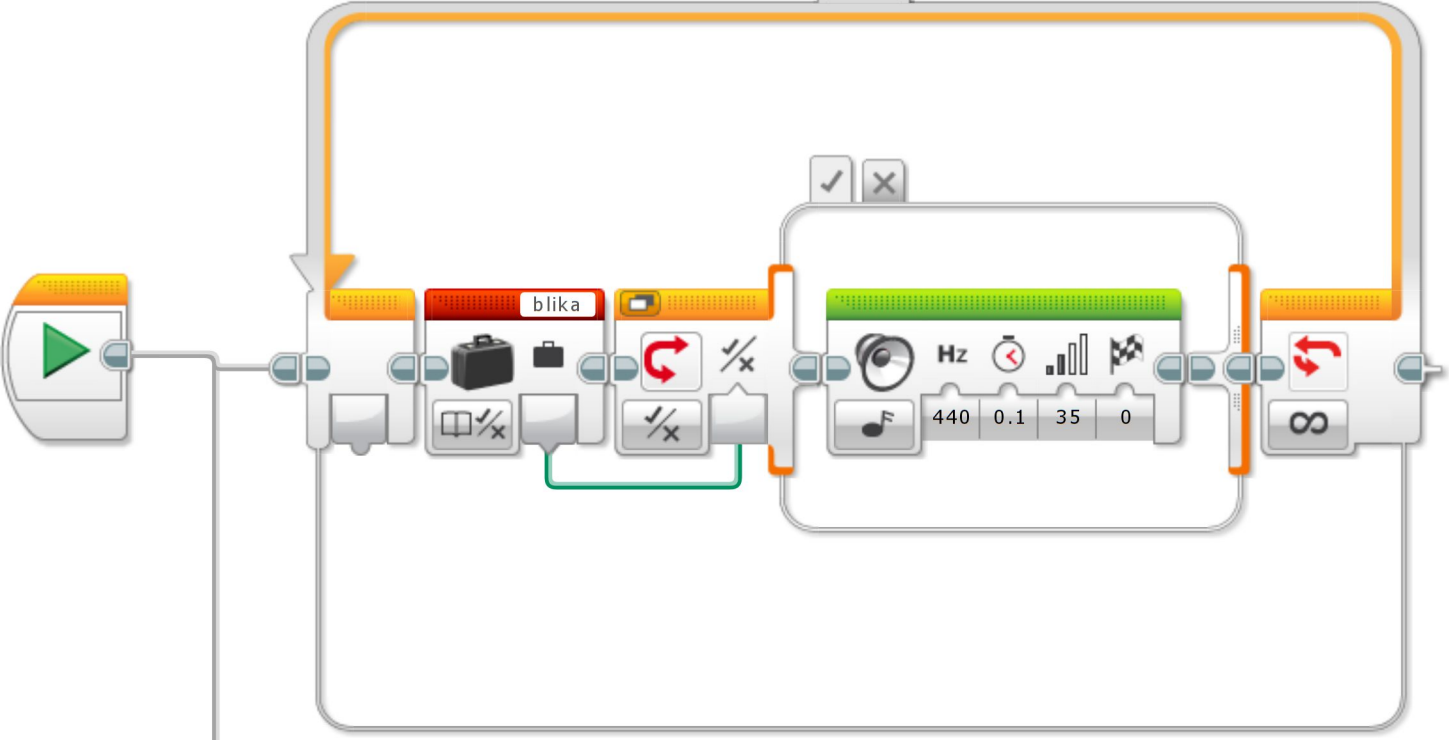
úloha: robot má pomocou svetelného senzora rozpoznať, či svetlo, ktoré do neho dopadá je blikajúce svetlo - ak vidí blikajúce svetlo (napr. cyklistické blikajúce svetlo), tak má vydávať zvukový signál.

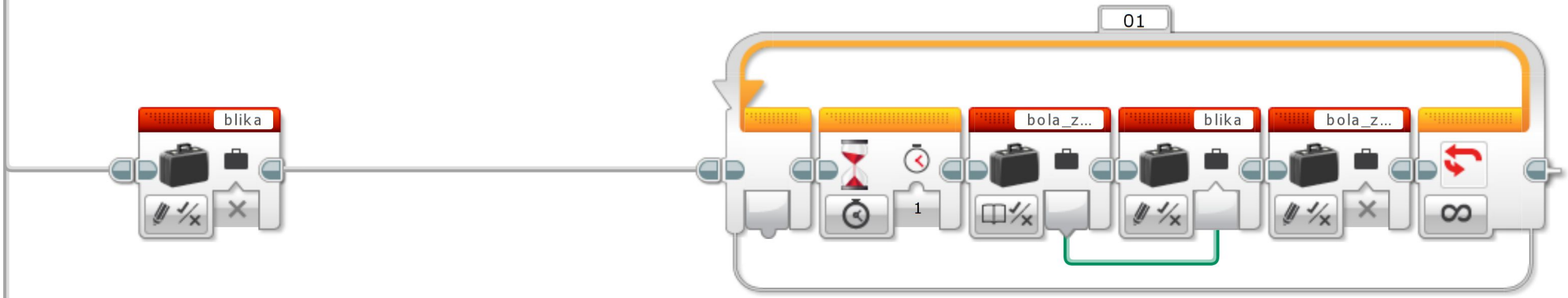
riešenie:
program je rozdelený na tri paralelné procesy:

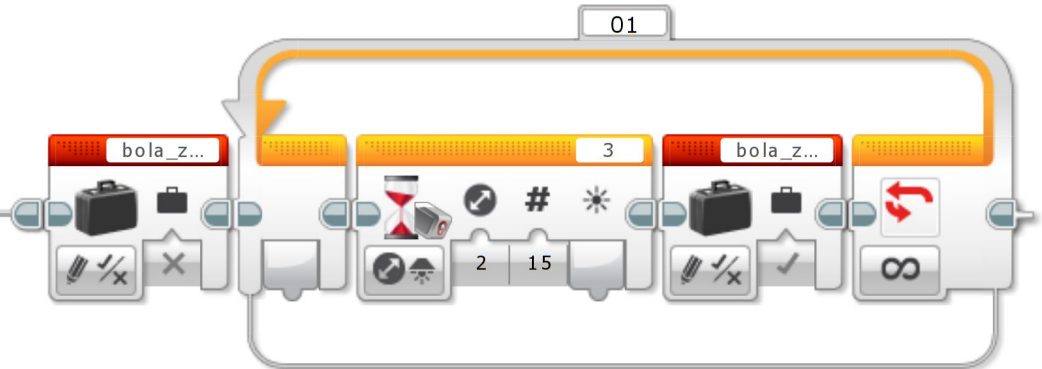
1. (celkom dolu) čaká pomocou čakacieho bloku na nasledujúcu zmenu intenzity svetla, ktoré dopadá do senzora (režim "change"). vždy keď dôjde k dostatočne silnej zmene osvetlenia (naznačujúcej blikajúce svetlo), nastaví logickú premennú bola_zmena.

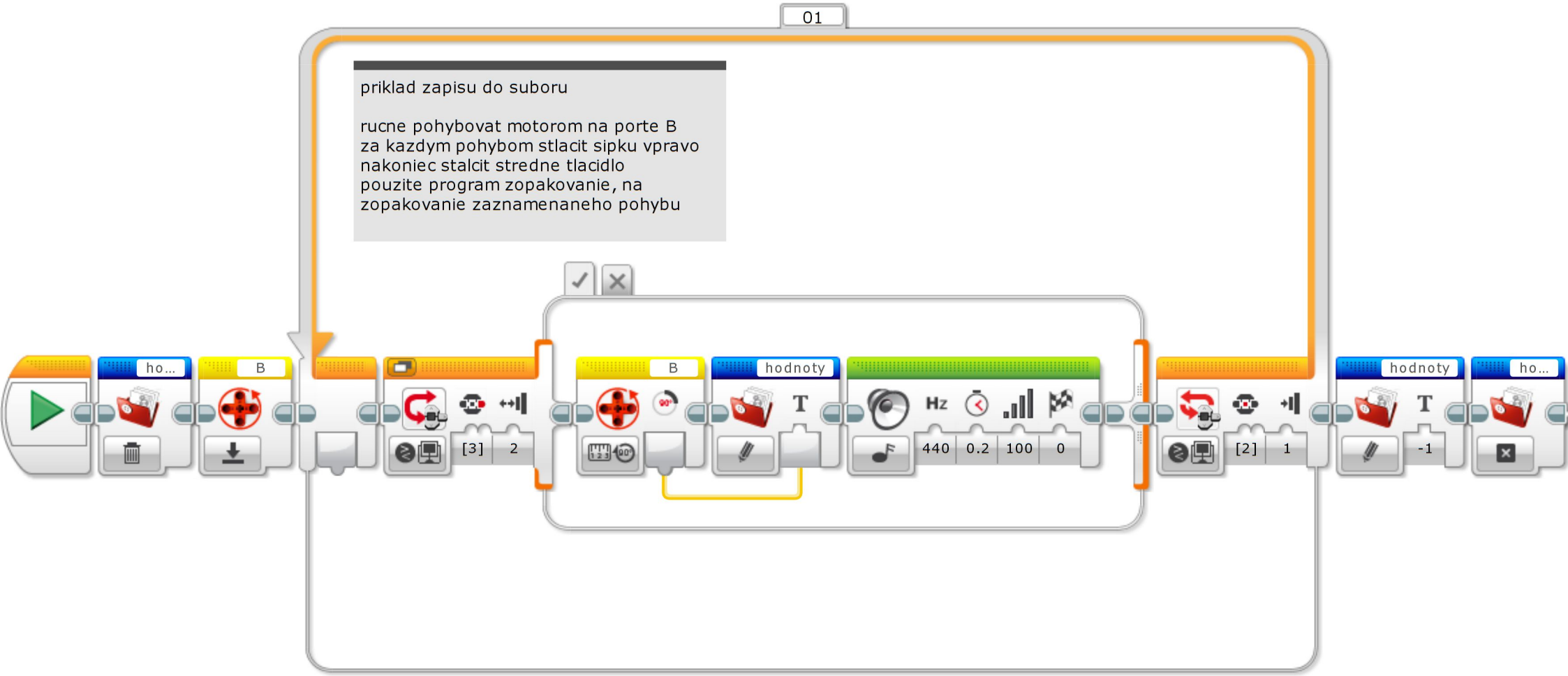
2. (v strede) - znovu a znovu čaká 1 sekundu a potom sa pozrie, či v priebehu poslednej sekundy bola zmena osvetlenia (indikovaná hodnotou "yes" v premennej bola_zmena - túto hodnotu len priamo skopíruje do premennej blika, potom vynuluje stav bola_zmena a znovu zaspí na 1 sekundu.

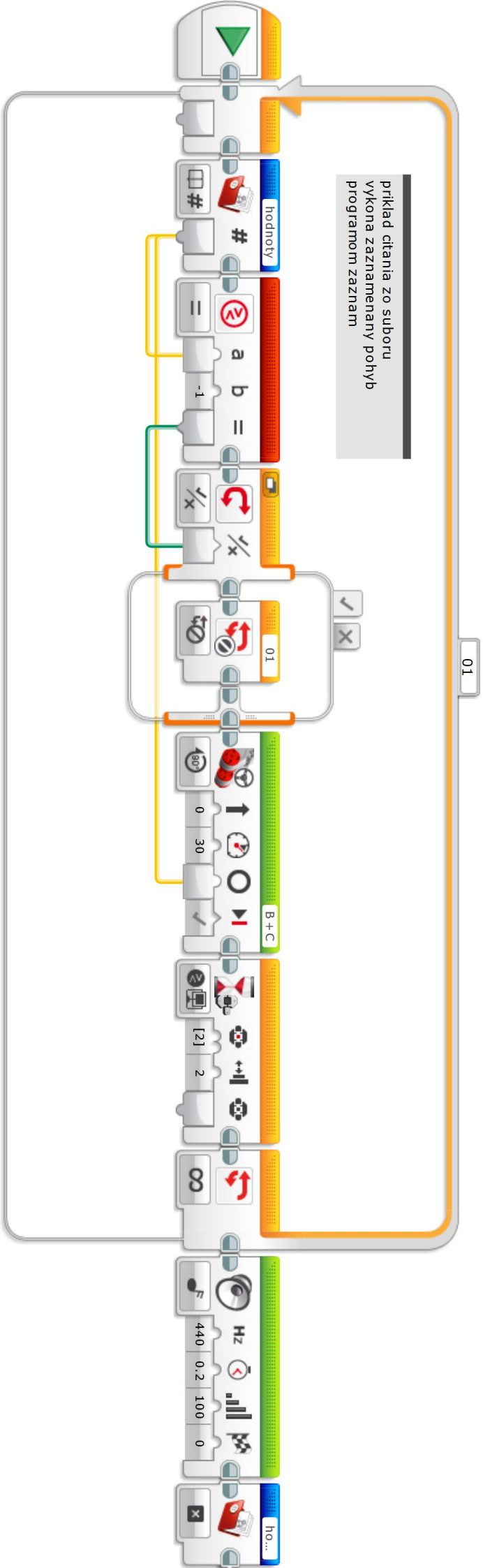
3. (hore) neustále monitoruje stav premennej blika - a v prípade, že obsahuje hodnotu "yes", tak vydáva zvukový signál.



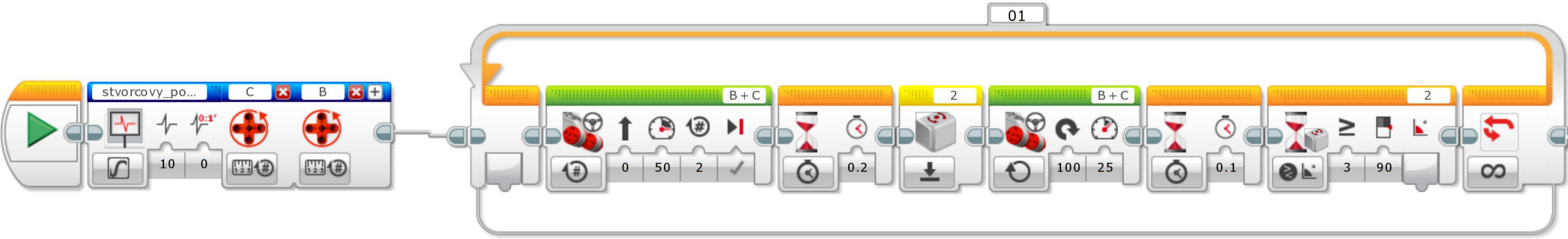





01
priklad zapisu do suboru
rucne pohybovat motorom na porte B
za kazdym pohybom stlacit sipku vpravo
nakoniec stalcit stredne tlacidlo
pouzite program zopakovanie, na
zopakovanie zaznamenaneho pohybu
ho...
B
B
hodnoty
hodnoty
ho...
[3]
2
440
0.2
100
0
[2]
1
-1

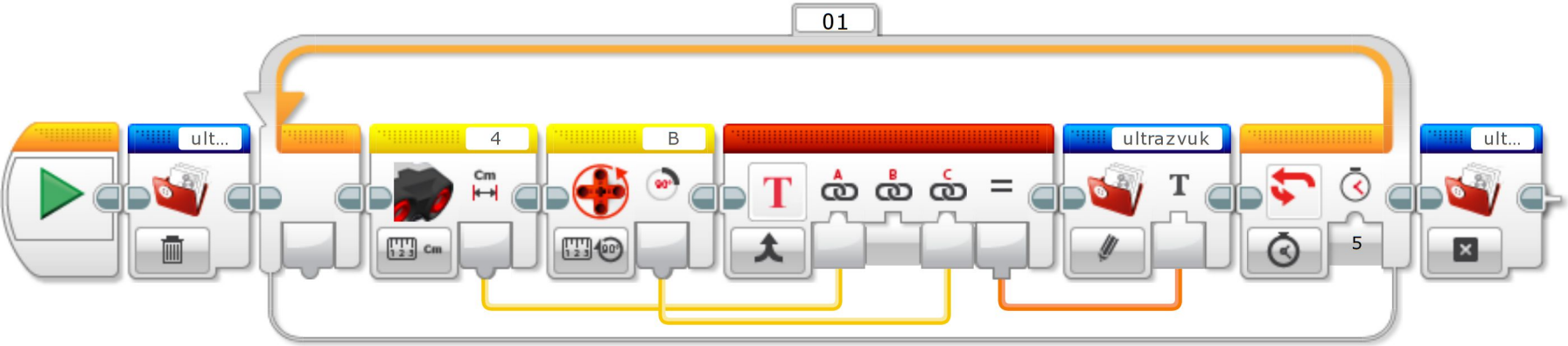

priklad citania zo suboru
vykona zaznamenany pohyb
programom zaznam
hodnoty
01
B+C
440
0.2
100
01

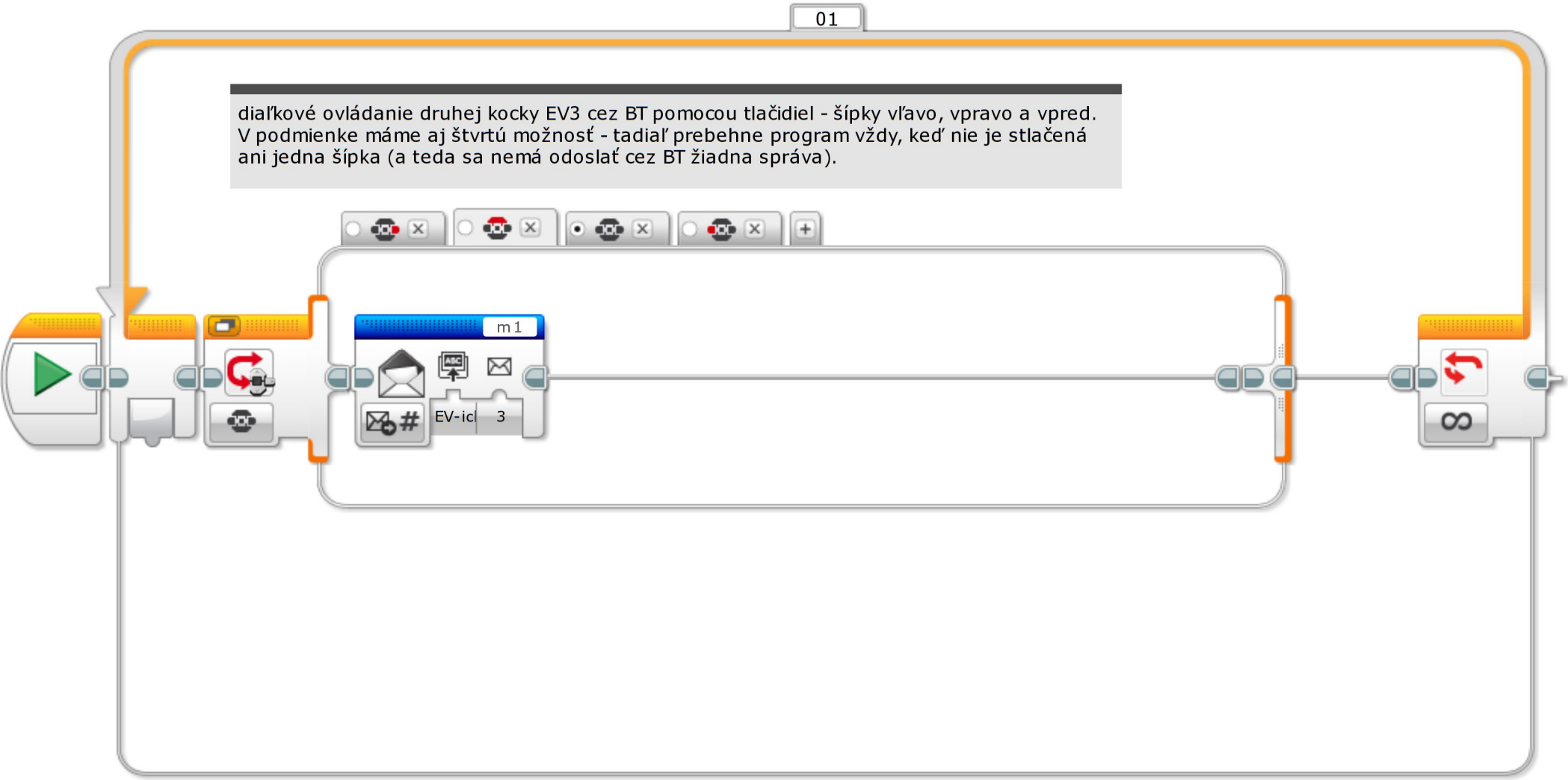
Príklad využitia datalogu - zaznamenávame hodnoty otáčkových senzorov motorov B a C počas pohybu do štvorca.
Na grafe potom bude vidieť, že vonkajší motor celkovo najazdí viac a hodnota otáčok stále rastie, zatiaľ čo vnútorný pri otočeniach v rohoch štvorca klesá a pri pohybe vpred rovnobežne rastie.

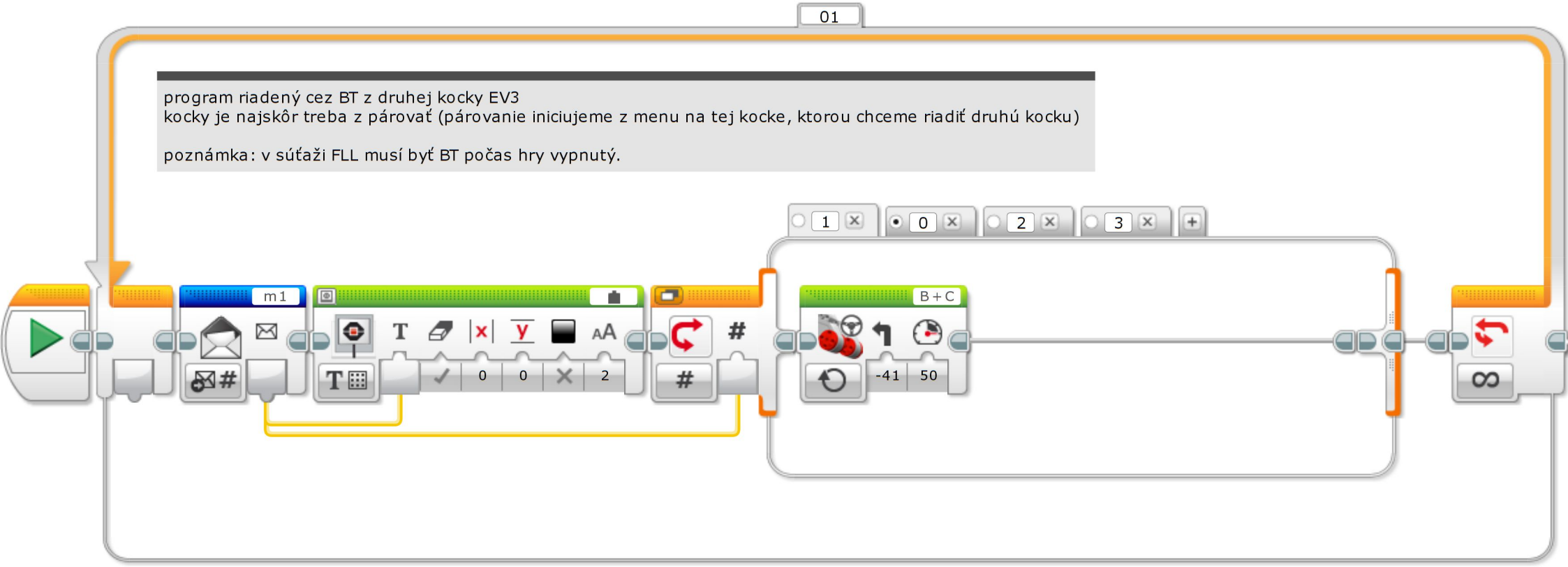


Jednoduchá ukážka záznamu údajov do súboru. Zaznamenávame ultrazvuk na porte 4 a otáčkový senzor motora B.

Najskôr súbor vymažeme - aby sme nepridávali na koniec už existujúceho súboru, potom čítanie zo senzorov zlepíme do jedného riadku, aby sme vytvorili textový súbor s dvomi hodnotami na každom riadku. Nakoniec vytvorený súbor uzavrieme.




01
diaľkové ovládanie druhej kocky EV3 cez BT pomocou tlačidiel - šípky vľavo, vpravo a vpred.
V podmienke máme aj štvrtú možnosť - tadiaľ prebehne program vždy, keď nie je stlačená
ani jedna šípka (a teda sa nemá odoslať cez BT žiadna správa).
m1
EV-ic
3


01
program riadený cez BT z druhej kocky EV3
kocky je najskôr treba z párovať (párovanie iniciujeme z menu na tej kocke, ktorou chceme riadiť druhú kocku)
poznámka: v súťaži FLL musí byť BT počas hry vypnutý.
m1
0
0
2
1
0
2
3
B+C
-41
50