**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**

**Fakulta chemickej a potravinárskej technológie**

Evidenčné číslo: FCHPT-5415-50183

# Programovanie NXT Robotov

**Bakalárska práca**

**2013** **Ivan Malíšek**

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**

**Fakulta chemickej a potravinárskej technológie**

Evidenčné číslo: FCHPT-5415-50183

# Programovanie NXT Robotov

**Bakalárska práca**

Študijný program: automatizácia, informatizácia a manažment v chémii a potravinárstve
Číslo študijného odboru: 2621
Názov študijného odboru: 5.2.14 automatizácia, 5.2.52 priemyselné inžinierstvo
Školiace pracovisko: Ústav informatizácie, automatizácie a matematiky
Vedúci záverečnej práce: Ing. Richard Valo
Konzultant: Ing. Martin Kalúz

**Bratislava 2013** **Ivan Malíšek**

Slovenská technická univerzita v Bratislave
Ústav informatizácie, automatizácie a matematiky

Fakulta chemickej a potravinárskej technológie
Akademický rok: 2012/2013
Evidenčné číslo: FCHPT-5415-50183

# ZADANIE BAKALÁRSKEJ PRÁCE

| | |
|---|---|
| Študent: | **Ivan Malíšek** |
| ID študenta: | 50183 |
| Študijný program: | automatizácia, informatizácia a manažment v chémii a potravinárstve |
| Kombinácia študijných odborov: | 5.2.14 automatizácia, 5.2.52 priemyselné inžinierstvo |
| Vedúci práce: | Ing. Richard Valo, PhD. |
| Konzultant: | Ing. Martin Kalúz |

Názov práce: **Programovanie NXT Robotov**

Špecifikácia zadania:

Projekt je zameraný na osvojenie si programovacích zručností pri tvorbe riadiacich algoritmov robotov LEGO-Mindstorms.

LEGO Mindstorms NXT Stavebnica s programovateľnou NXT kockou a rozličnými senzormi je zaujímavý nástroj na programovanie a navrhovanie rôznych druhov robotov, na ktorých sa dajú aplikovať vedomosti z viacerých predmetov vyučovaných na oddelení. Študenti môžu aplikovať svoje znalosti pri stavaní múdrych robotov, zbieraní dát a prezentácii nameraných výsledkov. Takto sú motivovaní riešiť praktické problémy či vytvárať a overovať vlastné námety.

Pomocou tejto stavebnice sa dajú jednoducho vytvárať roboty či vozidlá, ktoré si samé dokážu nájsť cestu, zaparkovať, prekonať prekážku či zbierať predmety roztrúsené po miestnosti.

Úlohy:
1. Zoznámenie sa s programovacím prostredím pre riadiace jednotky robotov.
2. Návrhy riadenia

Riešenie zadania práce od: 18. 02. 2013

Dátum odovzdania práce: 25. 05. 2013

**L. S.**

**Ivan Malíšek**
študent

**prof. Ing. Miroslav Fikar, DrSc.**
vedúci pracoviska

**prof. Ing. Miroslav Fikar, DrSc.**
garant študijného programu

**Poďakovanie**

Chcel som poďakovať vedúcemu bakalárskeho projektu Ing. Richardovi Valovy, PhD za odborné vedenie, pripomienky a cenné rady.

## Obsah

**Abstrakt**

Hlavným cieľom tejto práce je vytvorenie nového riadiaceho bloku v lego knižnici v Simulink-u a následné overenie jeho funkčnosti v praxi.

V prvých dvoch kapitolách sa oboznámime s lego stavebnicou LEGO Mindstorms NXT a prácou v interaktívnom dizajnovom simulačnom prostredí menom Stateflow.

V kapitole try popíšeme vytvorenie riadiaceho bloku v Simulink-u a v kapitolách päť a šesť podrobne popíšeme aplikáciu nami vytvoreného riadiaceho bloku. V kapitole sedem vytvoríme program, ktorý je nadstavbou predchádzajúceho programu.

Výsledkom práce je otestovanie programov ako aj nami vytvoreného senzoru v reálnom prostredí.

Kľúčové slová :LEGO Mindstorms NXT, MATLAB, Simulink, Stateflow

**Abstract**

The main aim of this bachelor thesis is to form a new control block in lego library in Simulink and to test its functionality in praxis.

In the first two chapters we are going to become familiar with lego parts LEGO Mindstorms NXT and with the interactive design simulation environment named Stateflow.

In the chapter three we are going to describe creation of controlling block in Simulink and in chapters five and six we are going to describe in detail the application of our formed control block.
In chapter seven we are going to create a program , that is the extension of the previous program.
The result of the work is to test of programs as well as control block, created by us in real environment.

Key words : LEGO Mindstorms NXT, MATLAB, Simulink, Stateflow

# Úvod

V posledných desaťročiach sa stala automatizácia nedoddeliteľnou súčsťou náško života. Stretávame sa s ňou na každom kroku od jedoduchých domácich spotrebyčov, až po zložité priemyselné alebo vedecké zariadenia. Všetko to začalo vynájdením kolesa, keď sa človek snažil odľahčiť si svoju prácu a pokračovalo to až do dnešnej doby, kde máme rôzne stroje, ktoré nám uľahčuju náš každodenný život a bezpochyby by sme si jeho prirodzený chod nevedeli bez nich predstaviť.

Prácu som si vybral. lebo ma zaujala daná problematika na predmete, kde bola jeho časť venovaná práci s lego-robotmi a vytváraní riadiacich algoritmov v prostredí Simulink, konkrétne za pomoci stavového riadenia.

Ing. Juraj Holaza a Ing. Takács Bálint rozpracovali vo svojej bakalárskej práci z roka 2010 podobnú tému v ktorej sa venovali preskúmaniu možnosti riadenia robotov a vytvoreniu algoritmu na ovládanie robotického auta. [4]

Moja práca je odlišná v tom, že som sa snažil navrhnúť riadiaci blok do lego knižnice v MATLAB-e.

Úlohu budem riešiť použitím princípu stavového riadenia, ktoré môžme bežne vidieť v každodennom živote. Týmto typom riadenia sa môžu riadiť jednoduché systémy (semafory, rampy,...), taktiež zložitejšie priemyselné procesy (plniace linky, zváracie roboty v automobilovom priemysle, ...).

## 1 Lego Mindstorms NXT

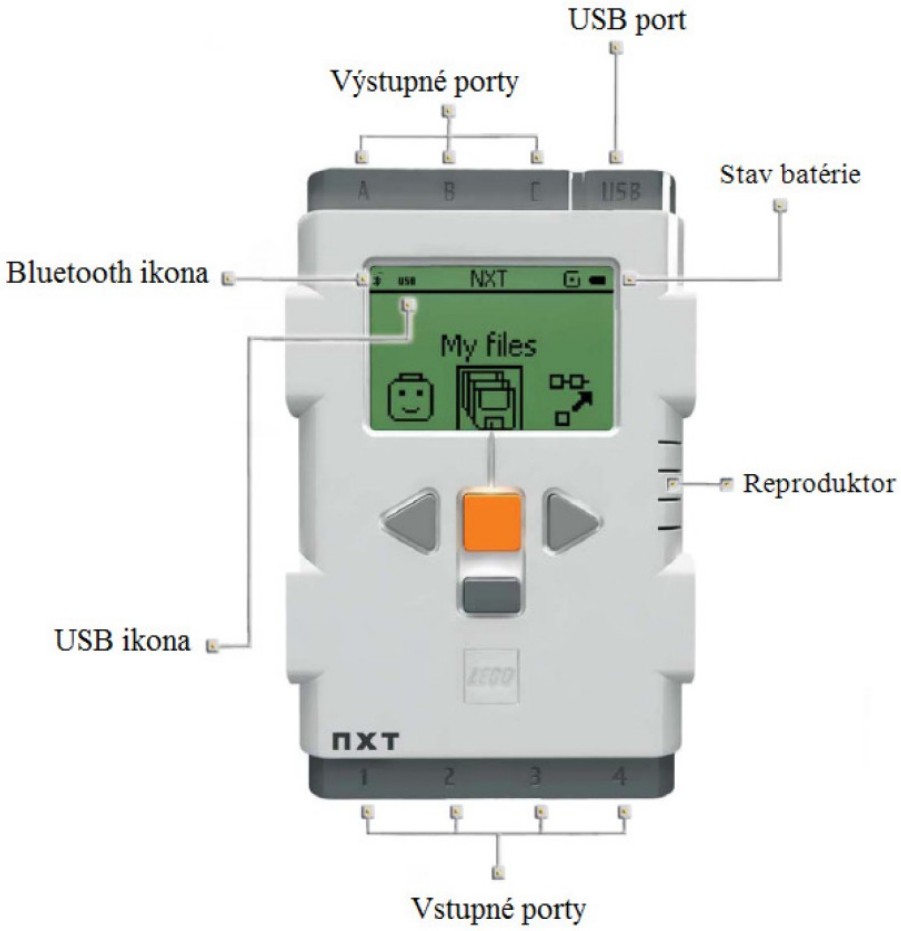
LEGO MINDSTORM Education je generaciou robotickych systomou LEGO, určených k podpore vzdelávania. Žiakov a študentov uvádza do sveta vedy a techniky.[1 ]

Súprava , ktorú máme k dispozícii, pozostáva z mikropočitača, elektrických motorv, rôznych senzorov, a taktiež z rôznich Lego kociek a Lego Technic časti, ktorým sa budeme blyžšie venovať v jednotlivých častiach tejto práce.


**Obr. 1 Stavebnica lego**

### 1.1 Kocka NXT

NXT kocka predstavuje hlavný komponent celej súpravy. Je to počítač v tvare kocky zvaný NXT inteligentná kocka. K nej sa môže pripojiť štyri senzory a môžeme ovládať tri motory cez upravenú verziu káblu typu RJ12, veľmi podobnom ale nekompatibilnom s RJ11 telefónnym káblom. Kocka má jednofarebný LCD displej s rozlíšením 100x60 pixelov. Pod displejom sú štyri ovládacie gombíky na navigáciu v hierarchickom menu kocky. Kocka má 32-bitové ARM7TDMI jadro Atmel AT91SAM7S265 mikro ovládač s 256KB FLASH pamäťou a 64KB RAM. Taktiež má 8-bitový Atnel AVR ATmega48 mikro ovládač, repráky s výkonom 8kHz a podporu pre bluetooth. Ako zdroj energie je Li-lon nabíjacia batéria s adaptérom alebo 6 AA bateriek (1.5V každá). [1 ]



**Obr. 2 Kocka NXT**

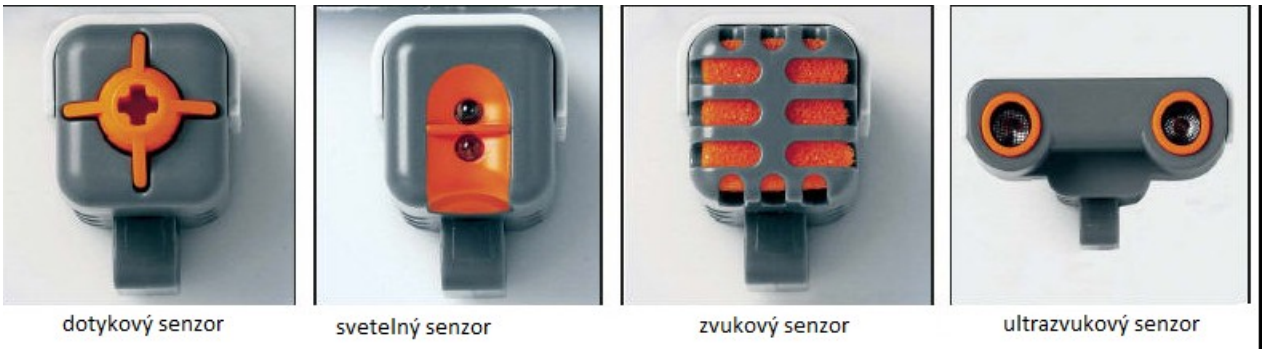
### 1.2 Senzory

**Dotykový senzor** (tlačidlo) - Slúži na zdetekovanie kontaktu s prekážkou. Pokiaľ má dotykový senzor kontakt s objektom, vracia hodnotu 1.

**Ultrazvukový senzor -** Ultrazvukový senzor meria vzdialenosť v centimetroch a inčoch. Je schopný naperať vzdialenost od 0 do 255 centimetrov s presnosťou +/- 3 cm

**Svetelný senzor -** Slúži na meranie jasu objektu pomocou rozličnej intenzity odrazeného svetla. Čim je väčšia hodnota výstupného signálu, tým je objekt tmavší (alebo nižšia odrazovosť svetla. )

**Zvukový senzor -** Zaznamenáva intenzitu zvukových vĺn, vydávaných objektom. Čím je nižšia hodnota výstupu, tým je nižšia hodnota výstupného údaja zo senzora, tým je vydávaný zvuk slabší. [5 ]

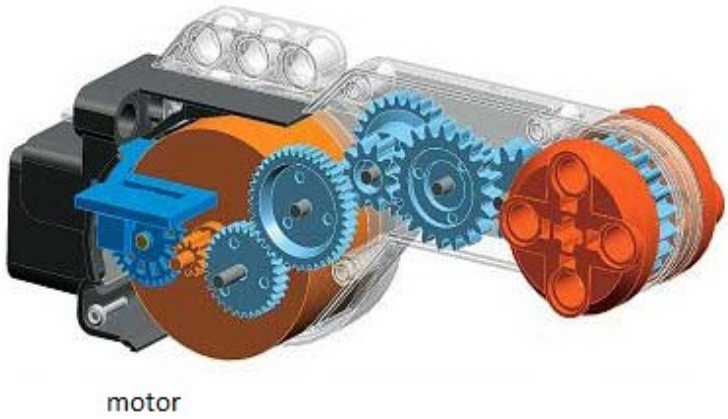



**Obr. 3 Senzor**

### 1.3 Akčný člen

**Motor -** Každý motor má vstavaný senzor otáčok. Senzor otáčok meria pootočenie motora v stupňoch alebo plných otáčkach s presnosťou +/- 1 stupeň. Jedna otáčka je rovná 360 stupňom. Pre rýchlosť otáčania sa väčšinou volia hodnoty z intervalu <0 , 100>. [5 ]

.




**Obr. 4 Akčný člen**

## 2 Software

Na vytvorenie programu, riadiaceho modela sa používa viacero softwarovo. Najčastejšie sa stretneme s RCX Code a ROBOLAB , ktorý je dostupný spolu s stavebnicou. Taktiež môžeme použiť MATLAB, NXC (Not Exactly C – opensouce C -typ programovacieho jazyka), NQC (Not Quiet C), Visual Basic, Pre účely riadenia zvoleného modela som použil MATLAB.

MATLAB je programové prostredie pre vedeckotechnické numerické výpočty, modelovanie, návrhy algoritmov, počítačových simulácií, analýzu a prezentáciu dát meranie a spracovanie signálov, navrch riadiacich a komunikačných systémov. Nadstavbou MATLAB-u je Simulink, programovacie prostredie pre simuláciu a modelovanie dynamických systémov, ktorý využíva algoritmy MATLAB-u pre numerické riešenia predovšetkým nelineárnych rovníc. [2]

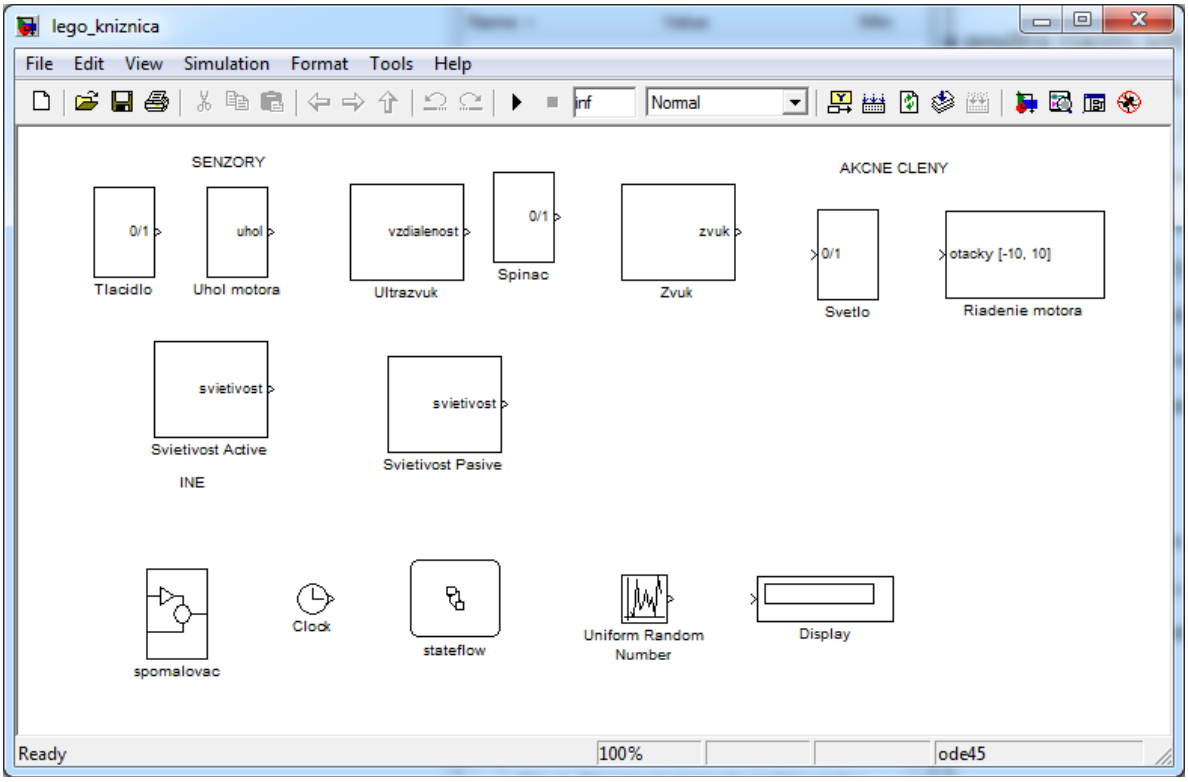
### 2.1 Simuling

Simulink slúži na simuláciu a modelovanie dynamických systémov. Pozostáva zo súboru knižníc, ktoré sa dajú v prípade potreby rozširovať (lego_kniznica). Každá z nich obsahuje špecifické bloky,ktoré môžete vidieť na obrázku 5, pomocou ktorých vieme vytvoriť blokové schémy. Jednotlivé bloky, ktoré obsahuje lego knižnica viď obr. 5, môžeme rozdeliť do troch skupín.[6]

• **Senzory**: tlačidlo, uhol motora, ultrazvuk, light, spínač, zvuk, Svietivost Active, Svietivost Pasive

• **Akčné členy**: riadenie motora, svetlo

• **Iné**: spomaľovač, stateflow, clock, display, uniform random number.

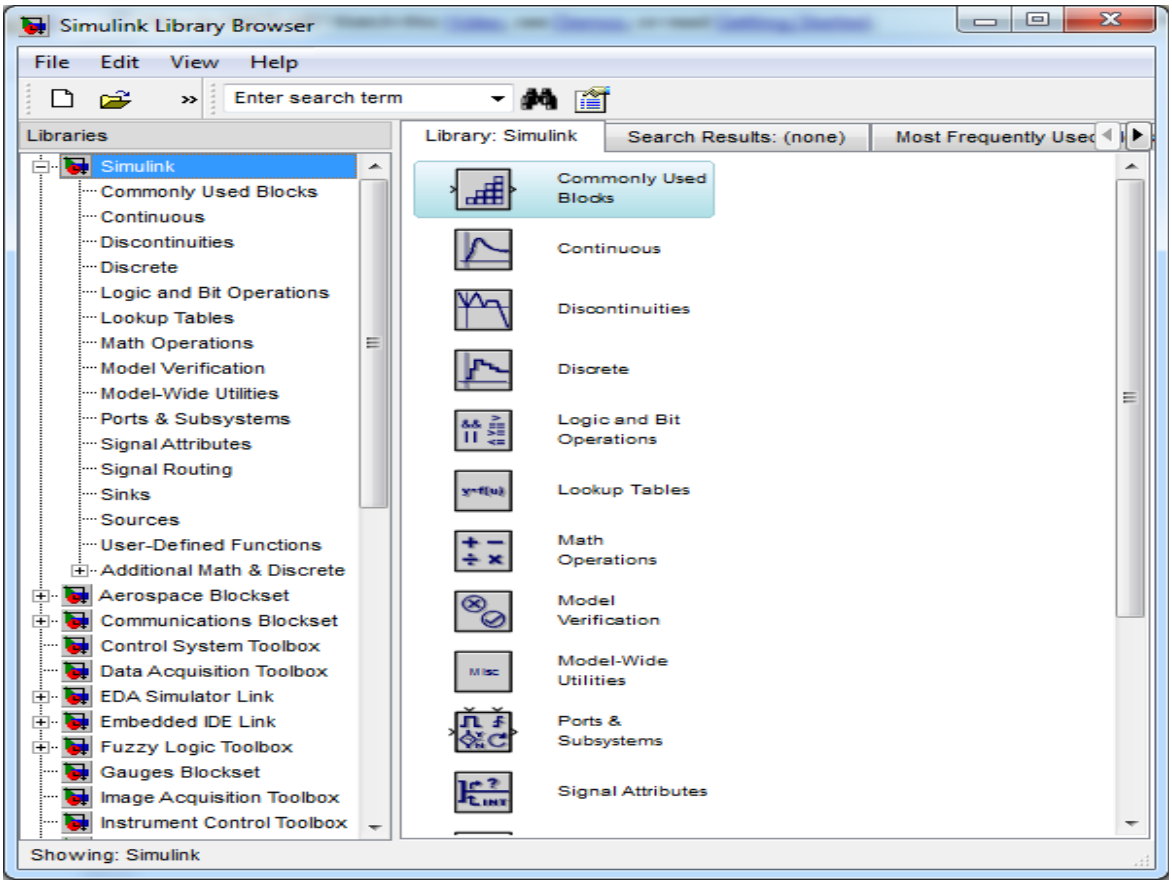



**Obr. 5 Lego knižnica**

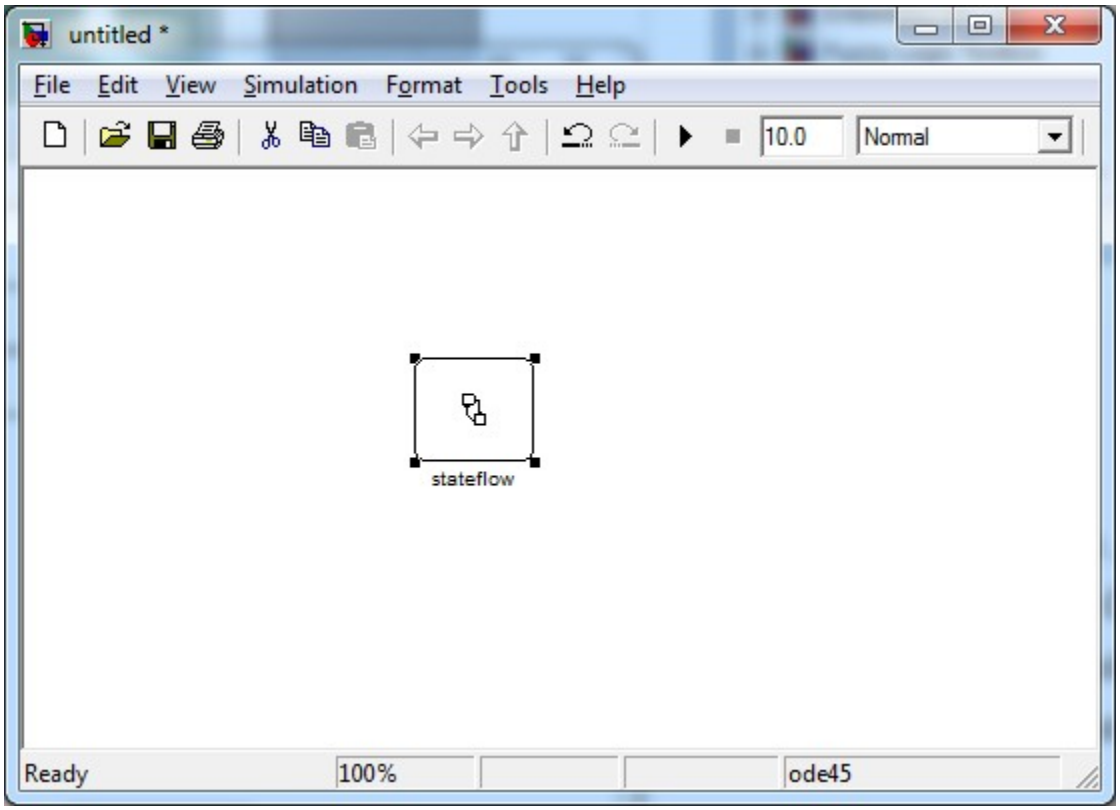
### 2.1.1 Práca v Simulinku

Vytvorenie nového modelu :

V MATLAB-e v príkazovom okne *command window* napíšeme Simulink, alebo kliknutím na tlačidlo Simulink. Spustí sa nám Simulink-ová knižnica viď obr. 6, kde môžme na pravej strane vidiet jednotlivé bloky knižnice a na ľavej strane jednotlive knižnice. V nej kliknutím na ikonu *Create a new model*, alebo stlačením kombinácie „Ctrl+N" vytvoríme nový model. Do takto vytvoreného modelu môžeme vkladať bloky z knižníc tak, že potiahneme želaný blok do nového modlu, a vytvoríme tak požadovanú blokovú schému. Z lego knižnice vyberieme blok *stateflow*, prenesie ho potiahnutím do okna nového modelu (dvojklikom na tento blok otvoríme Stateflow). Čas simulácie je prednastavený na 10 sekúnd. Po uplynutí tohto času od spustenia nastáva automatické vypnutie simulácie. Nakoľko nevieme odhadnúť čas našej simulácie, je vhodne nastaviť tuto hodnotu na nekonečno(„inf").[6]




**Obr. 6 Simulinkový prehliadač**

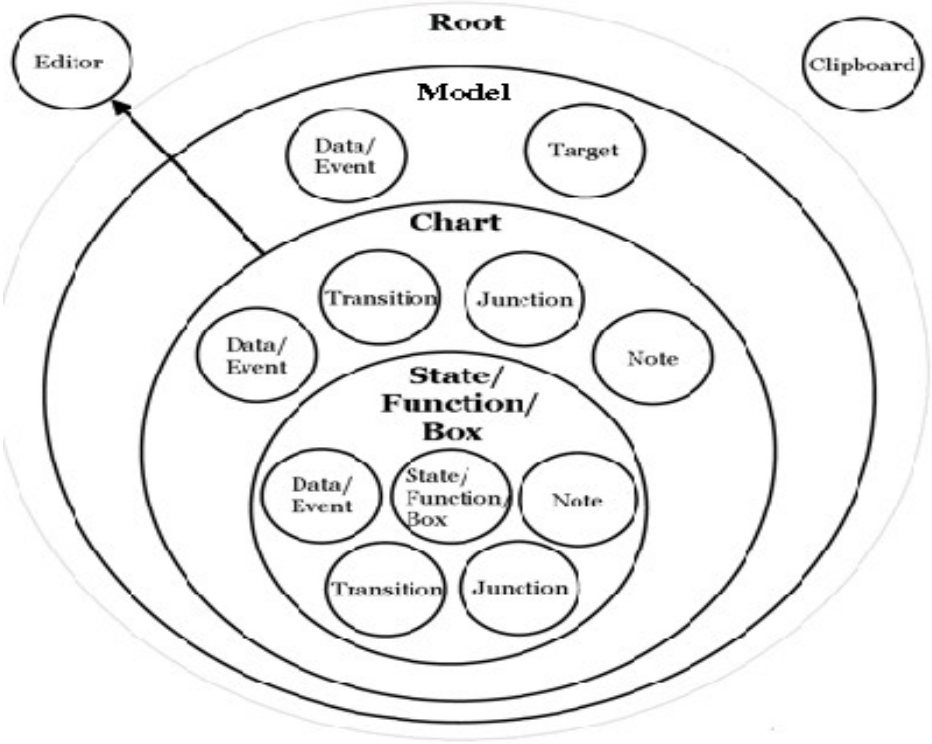



**Obr. 7 Nový model**

### 2.2 Stateflow

Stateflow používa variantu *finite-state machine* (FSM) - konečý automt. V skratke je to jednoduchý systém, ktorý môže byť v určitých preddefinovaných stavov, medzi ktorými prechádza na základe splnenia určitých podmienok. [6] Príkladom stavového automatu môže byť princíp fungovania klimatizácie. Predpokladajme, že máme klimatizáciu nastavenú na určitú teplotu napr. 20 °C. V miestnosti začne otepľovať. Senzor v zariadení zadetekuje zmenu teploty a vydá príkaz na zapnutie klímy, aby začala ochladzovať okolie. Ak sa teplota vráti na hodnotu nižšiu , alebo rovnajúcu sa požadovanej teplote, klimatizácia sa vypne.

Medzi jednotlivými stavmi existuje hierarchia viď obr.8. V Simulink-u vieme pripojiť bloky k príslušným udalostiam (vstupy a výstupy), čím zabezpečíme prepojenie medzi reálnym objektom a jeho riadiacim systémom (stateflow). Na nižšej úrovni sa nachádza schéma stateflowe-u (Stateflow diagram), ktorá je súborom stavov, uzlov ako aj prepojenia medzi nimi. Základným kmeňom sú samotne stavy a funkcienich. [3]




**Obr. 8 Anatómia stavového automatu**

Informácie získane v programe Stateflow sa využívajú aj v nasledovných odvetviach [3]:

• Programovanie a nastavovanie PLC zariadení.

• Pri vytváraní smerových algoritmov, ktoré sa používajú v telekomunikácii

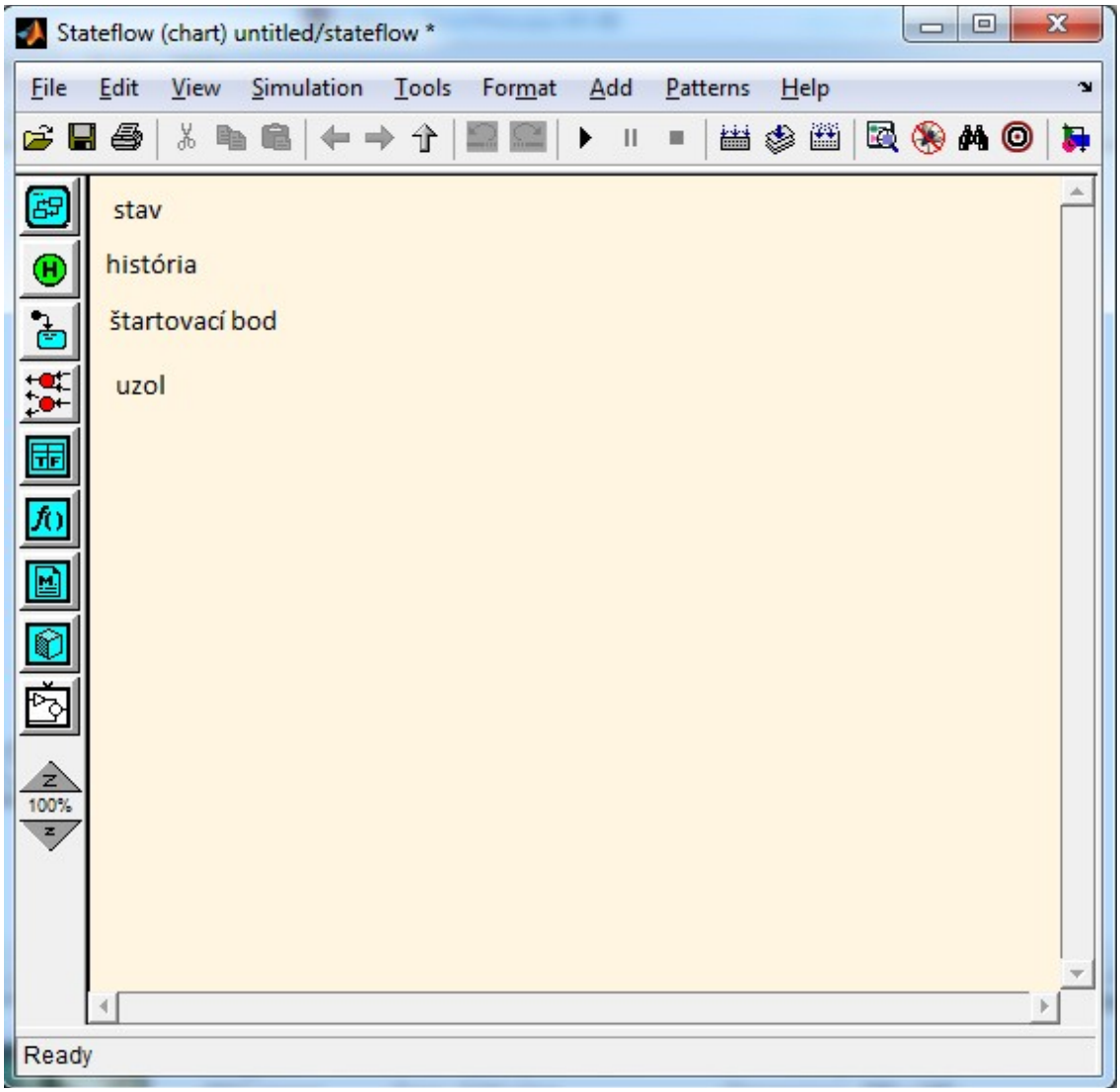
• V automobilovom priemysle (automatická prevodovka)

• Rôznych spotrebičoch (chladnička, rýchlovarná kanvica … )

• Systémy riadenia letovej prevádzky (Digital Signal Processing)

### 2.2.1 Práca so Stateflow

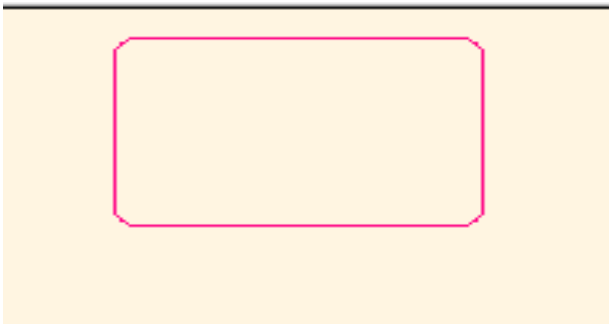
V užívateľskom prostredí Stateflowu viď obr. 9 sa na ľavej strane nachádza paleta, v ktorej si možno vybrať z ponuky: stav, história, štartovací bod, uzol, pravdivostnú tabuľku, funkciu, vloženú MATLAB funkciu a krabicu. [6]




**Obr. 9 Stateflow paleta**

### 2.2.1. 1 Stav

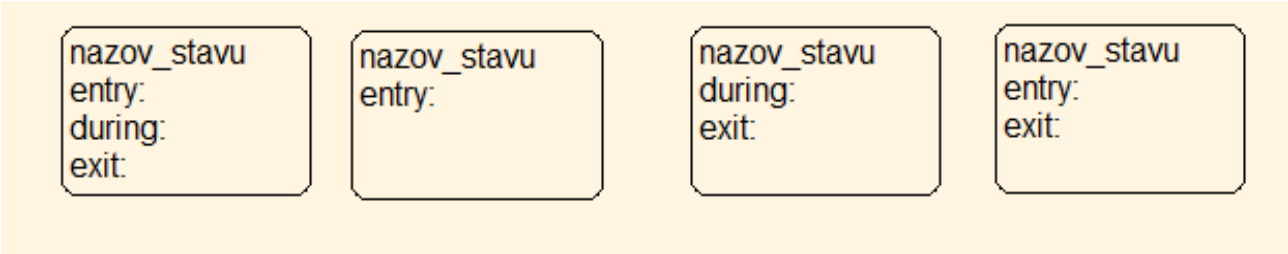
Na vytvorenie stavov sa používa prvé tlačidlo na ľavej strane. Po kliknutí na tlačidlo vieme umiestniť čistý stav na pracovnú plochu. Viď obr. 10. [6]



**Obr. 10 Nový stav**

### 2.2.1.2 Štruktúra stavu

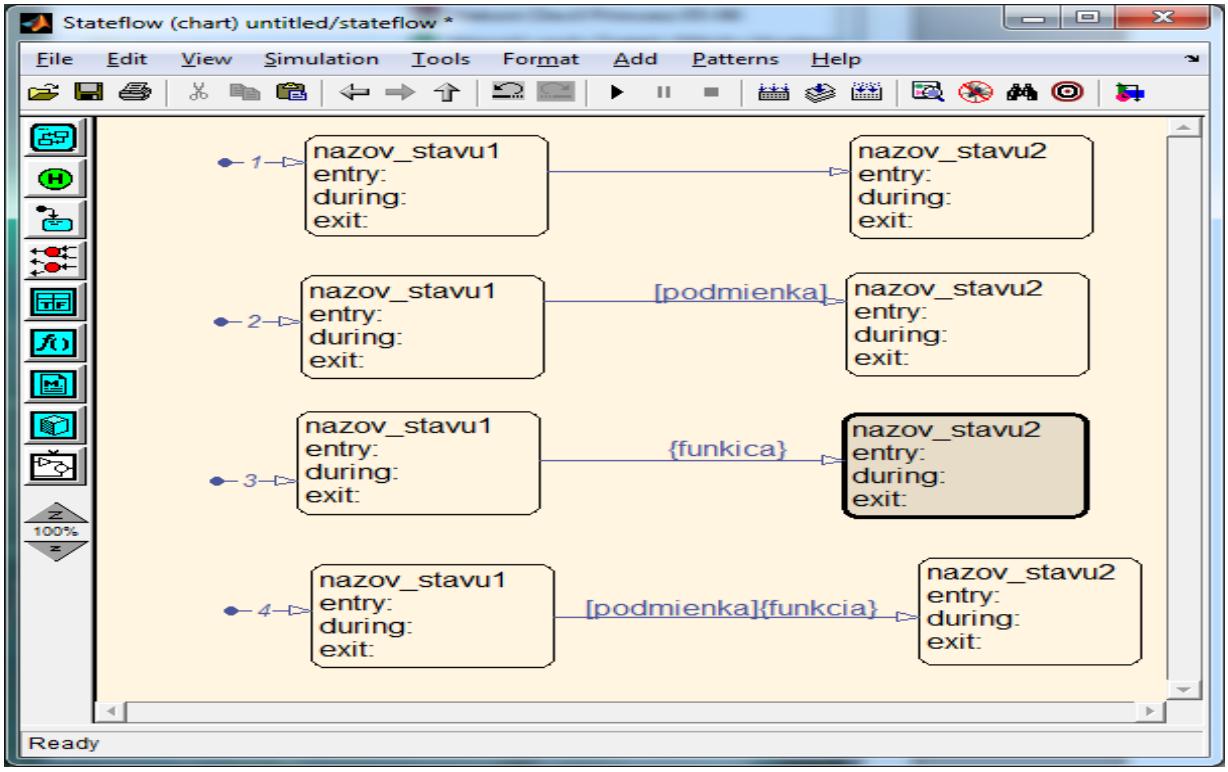
Každý stav ma svoju anatómiu. Do prvého riadku sa zadáva názov stavu, ktorý musí byť jedinečný, nemôže obsahovať diakritiku, medzeru medzi slovami a nesmie sa začínať číslom. V druhom sa zadáva *entry*:, v treťom *during*: a v štvrtom *exit:,* príkaz. Nemusia byť všetky definované, ale poradie musí byť dodržané. [6]




**Obr. 11 Štruktúra stavu**

### 2.2.1.3 Prepájanie stavov

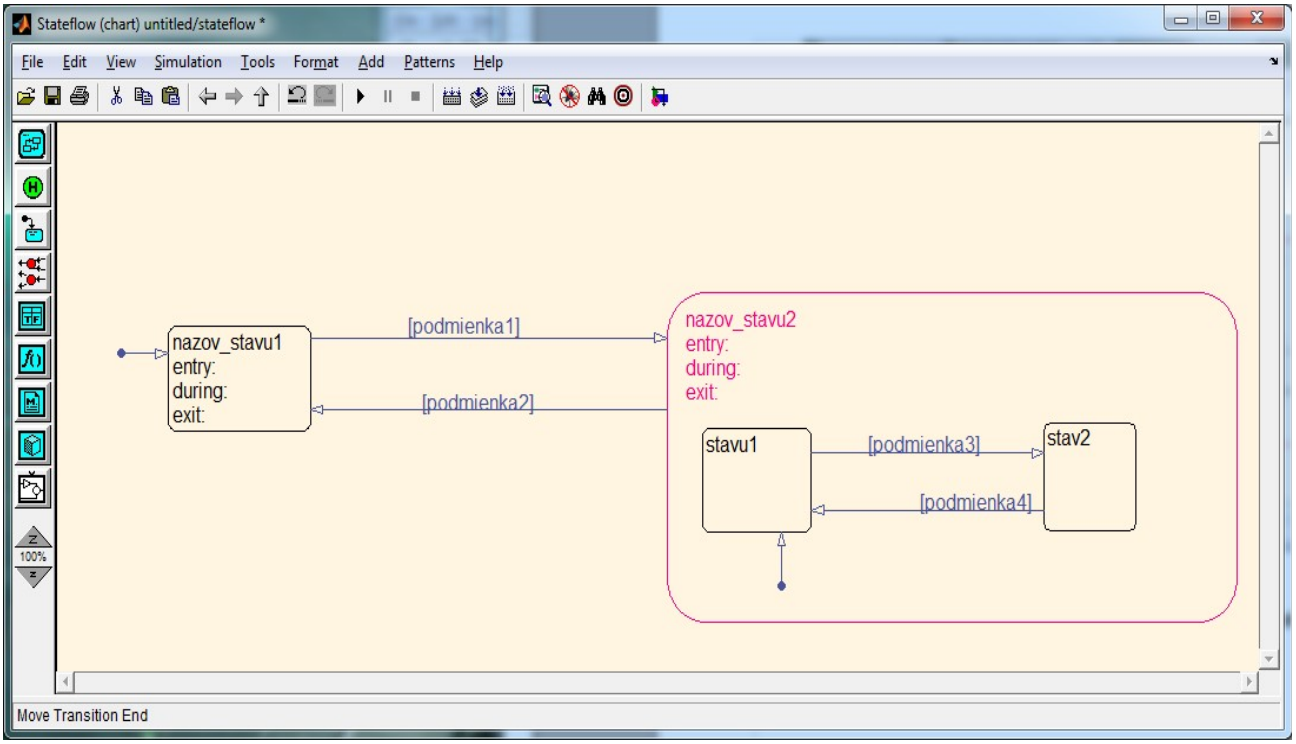
Aktivita stavov je dynamická závislosť, ktorá sa mení na základe splnenia podmienok. Prechod medzi stavmi zabezpečujú prepojenia, ktoré ukazujú smer prestupu a môžu obsahovať funkcie ({funkcia}) a podmienky na opustenie stavu ([podmienka]). [6]



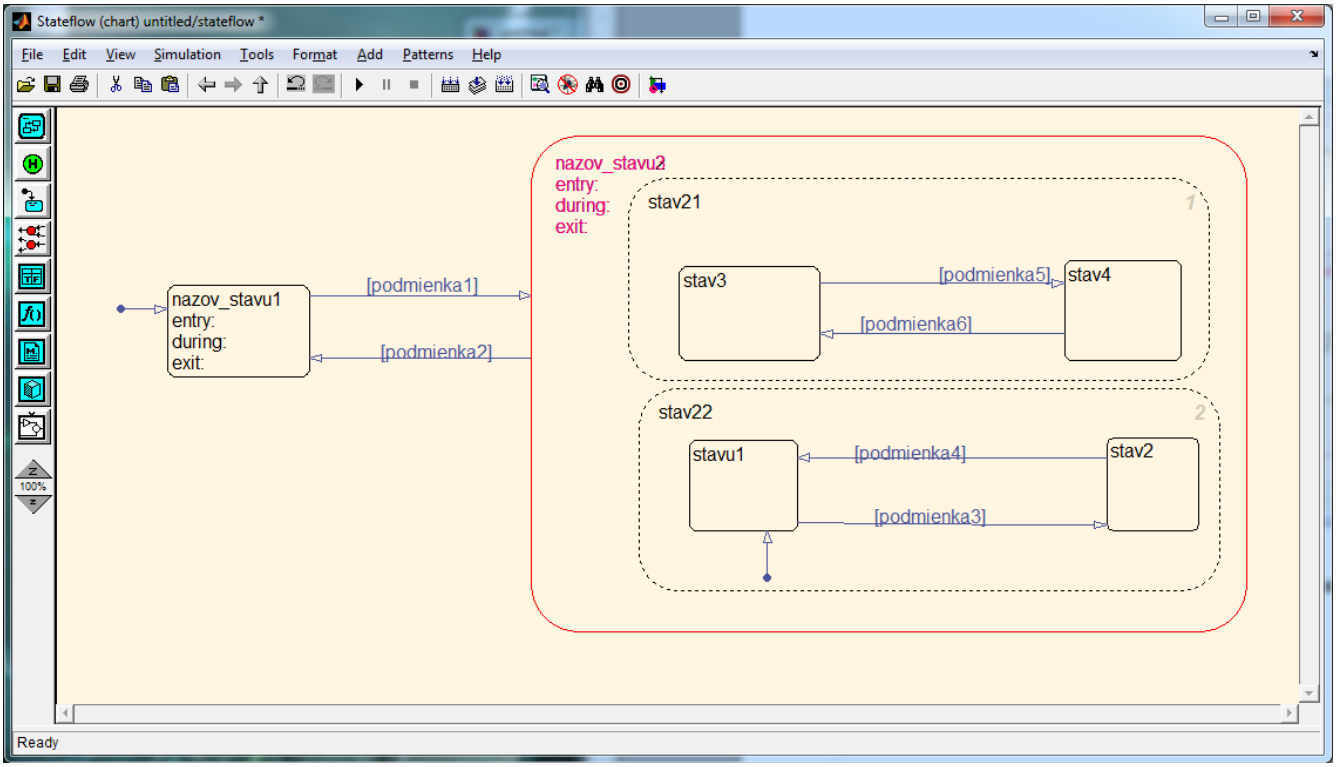

**Obr. 12 Prepájanie stavov**

### 2.2.1.4 Dekompozícia stavov

Stavy môžeme zapájať do dvoch rôznych dekompozícií. Východisková dekompozícia sa nazýva uzavretá (OR), ktorá nám dovoľuje nachádzať sa iba v jednom stave. Druhým typom je paralelná dekompozícia (AND). Toto zapojenie nám umožňuje, aby v stavovom diagrame mohlo byť aktivovaných viacej stavov súbežne. Pre paralelnú dekompozíciu klikneme pravým tlačítkom myši na bunku v ktorej chceme vytvoriť dekompozíciu tam na *decomposition* a tam *parallel* [6]




**Obr. 14 Uzavretá dekompozícia**

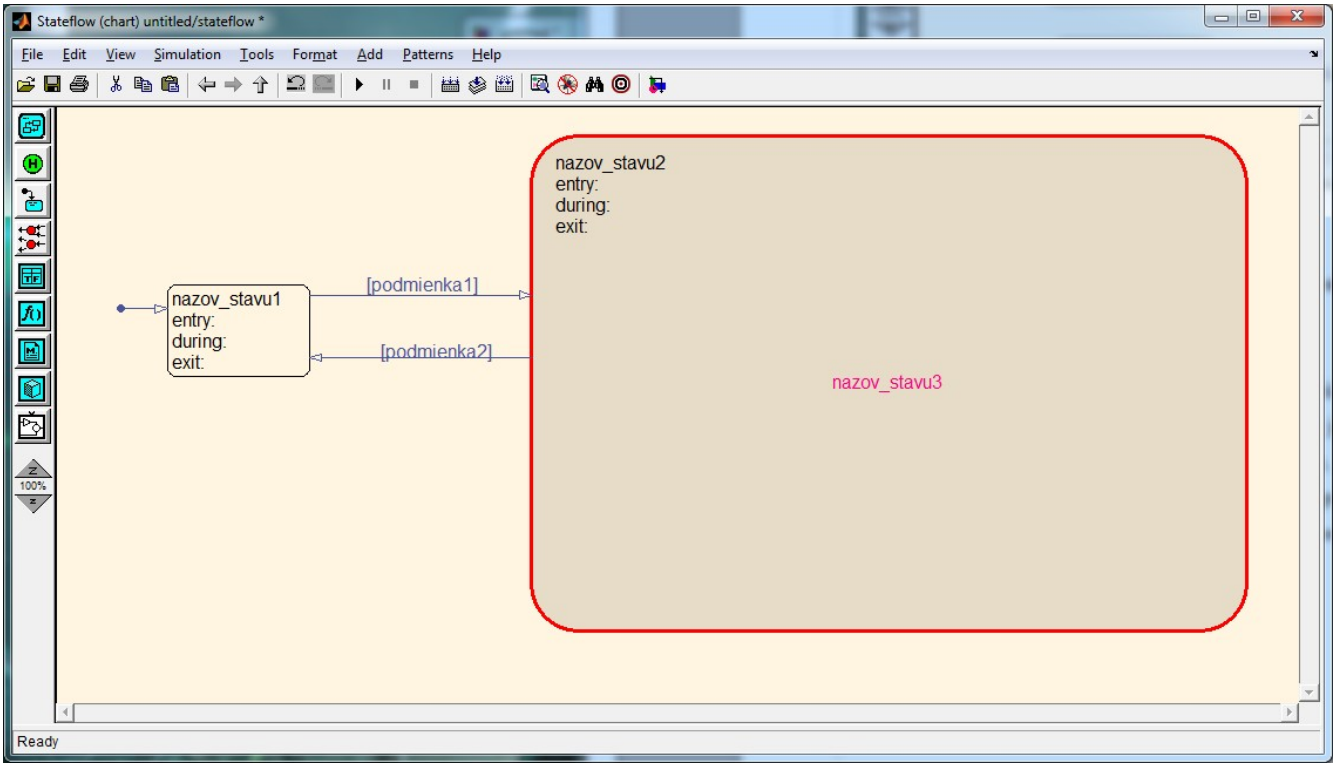



**Obr. 15 Paralelná dekompozícia**

### 2.2.1.5 Zakrytie obsahu stavov

Na zakrytie obsahu stavov nám slúži funkcia *subcharted*. Túto funkciu využívame pri zložitejších systémov, aby bol diagram prehľadnejší.

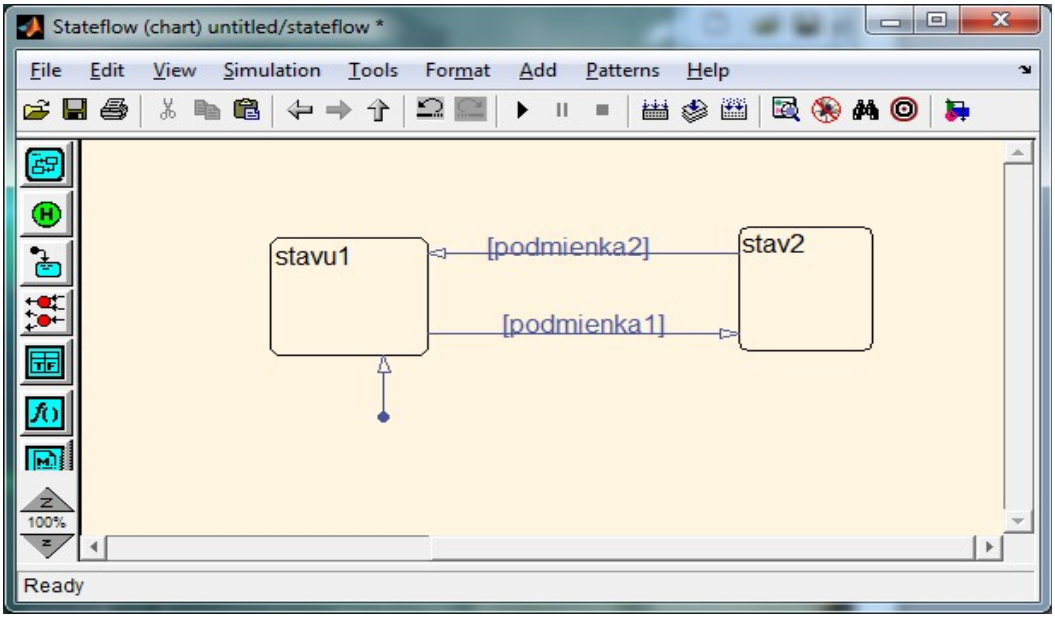
Ak chceme obsah stavu zakryť kvôli prehľadnosti tak treba kliknúť pravým na daný stav, tam na *Make content* a v nom na *Subcharted*. [6]




**Obr. 16 Zakrývanie obsahu stavov**

### 2.2.1.6 Štartovací bod

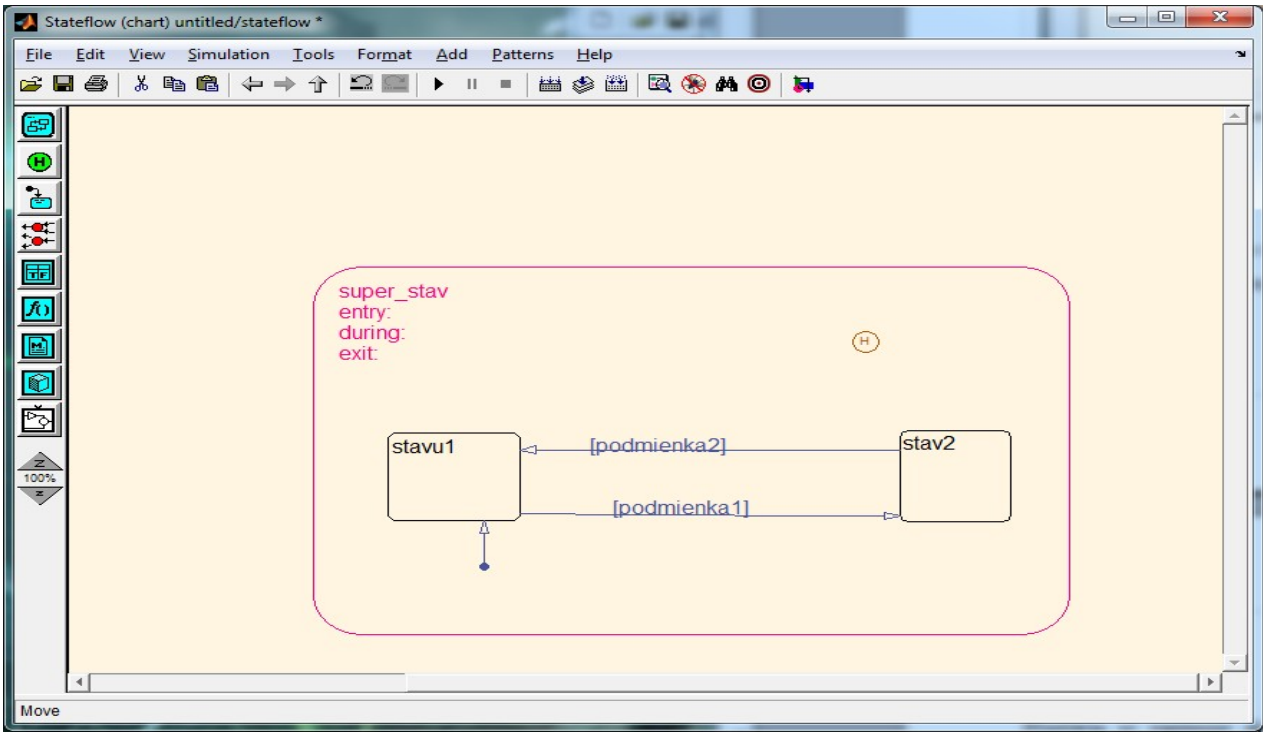
Štartovací bod slúži na určenie stavu, ktorý sa po spustení simulácie aktivuje ako prvý. Ide o prepojenie so smerom, ale bez zdroja. Prvý štartovací bod nesmie obsahovať podmienku. [6]



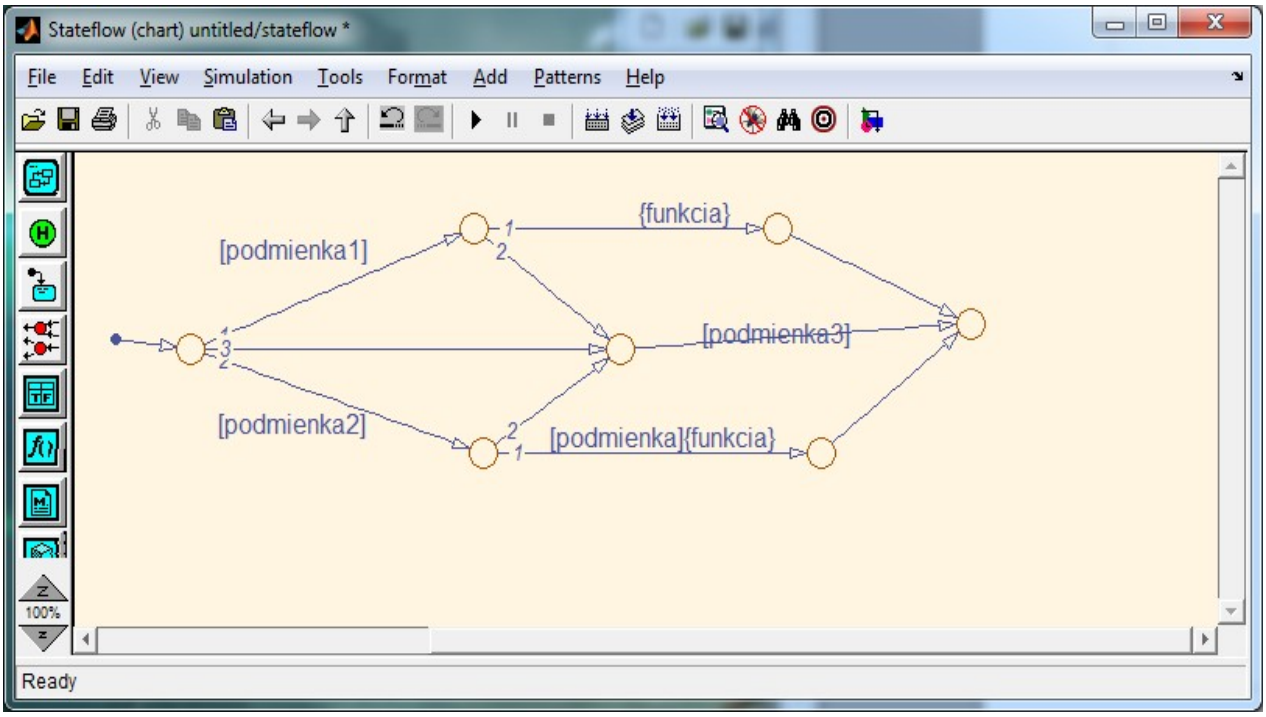

**Obr. 17 Štartovací bod**

### 2.2.1.7 História

História si zapisuje do pamäti stav, ktorý bol aktívny ako posledný pred opusteným super stavu. Vzťahuje sa len na úroveň v ktorom sa nachádza. [6]



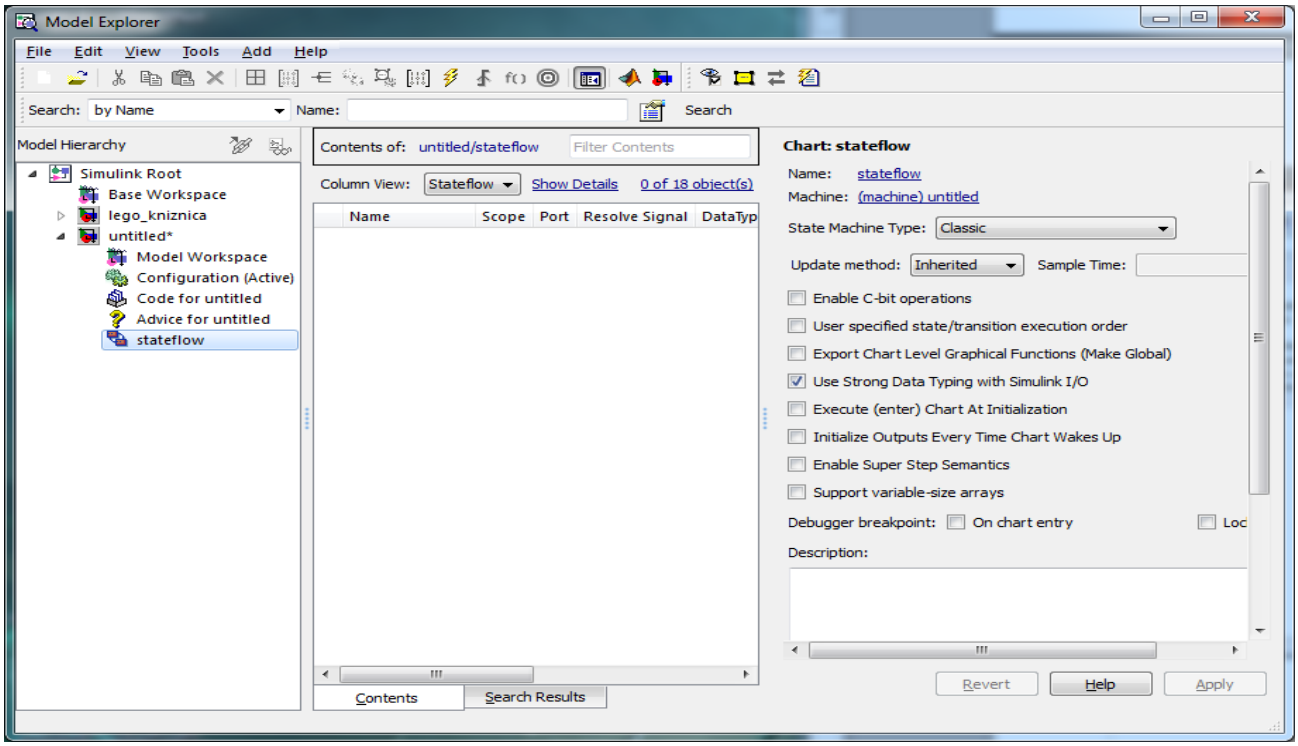

**Obr. 18 Použitie histórii v superstave**



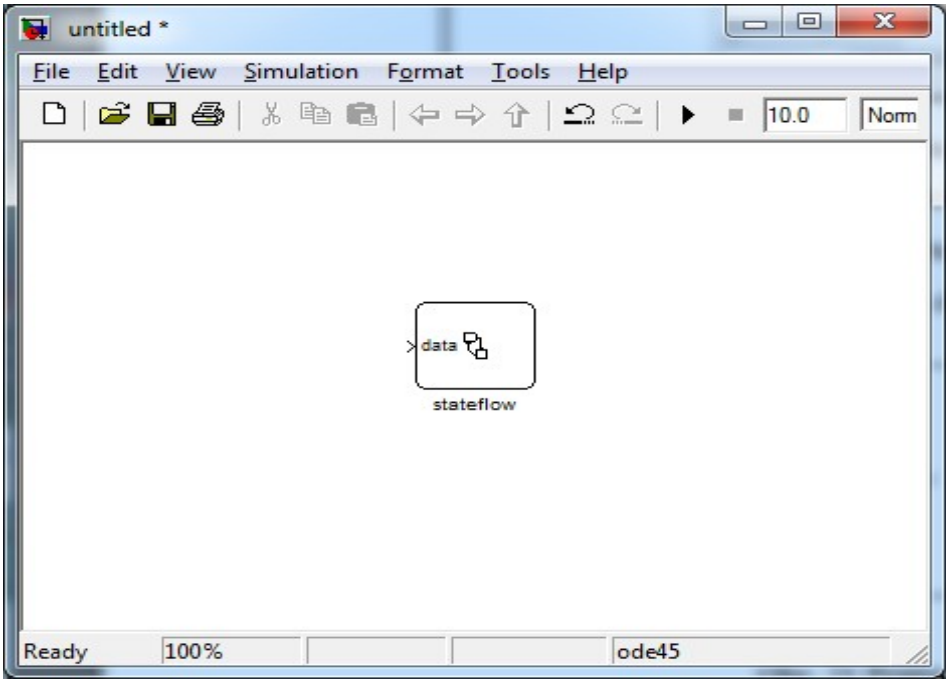

**Obr. 19 Využitie uzlov v stavovom riadení**

### 2.2.1.8 Pridávanie udalostí

Stateflow rozoznáva viac typov udalostí , najčastejšie sa používajú: vstupy, výstupy a lokálne premenné. Pridávajú sa cez nástroj *Model Explorer*. Umiestnený je v ponuke *View*, alebo ho môžeme pridať stlačením kombinácie „Ctrl+R“. Na hornej lište po kliknutí na *Add*, si vyberieme možnosť *Data* sa vytvorí udalosť s názvom „data”. V riadku „scope“ si môžeme nastaviť vlastnosti v závislosti od toho či to bude vstup, výstup či lokálna premenná. Po pridaní udalosti sa zmení blok stateflow, pribudne nám možnosť pripojenia blokov zo Simulink-u . Po pripojení môžeme spustiť simuláciu na reálnom objekte. [6]




**Obr. 20 Model Explorer**




**Obr. 21 Pridávanie udalostí v bloku Stateflow**

## 3. Inštalácia MATLAB-u a spustenie lego-robota

Pre spustenie lego-robota a následné riešenie danej úlohy v prostredí MATLAB-u je potrebné, aby na počítači bol nainštalovaný MATLAB, ktorý je kompatibilný so software-om operujúcim lego kocku. Operačný systém, pod ktorým táto práca bola vypracovaná je Windows 7 Ultimate s Servis Pack 1. Nižšie je stručne popísané ako bol MATLAB spustený a problém kompatibility medzi NXT kockou a nainštalovaným MATLAB-om riešený.

Najprv potrebujeme aby náš operačný systém rozoznal lego kocku ako prídavné zariadenie. Na to je treba nainštalovať program NXTUSBDriver_64bit (64bit kvôli tomu, že náš systém je 64 bitová verzia) daný program sa dá stiahnuť na stránke : http://www.robotc.net/download/nxt/ . Inštalácia programu prebieha nasledovne: Spustíme .exe súbor, ktorý sa nám stiahol do počítaču a riadime sa podľa inštrukciami, ktoré sa zobrazia pri inštalácii daného programu. Po nainštalovaní programu je potrebne nainštalovať Matlab. Matlab sa dá zakúpiť na stránke:
http://www.mathworks.com/academia/student_version/. Ak máme cd-nosič, tak sa nám automaticky spustí inštalačný program, v ktorom postupujeme podľa pokynov. Ak sa náhodou automaticky nespustí inštalácia, spustíte ju manuálne.

Po inštalácii Matlabu sa nám vyskytol problém. Verziu ktorú som nainštaloval ako prvú bola verzia MATLAB R2012a. Bohužiaľ, táto verzia nebola kompatibilná s NXT kockou. V minulosti som pracoval na počítači bola verziu R2010a. S touto verziou, ktorá bežala pod Windows XP Servis Pack 3 nebol žiaden problém. Robot reagoval na povely, i Stateflow išiel korektne, ale verzia pre Windows 7 nebola kompatibilná s kockou.

Po viacerých neúspešných pokusoch s rôznymi verziami som zistil, že najjednoduchším spôsobom, ako spustiť lego-robota pod Windows 7 je najprv inštalácia software-u MATLAB (konktrétne verziu R2010a ) spolu s nahratím potrebných lego knižníc na počítač s staršou verziou operačného systému Windows ako napr. Windows XP a následné úplné skopírovanie takto nainštalovaného MATLAB-u do počítača pracujúceho pod operačným systémom Windows 7.

**4. Vytvorenie senzora**

V prípade, ak sa v lego knižnici nenachádza potrebný senzor, je možne si ho vytvoriť viacerými spôsobmi. Jedným zo spôsobov je nájsť senzorový blok, ktorý dáva pôvodný výstup ako nami požadovaný senzor a upraviť ho, aby zodpovedal požiadavkám pre potrebnú funkciu. Najlepší spôsob ako zvoliť vhodný riadiaci blok na úpravu je pozrieť obsah m-súboru, v ktorom je požadovaná funkcia. M-súbor je jednoduchý textový dokument kde ukladáte MATLAB-ovske príkazy. Ak je súbor spustený MATLAB načíta príkazy a vykoná ich akoby boli jednotlive príkazy napísane separátne v príkazovom okne. Meno každého m-súboru musí byt zakončené príponou ".m" (test.m). V ňom je podrobne popísaná daná funkcia a taktiež vysvetlený jej priebeh. V tomto konkrétnom postupe je treba mať lego kocku pripojenú ku počítaču a zapnutú. Ak nie je lego kocka pripojena a zapnutá MATLAB vypisuje chybové hlasenia. Spustíme Matlab a v ňom napíšeme do príkazového okna(Command Window) lego_kniznica. Otvori sa nám lego knižnica a v nej klikneme pravý tlačidlom myši na vhodný riadiaci blok. Vysunie sa zoznam možných akcií. Tam vyberieme *look under mask.* Klikneme pravým tlačidlom na *matlab function* vyberieme *explore* , zobrazí sa nám okno *Model Explore,* kde zistime, ktorú funkciu daný senzor volá. Treba ju vyhľadať a zobraziť ju. V nej zistíme, kde sa nachádza funkcia v počítači, ktorá ovláda daný senzor. V tomto prípade som zistil, že sa v danom súbore nachádza už preddefinovaná funkcia.

Pre vytvorenie svetelného senzora sa javil ako najvhodnejší ekvivalent ultrazvukový senzor. Vytvorili sme jeho kópiu a premenovali sme ho. Vo funkcii nášho nového senzora sme zmenili obsah, bude volať novú funkciu. Teraz klikneme na senzor, vyberieme *look under mask* potom *Matlaba function*, tam zmeniť *Matlab function* na našu novo vytvorenú funkciu. Dané zmeny sme uložili. Ďalej klikneme pravým tlačidlom, vyberieme *edit mask,* potom *initialization*, kde prepíšeme v *switch senzor_id* v jednotlivých *case OpenUltrasonic* na *OpenLight* a taktiež treba pridáme stav pre každý senzor, ktorý môže byť *active* alebo *inactive.*

**5.Samo-riadiace vozidlo**

Cieľom úlohy je postavenie vozidla a vytvorenie programu, ktorý mu umožní si nájsť vyznačenú trasu a ísť po nej. Na uskutočnenie tejto úlohy máme ku dispozícii stavebnicu Lego Mindstorms NXT na zostrojenie vozidla a programovacie prostredie MATLAB konkrétne jeho nadstavbu Simulink. Program bude uskutočnený pomocou štruktúry stateflow.

Samotná práca bude rozdelená na 4 časti :

1. Analýza samotného problému
2. konštrukcia vozidla
3. Inštalácia software-u
4. Vytvorenie riadiaceho programu pomocou stateflow
5. Samotná simulácia programu

### 5.1 Analýza problému

Úlohou analýzy je vytvorenie osnovy programu, vďaka ktorej budeme vedieť, aké senzory a akčné členy budeme potrebovať. Dané poznatky budú ďalej využité pri riešení tejto o úlohy.

Úlohu som rozdelil na dva hlavne podúlohy. V prvej časti sa budem zaoberať problémom, v ktorom ubme riešiť, aby tripod našiel cestu a vedel sa po nej vydať. To znamená, že senzor umiestnený v prednej časti na tripode bude hľadať preddefinovaný signál. Ak ho nájde tak vyšle impulz do motorov a oni sa pohnú tripodom daným smerom, pokiaľ senzore bude zaznamenávať požadovaný signál. A opačnom prípade sa tripod prepne do vyhľadávacieho stavu a ostane v ňom pokiaľ znovu nenájde „cestu“ signál, alebo ho stlačením tlačidla neuvedieme do pokojového stavu.

Táto časť úlohy sa bude realizovať za podmienky, že tripod sa nachádza na ceste, to znamená, že signál je v o oblasti 360 ° od tripodu.

Druhá časť pozostáva z nadstavby na prvú časť. V druhej časti mame situáciu v ktorej je pred tripodom prekážka predstavovaná guľou. Tripod bude mať za úlohu zastaviť, danú prekážku odstrániť a potom postupovať v ceste ďalej.

#### 5.1.1 Analýza prvej úlohy

Štartovacím bodom bude stav, pri ktorom bude vozidlo v pokoji, tj akčné členy budu vykazovať hodnotu 0.

Po stlačení tlačítka, čo v našom prípade znázorňuje naštartovanie vozidla, merací člen (svetelný senzor) zistí, či sa nachádza na zvolenej trase, alebo nie.

Ak sa nachádza, aktivujú sa motory a vozidlo sa pohne ďalej. Ak senzor nebude mať požadovanú odozvu, nenachádza sa na zvolenej trase, vozidlo sa prepne do vyhľadávacieho módu. V ňom sa vozidlo snaží nájsť požadovanú trasu.

Z tejto krátkej analýzy mi vplynulo, že budeme potrebovať pohon, ktorý bude zabezpečovaný motorom. Náš model bude používať 2 motory, aby nedošlo ku preťaženiu motora, navyše bude potrebne ovládanie oboch kolies nezávisle z dôvodov korektného riadenia vozidla. Taktiež budeme potrebovať tlačilo a svetelný senzor.

### 5.1.2 Analýza druhej úlohy

Základom je požiadavka z prvej úlohy, a to je, aby vedel tripod postupovať po vyznačenej ceste. K tejto úlohe sa pripojí požiadavka, aby vozidlo bolo schopné detektovať prekážku v jeho ceste a následne bolo možné danú prekážku odstrániť.

Z toho nám vyplýva, že na už existujúcu konštrukciu musíme pripevniť servomotor s pohyblivým ramenom, rovnobežný s osou vozidla a rameno musí byť a ultrazvukový alebo svetelný senzor

## 5.2 Konštrukcia vozidla

Nakoľko sme potrebovali 2 nezávisle motory tlačidlo a svetelný senzor , ktoré budu pripojene na riadiacu jednotku (NXT kocku), potrebovali sme stabilnú konštrukciu.

Tejto požiadavke plne vyhovovala už preddefinovaná konštrukcia, ktorá bola v pridanom návode k stavebnici.

Hlavnou časťou vozidla je riadiaca kocka, ktorá je upevnená na konštrukcii, pozostávajúcej z dvoch prepojených servomotorov, nezávisle pohybujúcich sa od seba, na ktorých sú upevnené kolesá. Stabylitu celého vozidla – tripodu zabezpečuje tretie koleso, ktoré je sa voľne otáča. Svetelný senzor je umiestnený v prednej časti tesne nad zemou, pred servomotormi, na osi tripodu.

V druhej časti úlohy sme pripevnili na tripod ďalší servomotor na ktorom je upevnené rameno, ktoré slúži na odstránenie prekážky pred zariadením. Ďalej ultrazvukový detektor, ktorý slúži na zistenie prekážky v ceste vozidlu.

Tripod

**Obr.22 Tripod úloha 1**

Tripod nádstavba

**Obr.23 Tripod úloha 2**

## 6 Úloha 1 : Hľadanie cesty

V prvej úlohe je vozidlo postavené na cestu. Vozidlo je postavené tak, že senzor nedetekuje žiadnu cestu. úlohou je aby vozidlo cestu našlo a videlo sa po nej.

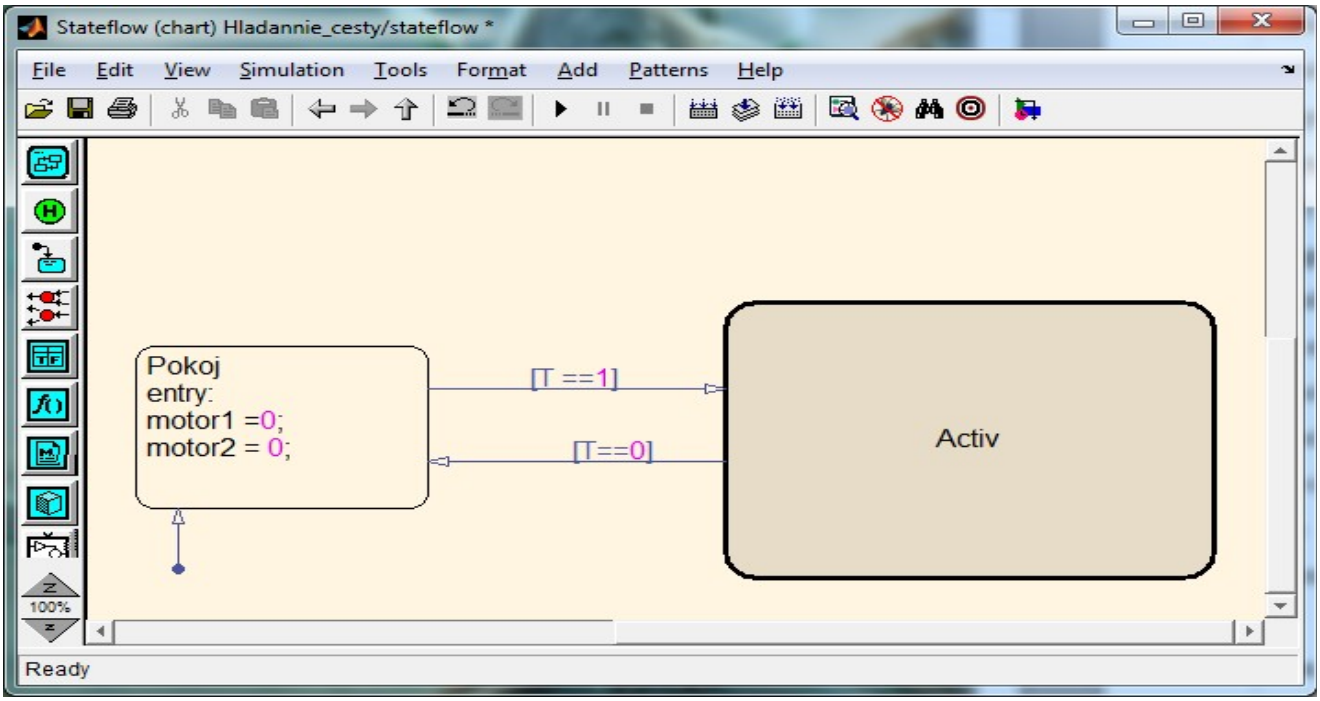
### 6.1. Pokoj

Vozidlo je najprv v pokojovom stave, obidva motory sú na hodnote nula. Po stlačení dotykového senzora, čo nám predstavuje naštartovanie vozidla sa tripod prepne do stavu Activ. V stave Activ sa nevykoná *entry* ale hneď sa prejde do stavu Hlada. (Obr. 23)

- Stav Pokoj

  -v tomto stave sa uvedú motory do pokojového stavu

  - je to východiskový stav a program sa do neho bude vracať pri spúštaní stavu, v ktorom sa budu obidva motory hýbať.




**Obr. 24 Pokoj a Activ**

## 6.2 Activ

V tomto stave sa vykonajú príkazy Hlada a Ide, ktoré sú podstavy stavu Activ. (Obr. 24)

V stave Hlada sa tripod pomaly začne otáčať, motory sa otáčajú do opačných smerov. Počas otáčania bude program kontrolovať, či výstup zo senzora nenabudol hodnoty opísané v podmienke 1 . V prípade, že senzor vyšle signál, že našiel cestu, čo je interpretované ako výstup z senzora nadobudne požadované hodnoty, ako je v podmienke, program prejde do stavu Ide, kde sa obidva motory začnú otáčať rovnakým smerom.

Z tohto stavu sa opäť pokračuje do stavu Hlada a to v prípade ak je splnená podmienka 2, čo znamená, že senzor nedetekuje nami vytvorenú cestu.

**podmienka 1 :** [svetlo >400 && svetlo < 550 ]

**podmienka 2 :** [svetlo < 400 || svetlo > 550]

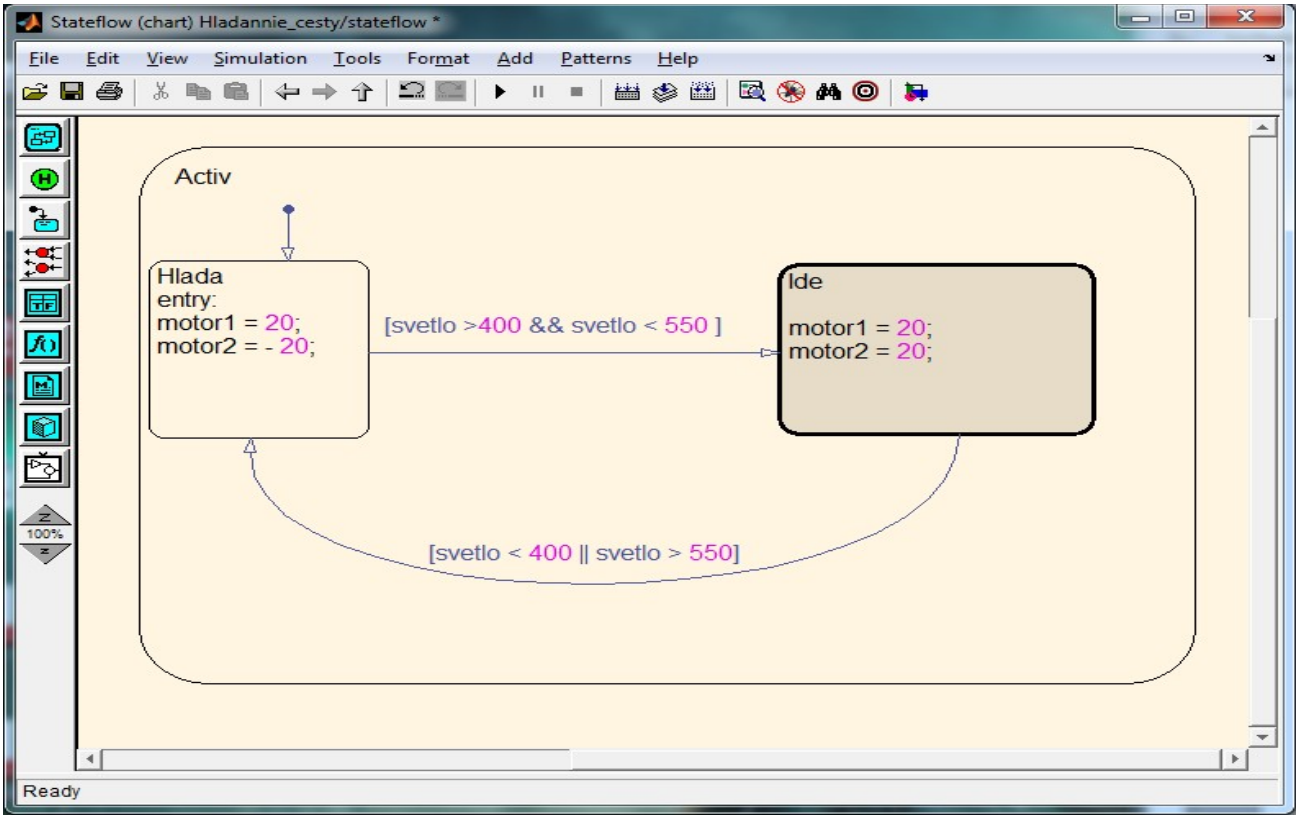
- Stav Hlada

  v tomto stave sa tripod bude otáčať okolo vlastnej osi. čo je zabezpečené príkazom motor1 = 20;motor2 = -20;

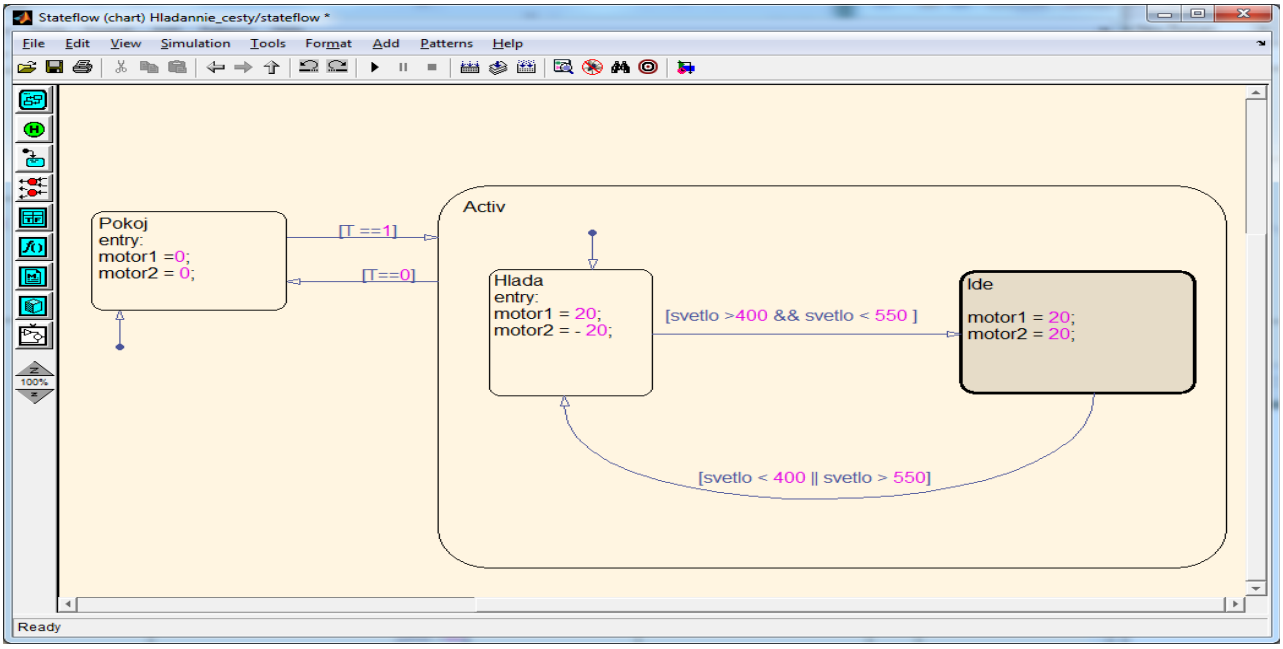
- Stav Ide

  -tu sa uskutoční príkaz, aby sa tripod vydal rovno

  - daná činnosť je zabezpečená príkazom motor1 = 20;motor2 = 20;




**Obr. 24 Hlada a Ide**




**Obr. 25 Stavový diagram celého riešenia úlohy1**

### 6.3 Simulink

Do Simulink-u sme definovali všetky udalosti (Obr. 26) :

vstupy : svetlo, T

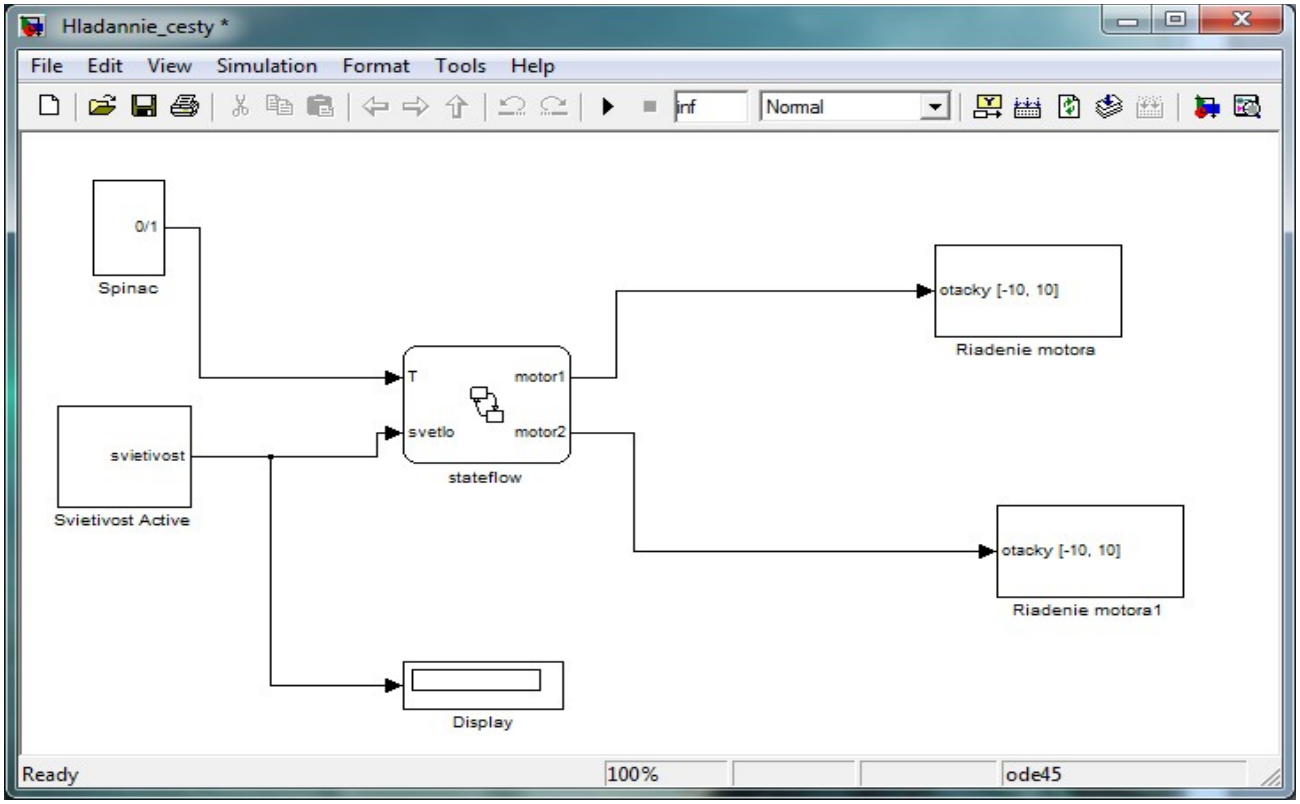
výstupy: motor1, motor2

Bloky z knižnice, potrebne na riešenie úlohy 1 :

Akčné členy : Riadenie motora , Riadenie motora1

Senzory : Svietivost Active, Spinac

Iné : Display




**Obr. 26 Simulink-ová schéma úlohy 1**

# 7 Úloha 2 : Odstránenie prekážky

V tejto kapitole sa budeme zaoberať len programom, v ktorom bude tripod musieť vykonať činnosť odstránenia prekážky z jeho cesty a následné pokračovanie v chode, nakoľko úloha 2 je nadstavbou na úlohu 1, ktorá je bližšie popísaná v predchádzajúcej kapitole.

## 7.1 Prekazka

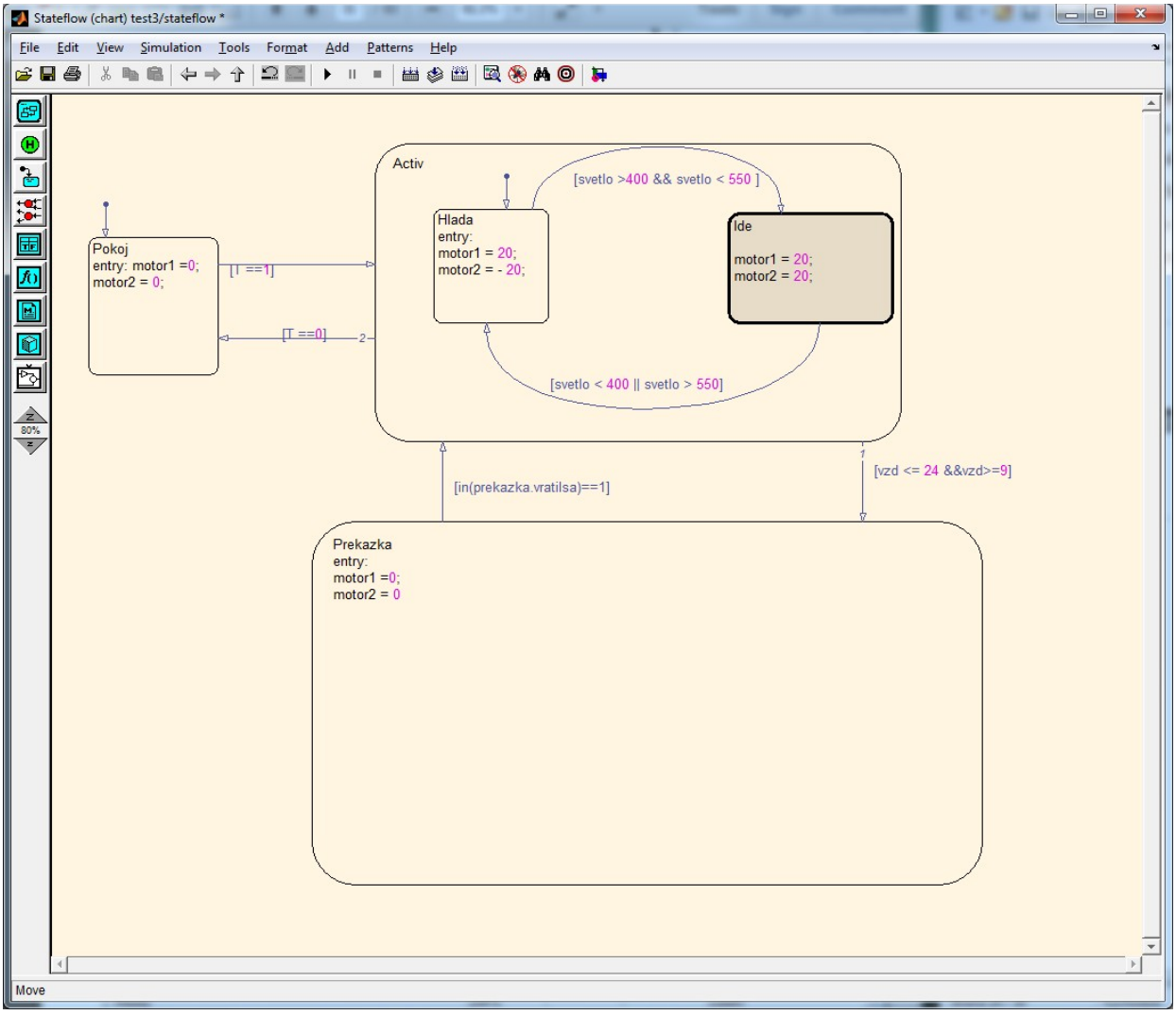
Program sa presunie do stavu Prekazka, ak je splnená podmienka 3. Fyzická interpretácia tejto podmienky je, že pred tripodom je prekážka.

V tomto stave sa vykonajú príkazy, ktoré zabezpečia odstránenie prekážky.

- Stav Prekazka

  -v tomto stave sa zastaví chod motorov

  - výstup z motor1 a motor1 bude vykazovať hodnotu nula




**Obr. 27 Odstránenie prekážky**

podmienka 3 : [vzd <= 24 &&vzd>=9]

podmieka pre prechod do stavu, v ktorom sa bude riešiť problém prekážky

podmienka 4 : [in(prekazka.vratilsa)==1]

podmienka pre návrat do stavu pre hľadanie cesty

Odstránenie prekážky :

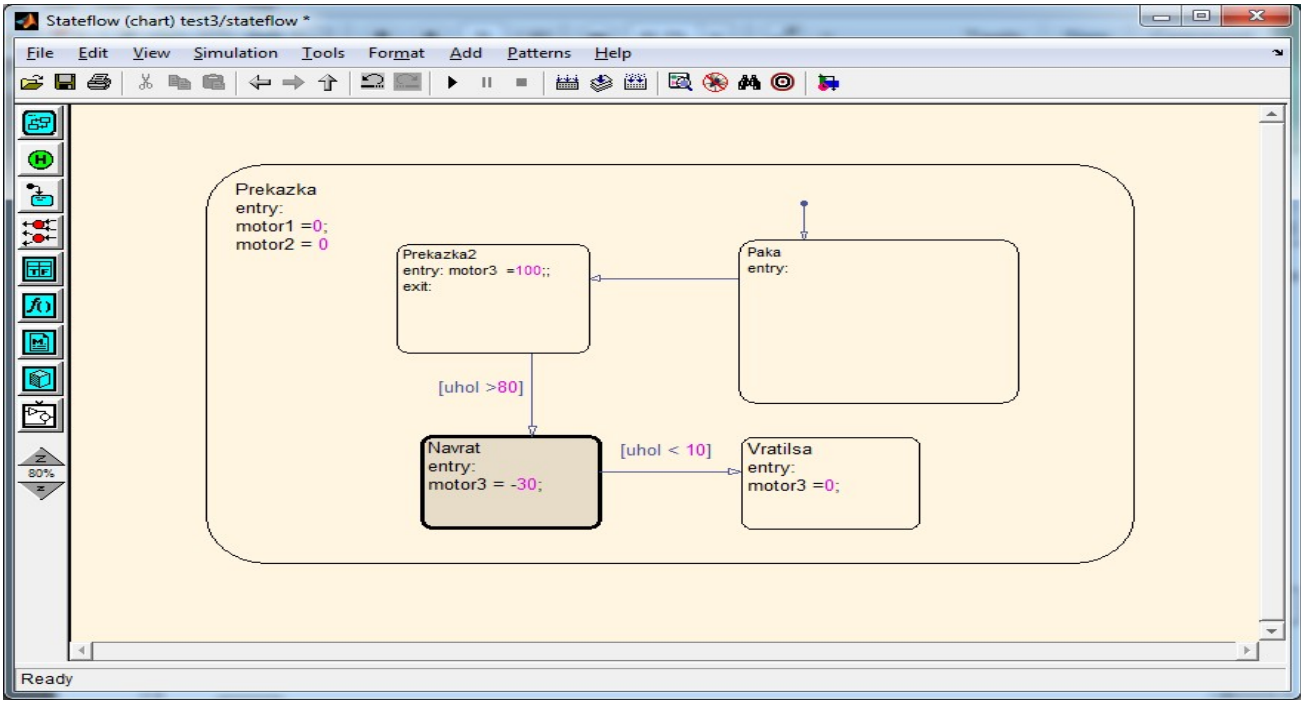
- stav Paka :

  v tomto stave sa nevykoná nič. Hneď sa prejde do stavu Prekazka2
- stav Prekazka2

  v tomto stave sa uvedie do chodu tretí servomotor, ktorý uvedie do pohybu rameno, ktoré odstráni prekážku v podobe guľe.
- Stav Navrat

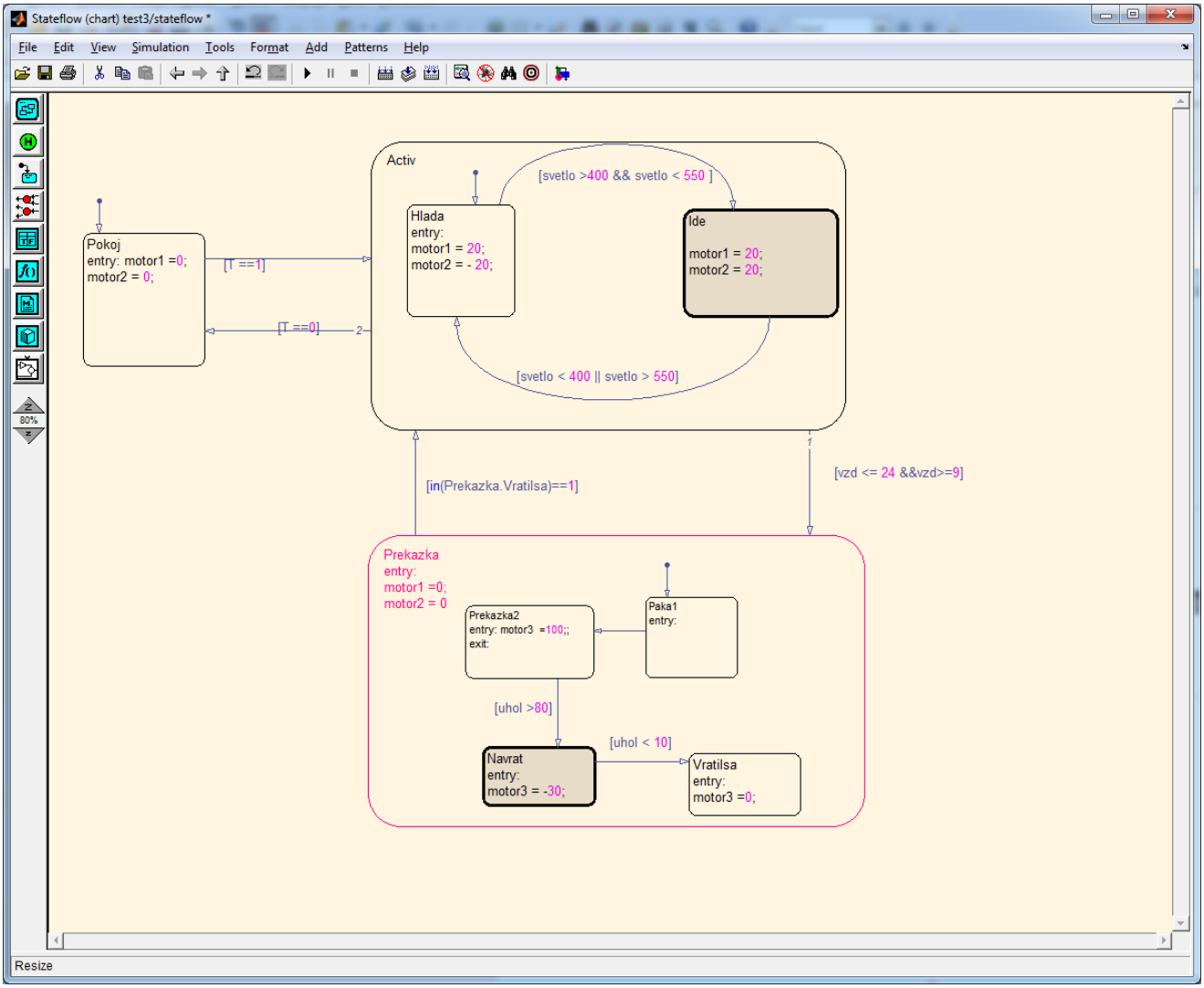
  ak dosiahne uhol pootočenia motora3 vyššiu hodnotu než 80°, vráti sa rameno do pôvodnej polohy
- Stav Vratilsa

  -tento stav nám slúži na to, aby zastavil motor3 a zároveň upozornil náš program na to, že prekážka bola odstránená a vozidlo môže pokračovať v ceste ďalej




**Obr. 28 Stavový diagram odstránenia prekážky**

## 7.2 Simulink




**Obr 29. Stavový diagram riešenia úlohy 1 a 2**

Do Simulink-u sme definovali všetky udalosti :

vstupy : svetlo, T, uhol , vzd

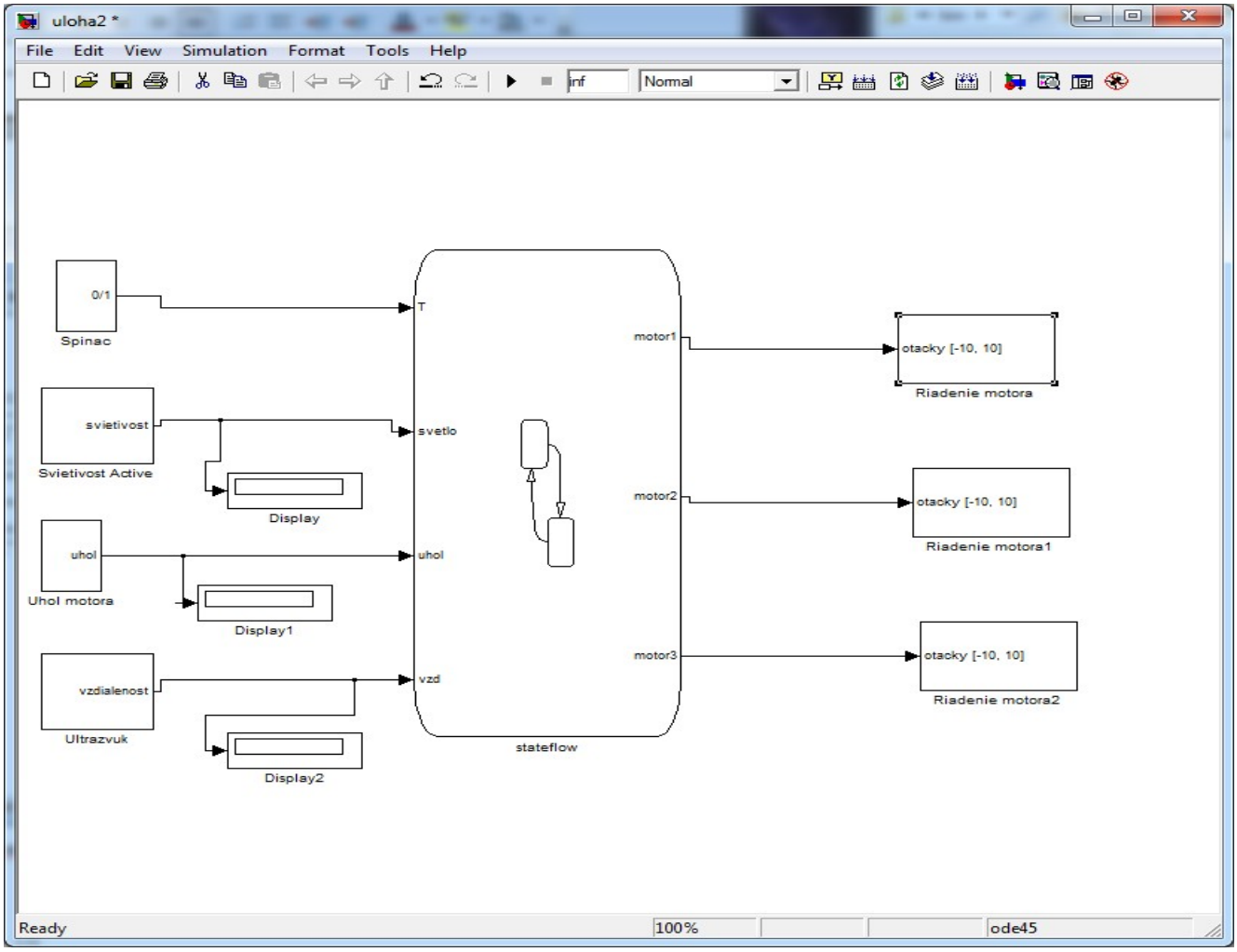
výstupy: motor1, motor2, motor3

Bloky z knižnice, potrebne na riešenie úlohy 1 :

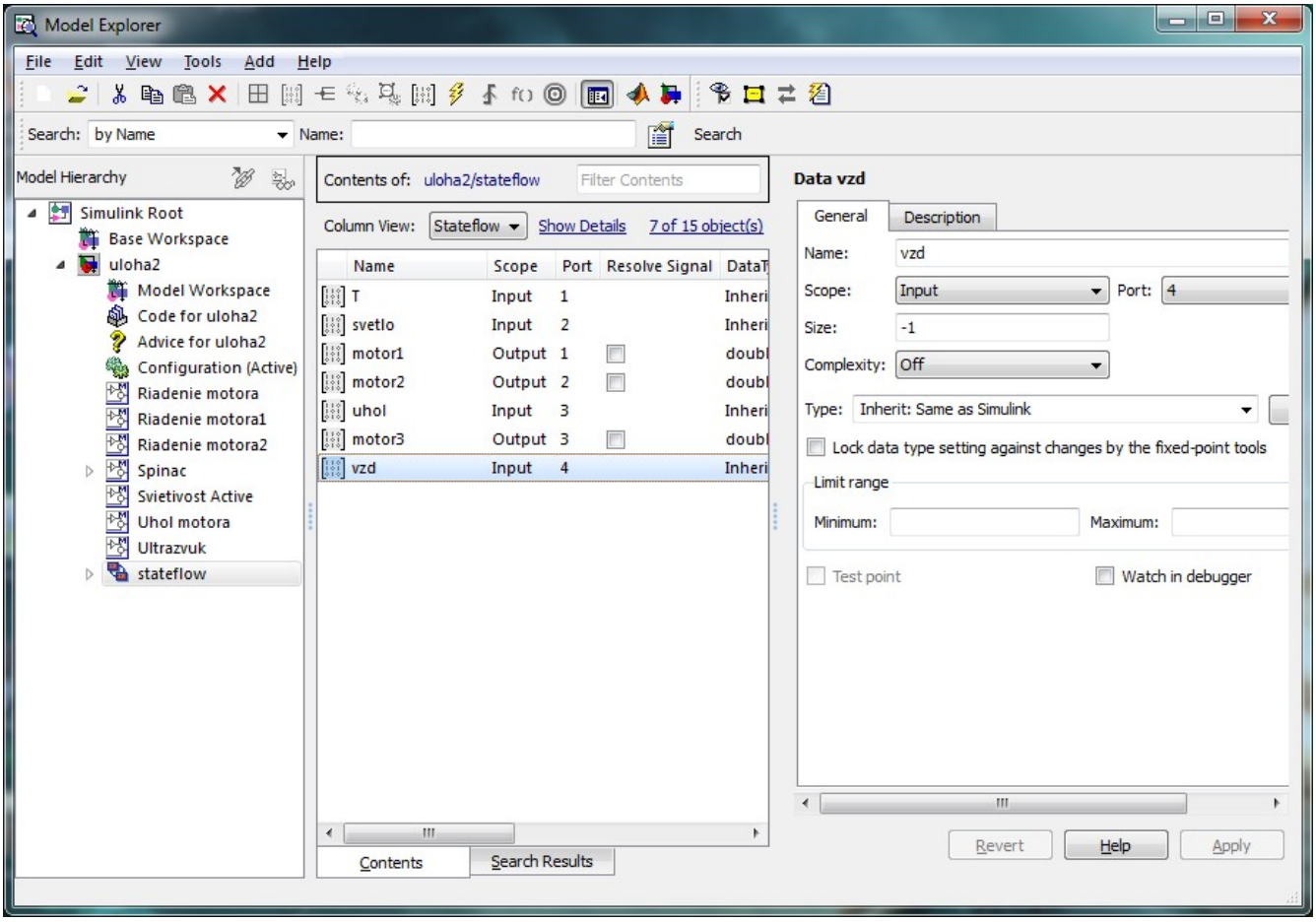
Akčné členy : Riadenie motora , Riadenie motora1 , Riadenie motora2

Senzory : Svietivost Active, Spinac ,Ultrazvuk ,Uhol motora

Iné : Display, Display1, Display2




**Obr. 30 Simulink-ová schéma úlohy 1 a 2**




**Obr. 31 Zoznam udalostí použitých v úlohe 1 a 2**

**Záver**

Cieľom našej práce boli dve hlavné úlohy. Prvá bola vytvorenie riadiaceho bloku v lego knižnici v programovacom prostredí Simulink a druhá sa skladala z dvoch hlavných častí, v ktorých sme testovali nami vytvorený riadiaci blok. Model sme riadili pomocou softvéru Stateflow, kde bol chod programu zabezpečený pomocou stavového riadenia. Prácu sme rozdelili na viacero kapitol.

V prvej kapitola bola venovaná stavebnicu LEGO Mindstorms NXT, jej súčastiam ako sú riadiaca kocka, jednotlivé senzory a akčné členy. V druhej kapitole sme predstavili Stateflow, ako balík programu MATLAB. Oboznámili sme sa s prácou v tomto programovacom prostredí. V posledných troch kapitolách sme riešili vytvorenie riadiaceho bloku v prostredí Simulink a overenie funkčnosti nami vytvoreného riadiaceho bloku na reálnom modely.

V prvej úlohe sme vytvorili riadiaci blok pre svetelný senzor. Jeho tvorbu sme podrobne opísali v kapitole tri.

Druhá úloha pozostávala z dvoch častí. V prvej časti sme overili funkčnosť nami vytvoreného riadiaceho bloku pre svetelný senzor. Test bol aplikovaný na tripod, vozidlo, ktoré malo hľadať signál, ktorý predstavoval cestu. Po nájdení cesty sa vozidlo malo vydať po nej.

Druhá časť pozostávala z vytvorenia programu, ktorý by riešil situáciu, v ktorej sa v ceste vozidla objavila prekážka. Tripod ju mal zaznamenať a odstrániť. Po odstránení mal pokračovať v ceste.

Najpodstatnejšou časťou v mojej práci bolo popísanie jednoduchého spôsobu ako si do simulinku konkrétniejšie do lego knižnice vytvorit riadiaci blok pre existujúce funkcie a testom som potvrdil správnosť daného riešenia.

**Literatúra**

[1] Mindstorms education - Metodika NXT © 2006 The Lego Group.

[2] **Mathworks, Inc**. Documentation MATLAB. Getting Started. [Online] 13. 05 2014 http://www.mathworks.com/access/helpdesk/help/techdoc/.

[3] **Mathworks, Inc.** Documentation Stateflow. User's Guide. [Online] 13. 05 2014. http://www.mathworks.com/access/helpdesk/help/toolbox/stateflow/.

[4] Holaza J., Bálint T. : Riadenie Lego Robotov : Bakalárska práca, STU FCHPT, Bratislava 2010

[5 ]TLEGOengineering, NXT Sensors [online] 13.5. 2014 http://www.legoengineering.com/nxt-sensors/

[6]The MathWorks, Statewlow and Stateflow Coder. User's Guide. The MathWorks. Inc, 2003 (1.vydanie 1997)