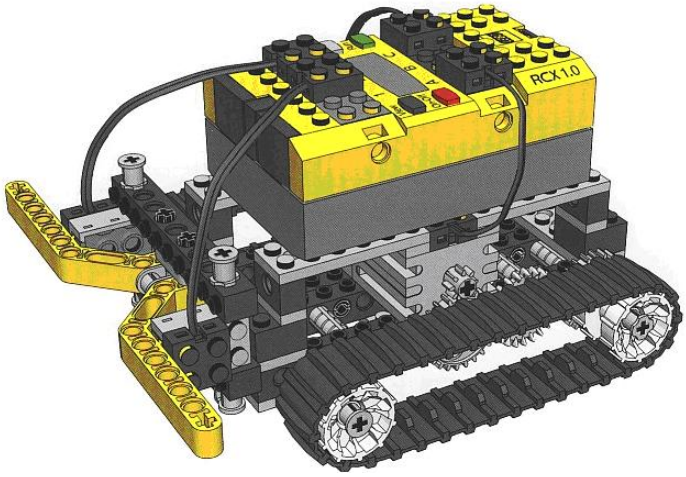
Πολυτεχνειο Κρητης
Τμημα Μηχανικων Παραγωγης και Διοικησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«ΧΡΗΣΗ ΤΩΝ LEGO MINDSTORMS ΣΤΗΝ ΔΙΔΑΣΚΑΛΙΑ ΤΩΝ ΣΥΣΤΗΜΑΤΩΝ ΑΥΤΟΜΑΤΟΥ ΕΛΕΓΧΟΥ»



ΜΥΛΩΝΑΚΗΣ ΝΙΚΟΛΑΟΣ

ΧΑΝΙΑ, 2007

Διπλωματική Εργασία Μυλωνάκη Νικολάου με θέμα:

«Χρήση των LEGO MINDSTORMS στην διδασκαλία των Συστημάτων Αυτομάτου Ελέγχου»

Επιβλέπων Καθηγητής:
Αναστάσιος Πουλιέζος

Εξεταστική Επιτροπή:
Αναστάσιος Πουλιέζος
Γεώργιος Σταυρακάκης
Νικόλαος Βλάσσης

ΕΥΧΑΡΙΣΤΙΕΣ

Καταρχήν, θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Αναστάσιο Πουλιέζο για την πολύτιμη βοήθεια του και την καθοδήγηση του καθ' όλη τη διάρκεια της εκπόνησης της παρούσας εργασίας.

Επίσης, ευχαριστώ θερμά τον κ. Αρναουτάκη Νεκτάριο, για την παραχώρηση του απαραίτητου εργαστηριακού εξοπλισμού αλλά και τη βοήθεια του, η οποία ήταν καταλυτική για να ξεπεραστούν διάφορες τεχνικές δυσκολίες που παρουσιάστηκαν.

Τέλος, θέλω να ευχαριστήσω θερμά την εξεταστική επιτροπή, τους κ. Γεώργιο Σταυρακάκη και κ. Νικόλαο Βλάσση, οι οποίοι δέχτηκαν με πολύ ευχαρίστηση να διορθώσουν την παρούσα εργασία και να μου δώσουν πολύτιμες συμβουλές για την βελτίωση της.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ

Στόχος της παρούσας εργασίας είναι η διερεύνηση των Lego Mindstorms σαν βοήθημα στην διδασκαλία των Συστημάτων Αυτομάτου Ελέγχου (Σ.Α.Ε.). Γίνεται μία προσπάθεια για την κατανόηση της λειτουργίας τους μέσω του λογισμικού MATLAB. Στη συνέχεια, γίνεται η εξαγωγή χρήσιμων συμπερασμάτων για την χρήση τους σαν εργαστηριακή διάταξη.

Η ιδέα αυτή προήλθε από την συνεχώς αυξανόμενη χρήση των Lego Mindstorms, από έναν μεγάλο αριθμό πανεπιστημίων παγκοσμίως, για την διδασκαλία μαθημάτων όπως της Ρομποτικής, του Προγραμματισμού και των Συστημάτων Αυτομάτου Ελέγχου. Η χρησιμότητα τους έγκειται στο γεγονός ότι η πλατφόρμα αυτή είναι πολύ ευέλικτη, δίνοντας την δυνατότητα κατασκευής μιας μεγάλης ποικιλίας διαφορετικών ρομπότ, που σε συνδυασμό με το μικρό κόστος τους, κάνει τα Lego Mindstorms ένα κατάλληλο εργαλείο για την διδασκαλία των Συστημάτων Αυτομάτου Ελέγχου.

Η διδασκαλία αυτή, θα βασίζεται στη χρήση φυσικών μηχανικών μοντέλων και στην εφαρμογή εννοιών και ιδεών από την πλευρά των φοιτητών, με σκοπό την επίλυση πραγματικών προβλημάτων. Οι φοιτητές, ατομικά ή ομαδικά, θα κατασκευάζουν φυσικά μοντέλα τα οποία θα συλλέγουν πληροφορίες από το περιβάλλον και θα αντιδρούν ανάλογα με τα ερεθίσματα που λαμβάνουν. Με τον τρόπο αυτό θα διδάσκονται τις βασικές έννοιες των Σ.Α.Ε. στο φυσικό τους περιβάλλον, όχι μόνο θεωρητικά, αλλά και πρακτικά, αποκτώντας εμπειρία επίλυσης πραγματικών προβλημάτων.

Η εργασία αυτή αποτελείται από 4 κεφάλαια. Στο κεφάλαιο 1, γίνεται μια πρώτη γνωριμία με την πλατφόρμα των Lego Mindstorms και αναφέρεται η χρήση τους σαν εκπαιδευτικό εργαλείο.

Στο κεφάλαιο 2, γίνεται η περιγραφή του απαραίτητου λογισμικού που χρησιμοποιείται για την κατασκευή και αποστολή των μοντέλων με τη χρήση του ηλεκτρονικού υπολογιστή. Επίσης, περιγράφεται εν συντομία η εγκατάσταση αυτού του λογισμικού.

Στο κεφάλαιο 3, γίνεται αρχικά η λεπτομερής περιγραφή των δύο μοντέλων που κατασκευάστηκαν από το λογισμικό της MATLAB. Στη συνέχεια περιγράφεται η διαδικασία που ακολουθείται για την αποστολή και εκτέλεση αυτών των μοντέλων στο ρομπότ.

Τέλος, στο κεφάλαιο 4, αναφέρονται τα πλεονεκτήματα της πλατφόρμας των Lego Mindstorms και λαμβάνονται τα τελικά συμπεράσματα για την χρησιμότητα της ως εκπαιδευτικό εργαλείο.

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

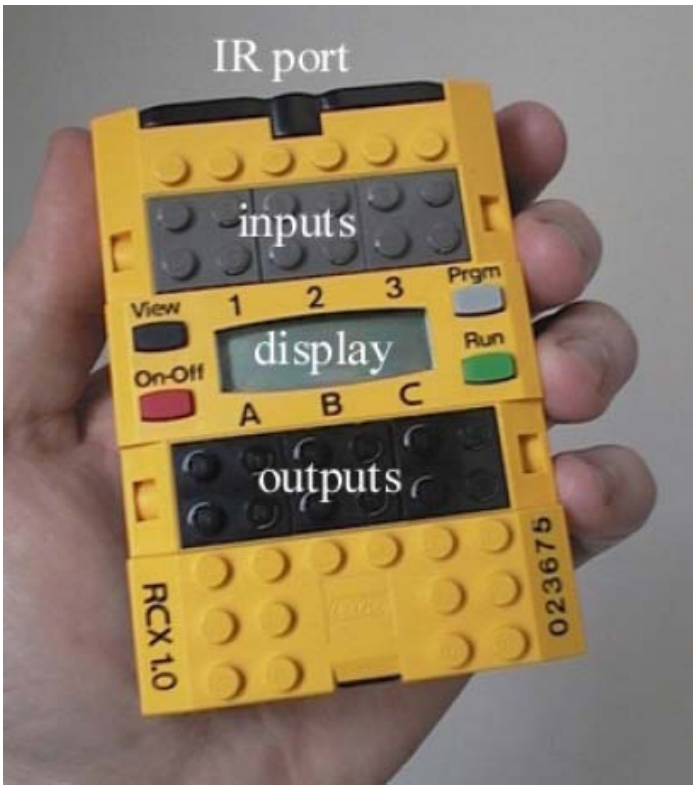
Το Lego Mindstorms Robotic Invention Kit διατέθηκε στην αγορά το 1998 από την εταιρία Lego. Δίνει τη δυνατότητα κατασκευής και προγραμματισμού διάφορων ρομπότ Lego, χρησιμοποιώντας τα πασίγνωστα ¨τουβλάκια¨ Lego. Αποτελείται από 717 κομμάτια και περιλαμβάνει, εκτός από τα τουβλάκια Lego, κινητήρες, γρανάζια, άξονες, διάφορους αισθητήρες και ένα κεντρικό ¨τουβλάκι RCX¨ (RCX Brick), το οποίο περιέχει τον μικροελεγκτή της Hitachi, H8/3292.

Εικόνα 1.1 Το Lego Mindstorms Robotic Invention Kit και τα κομμάτια από τα οποία αποτελείται.

1.1 Περιεχόμενα του Lego Mindstorms Robotic Invention Kit

1.1.1 RCX (Robotics Command eXplorer)

Το πιο σημαντικό τουβλάκι είναι το RCX (εικόνα 1.2), το οποίο αποτελεί τον πυρήνα κάθε ρομπότ που κατασκευάζεται με τα Lego Mindstorms.



Εικόνα 1.2 Το RCX.

Το RCX περιέχει τον μικροελεγκτή της Hitachi, H8/3292. Τα βασικά χαρακτηριστικά του ολοκληρωμένου του κυκλώματος βρίσκονται στο Παράρτημα Β. Επιπλέον, το τουβλάκι αυτό περιλαμβάνει (εικόνα 1.2):

- ✓ Θύρα υπερύθρων (IR Port) για την επικοινωνία με τον ηλεκτρονικό υπολογιστή.
- ✓ Οθόνη υγρών κρυστάλλων (LCD Display). Στην οθόνη αυτή μπορούν να εμφανιστούν μέχρι 4 ψηφία καθώς επίσης και διάφορες ενδείξεις όπως η στάθμη της μπαταρίας, η ύπαρξη υπέρυθρης επικοινωνίας με τον ηλεκτρονικό υπολογιστή και οι καταστάσεις των αισθητήρων και των κινητήρων (δηλαδή εάν λειτουργούν ή όχι).

- ✓ Περιβάλλον διεπαφής με το χρήστη (User Interface) που αποτελείται από 4 πλήκτρα: το πλήκτρο (On-Off) για την ενεργοποίηση και απενεργοποίηση του RCX, το πλήκτρο (Run) για το ξεκίνημα και το σταμάτημα των προγραμμάτων, το πλήκτρο (View) για την επιθεώρηση διαφόρων λειτουργιών και τέλος το πλήκτρο (Prgm) για την επιλογή του προγράμματος που θα εκτελεστεί από το RCX.
- ✓ Τρεις θύρες εισόδου (Inputs) στις οποίες συνδέονται οι διάφοροι αισθητήρες. Με τον τρόπο αυτό το RCX αλληλεπιδρά με το εξωτερικό περιβάλλον.
- ✓ Τρεις θύρες εξόδου (Outputs) στις οποίες συνδέονται οι ελέγξιμες, από το RCX, συσκευές (κινητήρες και λαμπτήρες).
- ✓ Μεγάφωνο (Speaker), το οποίο έχει την δυνατότητα αναπαραγωγής έως και εφτά διαφορετικών τόνων.

1.1.2 Πύργος Υπερύθρων (IR Tower)

Αντί για καλωδίωση, χρησιμοποιείται ένας πύργος υπερύθρων (εικόνα 1.3) για την αποστολή των προγραμμάτων στο RCX και την επικοινωνία με τον ηλεκτρονικό υπολογιστή. Τα δεδομένα μεταδίδονται μέσω της θύρας υπερύθρων του πύργου προς τη θύρα υπερύθρων του RCX. Ο πύργος αυτός συνδέεται στην θύρα USB του υπολογιστή.

Εικόνα 1.3 Πύργος Υπερύθρων (IR Tower).

1.1.3 Αισθητήρες (Sensors)

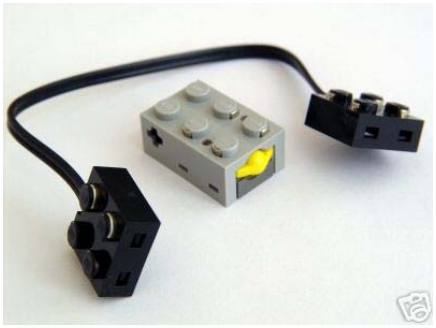
Ένα ρομπότ είναι ένας μηχανισμός του οποίου η συμπεριφορά προγραμματίζεται. Είναι αδύνατη η κατασκευή ενός ρομπότ χωρίς την παροχή εργαλείων μέσω των οποίων θα γίνεται εφικτή η απόκτηση πληροφοριών από τον πραγματικό κόσμο. Οι αισθητήρες παρέχουν στο ρομπότ πληροφορίες σχετικές με το εξωτερικό περιβάλλον. Η Lego έχει κατασκευάσει μερικούς τυπικούς αισθητήρες για την μέτρηση διαφόρων φυσικών ποσοτήτων.

Υπάρχουν 4 τυπικές κατηγορίες αισθητήρων για τα Lego Mindstorms. Αυτές είναι: αισθητήρες αφής (touch sensors), αισθητήρες φωτός (light sensors), αισθητήρες θερμοκρασίας (temperature sensors) και αισθητήρες περιστροφής (rotation sensors). Οι αισθητήρες αφής και θερμοκρασίας είναι παθητικοί αισθητήρες (passive sensors) δηλαδή απλοί διακόπτες on - off, ενώ οι αισθητήρες φωτός και περιστροφής είναι ενεργοί αισθητήρες (active sensors), δηλαδή χρειάζονται ηλεκτρική ενέργεια για την λειτουργία των εσωτερικών τους κυκλωμάτων.

Παρακάτω παρουσιάζονται οι 4 αυτές κατηγορίες αισθητήρων που χρησιμοποιούνται στα Lego Mindstorms.

1.1.3.1 Αισθητήρας αφής (Touch Sensor)

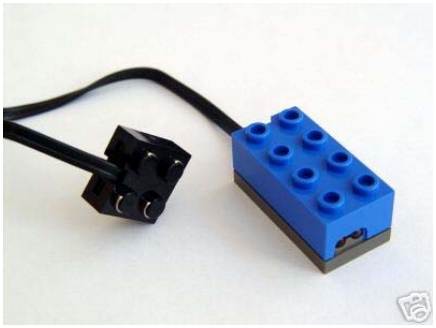
Ο αισθητήρας αφής (εικόνα 1.4) είναι ο πιο απλός αισθητήρας που περιλαμβάνεται στο πακέτο. Είναι ένας παθητικός αισθητήρας και χρησιμοποιείται για την ανίχνευση εμποδίων από το ρομπότ. Ο αισθητήρας αυτός λειτουργεί συνήθως σε κατάσταση Boolean επιστρέφοντας 0 (μη πιεσμένος) ή 1 (πιεσμένος).



Εικόνα 1.4 Αισθητήρας αφής (Touch Sensor).

1.1.3.2 Αισθητήρας φωτός (Light Sensor)

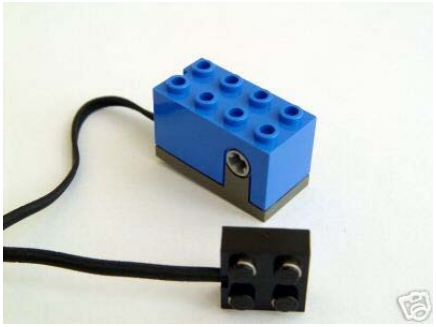
Ο αισθητήρας φωτός (εικόνα 1.5) είναι ένας ενεργός αισθητήρας που περιλαμβάνεται στο πακέτο των Lego Mindstorms. Αποτελείται από ένα LED κόκκινου φωτός και μία φωτοδίοδο. Αυτά τα δύο λειτουργούν σε ζευγάρια, η φωτοδίοδος μετράει την ποσότητα του περιβάλλοντος φωτός και το φως από το LED (εάν είναι αναμμένο) που αντανακλάται από τις κοντινές επιφάνειες. Ο αισθητήρας αυτός χρησιμοποιείται για την διάκριση περιοχών σε σκοτεινές και φωτεινές και για τον καθορισμό διαφορετικών χρωμάτων αντικειμένων. Η πιο συχνή και σημαντική του χρήση ωστόσο, είναι για να ακολουθήσει το ρομπότ μια προκαθορισμένη τροχιά.



Εικόνα 1.5 Αισθητήρας φωτός (Light Sensor).

1.1.3.3 Αισθητήρας περιστροφής (Rotation Sensor)

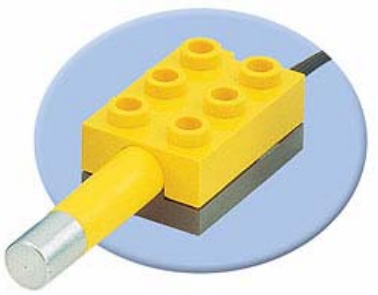
Ο αισθητήρας περιστροφής (εικόνα 1.6) είναι και αυτός ένας ενεργός αισθητήρας ο οποίος όμως δεν περιλαμβάνεται στο κύριο πακέτο των Lego Mindstorms. Χρησιμοποιείται για τον καθορισμό της περιστροφής ενός άξονα, ο οποίος συνδέεται στην οπή που βρίσκεται στη μία πλευρά του αισθητήρα. Η σχετική περιστροφή μετριέται σε τόξα των $22,5^{o}$, που σημαίνει ότι περιστροφή κατά $22,5^{o}$ δίνει την τιμή 1, ενώ μια πλήρης περιστροφή δίνει την τιμή 16 ($360/22,5^{o}$). Είναι ένας πολύ χρήσιμος αισθητήρας με πάρα πολλές εφαρμογές.



Εικόνα 1.6 Αισθητήρας περιστροφής (Rotation Sensor).

1.1.3.4 Αισθητήρας θερμοκρασίας (Temperature Sensor)

Ο αισθητήρας θερμοκρασίας (εικόνα 1.7), περιέχει έναν αντιστάτη με αρνητικό συντελεστή θερμοκρασίας, ο οποίος μεταβάλει την αντίσταση του ανάλογα με την θερμοκρασία. Ο αισθητήρας μετράει την θερμοκρασία του εξωτερικού περιβάλλοντος μέσω του μεταλλικού άκρου που έχει στο μπροστινό του μέρος. Μπορεί να μετρήσει θερμοκρασίες από -20^{o} έως 70^{o} Celsius ανά 0,1 βαθμό και με ακρίβεια περίπου 2 βαθμούς Celsius.



Εικόνα 1.7 Αισθητήρας θερμοκρασίας (Temperature Sensor).

1.1.4 Συσκευές εξόδου (Actuators)

1.1.4.1 Κινητήρες (Motors)

Ο κινητήρας Lego (εικόνα 1.8) έχει σχεδιαστεί για μέγιστη απόδοση. Περιλαμβάνει ένα εσωτερικό τροχό που αυξάνει την αδράνεια του κινητήρα, κάνοντας τον να δουλεύει ομαλότερα. Περιλαμβάνει επίσης ένα εσωτερικό σύνολο γραναζιών που μειώνει την ταχύτητα περιστροφής στην έξοδο, αυξάνοντας έτσι την ροπή του. Το γεγονός αυτό τον καθιστά κατάλληλο για ρομποτικές εφαρμογές.

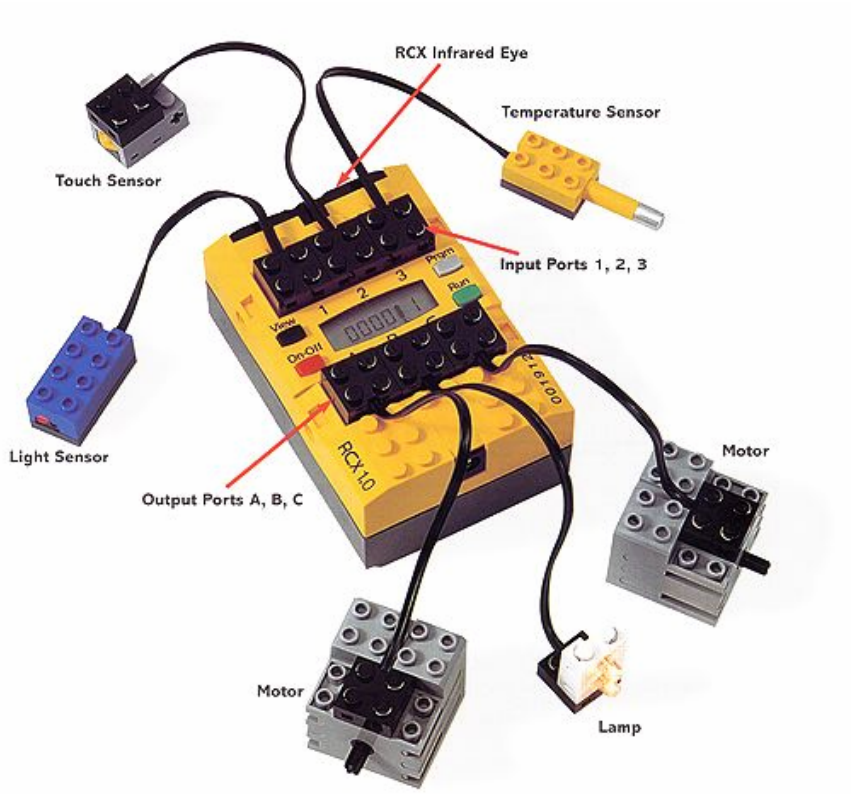
Εικόνα 1.8 Κινητήρας (Motor).

1.1.4.2 Λαμπτήρες (Lamps)

Η πιο απλή συσκευή εξόδου είναι οι λαμπτήρες. Είναι απλοί λαμπτήρες οι οποίοι έχουν δύο καταστάσεις, σβηστοί ή αναμμένοι (0 ή 1). Χρησιμοποιούνται στα Lego Mindstorms σαν ενδεικτικά λειτουργιών. Ένα τέτοιο σετ λαμπτήρων φαίνεται στην εικόνα 1.9.

Εικόνα 1.9 Σετ λαμπτήρων (Lamp Set).

Στην παρακάτω εικόνα 1.10, φαίνεται ο τρόπος συνδεσμολογίας των αισθητήρων, των κινητήρων και των λαμπτήρων με το RCX, χρησιμοποιώντας τα καλώδια που προσφέρονται στο πακέτο των Lego Mindstorms.



Εικόνα 1.10 Το RCX συνδεδεμένο με αισθητήρες, κινητήρες και λαμπτήρες.

1.2 Χρήσεις Lego Mindstorms

Το 1998, η εταιρία Lego διέθεσε στην αγορά ένα νέο προϊόν με την ονομασία Lego Mindstorms Robotic Invention Kit που αμέσως έγινε εμπορική επιτυχία. Ο λόγος για την επιτυχία αυτή είναι αφενός επειδή τα φτηνά ηλεκτρονικά έκαναν εφικτή την κατασκευή μικρών ρομπότ και αφετέρου, επειδή τα τουβλάκια Lego ανέκαθεν αποτελούσαν ένα εργαλείο μηχανικής προτυποποίησης. Σήμερα, υπάρχει μια μεγάλη κοινότητα από ερασιτέχνες και επαγγελματίες όλων των ηλικιών, οι οποίοι ανταλλάσουν σχέδια, προγραμματιστικές τεχνικές και άλλες ιδέες που σχετίζονται με τα Lego Mindstorms.

Τα Lego Mindstorms, προϊόν της συνεργασίας μεταξύ της Lego και του MIT Media Laboratory, αναπτύχθηκαν κυρίως σαν εκπαιδευτικό εργαλείο για την διδασκαλία μιας μεγάλης ποικιλίας θεμάτων όπως η ρομποτική, ο προγραμματισμός, τα συστήματα αυτομάτου ελέγχου και η τεχνητή νοημοσύνη. Ο συνδυασμός της μάθησης και της διασκέδασης, η ευελιξία του λογισμικού και το χαμηλό κόστος κάνουν τα ρομπότ αυτά πολύ ελκυστικά.

Τα Lego Mindstorms χρησιμοποιούνται σε πολλές τάξεις, από δημοτικά σχολεία μέχρι πανεπιστημιακά ιδρύματα. Οι λόγοι για την ευρεία χρήση τους σαν εργαλεία διδασκαλίας είναι πολλοί. Ο σημαντικότερος από αυτούς είναι ότι τα συναρμολογούμενα ρομπότ είναι πολύ ευέλικτα, όσον αφορά την κατασκευή και τον προγραμματισμό τους. Αποτελούν έναν ευχάριστο και διασκεδαστικό τρόπο διδασκαλίας, συνδυάζοντας την μάθηση με την πρακτική εμπειρία. Επίσης, αποτελούν μια πολύ οικονομική λύση, σε σχέση με τις δυνατότητες που προσφέρουν.

Το εκπαιδευτικό τμήμα της Lego (Lego Education) συνεργάζεται στενά με μεγάλο αριθμό ιδρυμάτων και εταιριών, που έχουν ως στόχο την ενίσχυση της εκπαίδευσης με τη χρήση της πλατφόρμας των Lego Mindstorms.

Ένα από αυτά τα ιδρύματα είναι το Tufts University College of Engineering, στη Μασαχουσέτη των Η.Π.Α. Το πανεπιστήμιο αυτό είναι αφοσιωμένο στην προσαρμογή της εκπαίδευσης της μηχανικής στην τάξη. Συνδυάζει την πείρα του στην μηχανική και στην εκπαίδευση, για το σχεδιασμό μιας ολοκληρωμένης σειράς μαθημάτων για τη διδασκαλία της μηχανικής και της ρομποτικής σε μαθητές

δημοτικού. Μέσα από προγράμματα, έρευνες και εργαστήρια, το πανεπιστήμιο αυτό προσπαθεί να βελτιώσει την εκμάθηση των επιστημών αυτών σε όλες τις ηλικίες.

Ένα άλλο ίδρυμα που συνεργάζεται με την Lego είναι το Carnegie Mellon University και πιο συγκεκριμένα η ανεξάρτητη ακαδημία Robotics Academy που εργάζεται για το πανεπιστήμιο αυτό. Η συγκεκριμένη ακαδημία έχει πολλά χρόνια εμπειρίας στη διδασκαλία της ρομποτικής στα σχολεία και είναι επιφορτισμένη για την κατασκευή σειράς μαθημάτων με την βοήθεια ψηφιακών μέσων για την καλύτερη εκπαίδευση των μαθητών.

1.2.1 Τα Lego Mindstorms στην Πρωτοβάθμια Εκπαίδευση

Η ρομποτική είναι ένας δημοφιλής και αποτελεσματικός τρόπος για τους δασκάλους να καλύψουν σημαντικούς τομείς της επιστήμης, της τεχνολογίας, της μηχανικής και των μαθηματικών στα προγράμματα μαθημάτων των σχολείων τους. Η σειρά Lego Mindstorms for Schools είναι κατάλληλη για χρήση σε σχολικό αλλά και σε εξωσχολικό περιβάλλον σε μαθητές από 8 έως 12 ετών. Περιλαμβάνει τα σύνολα κατασκευών (construction sets), τα εργαλεία προγραμματισμού (programming tools) και πακέτα δραστηριοτήτων (activity packs).

Με τα Lego Mindstorms for Schools οι μαθητές ασχολούνται με τεχνικές που χρησιμοποιούνται στον πραγματικό κόσμο της επιστήμης, της μηχανικής και του σχεδιασμού. Σχεδιάζουν, κατασκευάζουν και προγραμματίζουν πλήρη λειτουργικά μοντέλα. Μαθαίνουν να συμπεριφέρνονται σαν νέοι επιστήμονες, διεξάγοντας απλές έρευνες, υπολογίζοντας και μετρώντας τις συμπεριφορές των μοντέλων, καταγράφοντας και παρουσιάζοντας τα αποτελέσματά τους.

Τα σύνολα είναι για ομάδες 2 έως 3 μαθητών που συνεργάζονται με στόχο τη λύση ενός προβλήματος. Οι μαθητές αντιμετωπίζουν προκλήσεις που πρέπει να λύσουν δουλεύοντας από κοινού. Τα πακέτα δραστηριοτήτων τους εισάγουν στις εφαρμογές της αυτοματοποιημένης τεχνολογίας του πραγματικού κόσμου και τους παρέχουν τις δεξιότητες που χρειάζονται ώστε να είναι σε θέση να βρουν τις δικές τους λύσεις στα προβλήματα.

Στην Ελλάδα, γίνονται έρευνες σχετικά με την εισαγωγή των Lego Mindstorms στην πρωτοβάθμια εκπαίδευση. Σύμφωνα με άρθρο της Ελευθεροτυπίας (9/10/2006), δύο πιλοτικά προγράμματα που παρουσιάστηκαν στο 5ο Πανελλήνιο Συνέδριο της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών, Πληροφορίας και Επικοινωνιών στην Εκπαίδευση (ΕΤΠΕ), ήρθαν ν' αποδείξουν ότι το μάθημα με δύσκολες έννοιες μπορεί να γίνει παιχνίδι. Τα προγράμματα αυτά πραγματοποιήθηκαν στα 47^{ο} και 61^{ο} Δημοτικά Σχολεία της Πάτρας με τη συμμετοχή 4 αγοριών και κοριτσιών της ΣΤ' Τάξης, τυχαία επιλεγμένων, που εργάστηκαν σε δύο ομάδες. Μέσω του εκπαιδευτικού πακέτου ρομποτικών κατασκευών Lego Mindstorms οι μαθητές διδάχτηκαν έννοιες σχετικές με θερμότητα, θερμοκρασία, πήξη και τήξη, επέδειξαν αυξημένο ενδιαφέρον, ικανότητες και ανέπτυξαν πνεύμα ομαδικότητας.

1.2.2 Τα Lego Mindstorms στη Δευτεροβάθμια Εκπαίδευση

Τα Lego Mindstorms μπορούν να χρησιμοποιηθούν και στην δευτεροβάθμια εκπαίδευση.

Σύμφωνα με μια έρευνα των τμημάτων Εκπαιδευτικής και Κοινωνικής Πολιτικής και Εφαρμοσμένης Πληροφορικής του Πανεπιστημίου Μακεδονίας, οι δυσκολίες που συναντούν οι αρχάριοι προγραμματιστές αποτέλεσαν το κίνητρο για την αναζήτηση νέων μεθόδων διδασκαλίας για τα εισαγωγικά μαθήματα προγραμματισμού. Προτείνεται μια μέθοδος διδασκαλίας του προγραμματισμού που στηρίζεται στην καθοδήγηση των ρομπότ Lego Mindstorms, από προγράμματα που συντάσσονται με τη βοήθεια του γραφικού προγραμματιστικού περιβάλλοντος ROBOLAB.

Στην έρευνα αυτή επιχειρείται αρχικά η καταγραφή των αδυναμιών που παρουσιάζει η παραδοσιακή μέθοδος διδασκαλίας και στη συνέχεια η περιγραφή της εναλλακτικής προσέγγισης με τη χρήση των φυσικών μοντέλων Lego Mindstorms. Ακολουθεί η περιγραφή μιας σειράς μαθημάτων για τη διδασκαλία του εισαγωγικού μαθήματος στον προγραμματισμό της Γ’ Τάξης Γυμνασίου ή της Α’ Τάξης Ενιαίου Λυκείου με χρήση των Lego Mindstorms.

Τα συμπεράσματα της έρευνας αυτής είναι ότι η προσέγγιση της διδασκαλίας του προγραμματισμού με τα Lego Mindstorms, μπορεί να συμβάλλει στην εξάλειψη των αδυναμιών που συνεπάγεται η παραδοσιακή μέθοδος και να δημιουργήσει τις κατάλληλες συνθήκες μάθησης, ώστε να γίνει αποτελεσματικότερη η διδασκαλία.

1.2.3 Τα Lego Mindstorms στη Τριτοβάθμια Εκπαίδευση

Λιγότερο από 4 χρόνια πριν, τα περισσότερα προκαταρκτικά τμήματα ρομποτικής ήταν σχεδόν εξολοκλήρου θεωρητικά, επειδή ήταν πολύ δαπανηρή η προμήθεια αρκετού εξοπλισμού για τον κάθε φοιτητή, για την απόκτηση άμεσης πρακτικής εμπειρίας. Χάρη στη συνεχή μείωση του κόστους των υπολογιστών, ωστόσο, ευέλικτα ρομπότ είναι τώρα αρκετά οικονομικά και είναι εφικτός ο εφοδιασμός μιας ολόκληρης τάξης με προπτυχιακούς φοιτητές με αρκετό εξοπλισμό ώστε να επιτραπεί σε όλους να έχουν πρακτικές εμπειρίες στην κατασκευή ρομπότ. Από τα πολλά πακέτα που είναι διαθέσιμα στην αγορά, τα Lego Mindstorms είναι τα πιο δημοφιλή.

Στις μέρες μας, τα Lego Mindstorms χρησιμοποιούνται από έναν μεγάλο αριθμό πανεπιστήμιων παγκοσμίως για την διδασκαλία της ρομποτικής. Επίσης, χρησιμοποιούνται και από καθηγητές πανεπιστημίων σε διεθνή επιστημονικά συνέδρια στα οποία περιγράφουν πως, με τη χρήση των Lego Mindstorms, διεξάγουν προηγμένες έρευνες στον τομέα της ρομποτικής.

Παρότι η χρήση μιας απλής πλατφόρμας σε όλη τη διάρκεια των μαθημάτων εξασφαλίζει έναν βαθμό συνοχής, υπάρχει κίνδυνος ότι μπορεί να περιορίσει την πιθανή μάθηση του φοιτητή. Αυτό το ρίσκο μειώνεται επειδή η μεταβαλλόμενη φύση των Lego Mindstorms δίνει την δυνατότητα κατασκευής μιας μεγάλης ποικιλίας διαφορετικών ρομπότ. Αυτό δίνει ένα αξιοσημείωτο πλεονέκτημα στα Lego Mindstorms έναντι των περισσότερων άλλων ρομπότ που χρησιμοποιούνται στην διδασκαλία, τα οποία γενικά έχουν αμετάβλητα σώματα και επομένως είναι κατάλληλα μόνο για την διδασκαλία προγραμματισμού. Με τα Lego Mindstorms, οι φοιτητές μπορούν να εξερευνήσουν τις προκλήσεις της κατασκευής του σώματος ενός ρομπότ καθώς επίσης και τις προκλήσεις του προγραμματισμού του ¨εγκεφάλου¨ του. Γι' αυτό το λόγο, επιτρέπουν στους φοιτητές να εξερευνήσουν μία από τις πιο

συναρπαστικές και σημαντικές πλευρές της ρομποτικής – την αλληλεπίδραση μεταξύ του φυσικού σχεδίου του ρομπότ και του λογισμικού του. Οι φοιτητές ανακαλύπτουν ότι πολλά προβλήματα της ρομποτικής λύνονται κάνοντας μικρές μηχανικές αλλαγές στο σώμα του ρομπότ παρά κατασκευάζοντας περίπλοκες υπορουτίνες μέσα στο πρόγραμμα.

Αυτή η προσέγγιση της διδασκαλίας δίνει έμφαση στο ρόλο του ενεργού πειραματισμού και της πραγματικής εμπειρίας. Επίσης, τα εργαστήρια ενθαρρύνουν τους φοιτητές να συλλογιστούν πάνω σε αυτό που κάνουν και να εξάγουν περισσότερα πράγματα από τις πρακτικές δραστηριότητες που ασχολούνται.

1.2.4 Τα Lego Mindstorms σε εξωσχολικό περιβάλλον

Τα Lego Mindstorms after School είναι ιδανικά για τη χρήση σε εξωσχολικό περιβάλλον, όπου τα παιδιά έχουν χρόνο να ερευνήσουν και να κατασκευάσουν τα δικά τους ρομπότ. Πολλά σχολεία χρησιμοποιούν τα Lego Mindstorms εκτός από τις τάξεις και σε εξωσχολικές δραστηριότητες, που σημαίνει ότι οι μαθητές συνεχίζουν να βελτιώνουν τις ικανότητες τους στην επιστήμη και την τεχνολογία και μετά από τις ώρες του σχολείου, μαθαίνοντας ότι η επιστήμη και η τεχνολογία είναι πάνω από όλα διασκέδαση.

Η σχεδίαση, η κατασκευή και ο προγραμματισμός ρομπότ είναι δημοφιλείς δραστηριότητες για διαγωνισμούς. Τα παιδία αγαπούν τις προκλήσεις - ειδικά εάν πρόκειται να ανταγωνιστούν με άλλα.

Ένας από τους πρώτους διεθνείς διαγωνισμούς που έλαβαν μέρος, και έκτοτε έχει γίνει θεσμός, είναι η FIRST Lego League (FLL). Η FIRST Lego League είναι ένα

διεθνές πρόγραμμα για παιδιά ηλικίας από 9 έως 16 ετών, που συνδυάζει ένα αλληλεπιδραστικό πρόγραμμα ρομποτικής με μια αθλητική ατμόσφαιρα χρησιμοποιώντας τα Lego Mindstorms. Οι ομάδες αποτελούνται μέχρι και από 10 άτομα και εστιάζουν την προσοχή τους στην ομαδική κατασκευή, την επίλυση προβλημάτων, στη δημιουργικότητα και στην αναλυτική σκέψη. Οι ομάδες αντιμετωπίζουν μια ετήσια πρόκληση εξομοιώνοντας μια κατάσταση του πραγματικού κόσμου και πρέπει να ερευνήσουν, να σχεδιάσουν, να κατασκευάσουν, να προγραμματίσουν και να δοκιμάσουν ένα πλήρως αυτόνομο ρομπότ ικανό να ανταπεξέλθει στην εκάστοτε πρόκληση.

Οι διαγωνισμοί αυτοί οργανώνονται από μη κερδοσκοπικές οργανώσεις και χρηματοδοτούνται από χορηγούς που επιθυμούν να προωθήσουν τη συμμετοχή των παιδιών στην επιστήμη και την τεχνολογία. Τα Lego Mindstorms είναι βασικός παράγοντας σε αυτά τα γεγονότα. Το 2005, περισσότερα από 100.000 παιδιά και περίπου 8.000 ομάδες συμμετείχαν σε διαγωνισμούς ρομποτικής παγκοσμίως.

ΚΕΦΑΛΑΙΟ 2: ΠΕΡΙΒΑΛΛΟΝ ΕΡΓΑΣΙΑΣ

Σε αυτό το κεφάλαιο παρουσιάζεται περιληπτικά το λογισμικό που θα χρησιμοποιηθεί στη συνέχεια της παρούσας εργασίας για τον προγραμματισμό του RCX μέσω του ηλεκτρονικού υπολογιστή. Αρχικά, γίνεται η περιγραφή του απαραίτητου λογισμικού της Mathworks και στην συνέχεια οι πρόσθετες εφαρμογές οι οποίες είναι απαραίτητες, για την μεταγλώττιση των κατασκευασμένων μοντέλων Simulink σε γλώσσα C και στην ακόλουθη αποστολή τους μέσω υπερύθρων στο RCX.

2.1 Λογισμικό Mathworks

2.1.1 MATLAB

Η MATLAB είναι μία γλώσσα προγραμματισμού υψηλού επιπέδου. Ενοποιεί τον υπολογισμό, την απεικόνιση και τον προγραμματισμό σε ένα περιβάλλον, εύκολο στη χρήση, όπου τα προβλήματα και οι λύσεις, εκφράζονται σε μία μαθηματική σημειολογία. Με τη χρήση της MATLAB είναι δυνατή η επίλυση τεχνικών προγραμματιστικών προβλημάτων γρηγορότερα από άλλες παραδοσιακές γλώσσες προγραμματισμού όπως η C, η C++ και η Fortran. Τυπικές χρήσεις της MATLAB περιλαμβάνουν:

- Μαθηματικά και υπολογισμούς (Math and computation)
- Ανάπτυξη αλγορίθμων (Algorithm development)
- Μοντελοποίηση, προσομοίωση και προτυποποίηση (Modeling, simulation, and prototyping)
- Ανάλυση δεδομένων, εξερεύνηση και οπτική απεικόνιση (Data analysis, exploration, and visualization)
- Επιστημονικά και μηχανολογικά γραφικά (Scientific and engineering graphics)
- Ανάπτυξη εφαρμογών, συμπεριλαμβανομένης της κατασκευής γραφικού περιβάλλοντος διεπαφής (Application development, including graphical user interface building)

Η MATLAB έχει αναπτυχθεί κατά τη διάρκεια αρκετών ετών μέσω ανατροφοδότησης πολλών χρηστών. Στα πανεπιστήμια, αποτελεί το βασικό εκπαιδευτικό εργαλείο τόσο για εισαγωγικά όσο και για προχωρημένα μαθήματα μαθηματικών και μηχανικής. Στη βιομηχανία, η MATLAB αποτελεί ένα εργαλείο για έρευνα υψηλής παραγωγικότητας, ανάπτυξης και ανάλυσης.

Πρόσθετες εργαλειοθήκες (toolboxes) επεκτείνουν το περιβάλλον της MATLAB επιτρέποντας την επίλυση συγκεκριμένων κατηγοριών προβλημάτων σε πολλούς τομείς εφαρμογών. Αυτές είναι πολύ σημαντικές στους περισσότερους χρήστες της MATLAB και επιτρέπουν την εκμάθηση και εφαρμογή εξειδικευμένης τεχνολογίας. Περιοχές εφαρμογών στις οποίες είναι διαθέσιμες οι εργαλειοθήκες αυτές περιλαμβάνουν επεξεργασία σημάτων (signal processing), συστήματα ελέγχου (control systems), νευρωνικά δίκτυα (neural networks), ασαφής λογική (fuzzy logic), προσομοίωση (simulation) και πολλές άλλες.

2.1.2 Simulink

Το Simulink είναι ένα πακέτο λογισμικού που επιτρέπει την γραφική μοντελοποίηση, την προσομοίωση και την ανάλυση δυναμικών συστημάτων. Υποστηρίζει γραμμικά και μη γραμμικά συστήματα, μοντελοποιημένα σε συνεχή ή διακριτό χρόνο, ή ακόμη και υβριδικά συστήματα (εν μέρει μοντελοποιημένα σε συνεχή και εν μέρει σε διακριτό χρόνο). Υποστηρίζονται ακόμη συστήματα με τμηματικά διαφορετικούς χρόνους δειγματοληψίας. Χρησιμοποιείται για την εξέταση της συμπεριφοράς μιας μεγάλης έκτασης δυναμικών συστημάτων πραγματικού κόσμου, συμπεριλαμβανομένων ηλεκτρικών κυκλωμάτων, συστημάτων πέδησης, και πολλών άλλων ηλεκτρικών, μηχανικών και θερμοδυναμικών συστημάτων.

Για τη μοντελοποίηση, το Simulink παρέχει ένα γραφικό περιβάλλον διεπαφής (GUI) που επιτρέπει την κατασκευή μοντέλων ως δομικών διαγραμμάτων, χρησιμοποιώντας λειτουργίες click-and-drag του ποντικιού. Το Simulink περιλαμβάνει ένα πλήθος βιβλιοθηκών δομικών στοιχείων (blocks), τα βασικότερα από τα οποία είναι οι πηγές (sources), τα στοιχεία «απορρόφησης» (sinks), τα συνεχή γραμμικά στοιχεία, τα μη γραμμικά στοιχεία και τα στοιχεία σημάτων και συστημάτων.

Επίσης είναι δυνατή η τροποποίηση και η δημιουργία νέων δομικών στοιχείων από το χρήστη με τη χρήση των συναρτήσεων συστήματος, S-functions (System-functions). Οι S-functions χρησιμοποιούνται για τη δημιουργία νέων δομικών διαγραμμάτων Simulink. Ο χρήστης μπορεί να χρησιμοποιήσει τις S-functions σε μια ποικιλία εφαρμογών που περιλαμβάνουν:

- ✓ Προσθήκη νέων δομικών διαγραμμάτων γενικού σκοπού (general purpose blocks) στο Simulink
- ✓ Προσθήκη δομικών διαγραμμάτων που αντιπροσωπεύουν οδηγούς συσκευών υλικού (hardware device drivers)
- ✓ Περιγραφή ενός συστήματος σαν ένα σύνολο μαθηματικών εξισώσεων

Οι S-functions, επιτρέπουν στον χρήστη να προσθέσει τους δικούς του αλγόριθμους στα μοντέλα του Simulink. Οι αλγόριθμοι αυτοί μπορούν να γραφτούν στη MATLAB ή σε γλώσσες προγραμματισμού όπως η C, η C++, η Fortran και η Ada. Ακολουθώντας ένα σύνολο απλών κανόνων, ο χρήστης μπορεί να εφαρμόσει τους αλγόριθμους αυτούς σε μια συνάρτηση συστήματος (S-function). Μετά την κατασκευή της S-function και την ονομασία του δομικού διαγράμματος της (S-function block), που μετά είναι διαθέσιμο στην υπό-βιβλιοθήκη Nonlinear Block sublibrary του Simulink, ο χρήστης μπορεί να την τροποποιεί χρησιμοποιώντας μάσκα (masking).

Ένα πλεονέκτημα της χρήσης S-functions είναι η δυνατότητα κατασκευής ενός δομικού διαγράμματος γενικού σκοπού το οποίο στη συνέχεια μπορούμε να το χρησιμοποιήσουμε πολλές φορές μέσα σε ένα μοντέλο, διαφοροποιώντας τις παραμέτρους του κάθε φορά και βλέποντας τα αποτελέσματα.

Μετά τη δημιουργία του μοντέλου, είναι δυνατή η προσομοίωση του, χρησιμοποιώντας μια από τις διάφορες μεθόδους ολοκλήρωσης που παρέχει το Simulink. Χρησιμοποιώντας παλμογράφους (scopes) και άλλα μπλοκ απεικόνισης, είναι δυνατή η παρακολούθηση των αποτελεσμάτων της προσομοίωσης καθώς αυτή εξελίσσεται. Επιπλέον, είναι δυνατή η εξαγωγή αποτελεσμάτων της προσομοίωσης στο χώρο εργασίας της MATLAB για περαιτέρω επεξεργασία.

2.1.3 Real-Time Workshop

Χρησιμοποιώντας την εργαλειοθήκη πραγματικού χρόνου (Real-Time Workshop) είναι δυνατή η προσομοίωση αλλά και ο έλεγχος συστημάτων σε πραγματικό χρόνο.

Το Real-Time Workshop παράγει κώδικα C απευθείας από τα μοντέλα του Simulink και κατασκευάζει αυτόματα προγράμματα τα οποία μπορούν να τρέξουν, σε πραγματικό χρόνο, σε μία ποικιλία από περιβάλλοντα, συμπεριλαμβανομένων συστημάτων πραγματικού χρόνου (real-time systems) και αυτοδύναμων προσομοιώσεων (stand-alone simulations).

Με το Real-Time Workshop, μπορούμε να τρέξουμε το μοντέλο Simulink σε έναν απομακρυσμένο επεξεργαστή (remote processor) σε πραγματικό χρόνο. Μπορούμε επίσης να τρέξουμε αυτοδύναμες προσομοιώσεις μεγάλης ταχύτητας (high-speed stand-alone simulations) σε έναν κεντρικό ή εξωτερικό υπολογιστή.

2.1.4 Real-Time Workshop Embedded Coder

Το Real-Time Workshop Embedded Coder είναι ένα ξεχωριστό, επιπρόσθετο προϊόν για χρήση με το Real-Time Workshop.

Παρέχει ένα πλαίσιο εργασίας για την ανάπτυξη ενός βελτιστοποιημένου κώδικα όσον αφορά την ταχύτητα, την χρήση μνήμης και την απλότητα του. Το Real-Time Workshop Embedded Coder προτείνεται για χρήση με ενσωματωμένα συστήματα (embedded systems).

Επίσης, παράγει κώδικα που είναι εύκολο να διαβαστεί, να σχεδιαστεί και να τροποποιηθεί για οποιοδήποτε περιβάλλον παραγωγής (production environment).

2.1.5 Target Language Compiler

Για την παραγωγή κώδικα, το Real-Time Workshop υποστηρίζεται από τον Target Language Compiler (TLC). Ο Target Language Compiler μετασχηματίζει μία ενδιάμεση περιγραφή του μοντέλου, που έχει παραχθεί από το Real-Time Workshop, του διαγράμματος Simulink σε κώδικα σχετικό με τον στόχο. Αυτή η ενδιάμεση περιγραφή του μοντέλου αποθηκεύεται σε ένα αρχείο ASCII που ονομάζεται *model*.rtw. Ο Target Language Compiler επιτρέπει την τροποποίηση των περισσότερων πλευρών του παραγόμενου κώδικα. Ο μεταγλωττιστής διαβάζει το αρχείο *model*.rtw και εκτελεί ένα πρόγραμμα TLC που αποτελείται από ένα σύνολο αρχείων στόχων [target files (.tlc files)]. Αυτά είναι αρχεία ASCII γραμμένα για τον Target Language Compiler. Το πρόγραμμα TLC καθορίζει πώς να μετασχηματιστεί το αρχείο *model*.rtw σε κώδικα.

Ένα πρόγραμμα TLC αποτελείται από:

- Το σημείο εκκίνησης (entry point) ή το κύριο αρχείο (main file)
- Ένα σύνολο από δομικά στοιχεία αρχείων στόχων (block target files). Αυτά καθορίζουν πως θα μεταφραστεί κάθε δομικό στοιχείο (block) του μοντέλου σε κώδικα σχετικό με τον στόχο.
- Μία βιβλιοθήκη συναρτήσεων (function library). Αυτή αποτελεί ένα σύνολο από συναρτήσεις βιβλιοθήκης που χρησιμοποιεί το πρόγραμμα TLC κατά την μετατροπή του αρχείου *model*.rtw σε κώδικα.

2.1.6 Stateflow

Το Stateflow είναι ένα εργαλείο ανάπτυξης και γραφικού σχεδιασμού που χρησιμοποιείται σε συνδυασμό με το Simulink. Παρέχει καθαρή και σαφή περιγραφή της συμπεριφοράς σύνθετων συστημάτων χρησιμοποιώντας, στο ίδιο διάγραμμα Stateflow, την θεωρία των πεπερασμένων καταστάσεων μηχανών (finite state machine theory), την σημειολογία των διαγραμμάτων ροής (flow diagram notations) και τα διαγράμματα καταστάσεων-μεταβάσεων (state-transition diagrams).

Με την χρήση του Stateflow Coder, μπορούμε να παράγουμε κώδικα για εφαρμογές που κατασκευάζουμε σε άλλα περιβάλλοντα, όπως σε ένα ενσωματωμένο περιβάλλον (embedded environment). Μπορούμε ακόμα να συμπεριλάβουμε τον δικό μας, κατασκευασμένο κώδικα, ο οποίος χρησιμοποιείται από το Stateflow για την κατασκευή του στόχου.

Με την εργαλειοθήκη του Real-Time Workshop είναι δυνατή η λήψη κώδικα από το Simulink και το Stateflow και η εκτέλεση του σαν εφαρμογή σε ένα άλλο περιβάλλον, για τον έλεγχο μιας διαδικασίας. Επίσης, είναι εφικτή η παραγωγή κώδικα από τα διαγράμματα του Stateflow και η χρησιμοποίηση του με κάθε δυνατό τρόπο. Το Stateflow συνεργάζεται με το Real-Time Workshop με τους παρακάτω τρόπους:

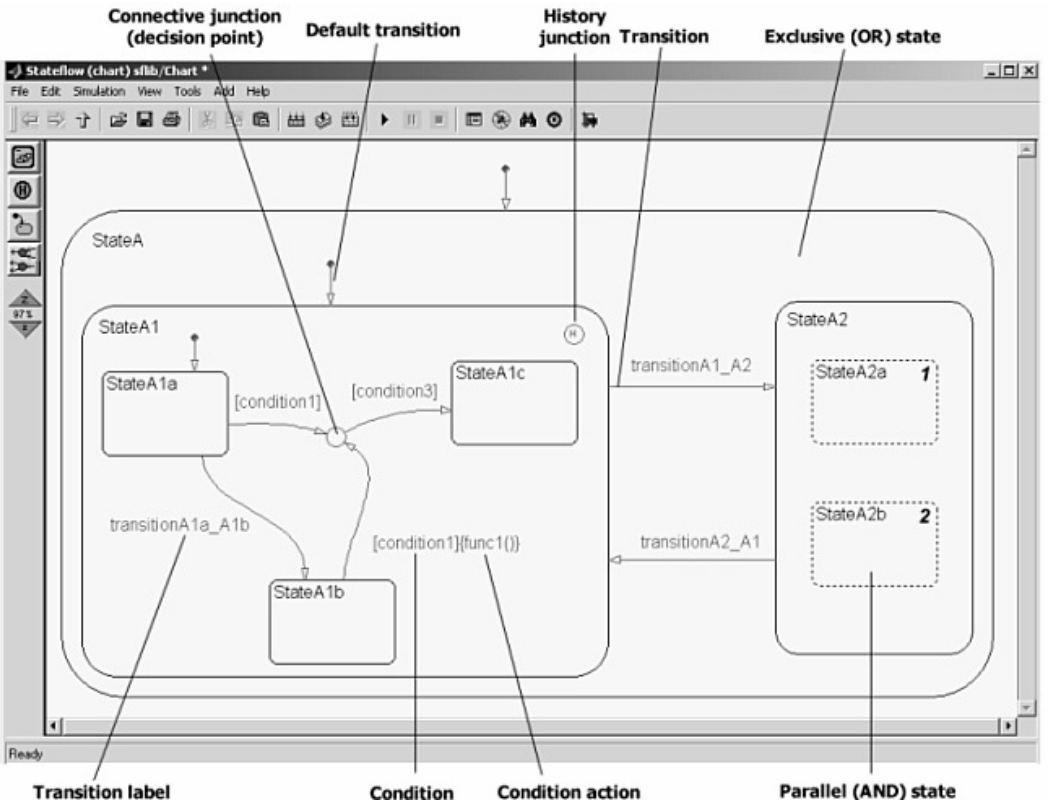
- Το Stateflow επιτρέπει την δημιουργία και προσομοίωση των μοντέλων του Simulink που περιέχουν μηχανές καταστάσεων (state machines).
- Το Stateflow επιτρέπει την δημιουργία αυτοδύναμων εκτελέσιμων (stand-alone executables) που καθορίζονται από μηχανές καταστάσεων.
- Το Real-Time Workshop παράγει αυτοδύναμα εκτελέσιμα (stand-alone executables) από τα μοντέλα του Simulink, συμπεριλαμβάνοντας αυτές τις μηχανές καταστάσεων που καθορίζονται από το Stateflow.

Η χρήση του Stateflow επιτρέπει:

- Την παραστατική μοντελοποίηση και προσομοίωση σύνθετων συστημάτων που βασίζονται στην θεωρία της μηχανής πεπερασμένων καταστάσεων (finite state machine theory). Σύμφωνα με την θεωρία αυτή, η μετάβαση από μία κατάσταση (state) σε μία άλλη προκαθορισμένη κατάσταση, γίνεται εάν η συνθήκη που καθορίζει την μετάβαση είναι αληθής. Σαν παράδειγμα, η θεωρία της μηχανής πεπερασμένων καταστάσεων μπορεί να εφαρμοστεί στην αναπαράσταση της κίνησης του RCX. Η κίνηση του RCX έχει έναν αριθμό από λειτουργικές καταστάσεις: την κίνηση προς τα εμπρός (forwards), την κίνηση προς τα πίσω (backwards), τη στροφή αριστερά (left) και τη στροφή δεξιά (right). Οι μεταβάσεις στο σύστημα από την μία κατάσταση στην άλλη συμβαίνουν όταν ενεργοποιηθούν οι αισθητήρες αφής, για παράδειγμα από κίνηση προς τα εμπρός σε κίνηση προς τα πίσω.

- Την σχεδίαση και ανάπτυξη αιτιοκρατικών συστημάτων εποπτικής λογικής (deterministic, supervisory control systems). Τα συστήματα αυτά δημιουργούνται με την βοήθεια διαγραμμάτων. Ένα διάγραμμα Stateflow (Stateflow diagram) είναι μια γραφική αναπαράσταση ενός μηχανισμού πεπερασμένων καταστάσεων όπου οι καταστάσεις και οι μεταβάσεις σχηματίζουν τα βασικά δομικά στοιχεία του συστήματος. Στα διαγράμματα αυτά γίνεται ταυτόχρονη χρήση της σημειολογίας διαγραμμάτων ροής (flow diagram notation) και σημειολογίας μεταβάσεων καταστάσεων (state transition notation). Η σημειολογία των διαγραμμάτων ροής αποτελεί έναν αποτελεσματικό τρόπο αναπαράστασης κοινών δομών όπως οι επαναλήψεις for ή η δομές if-then-else.
- Την εύκολη τροποποίηση του συστήματος, την αξιολόγηση των αποτελεσμάτων και την επιβεβαίωση της συμπεριφοράς του σε οποιοδήποτε στάδιο της σχεδίασης.
- Την αυτόματη παραγωγή κώδικα C απευθείας από το μοντέλο, με χρήση του Stateflow Coder.
- Την μοντελοποίηση, προσομοίωση και ανάλυση του συστήματος με χρήση του Simulink.
- Την αυτόματη κατασκευή προγραμμάτων τα οποία μπορούν να τρέξουν, σε πραγματικό χρόνο (real time), σε μια μεγάλη ποικιλία από περιβάλλοντα, στην περίπτωση μας το μικροελεγκτή του RCX, με χρήση του Real-Time Workshop (RTW) και του Real-Time Workshop Embedded Coder.

Η λειτουργία του Stateflow στηρίζεται στην κατασκευή διαγραμμάτων. Ένα διάγραμμα Stateflow αποτελείται από ένα σύνολο γραφικών αντικειμένων, το οποίο φαίνεται στην εικόνα 2.1. Στις παρακάτω παραγράφους γίνεται μία σύντομη περιγραφή των πιο σημαντικών στοιχείων που θα χρησιμοποιηθούν για την κατασκευή των μοντέλων που θα εκτελεστούν στο ρομπότ.



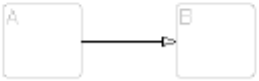
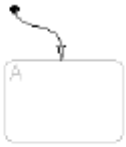
Εικόνα 2.1 Το σύνολο των γραφικών αντικειμένων του Stateflow.

Το Stateflow έχει τη δική του σημειολογία. Η σημειολογία αυτή καθορίζει ένα σύνολο αντικειμένων και τους κανόνες που ελέγχουν τις σχέσεις μεταξύ αυτών των αντικειμένων. Η σημειολογία του Stateflow αποτελείται από:

- ✓ Ένα σύνολο γραφικών αντικειμένων (Graphical objects).
- ✓ Ένα σύνολο μη γραφικών αντικειμένων (Non Graphical objects).
- ✓ Καθορισμένες σχέσεις μεταξύ αυτών των αντικειμένων.

2.1.6.1 Γραφικά αντικείμενα (Graphical objects)

Τα πιο σημαντικά γραφικά αντικείμενα του Stateflow είναι οι καταστάσεις (states), οι μεταβάσεις (transitions), οι προκαθορισμένες μεταβάσεις (Default transitions) και οι συνδετικοί σύνδεσμοι (connective junctions). Οι γραφικές τους αναπαραστάσεις στο περιβάλλον του Stateflow φαίνονται στην εικόνα 2.2.

Name	Notation	Toolbar Icon
State		
Transition		NA
Default Transition		
Connective Junction		

Εικόνα 2.2: Γραφικά αντικείμενα του Stateflow.

Οι **καταστάσεις (states)** περιγράφουν τις δυνατές καταστάσεις που μπορεί να βρεθεί το σύστημα. Στο Stateflow απεικονίζονται όπως στην εικόνα 2.3.

Εικόνα 2.3 Γραφική απεικόνιση της κατάστασης (state) στο Stateflow.

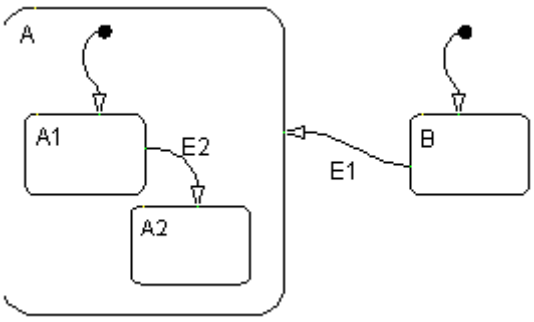
Οι καταστάσεις μπορούν να είναι **υπέρ-καταστάσεις (superstates)**, **υπό-καταστάσεις (substates)** ή απλά **καταστάσεις (states)**. Μία κατάσταση καλείται υπέρ-κατάσταση εάν περιέχει στο εσωτερικό της άλλες καταστάσεις, που καλούνται υπό-καταστάσεις. Μία κατάσταση καλείται υπό-κατάσταση εάν περιέχεται σε μία άλλη κατάσταση. Μία κατάσταση η οποία δεν είναι ούτε υπέρ-κατάσταση ούτε υπό-κατάσταση καλείται απλά κατάσταση.

Οι καταστάσεις διακρίνονται σε **ενεργές (active)** και **μη ενεργές (inactive)**. Μία κατάσταση είναι ενεργή όταν το διάγραμμα βρίσκεται στην συγκεκριμένη

κατάσταση, ενώ είναι μη ενεργή όταν δεν βρίσκεται σε αυτήν. Η ενεργοποίηση ή όχι μίας κατάστασης αλλάζει δυναμικά και βασίζεται στα γεγονότα (events) και τις συνθήκες (conditions). Η ύπαρξη των γεγονότων οδηγεί στην εκτέλεση του διαγράμματος Stateflow κάνοντας τις καταστάσεις ενεργές ή μη ενεργές. Κατά τη διάρκεια εκτέλεσης ενός διαγράμματος, υπάρχει ένας συνδυασμός ενεργών και μη ενεργών καταστάσεων.

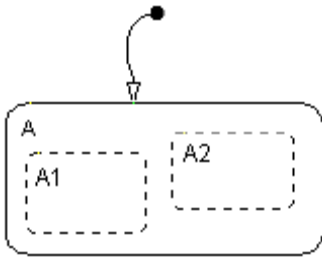
Κάθε κατάσταση (και διάγραμμα) αναλύεται στα στοιχεία που το αποτελούν και που υποδεικνύουν το είδος των υπό-καταστάσεων που μπορεί να περιέχει. Όλες οι υπό-καταστάσεις μίας υπέρ-κατάστασης πρέπει να είναι του ίδιου τύπου με αυτόν της υπέρ-κατάστασης. Μπορούμε να έχουμε δύο ειδών ανάλυση μίας κατάστασης: την παράλληλη (parallel ή AND) ή την αποκλειστική (exclusive ή OR).

Η αποκλειστική ανάλυση (exclusive ή OR decomposition) μίας υπέρ-κατάστασης χρησιμοποιείται για να περιγράψει καταστάσεις συστημάτων οι οποίες είναι αμοιβαία αποκλεισμένες. Στην περίπτωση αυτή μόνο μία υπό-κατάσταση μπορεί να είναι ενεργή κάθε χρονική στιγμή. Στο παράδειγμα τις εικόνας 2.4, οποιαδήποτε από τις καταστάσεις A ή B μπορεί να είναι ενεργή. Εάν η A είναι ενεργή τότε οποιαδήποτε από τις καταστάσεις A1 ή A2 μπορεί να είναι ενεργή, αλλά όχι και οι δύο ταυτόχρονα.



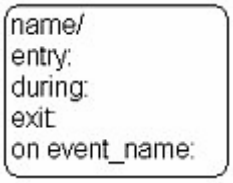
Εικόνα 2.4: Παράδειγμα ανάλυσης OR.

Η παράλληλη ανάλυση μίας υπέρ-κατάστασης χρησιμοποιείται για να περιγράψει καταστάσεις συστημάτων οι οποίες μπορούν να συμβούν ταυτόχρονα. Στο παράδειγμα της εικόνας 2.5, όταν η κατάσταση A είναι ενεργή, οι καταστάσεις A1 και A2 είναι ενεργές ταυτόχρονα.



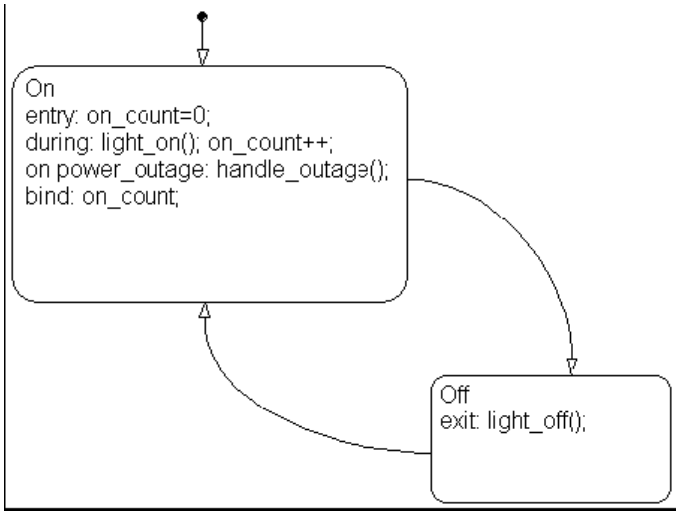
Εικόνα 2.5 Παράδειγμα ανάλυσης AND.

Κάθε κατάσταση περιγράφεται από μία **επιγραφή (label)** που εμφανίζεται στην πάνω αριστερή γωνία του σχήματος της (εικόνα 2.6). Κάθε τμήμα της επιγραφής είναι προαιρετικό. Η επιγραφή έχει την παρακάτω γενική μορφή:



Εικόνα 2.6 Τα μέρη της επιγραφής μιας κατάστασης.

Ένα παράδειγμα φαίνεται παρακάτω (εικόνα 2.7):



Εικόνα 2.7 Παράδειγμα επιγραφής μίας κατάστασης.

Η επιγραφή μίας κατάστασης ξεκινάει με το **όνομα της κατάστασης (name)**. Στο παραπάνω παράδειγμα, τα ονόματα των καταστάσεων είναι On και Off.

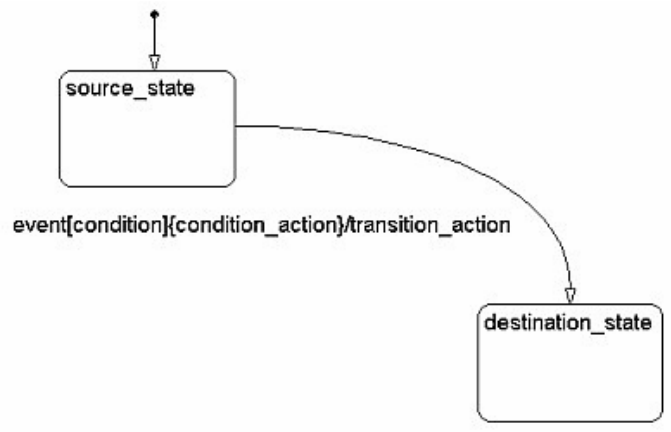
Ακολουθεί η δήλωση των **ενεργειών εισόδου (entry)**. Στο παράδειγμα, η κατάσταση On έχει σαν ενέργεια εισόδου on_count=0. Αυτό σημαίνει ότι η τιμή της μεταβλητής on_count θα γίνεται 0 κάθε φορά που η κατάσταση On θα γίνεται ενεργή.

Στη συνέχεια δηλώνουμε τις ενέργειες που θα συμβαίνουν όταν μία κατάσταση είναι ενεργή και συμβαίνει οποιοδήποτε γεγονός. Αυτές δηλώνονται μετά από το **during**. Στο παράδειγμα, για την κατάσταση On, οι ενέργειες light_on() και on_count++ θα εκτελεστούν όταν η κατάσταση On είναι ενεργή και συμβεί οποιοδήποτε γεγονός.

Κατόπιν δηλώνονται οι **ενέργειες εξόδου (exit)**. Οι ενέργειες αυτές εκτελούνται όταν η κατάσταση απενεργοποιείται. Στο παράδειγμα, για την κατάσταση Off, η ενέργεια εξόδου είναι η light_off().

Τέλος, ακολουθούν οι ενέργειες οι οποίες εκτελούνται αφού συντελεστεί ένα γεγονός **(on όνομα_γεγονότος).** Στο παράδειγμα, στην κατάσταση On δηλώνεται ότι on power_outage:handle_outage(). Εάν η κατάσταση On είναι ενεργή και συμβεί το γεγονός power_outage, τότε η ενέργεια handle_outage εκτελείται.

Οι **μεταβάσεις (transitions)** είναι καμπύλες γραμμές με βέλη που ενώνουν ένα γραφικό αντικείμενο με ένα άλλο. Στις περισσότερες περιπτώσεις, μία μετάβαση αναπαριστά το πέρασμα του συστήματος από μία κατάσταση σε μία άλλη. Μία μετάβαση ξεκινάει από την κατάσταση πηγή (source state) και καταλήγει στην κατάσταση προορισμού (destination state). Στην εικόνα 2.8 βλέπουμε ένα απλό παράδειγμα μετάβασης. Σε αυτό το παράδειγμα, έχουμε μία μετάβαση από την κατάσταση πηγή source_state στην κατάσταση προορισμού destination_state.



Εικόνα 2.8 Απεικόνιση απλής μετάβασης.

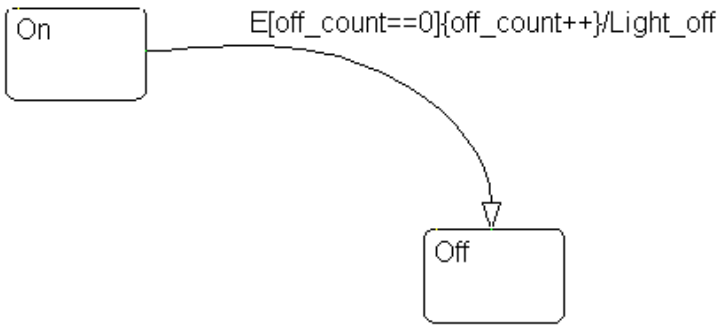
Κάθε μετάβαση περιγράφεται από μία επιγραφή, η οποία έχει την παρακάτω γενική μορφή:

event[condition]{condition_action}/transition_action

γεγονός[συνθήκη]{ενέργεια συνθήκης}/ενέργεια μετάβασης

Το κάθε τμήμα της επιγραφής είναι προαιρετικό.

Για να γίνει κατανοητή η έννοια αυτή θα δώσουμε ένα παράδειγμα (εικόνα 2.9).



Εικόνα 2.9 Παράδειγμα επιγραφής μίας μετάβασης.

Πρώτα δηλώνεται το **γεγονός (event).** Το καθορισμένο γεγονός είναι αυτό που προκαλεί την μετάβαση, εάν η συνθήκη, είναι αληθής. Ο καθορισμός του γεγονότος είναι προαιρετικός. Έλλειψη γεγονότος δηλώνει ότι η μετάβαση θα γίνεται με την εμφάνιση οποιουδήποτε γεγονότος. Πολλαπλά γεγονότα καθορίζονται με τη χρήση

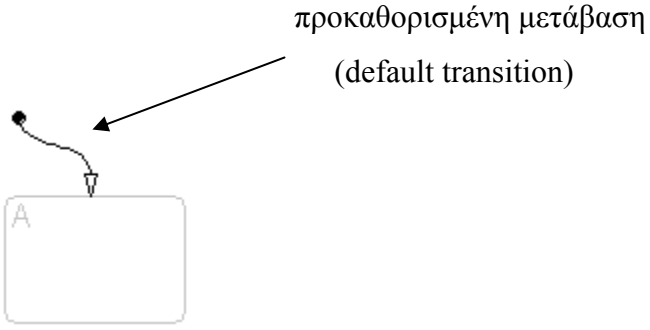
του λογικού τελεστή OR (|). Στο παραπάνω παράδειγμα, η εμφάνιση του γεγονότος E ενεργοποιεί την μετάβαση από το On στο Off εάν η συνθήκη [off_count==0] είναι αληθής.

Στη συνέχεια δηλώνεται η **συνθήκη (condition)**. Μία συνθήκη είναι μία Boolean έκφραση για να καθορίσουμε ότι η μετάβαση θα συμβεί δεδομένου ότι η καθορισμένη έκφραση είναι αληθής. Η συνθήκη αυτή γράφεται μέσα σε [].

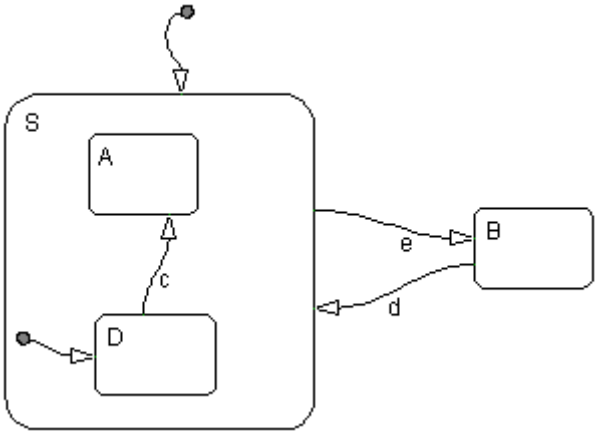
Κατόπιν, δηλώνονται οι **ενέργειες συνθήκης (condition_action)** μέσα σε {}. Οι ενέργειες αυτές εκτελούνται αμέσως μόλις η συνθήκη εκτιμηθεί ως αληθής και πριν καθοριστεί σαν έγκυρη η μετάβαση προς την κατάσταση προορισμού. Εάν δεν καθοριστεί συνθήκη, τότε υπονοείται μία συνθήκη, εκτιμάται ως αληθής και η ενέργεια συνθήκης εκτελείται. Στο παράδειγμα, εάν η συνθήκη [off_count==0] είναι αληθής, η ενέργεια συνθήκης, off_count++, εκτελείται αμέσως.

Τέλος, δηλώνονται οι **ενέργειες μετάβασης (transition action)**. Αυτές εκτελούνται αφού έχει καθοριστεί έγκυρη η μετάβαση προορισμού, και εάν η συνθήκη είναι αληθής. Στο παράδειγμα, εάν η συνθήκη [off_count==0] είναι αληθής και η κατάσταση προορισμού είναι έγκυρη, τότε εκτελείται η ενέργεια μετάβασης Light_off.

Οι **προκαθορισμένες μεταβάσεις (default transitions)** (εικόνα 2.10), χρησιμοποιούνται για να καθορίσουν σε ποια κατάσταση θα εισέλθει το σύστημα εξ' ορισμού. Είναι μία μετάβαση η οποία καταλήγει μόνο σε μία κατάσταση προορισμού. Επίσης, χρησιμοποιείται για να καθορίσει σε ποια κατάσταση OR θα πρέπει να εισέλθει το σύστημα όταν υπάρχουν δύο ή περισσότερες γειτονικές καταστάσεις OR στο ίδιο επίπεδο ιεραρχίας.



Εικόνα 2.10 Σχηματική απεικόνιση προκαθορισμένης μετάβασης (default transition).



Εικόνα 2.11 Παράδειγμα χρήσης προκαθορισμένων μεταβάσεων.

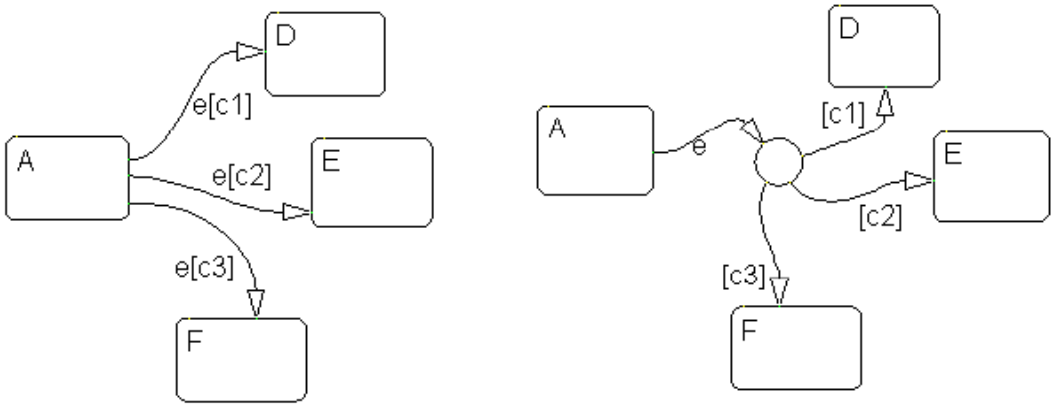
Στην εικόνα 2.11, δίνεται ένα παράδειγμα για τις χρήσεις των προκαθορισμένων καταστάσεων. Όταν το διάγραμμα ενεργοποιείται για πρώτη φορά, θα πρέπει να αποφασίσει ποια από τις καταστάσεις S ή B πρέπει να ενεργοποιηθεί, αφού είναι καταστάσεις OR. Επειδή έχουμε προκαθορισμένη μετάβαση στην S, αυτή ενεργοποιείται. Η κατάσταση S, που είναι τώρα ενεργή, έχει δύο υπό-καταστάσεις, την A και την D. Η κατάσταση που πρέπει να ενεργοποιηθεί εξ' ορισμού και πάλι δίνεται από την προκαθορισμένη μετάβαση και είναι η κατάσταση D.

Οι **συνδετικοί σύνδεσμοι (connective junctions)** χρησιμοποιούνται για την αναπαράσταση κοινών δομών όπως οι επαναλήψεις for και οι δομές if-then-else, χωρίς τη χρήση καταστάσεων. Με την μείωση του αριθμού των καταστάσεων τα διαγράμματα Stateflow παράγουν αποδοτικότερο κώδικα που βοηθάει στην βελτιστοποίηση της χρήσης της μνήμης. Η απεικόνιση ενός συνδετικού συνδέσμου δίνεται στην εικόνα 2.12. Υπάρχουν οι ακόλουθοι συνδυασμοί:

- Μεταβάσεις από και προς συνδετικούς συνδέσμους.
- Επαναλήψεις σε συνδετικούς συνδέσμους.
- Εσωτερικές μεταβάσεις σε συνδετικούς συνδέσμους.

Εικόνα 2.12 Γραφική απεικόνιση συνδετικού συνδέσμου.

Στο παρακάτω παράδειγμα (εικόνα 2.13), στα αριστερά, εάν η κατάσταση A είναι ενεργή όταν το γεγονός e συμβεί, τότε η μετάβαση από την κατάσταση A σε οποιαδήποτε κατάσταση D, E, F θα λάβει χώρα εάν πραγματοποιηθεί μία από τις συνθήκες [c1], [c2] ή [c3]. Στην ισοδύναμη αναπαράσταση στα δεξιά, η μετάβαση από την κατάσταση πηγή (source state) στον συνδετικό σύνδεσμο χαρακτηρίζεται από την ετικέτα του γεγονότος. Οι μεταβάσεις από τον σύνδεσμο προς κάθε κατάσταση προορισμό (destination state) χαρακτηρίζονται από τις συνθήκες. Εάν η κατάσταση A είναι ενεργή όταν συμβεί το γεγονός e, πρώτα συμβαίνει η μετάβαση από την κατάσταση A στον σύνδεσμο. Ακολουθεί η μετάβαση από τον σύνδεσμο σε κάποια κατάσταση προορισμό, βασιζόμενη στο ποια από τις συνθήκες [c1], [c2] ή [c3] είναι αληθής. Εάν καμία από τις συνθήκες δεν είναι αληθής, τότε δεν συμβαίνει μετάβαση και η κατάσταση A παραμένει ενεργή.



Εικόνα 2.13 Παράδειγμα χρήσης συνδετικών συνδέσμων.

2.1.6.2 Μη γραφικά αντικείμενα (Non Graphical objects)

Τα γεγονότα (events), τα δεδομένα (data) και τα αντικείμενα στόχου (target objects), δεν έχουν γραφική αναπαράσταση σε ένα διάγραμμα Stateflow.

Τα **γεγονότα (events)** οδηγούν στην εκτέλεση των διαγραμμάτων. Η εμφάνιση ενός γεγονότος μπορεί να προκαλέσει την εμφάνιση μίας μετάβασης ή την εκτέλεση μίας ενέργειας. Επειδή τα διαγράμματα του Stateflow ενεργοποιούνται αντιδρώντας στα γεγονότα, πρέπει να καθορίζονται και να προγραμματίζονται όλα τα γεγονότα που επηρεάζουν το διάγραμμα Stateflow. Μπορούμε να ορίσουμε άμεσα (explicit) γεγονότα, τα οποία δηλώνονται αμέσως. Επίσης, μπορούμε να δηλώσουμε έμμεσα (implicit) γεγονότα, τα οποία λαμβάνουν χώρα όταν εκτελούνται συγκεκριμένες ενέργειες.

Ένα διάγραμμα Stateflow αποθηκεύει και ανακτά **δεδομένα (data)** τα οποία τα χρησιμοποιεί για τον έλεγχο της εκτέλεσης του. Τα δεδομένα αυτά ανήκουν στο χώρο εργασίας του Stateflow, αλλά μπορούμε να έχουμε πρόσβαση και σε δεδομένα τα οποία ανήκουν εξωτερικά στο μοντέλο του Simulink ή στην εφαρμογή που ενσωματώνει ο μηχανισμός του Stateflow. Κατά την δημιουργία ενός μοντέλου στο Stateflow, πρέπει να ορίζονται όλα τα εσωτερικά και εξωτερικά δεδομένα (internal and external data) που χρησιμοποιούνται στο διάγραμμα του Stateflow.

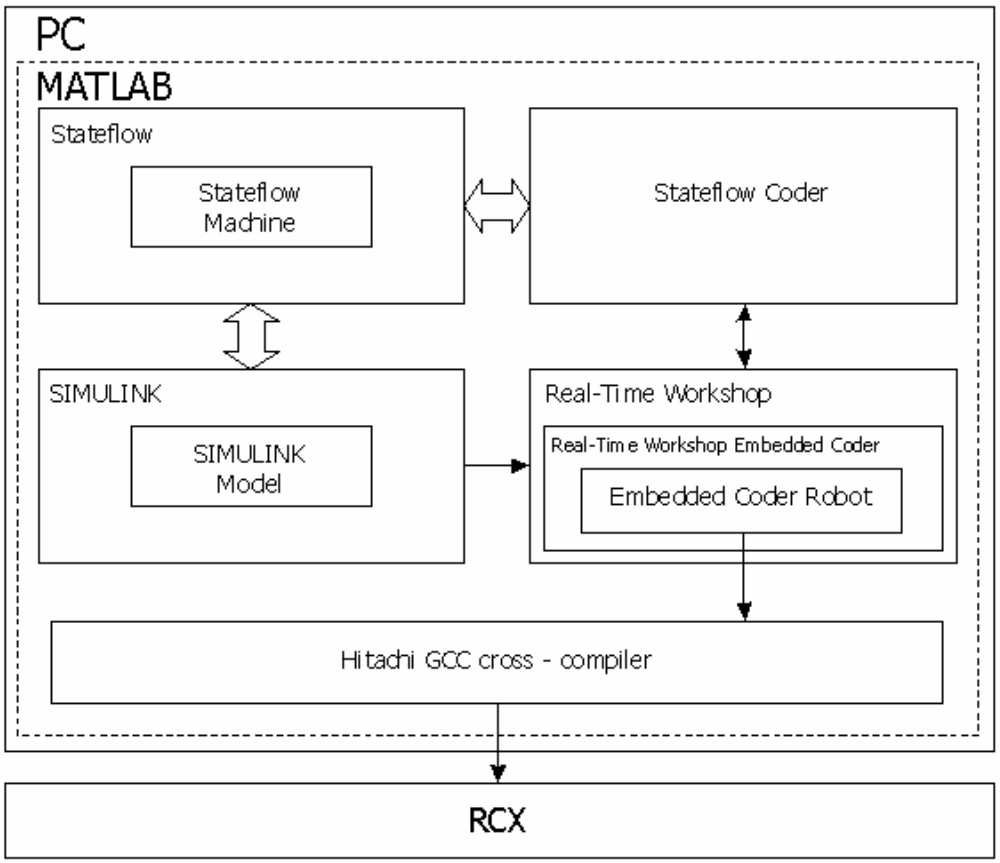
Τέλος, υπάρχουν και τα **αντικείμενα στόχου (target objects)**. Κατασκευάζουμε στόχους (targets) στο Stateflow για να εκτελέσουμε τις εφαρμογές που κατασκευάσαμε στα διαγράμματα του Stateflow και στα μοντέλα του Simulink που τα περιέχουν. Ένας στόχος είναι ένα πρόγραμμα που εκτελεί ένα μοντέλο Stateflow ή ένα μοντέλο Simulink που περιέχει τον μηχανισμό του Stateflow. Είναι δυνατή η κατασκευή ενός στόχου προσομοίωσης (simulation target) για την εκτέλεση μίας προσομοίωσης του μοντέλου και ενός στόχου Real-Time Workshop (Real-Time Workshop target) για την εκτέλεση του μοντέλου Simulink σε ένα υποστηριζόμενο περιβάλλον (εδώ στο RCX).

2.1.6.3 Διαδικασία παραγωγής κώδικα στη MATLAB

Ένας στόχος (target) είναι ένα πρόγραμμα που εκτελεί ένα μοντέλο Stateflow ή Simulink, που περιέχει τον μηχανισμό του Stateflow. Το Stateflow και τα εργαλεία που λειτουργούν σε συνεργασία με αυτό, μπορούν να κατασκευάσουν στόχους σχεδόν για κάθε υπολογιστή.

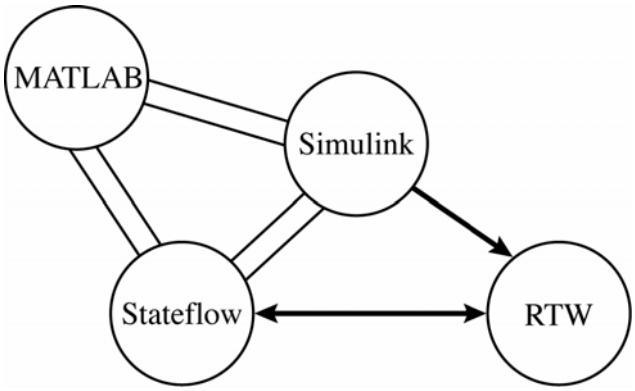
Το Stateflow κατασκευάζει έναν στόχο για τα διαγράμματα του με την εξής διαδικασία. Πρώτα, το Stateflow αναλύει τα διαγράμματα για να εξασφαλίσει ότι η λογική τους είναι έγκυρη. Εάν βρεθούν λάθη, τότε το Stateflow τα εμφανίζει στο παράθυρο εντολών της MATLAB και σταματάει την διαδικασία. Εάν τα διαγράμματα αναλυθούν και δεν βρεθούν λάθη, τότε το Stateflow επικαλείται μία γεννήτρια κώδικα για την μετατροπή των διαγραμμάτων Stateflow σε πηγαίο κώδικα C. Η γεννήτρια κώδικα παράγει ένα makefile για την κατασκευή του παραγόμενου πηγαίου κώδικα σε ένα εκτελέσιμο πρόγραμμα. Τέλος, το Stateflow κατασκευάζει τον στόχο χρησιμοποιώντας έναν C compiler (μεταγλωττιστή C) και μία make utility. Και οι δύο αυτές εφαρμογές εγκαθίστανται με την παρούσα έκδοση της MATLAB.

Το Real-Time Workshop επιτρέπει την μετατροπή του μοντέλου Simulink, που περιέχει τον μηχανισμό του Stateflow, σε κώδικα C, χρησιμοποιώντας τον Real-Time Workshop Embedded Coder. Το rcx-standalone-0.95 αποτελεί ένα παράδειγμα ενός κατασκευασμένου στόχου (custom target) που βασίζεται στο Real-Time Workshop Embedded Coder και παράγει κώδικα για ενσωματωμένα συστήματα όπως το BrickOS. Ο κώδικας C, που παράγεται από το Real-Time Workshop Embedded Coder, μεταγλωττίζεται από τον GCC cross-compiler για τον μικροελεγκτή της Hitachi, h8300-hms-gcc. Τα προγράμματα που παράγονται από τον στόχο rcx-standalone-0.95, τρέχουν στη μονάδα RCX των Lego Mindstorms Robotics Invention System 2.0. Η αυτόματη διαδικασία παραγωγής κώδικα της MATLAB για το RCX φαίνεται παρακάτω (εικόνα 2.14):



Εικόνα 2.14 Η διαδικασία παραγωγής κώδικα στην MATLAB.

Τα παραπάνω πακέτα λογισμικού της Mathworks δεν λειτουργούν μεμονωμένα αλλά συνεργάζονται μεταξύ τους και η σχέση τους φαίνεται στην εικόνα 2.15.



Εικόνα 2.15 Σχέση μεταξύ των πακέτων λογισμικού Mathworks.

2.2 Πρόσθετο λογισμικό

2.2.1 BrickOS (LegOS)

Το λειτουργικό σύστημα που υπάρχει στο RCX και διατίθεται από την Lego περιλαμβάνει έναν διερμηνέα δυαδικού κώδικα (bytecode interpreter), που καθιστά το RCX ικανό να καταλαβαίνει τον κώδικα που παράγεται από το γραφικό περιβάλλον του λογισμικού της Lego (ROBOLAB). Επιτρέπει τον προγραμματισμό του RCX σε κώδικα του RCX, είναι όμως ένα πολύ απλό και περιορισμένο εργαλείο. Αντικαθιστώντας το λειτουργικό αυτό σύστημα με ένα άλλο, ο προγραμματιστής αποκτά εξ' ολοκλήρου τον έλεγχο χαμηλού επιπέδου του RCX. Μετά την κυκλοφορία των Lego Mindstorms, έγιναν διαθέσιμα αρκετά ανεξάρτητα εργαλεία ανάπτυξης όπως το NQC (Not Quiet C), το pbForth (Programmable Brick FORTH) και το BrickOS.

Το BrickOS είναι το πιο δυναμικό εργαλείο ανάπτυξης που έχει σχεδιαστεί να λειτουργεί στο RCX των Lego Mindstorms που βασίζεται στο μικροελεγκτή Hitachi H8/3292. Ξεκίνησε από τον Markus Noga (με την ονομασία LegOS) τον Οκτώβριο του 1998 και συνεχίζει να αναπτύσσεται σαν λογισμικό ανοικτού κώδικα (open source).

Το λειτουργικό αυτό σύστημα αντικαθιστά το σύστημα που είναι εγκατεστημένο στο RCX. Προσφέρει μεγάλη βελτίωση σε σχέση με το κλασσικό λειτουργικό σύστημα της Lego, και στην απόδοση αλλά και στην ευελιξία. Είναι ένα ολοκληρωμένο πακέτο το οποίο δίνει πρόσβαση στους κινητήρες, στους αισθητήρες και στα άλλα μέρη του RCX. Το BrickOS προσφέρει στους προγραμματιστές την δυνατότητα να αναπτύξουν προγράμματα για το RCX σε γλώσσες προγραμματισμού όπως η C και η C++.

Αποτελεί ένα ισχυρό εργαλείο, το οποίο όμως δεν είναι εύκολο στη χρήση. Στην αρχή κατασκευάστηκε για τα Linux και ακόμη και αν έχει γίνει αρκετή δουλεία για να μετασχηματιστεί και να τρέξει σε περιβάλλον Windows, χρειάζεται η εγκατάσταση ενός προσομοιωτή περιβάλλοντος Linux (Linux emulator) όπως το Cygwin, για να λειτουργήσει.

Επίσης, απαιτείται ένας δια-μεταγλωττιστής C (C cross-compiler) για την μεταγλώττιση των προγραμμάτων πριν το τρέξιμο τους στο RCX. Ο δια-μεταγλωττιστής τρέχει στον ηλεκτρονικό υπολογιστή αλλά παράγει κώδικα για την πλατφόρμα του μικροελεγκτή της Hitachi. Επιτρέπει στον προγραμματιστή να γράψει πηγαίο κώδικα (source code) στη C σε ένα οποιοδήποτε περιβάλλον, ο οποίος στη συνέχεια μεταγλωττίζεται σε εκτελέσιμο πρόγραμμα για έναν διαφορετικό στόχο, δηλαδή τον μικροελεγκτή H8/3292.

2.2.2 Cygwin

Το Cygwin είναι μια συλλογή από εργαλεία ανοικτού κώδικα που επιτρέπει σε εφαρμογές Linux να μεταγλωττίζονται και να τρέχουν σε λειτουργικά συστήματα Windows, μέσα από ένα περιβάλλον διεπαφής που ομοιάζει με Linux. Η δυνατότητα αυτή, επιτρέπει στους προγραμματιστές να μεταφέρουν τις εφαρμογές τους από το λειτουργικό σύστημα Linux στο λειτουργικό σύστημα Windows, κάνοντας ευκολότερη την υποστήριξη τους.

2.2.3 GCC cross compiler

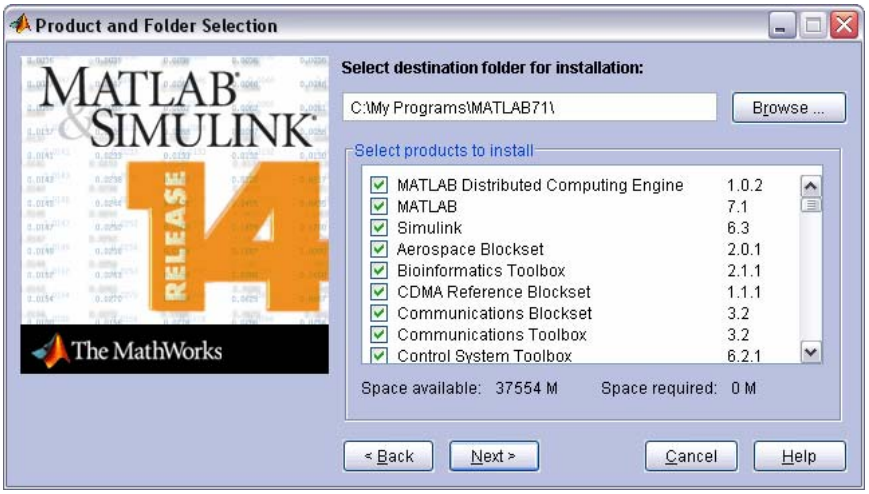
Ένας δια-μεταγλωττιστής (cross compiler) είναι ένας μεταγλωττιστής που τρέχει σε έναν υπολογιστή αλλά είναι ικανός να δημιουργήσει εκτελέσιμο κώδικα για έναν διαφορετικό υπολογιστή. Οι δια-μεταγλωττιστές χρησιμοποιούνται για την παραγωγή λογισμικού που μπορεί να τρέχει σε υπολογιστές με μια νέα αρχιτεκτονική ή σε συσκευές ειδικού σκοπού, οι οποίες δεν μπορούν να φιλοξενήσουν τους δικούς τους μεταγλωττιστές. Ένα τέτοιο εργαλείο είναι βολικό όταν χρειαζόμαστε να μεταγλωττίσουμε κώδικα για μία πλατφόρμα στην οποία δεν έχουμε πρόσβαση ή επειδή είναι αδύνατη η μεταγλώττιση για την συγκεκριμένη πλατφόρμα. Ο δια-μεταγλωττιστής που θα χρησιμοποιηθεί στην παρούσα εργασία είναι ο Gcc cross compiler, ο οποίος προσφέρεται δωρεάν στο διαδίκτυο, είναι λογισμικό ανοικτού κώδικα και υποστηρίζει δεκάδες πλατφόρμες και γλώσσες προγραμματισμού.

2.3 Εγκατάσταση λογισμικού

Μετά την περιγραφή του απαραίτητου λογισμικού, ακολουθεί μία σύντομη περιγραφή της διαδικασίας εγκατάστασης του. Τα βήματα που πρέπει να ακολουθηθούν είναι:

1. **Εγκατάσταση της MATLAB.** Για την δημιουργία, την τροποποίηση και την αποστολή των μοντέλων Simulink, μέσω του πύργου υπερύθρων στο RCX, είναι απαραίτητη η εγκατάσταση της MATLAB έκδοση R14 SP3 ή νεότερη (εικόνα 2.16). Επιλέγουμε τα παρακάτω πακέτα λογισμικού, καθώς είναι απαραίτητα τόσο για την επεξεργασία των μοντέλων στο γραφικό περιβάλλον του Simulink όσο και για την επικοινωνία με το RCX:

    - Simulink (έκδοση 6.3)[1]
    - Stateflow (έκδοση 6.3)
    - Stateflow Coder (έκδοση 6.3)
    - Real-Time Workshop (έκδοση 6.3)
    - Real-Time Workshop Embedded Coder (έκδοση 4.3)



Εικόνα 2.16 Παράθυρο εγκατάστασης πακέτων λογισμικού της MATLAB.

[1] ΠΡΟΣΟΧΗ! Τα μοντέλα έχουν κατασκευαστεί με την έκδοση 6.3 του Simulink. Η χρήση οποιασδήποτε προηγούμενης έκδοσης θα εμφανίσει μηνύματα σφάλματος.

2. **Εγκατάσταση του λογισμικού rcx-standalone-0.95.** Αποσυμπιέζουμε το αρχείο rcx-standalone-0.95.zip σε έναν φάκελο, κατά προτίμηση έξω από τον κύριο φάκελο της MATLAB (για παράδειγμα στον C:\MY_RCX). Κατά την αποσυμπίεση, εμφανίζονται οι παρακάτω φάκελοι:

   - **blocks:** Περιέχει τα δομικά στοιχεία των εισόδων και εξόδων του RCX τα οποία χρησιμοποιούνται στο περιβάλλον εργασίας του Simulink.
   - **brickos-0.2.6.10.6:** Περιέχει τα αρχεία του λειτουργικού συστήματος BrickOS.
   - **cygwin:** Περιέχει τα αρχεία που χρειάζονται για την εξομοίωση του περιβάλλοντος εργασίας Linux σε περιβάλλον Windows. Επίσης, περιέχει τον δια-μεταγλωττιστή GCC (GCC cross compiler) για τον μικροελεγκτή της Hitachi, h8300-hms-gcc.
   - **demos:** Περιέχει διάφορα μοντέλα επίδειξης.
   - **doc:** Περιέχει τα αρχεία τεκμηρίωσης της MATLAB, καθώς και εγχειρίδια για τον μικροεπεξεργαστή της Hitachi.
   - **environment:** Περιέχει τα αρχεία που χρειάζονται από το Simulink για την προσομοίωση των μοντέλων στον υπολογιστή.
   - **rcx:** Περιέχει τα αρχεία που είναι απαραίτητα για την εγκατάσταση και λειτουργία του στόχου στο Real-Time Workshop.

   Μετά από την αποσυμπίεση, ανοίγουμε την MATLAB και θέτουμε σαν ενεργό κατάλογο (Current Directory), τον κατάλογο στον οποίο αποσυμπιέσαμε το παραπάνω αρχείο (στο παράδειγμά μας ο C:\MY_RCX). Στη γραμμή εντολών της MATLAB γράφουμε την παρακάτω σειρά εντολών:

   ```
   >> cd rcx
   >> rcxsetup
   ```

   Στη συνέχεια εμφανίζονται τα παρακάτω μηνύματα εγκατάστασης του στόχου. Δύο διάλογοι μας ρωτούν που είναι εγκατεστημένες οι εφαρμογές Cygwin και BrickOS. Αφήνουμε τις προεπιλεγμένες ρυθμίσεις όπως φαίνεται και παρακάτω:

starting rcx target installation.

Set path where CYGWIN (version 1.3 or later) is installed.
Change current value 'C:\MY_RCX\cygwin' (Y/N) ? [N]

Set path where brickOS is installed.
Change current value 'C:\MY_RCX\brickos-0.2.6.10.6' (Y/N) ? [N]

3. **Εγκατάσταση του ecrobot2_demo.** Παίρνουμε τον φάκελο ecrobot2_demo και τον τοποθετούμε μέσα στον φάκελο που δημιουργήσαμε προηγουμένως (C:\MY_RCX\ecrobot2_demo). Πηγαίνουμε στη MATLAB, κάνουμε κλικ διαδοχικά στο File→ Set Path… και στον κατάλογο που εμφανίζεται πατάμε Add Folder… και προσθέτουμε τους φακέλους C:\MY_RCX\ ecrobot2_demo\devices, C:\MY_RCX\ecrobot2_demo\environment και C:\MY_RCX\ecrobot2_demo\demos.

4. **Ρυθμίσεις Real-Time Workshop.** Λεπτομερείς οδηγίες για τις ρυθμίσεις που πρέπει να γίνουν στο Real-Time Workshop, δίνονται στο κεφάλαιο 3.2.

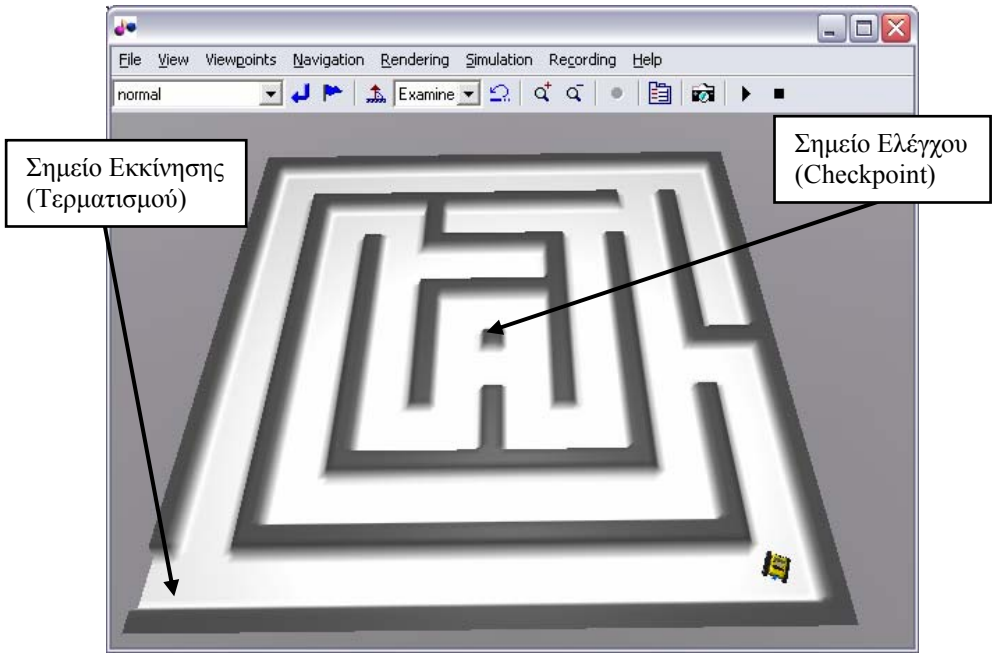
ΚΕΦΑΛΑΙΟ 3: ΚΑΤΑΣΚΕΥΗ ΚΑΙ ΑΠΟΣΤΟΛΗ ΜΟΝΤΕΛΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΗΣ MATLAB

3.1. Παρουσίαση των δύο μοντέλων

Στο κεφάλαιο αυτό, παρουσιάζονται τα δύο μοντέλα, Explorer.mdl και Linetracker.mdl, κατασκευασμένα στο Simulink με χρήση και διαγραμμάτων Stateflow. Ένα πλεονέκτημα που έχουμε με αυτά τα μοντέλα είναι ότι μέσα από ένα περιβάλλον εικονικής πραγματικότητας μπορούμε να προσομοιώσουμε και να παρατηρήσουμε την συμπεριφορά του ρομπότ στην οθόνη του υπολογιστή. Στην αρχή, παρουσιάζονται περιληπτικά τα δύο πλήρη μοντέλα, μαζί με τον μηχανισμό της προσομοίωσης τους. Στη συνέχεια μπορούμε με τη χρήση του Real-Time Workshop να τα κωδικοποιήσουμε και να τα στείλουμε στο RCX, ώστε να τρέξουν σε πραγματικό χρόνο στο RCX.

3.1.1 Το μοντέλο Explorer.mdl

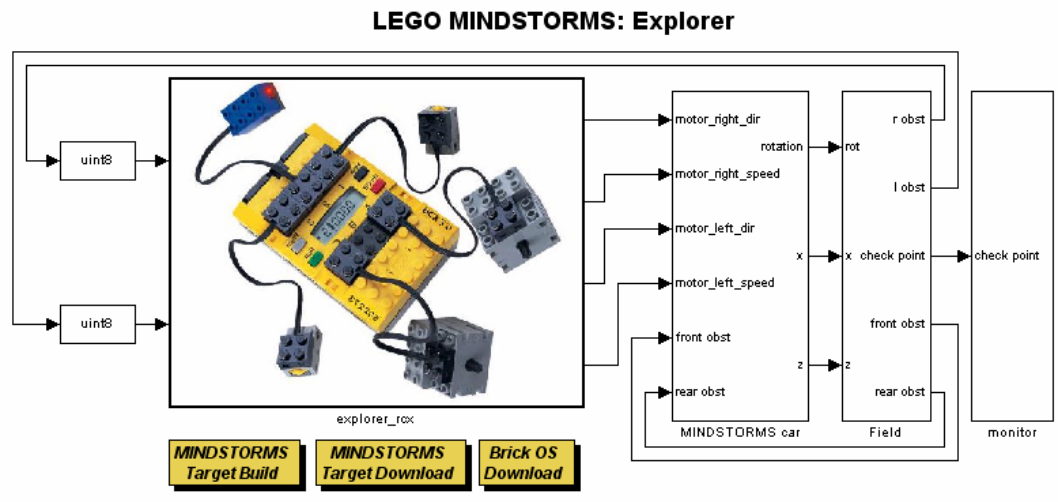
Το μοντέλο Explorer, αποτελεί έναν ¨ερευνητή λαβυρίνθων¨. Με το μοντέλο αυτό, το αυτοκίνητο που κατασκευάζεται, ξεκινάει από το σημείο εκκίνησης (εικόνα 3.1) και ακολουθώντας την εκάστοτε διαδρομή φτάνει στο σημείο ελέγχου. Στη συνέχεια, προσπαθεί να επιστρέψει στο σημείο από το οποίο ξεκίνησε.



Εικόνα 3.1 Εικόνα από την προσομοίωση του μοντέλου Linetracker στη MATLAB.

Το πλήρες διάγραμμα Simulink για το μοντέλο αυτό δίνεται στην εικόνα 3.2. Μπορούμε να παρατηρήσουμε ότι το διάγραμμα αυτό αποτελείται από 4 τμήματα. Το πρώτο τμήμα, που ονομάζεται **explorer_rcx** ουσιαστικά αποτελεί την στρατηγική ελέγχου του ρομπότ. Το τμήμα αυτό είναι υπεύθυνο για την κίνηση του ρομπότ μέσα στο λαβύρινθο. Το δεύτερο τμήμα του διαγράμματος, που ονομάζεται **MINDSTORMS car** και αποτελεί το εικονικό ρομπότ που χρησιμοποιείται κατά τη διάρκεια της προσομοίωσης στη MATLAB. Το τρίτο τμήμα, με την ονομασία **Field**, αποτελεί τον εικονικό λαβύρινθο στον οποίο θα τρέξει η προσομοίωση. Σαν λαβύρινθος μπορεί να χρησιμοποιηθεί οποιοδήποτε μονόχρωμο αρχείο bitmap (.bmp) το οποίο θα αποτελέσει τον χώρο μέσα στον οποίο θα κινηθεί το εικονικό ρομπότ MINDSTORMS car. Τέλος, το τελευταίο τμήμα του διαγράμματος, με την ονομασία **monitor**, χρησιμοποιείται για τον έλεγχο του συνολικού χρόνου από την αρχή της προσομοίωσης, του χρόνου που χρειάστηκε το ρομπότ για να πάει από το σημείο εκκίνησης στο σημείο ελέγχου και από το σημείο ελέγχου στον τερματισμό.

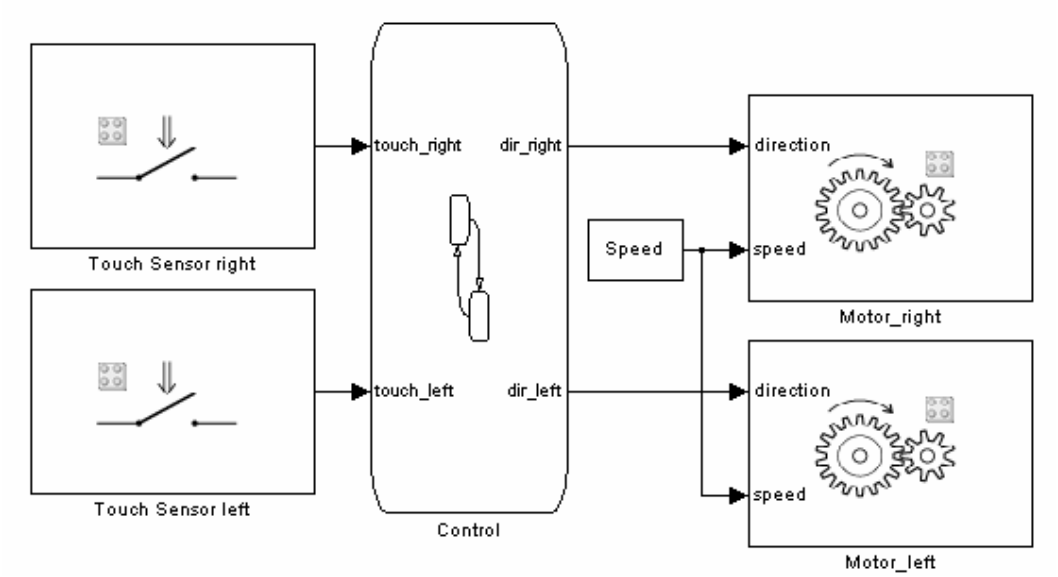
Τα διαγράμματα Simulink και Stateflow που χρησιμοποιούνται για την προσομοίωση αυτού του μοντέλου στον υπολογιστή δίνονται αναλυτικά στο παράρτημα Γ.



Εικόνα 3.2 Το πλήρες διάγραμμα Simulink για το μοντέλο Explorer.mdl.

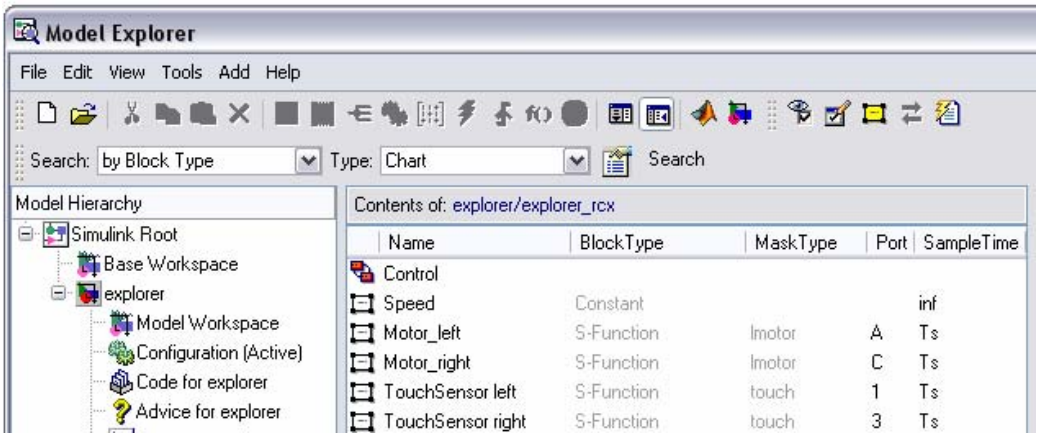
Ακολουθεί μια πιο λεπτομερής περιγραφή των διαγραμμάτων Simulink τα οποία χρησιμοποιούνται για τον έλεγχο της κίνησης του πραγματικού ρομπότ. Στη συνέχεια κωδικοποιούνται από το Real Time Workshop και αποστέλλονται μέσω του πύργου υπερύθρων στο RCX.

Το διάγραμμα Simulink το οποίο είναι υπεύθυνο για τον έλεγχο της κίνησης του ρομπότ φαίνεται στην εικόνα 3.3.



Εικόνα 3.3 Διάγραμμα Simulink για την στρατηγική ελέγχου του μοντέλου Explorer.

Στο κεντρικό παράθυρο του Simulink κάνουμε κλικ στο View→ Model Explorer και εμφανίζεται το παρακάτω παράθυρο (εικόνα 3.4):

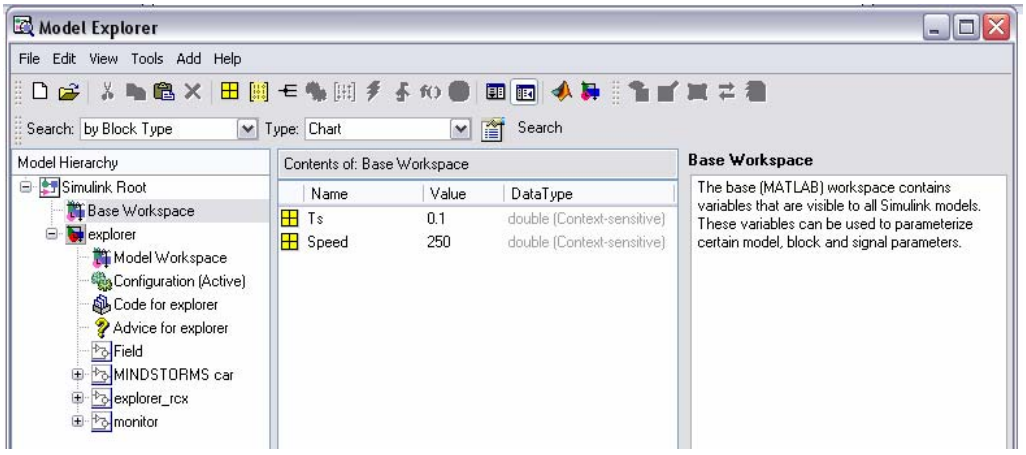


Εικόνα 3.4 Τα στοιχεία του μοντέλου Explorer.mdl.

Στο παράθυρο αυτό βλέπουμε όλα τα στοιχεία από τα οποία αποτελείται το διάγραμμα Simulink του Explorer.mdl. Αυτά είναι ένα δομικό στοιχείο Stateflow (Control), μία σταθερά (Speed) και τέσσερις κατασκευασμένες συναρτήσεις συστήματος (S-Functions), που αποτελούν και αυτές δομικά στοιχεία του Simulink (Motor_left, Motor_right, TouchSensor left και TouchSensor right). Η λειτουργία όλων αυτών των στοιχείων περιγράφεται αναλυτικά στη συνέχεια.

Κάνοντας κλικ στο Base Workspace, βλέπουμε το βασικό χώρο εργασίας της MATLAB (εικόνα 3.5). Σε αυτόν περιέχονται όλες οι χρησιμοποιούμενες μεταβλητές που εμφανίζονται στο διάγραμμα Simulink. Αυτές οι μεταβλητές χρησιμοποιούνται για την παραμετροποίηση του μοντέλου, των δομικών στοιχείων του και του σήματος του.

Εδώ, χρησιμοποιούμε δύο μεταβλητές, την Ts και την Speed. Η **μεταβλητή Ts** θα χρησιμοποιηθεί σαν χρόνος δειγματοληψίας (Sample Time). Ο χρόνος δειγματοληψίας καθορίζει τον ρυθμό που θα συμβαίνει η διαδικασία ανάγνωσης των τιμών που δίνει ο αισθητήρας. Αυτό σημαίνει ότι κάθε χρόνο ίσο με Ts θα ελέγχονται οι τιμές που δίνουν οι αισθητήρες επαφής. Η **μεταβλητή Speed,** παίρνει τιμές από 0-255 και καθορίζει την ταχύτητα περιστροφής (Speed) των κινητήρων του αυτοκινήτου (εδώ είναι 250).

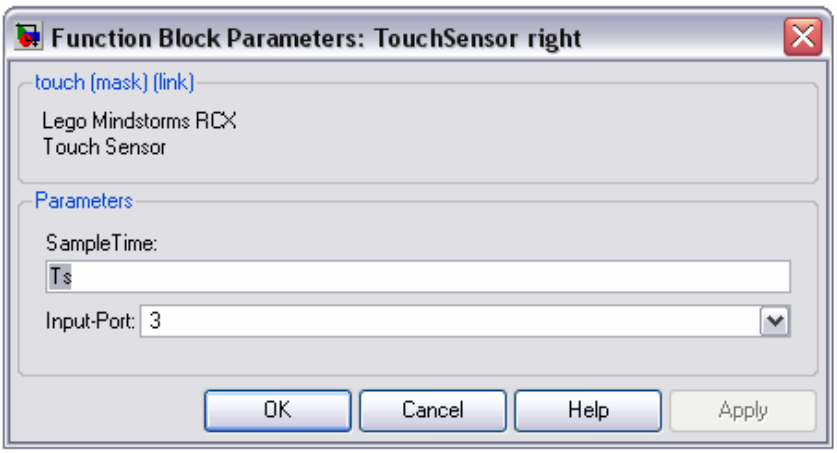


Εικόνα 3.5 Ο βασικός χώρος εργασίας (Base Workspace) στον Model Explorer.

Στη συνέχεια θα δούμε αναλυτικά τα δομικά στοιχεία (blocks) από τα οποία αποτελείται το διάγραμμα Simulink. Αυτά είναι:

1. **Δομικά Στοιχεία Αισθητήρων Επαφής (Inputs):** Σαν είσοδοι στο μοντέλο χρησιμοποιούνται δύο συναρτήσεις συστήματος (S-Functions), η TouchSensor right και η TouchSensor left. Και οι δύο συναρτήσεις συστήματος των αισθητήρων επαφής έχουν σαν παραμέτρους τον χρόνο δειγματοληψίας (Sample Time) και τον αριθμό της θύρας εισόδου (Input Port) όπως φαίνεται στην εικόνα 3.6. Ο χρόνος δειγματοληψίας καθορίζεται ως Ts (δηλαδή 0,1 sec). Στη συνέχεια, ο αριστερός αισθητήρας επαφής συνδέεται στην θύρα εισόδου με τον αριθμό 1 στο RCX και ο δεξιός στη θύρα με τον αριθμό 3.

Οι τιμές που δίνουν οι αισθητήρες είναι 0 (ο αισθητήρας μη ενεργοποιημένος) ή 1 (ο αισθητήρας ενεργοποιημένος). Αυτές οι τιμές λειτουργούν σαν γεγονότα μέσα στο δομικό διάγραμμα του Stateflow.

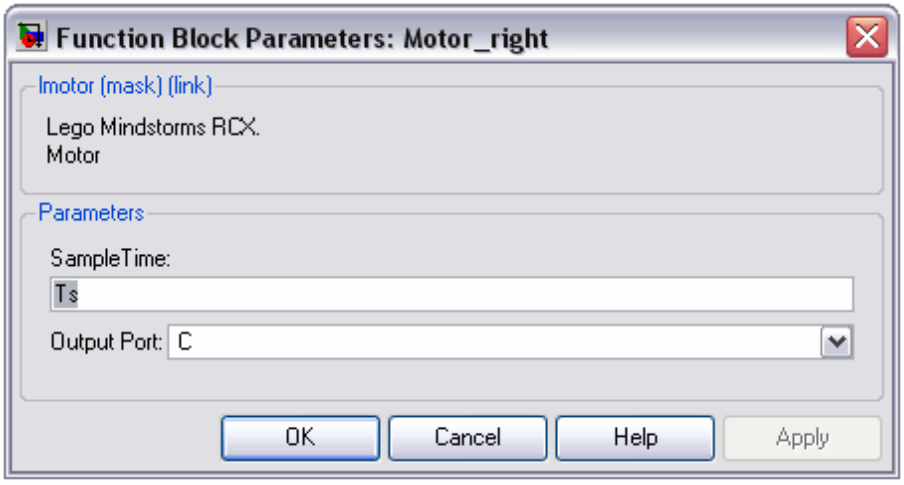


Εικόνα 3.6 Παράμετροι των δομικών στοιχείων των αισθητήρων.

2. **Δομικά Στοιχεία Κινητήρων (Outputs):** Σαν έξοδοι στο μοντέλο χρησιμοποιούνται δύο συναρτήσεις συστήματος, η Motor_right και η Motor_left. Και οι δύο συναρτήσεις συστήματος των κινητήρων έχουν σαν παραμέτρους τον χρόνο δειγματοληψίας (Sample Time) και τον αριθμό της θύρας εξόδου (Output Port) όπως φαίνεται στην εικόνα 3.7. Ο χρόνος δειγματοληψίας καθορίζεται πάλι ως Ts. Στη

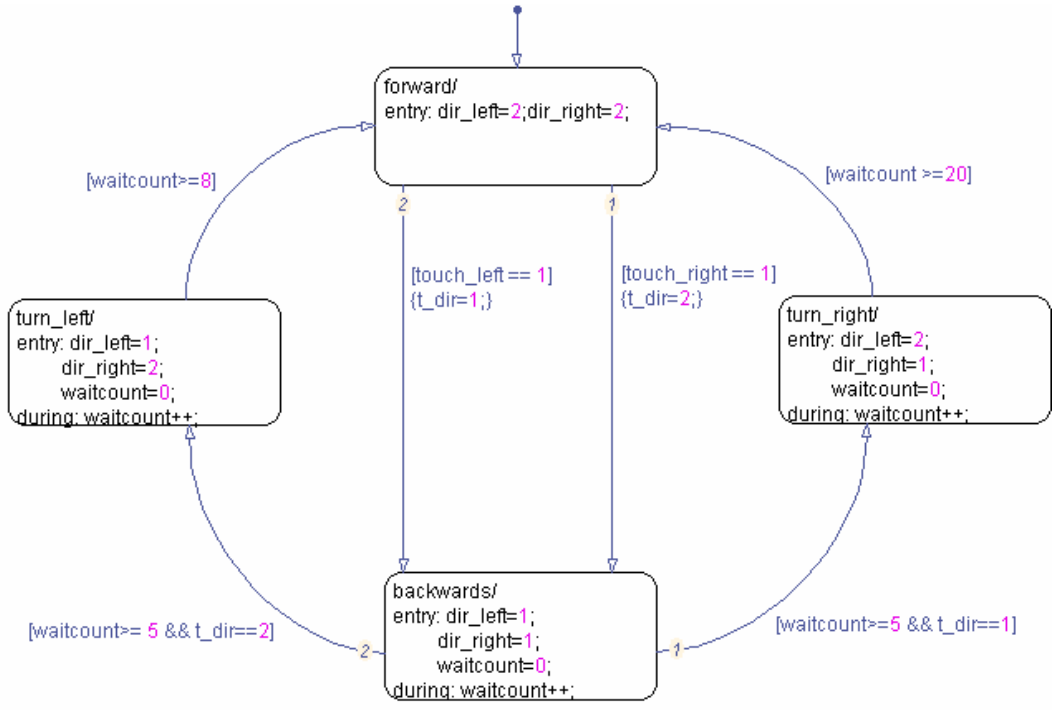
συνέχεια, ο αριστερός κινητήρας συνδέεται στην θύρα εξόδου με το γράμμα A στο RCX και ο δεξιός στη θύρα με το γράμμα C.

Οι συναρτήσεις συστήματος των κινητήρων περιγράφονται από δύο εισόδους που είναι η ταχύτητα (Speed) και η διεύθυνση περιστροφής τους (Direction). Η ταχύτητα είναι μία μη προσημασμένη (unsigned) σταθερά 8-bit, δηλαδή παίρνει τιμές από 0-255, και δηλώνεται στη μεταβλητή Speed στον Model Explorer (εικόνα 3.4). Η διεύθυνση περιστροφής είναι και αυτή μία μη προσημασμένη σταθερά 8-bit και παίρνει τιμές 1 (κίνηση προς τα εμπρός), 2 (κίνηση προς τα πίσω), 3 (φρένο) ή 0 (σταματημένη).



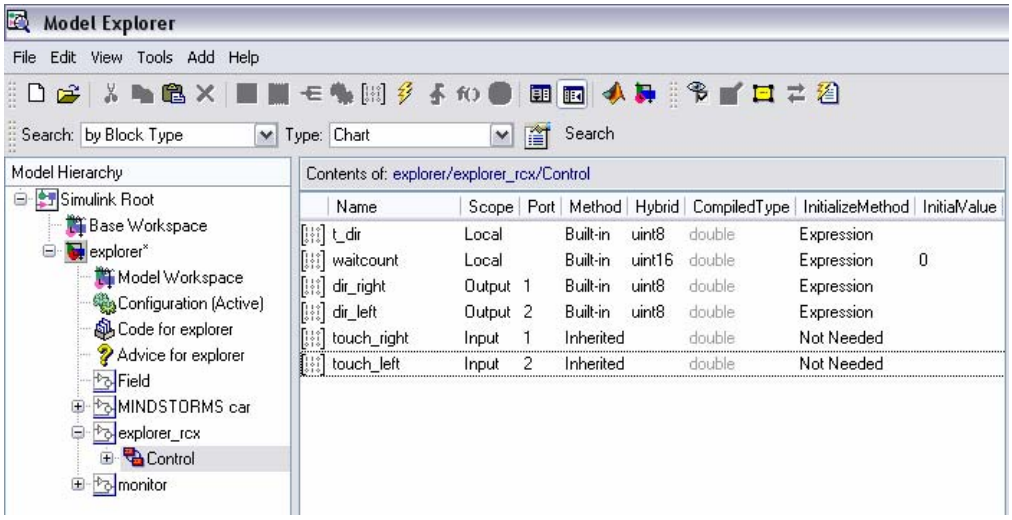
Εικόνα 3.7 Παράμετροι των δομικών στοιχείων των κινητήρων.

3. **Δομικό στοιχείο Stateflow (Control):** Το μοντέλο Simulink, Explorer.mdl, περιλαμβάνει και ένα δομικό στοιχείο Stateflow, που ονομάζεται Control. Το δομικό αυτό στοιχείο περιέχει ένα διάγραμμα Stateflow (εικόνα 3.8). Το διάγραμμα αυτό αναπαριστά την στρατηγική ελέγχου του αυτοκινήτου RCX.



Εικόνα 3.8 Το διάγραμμα Stateflow του μοντέλου Explorer.mdl.

Από το βασικό παράθυρο του Simulink ανοίγουμε τον Model Explorer (κάνοντας κλικ στο View →Model Explorer). Κάνοντας διπλό κλικ στο explorer_rcx, ανοίγει το παρακάτω παράθυρο (εικόνα 3.9).



Εικόνα 3.9 Ο Model Explorer για το δομικό στοιχείο Control.

Στο παράθυρο αυτό βλέπουμε τις χρησιμοποιούμενες μεταβλητές στο δομικό στοιχείο Control του Stateflow. Αποτελείται από δύο εισόδους (inputs), δύο εξόδους (outputs) και δύο τοπικές μεταβλητές (local).

Οι είσοδοι του διαγράμματος Stateflow, touch_right και touch_left, είναι οι τιμές που παίρνει το διάγραμμα Stateflow από τις δύο συναρτήσεις συστήματος των αισθητήρων, TouchSensor right και TouchSensor left αντίστοιχα (0 όταν ο αισθητήρας δεν είναι ενεργοποιημένος και 1 όταν είναι ενεργοποιημένος). Οι τιμές αυτές αποκτώνται (inherited) από τους δύο αισθητήρες επαφής που υπάρχουν στον προφυλακτήρα του αυτοκινήτου.

Οι τοπικές μεταβλητές (local), είναι η t_dir και η waitcount, κάθε μία από τις οποίες εκτελεί και μία διαφορετική λειτουργία. Η **τοπική μεταβλητή t_dir** (touch direction: left sensor=1, right sensor=2), παίρνει μη προσημασμένες ακέραιες τιμές 8 bit (uint8) και χρησιμοποιείται για τον καθορισμό του αισθητήρα που ενεργοποιεί το διάγραμμα Stateflow. Η **τοπική μεταβλητή waitcount**, παίρνει μη προσημασμένες ακέραιες τιμές 16 bit (uint16) και χρησιμοποιείται για τον καθορισμό της διάρκειας παραμονής σε μία ορισμένη κατάσταση. Η τιμή της αρχικοποιείται στο 0 (InitialValue = 0).

Οι δύο έξοδοι του δομικού στοιχείου του Stateflow, καθορίζουν τις τιμές της διεύθυνσης περιστροφής του αριστερού και δεξιού κινητήρα που θα σταλούν στις δύο συναρτήσεις συστήματος Motor_left και Motor_right του Simulink αντίστοιχα.

Το διάγραμμα Stateflow αποτελείται από 4 καταστάσεις:

- **Κίνηση προς τα εμπρός (forward):** Μόλις η κατάσταση αυτή ενεργοποιηθεί, εκτελούνται οι ενέργειες εισόδου (entry), δηλαδή οι διευθύνσεις περιστροφής των κινητήρων γίνονται dir_left=2 και dir_right=2, έτσι ώστε και οι δύο κινητήρες να κινούνται προς την ίδια κατεύθυνση. Στην συγκεκριμένη περίπτωση βάζουμε τους κινητήρες να έχουν διεύθυνση περιστροφής προς τα πίσω (δηλαδή 2), επειδή για να κινηθεί το αυτοκίνητο προς τα εμπρός με την σύνδεση των γραναζιών και των κινητήρων που έχουμε χρησιμοποιήσει (βλ. Παράρτημα Α), θα πρέπει οι κινητήρες να έχουν την συγκεκριμένη διεύθυνση περιστροφής. Η κατάσταση αυτή αποτελεί τον προορισμό μίας

προκαθορισμένης μετάβασης (Default transition), που σημαίνει ότι ενεργοποιείται όταν ξεκινήσει για πρώτη φορά το μοντέλο.

- **Κίνηση προς τα πίσω (backwards):** Μόλις η κατάσταση αυτή ενεργοποιηθεί, εκτελούνται οι ενέργειες εισόδου (entry), δηλαδή οι διευθύνσεις περιστροφής των κινητήρων γίνονται dir_left=1 και dir_right=1, έτσι ώστε και οι δύο κινητήρες να κινούνται με κατεύθυνση αντίθετη από αυτήν της προηγούμενης κατάστασης (δηλαδή το αυτοκίνητο κινείται προς τα πίσω). Επιπλέον, η μεταβλητή waitcount αρχικοποιείται στην τιμή 0 (waitcount=0). Η μεταβλητή αυτή χρησιμοποιείται για τον καθορισμό της διάρκειας παραμονής σε αυτήν την κατάσταση. Εκτός από τις ενέργειες εισόδου, έχουμε και μία ενέργεια που εκτελείται όταν η κατάσταση backwards είναι ενεργή και συμβεί οποιοδήποτε γεγονός. Η ενέργεια αυτή (**during: waitcount++)** είναι η αύξηση κατά μία μονάδα της μεταβλητής waitcount σε κάθε χρόνο δειγματοληψίας του μοντέλου.
- **Στροφή προς τα αριστερά (turn_left):** Μόλις ενεργοποιηθεί αυτή η κατάσταση, εκτελούνται οι ενέργειες εισόδου (entry), δηλαδή οι κινητήρες περιστρέφονται με αντίθετες κατευθύνσεις. Ο αριστερός κινητήρας κινείται προς τα πίσω (dir_left=1), ενώ ο δεξιός κινητήρας κινείται προς τα εμπρός (dir_right=2) και με τον τρόπο αυτό, το αυτοκίνητο στρίβει προς τα αριστερά. Γίνεται πάλι η αρχικοποίηση της μεταβλητής waitcount (waitcount=0). Η μεταβλητή αυτή ελέγχεται σε κάθε χρόνο δειγματοληψίας του μοντέλου και η τιμή της αυξάνεται κατά ένα κατά τη διάρκεια του παραπάνω χρόνου (**during: waitcount**++).
- **Στροφή προς τα δεξιά (turn_right):** Μόλις ενεργοποιηθεί αυτή η κατάσταση, εκτελούνται οι ενέργειες εισόδου (entry), δηλαδή οι κινητήρες περιστρέφονται με αντίθετες διευθύνσεις. Ο αριστερός κινητήρας κινείται προς τα εμπρός (dir_left=2), ενώ ο δεξιός κινητήρας κινείται προς τα πίσω (dir_right=1) και με τον τρόπο αυτό, το αυτοκίνητο στρίβει προς τα δεξιά. Γίνεται πάλι η αρχικοποίηση της μεταβλητής waitcount (waitcount=0). Η μεταβλητή αυτή ελέγχεται σε κάθε χρόνο δειγματοληψίας του μοντέλου και η τιμή της αυξάνεται κατά ένα κατά τη διάρκεια του παραπάνω χρόνου (**during: waitcount**++).

Όλες οι καταστάσεις ανήκουν στο ίδιο επίπεδο ιεραρχίας και έχουν αποκλειστική ανάλυση (exclusive decomposition). Αυτό σημαίνει ότι μόνο μία από τις καταστάσεις αυτές μπορεί να είναι ενεργή κάθε χρονική στιγμή.

Οι συνθήκες (conditions) που ενεργοποιούν τις μεταβάσεις από την μία κατάσταση στην άλλη είναι δύο:

- **[touch_right==1]:** Η συνθήκη αυτή είναι αληθής όταν το αυτοκίνητο συναντήσει κάποιο εμπόδιο και ενεργοποιηθεί ο δεξιός αισθητήρας αφής που βρίσκεται στον προφυλακτήρα του, δηλαδή δώσει την τιμή 1.
- **[touch_left==1]:** Η συνθήκη αυτή είναι αληθής όταν το αυτοκίνητο συναντήσει κάποιο εμπόδιο και ενεργοποιηθεί ο αριστερός αισθητήρας αφής που βρίσκεται στον προφυλακτήρα του, δηλαδή δώσει την τιμή 1.

Στο σημείο αυτό θα αναλύσουμε την λογική ελέγχου (control logic) του μοντέλου.

Εκκινώντας το RCX, πατώντας το κόκκινο πλήκτρο (On - Off) και μετά το πλήκτρο εκκίνησης (Run), η κατάσταση forward, που αποτελεί τον προορισμό μίας προκαθορισμένης μετάβασης (Default transition), ενεργοποιείται. Κατά συνέπεια, το αυτοκίνητο αρχίζει να κινείται προς τα εμπρός. Η ενέργεια αυτή συνεχίζεται μέχρι να συμβεί κάποιο από τα δύο πιθανά γεγονότα, δηλαδή η ενεργοποίηση ενός από τους δύο αισθητήρες αφής. Στην περίπτωση που ενεργοποιηθεί ο αριστερός αισθητήρας (το αυτοκίνητο συναντά κάποιο εμπόδιο), τότε η συνθήκη [touch_left ==1] γίνεται αληθής και πραγματοποιείται μετάβαση από την κατάσταση forward στην κατάσταση backwards. Ταυτόχρονα, επειδή έχουμε και την ενέργεια συνθήκης {t_dir=1}, η τοπική μεταβλητή t_dir παίρνει την τιμή 1. Παρόμοια γεγονότα συμβαίνουν όταν ενεργοποιηθεί ο δεξιός αισθητήρας, με μόνη διαφορά ότι η μεταβλητή t_dir παίρνει την τιμή 2.

Στην κατάσταση backwards, οι κινητήρες αρχίζουν να περιστρέφονται κατά την αντίθετη κατεύθυνση από αυτήν προηγουμένως και έτσι το αυτοκίνητο θα κινηθεί προς τα πίσω. Γίνεται αρχικοποίηση της μεταβλητής waitcount και αύξηση της κατά μία μονάδα κάθε 20 msec. Η μεταβλητή αυτή λειτουργεί σαν ρολόι και με τον τρόπο

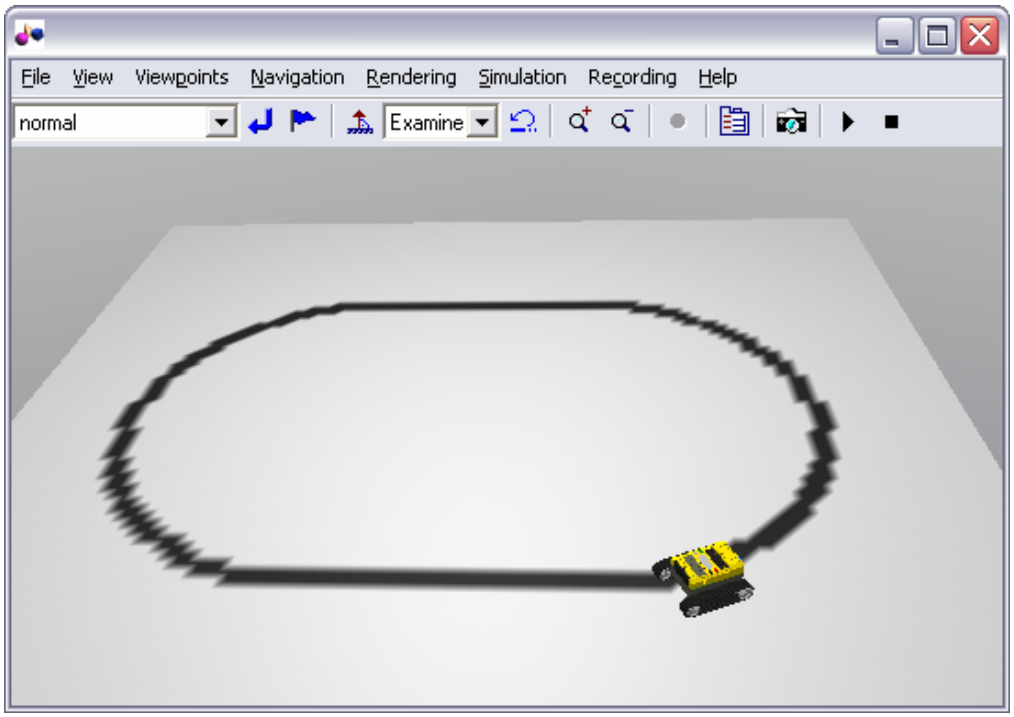
αυτό μπορούμε να καθορίσουμε την διάρκεια παραμονής στην συγκεκριμένη κατάσταση.

Για να στρίψει το αυτοκίνητο προς τα δεξιά, θα πρέπει να είναι αληθής η συνθήκη μετάβασης [waitcount>=5 && t_dir=1]. Η συνθήκη αυτή γίνεται αληθής όταν η μεταβλητή waitcount πάρει τιμή μεγαλύτερη από 100 msec (5*20 msec) ΚΑΙ (το σύμβολο && αντιπροσωπεύει τον λογικό τελεστή AND) η τιμή της μεταβλητής t_dir να είναι 1 (δηλαδή έχει ενεργοποιηθεί ο αριστερός αισθητήρας). Στην κατάσταση turn_right, οι κινητήρες περιστρέφονται με αντίθετες διευθύνσεις και έτσι το αυτοκίνητο στρίβει δεξιά. Γίνεται αρχικοποίηση της μεταβλητής waitcount και το αυτοκίνητο μένει σε αυτήν την κατάσταση για 400 msec (20*20 msec) αφού πρέπει να ικανοποιηθεί η συνθήκη [waitcount>=20]. Όμοια, για να στρίψει το αυτοκίνητο, προς τα αριστερά, θα πρέπει να είναι αληθής η συνθήκη μετάβασης [waitcount>=5 && t_dir=2]. Η συνθήκη αυτή γίνεται αληθής όταν η μεταβλητή waitcount πάρει τιμή μεγαλύτερη από 100 msec (5*20 msec) ΚΑΙ η τιμή της μεταβλητής t_dir να είναι 2 (δηλαδή έχει ενεργοποιηθεί ο δεξιός αισθητήρας). Στην κατάσταση turn_left το αυτοκίνητο στρίβει προς τα αριστερά για 160 msec (8*20 msec), αφού πρέπει να ικανοποιείται η συνθήκη [waitcount>=8].

Σύμφωνα λοιπόν με τα παραπάνω, το αυτοκίνητο όταν συναντήσει ένα εμπόδιο και ενεργοποιηθεί ένας από τους δύο αισθητήρες, κάνει λίγο πίσω (στην περίπτωση που εξετάζουμε 100 msec), στη συνέχεια στρίβει προς την αντίθετη κατεύθυνση από τον αισθητήρα που ενεργοποιήθηκε και μετά από έναν προκαθορισμένο χρόνο συνεχίζει να κινείται προς τα εμπρός. Αν για παράδειγμα ενεργοποιηθεί ο αριστερός αισθητήρας, το αυτοκίνητο θα πάει πίσω για 100 msec, θα στρίψει δεξιά για 400 msec και κατόπιν θα συνεχίσει να κινείται προς τα εμπρός μέχρι να ξανασυναντήσει άλλο εμπόδιο και ενεργοποιηθεί πάλι ένας αισθητήρας. Με τον τρόπο αυτόν, το αυτοκίνητο συνεχίζει να κινείται μέχρι να ολοκληρώσει τον στόχο του, που είναι η εκκίνηση από το σημείο εκκίνησης του λαβύρινθου, η επαφή ενός από τους αισθητήρες του στο σημείο ελέγχου (εικόνα 3.1) και η επιστροφή του στο σημείο από το οποίο ξεκίνησε.

3.1.2 Το μοντέλο Linetracker.mdl

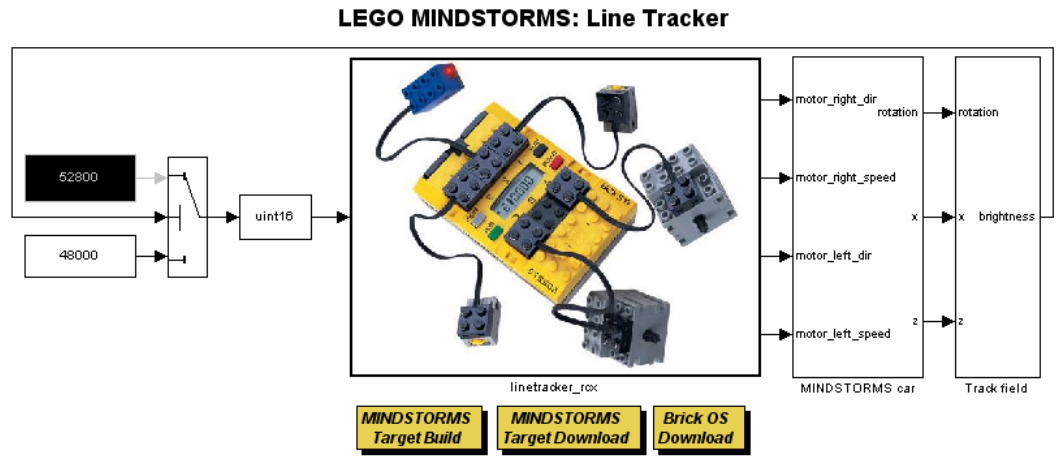
Το μοντέλο Linetracker αποτελεί έναν ¨ανιχνευτή γραμμής¨. Χρησιμοποιώντας μία πίστα, όπως για παράδειγμα αυτή που φαίνεται στην εικόνα 3.10, το αυτοκίνητο την ακολουθεί.



Εικόνα 3.10 Εικόνα από την προσομοίωση του μοντέλου Linetracker στη MATLAB.

Το πλήρες διάγραμμα Simulink για το μοντέλο αυτό δίνεται στην εικόνα 3.11. Μπορούμε να παρατηρήσουμε ότι το διάγραμμα αυτό αποτελείται από 3 τμήματα. Το πρώτο τμήμα, που ονομάζεται **linetracker_rcx** ουσιαστικά αποτελεί την στρατηγική ελέγχου του ρομπότ. Το τμήμα αυτό είναι υπεύθυνο για την κίνηση του ρομπότ πάνω από στην πίστα. Το δεύτερο τμήμα του διαγράμματος, που ονομάζεται **MINDSTORMS car** και αποτελεί το εικονικό ρομπότ που χρησιμοποιείται κατά τη διάρκεια της προσομοίωσης στη MATLAB. Το τρίτο τμήμα, με την ονομασία **Track field**, αποτελεί την εικονική πίστα στην οποία θα τρέξει η προσομοίωση. Σαν πίστα μπορεί να χρησιμοποιηθεί οποιοδήποτε μονόχρωμο αρχείο bitmap (.bmp) το οποίο θα αποτελέσει τον χώρο μέσα στον οποίο θα κινηθεί το εικονικό ρομπότ MINDSTORMS car.

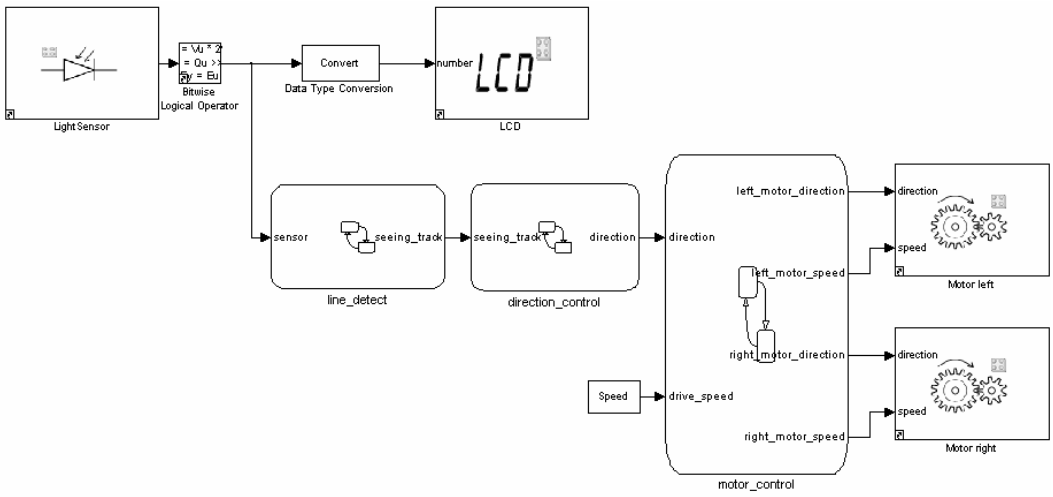
Τα διαγράμματα Simulink και Stateflow που χρησιμοποιούνται για την προσομοίωση αυτού του μοντέλου στον υπολογιστή δίνονται αναλυτικά στο παράρτημα Γ.



Εικόνα 3.11 Το πλήρες διάγραμμα Simulink του μοντέλου Linetracker.mdl.

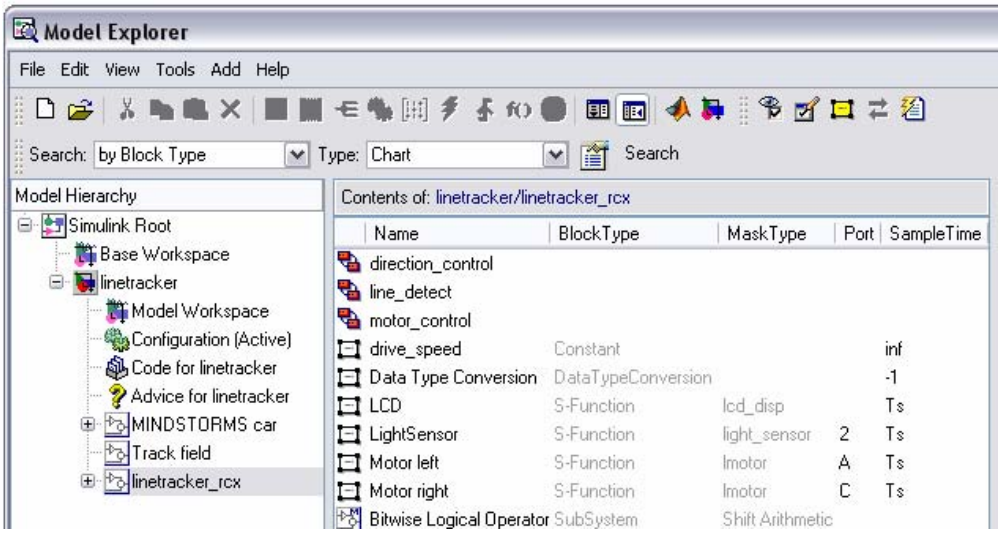
Ακολουθεί μια πιο λεπτομερής περιγραφή των διαγραμμάτων Simulink τα οποία χρησιμοποιούνται για τον έλεγχο της κίνησης του πραγματικού ρομπότ. Στη συνέχεια κωδικοποιούνται από το Real Time Workshop και αποστέλλονται μέσω του πύργου υπερύθρων στο RCX.

Το διάγραμμα Simulink που είναι υπεύθυνο για τον έλεγχο της κίνησης του ρομπότ φαίνεται στην εικόνα 3.12.



Εικόνα 3.12 Διάγραμμα Simulink για την στρατηγική ελέγχου μοντέλου Linetracker.

Στο κεντρικό παράθυρο του Simulink κάνουμε διπλό κλικ στο View→ Model Explorer και εμφανίζεται το παρακάτω παράθυρο (εικόνα 3.13).



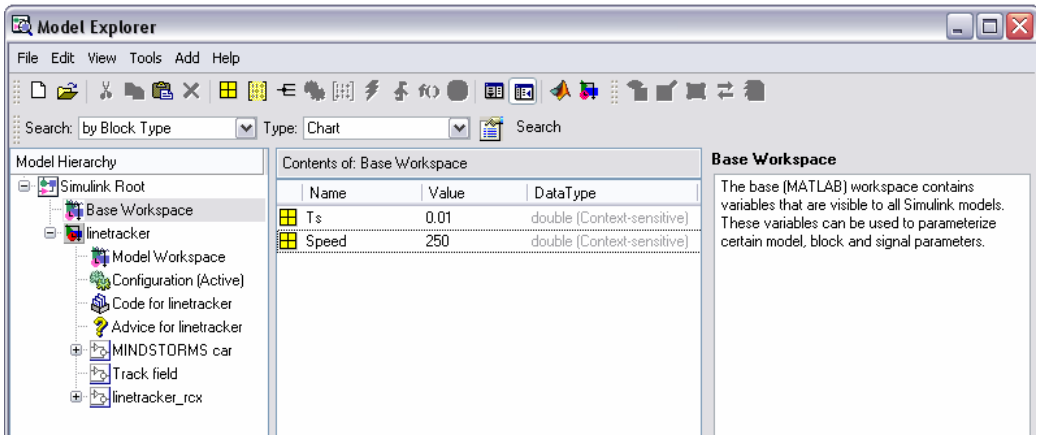
Εικόνα 3.13 Τα στοιχεία του μοντέλου Linetracker.mdl..

Στο παράθυρο αυτό βλέπουμε τα όλα τα στοιχεία από τα οποία αποτελείται το μοντέλο Linetracker.mdl. Αυτά είναι τρία δομικά στοιχεία Stateflow (line_detect, direction_control και motor_control), μία σταθερά (drive_speed), δύο δομικά στοιχεία Simulink (Data Type Conversion και Bitwise Logical Operator) και τέσσερις κατασκευασμένες συναρτήσεις συστήματος (S-Functions), που αποτελούν και αυτές δομικά στοιχεία του Simulink (LCD, LightSensor, Motor left και Motor right). Η λειτουργία όλων αυτών των στοιχείων περιγράφεται αναλυτικά στη συνέχεια.

Κάνοντας κλικ στο Base Workspace, βλέπουμε το βασικό χώρο εργασίας της MATLAB (εικόνα 3.14). Σε αυτόν περιέχονται όλες οι χρησιμοποιούμενες μεταβλητές που εμφανίζονται στο διάγραμμα Simulink. Αυτές οι μεταβλητές χρησιμοποιούνται για την παραμετροποίηση του μοντέλου, των δομικών στοιχείων του και του σήματος του.

Εδώ, χρησιμοποιούμε δύο μεταβλητές, την Ts και την Speed. Η **μεταβλητή Ts**, θα χρησιμοποιηθεί σαν το χρόνο δειγματοληψίας (Sample Time). Ο χρόνος δειγματοληψίας καθορίζει τον ρυθμό που θα συμβαίνει η διαδικασία ανάγνωσης των τιμών που δίνει ο αισθητήρας. Αυτό σημαίνει ότι κάθε χρόνο Ts (εδώ κάθε 10 msec) θα ελέγχεται η τιμή του αισθητήρα φωτός. Η **μεταβλητή Speed,** παίρνει τιμές από 0-

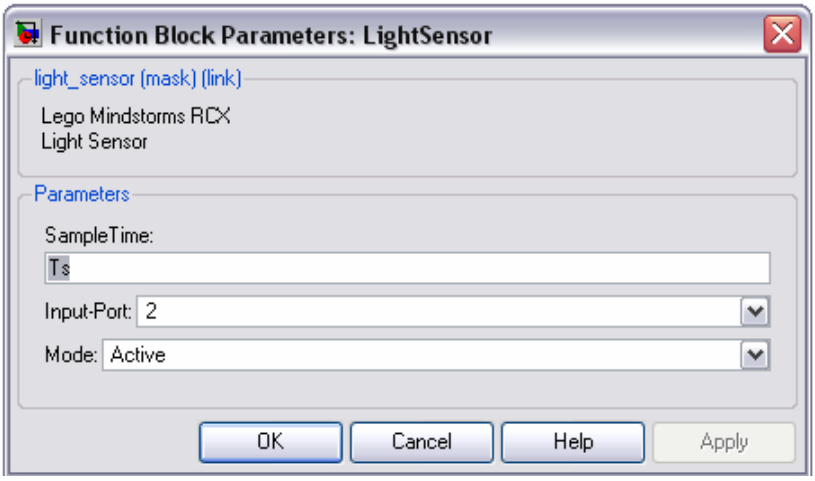
255 και καθορίζει την ταχύτητα περιστροφής των κινητήρων του αυτοκινήτου (drive_speed) στο δομικό στοιχείο motor_control του Simulink (εδώ είναι 250).



Εικόνα 3.14 Ο βασικός χώρος εργασίας (Base Workspace) στον Model Explorer.

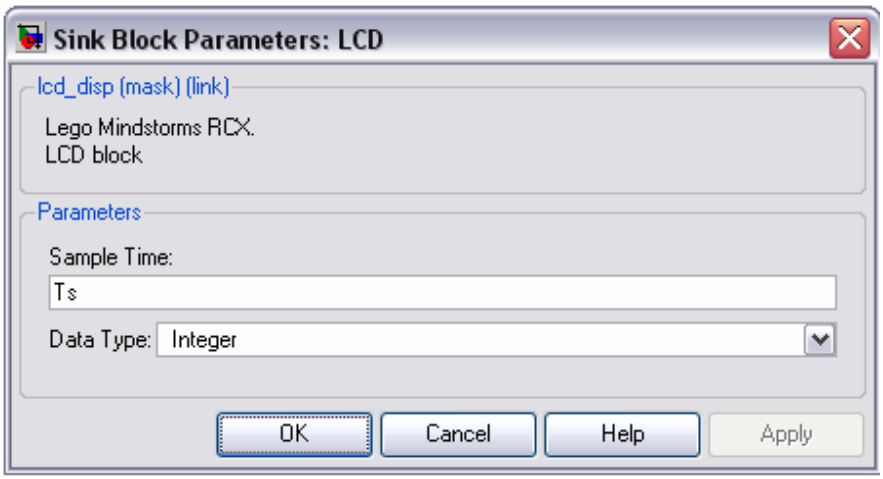
Στη συνέχεια θα δούμε αναλυτικά τα δομικά στοιχεία (blocks) από τα οποία αποτελείται το διάγραμμα Simulink. Αυτά είναι:

1. **Δομικό Στοιχείο Αισθητήρα Φωτός (Input):** Σαν είσοδος στο μοντέλο αυτό χρησιμοποιείται η συνάρτηση συστήματος (S-Function), LightSensor (αισθητήρας φωτός). Έχει σαν παραμέτρους, το χρόνο δειγματοληψίας (Sample Time), την θύρα εισόδου (Input-Port) και την μέθοδο λειτουργίας (Mode), όπως φαίνεται στην εικόνα 3.15. Ο χρόνος δειγματοληψίας καθορίζεται ως Ts και είναι ο χρόνος που το Simulink θα ελέγχει την τιμή που δίνει ο αισθητήρας φωτός (εδώ κάθε 10 msec). Η μέθοδος λειτουργίας περιγράφεται από δύο πιθανές καταστάσεις λειτουργίας του αισθητήρα, την Active και την Passive. Στην κατάσταση Active, το κόκκινο LED που υπάρχει στον αισθητήρα είναι αναμμένο ενώ αντίθετα στην κατάσταση Passive, είναι σβηστό. Η σύνδεση του αισθητήρα γίνεται στην θύρα εισόδου με τον αριθμό 2 στο RCX. Το δομικό στοιχείο αυτό διαβάζει τις τιμές σε κάθε χρόνο δειγματοληψίας (Sample Time) του μοντέλου. Η τιμή εξόδου (Output value) του δομικού στοιχείου αυτού είναι μία μη προσημασμένη ακέραια μεταβλητή 16 bit (uint16). Αυτό σημαίνει ότι οι τιμές που δίνει το δομικό στοιχείο κυμαίνονται από 0 (πολύ φως) έως 65535 (καθόλου φως). Πρακτικά, οι τιμές που δίνει το δομικό αυτό στοιχείο κυμαίνονται από 32000 έως 56000 (όσο μικρότερη η τιμή τόσο περισσότερο φως).



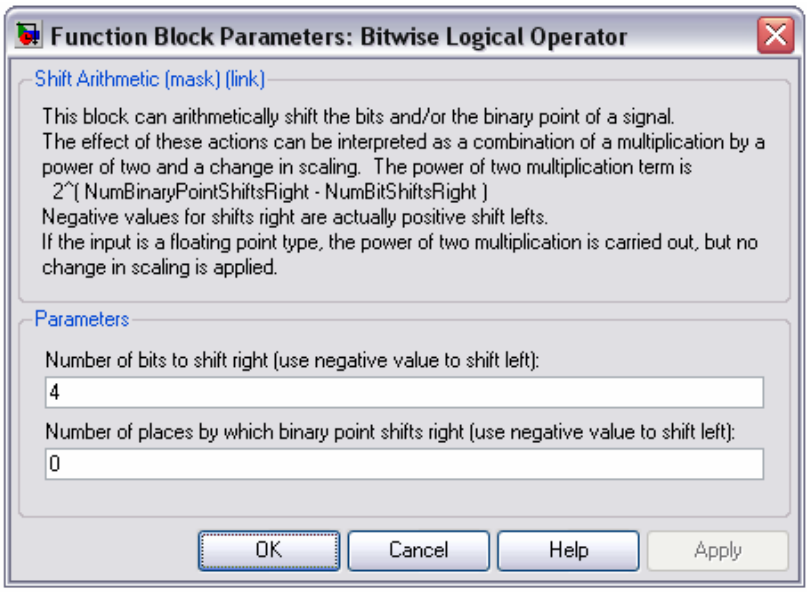
Εικόνα 3.15 Παράμετροι δομικού στοιχείου του αισθητήρα φωτός.

2. **Δομικό Στοιχείο Οθόνης LCD:** Η συνάρτηση συστήματος αυτή του Stateflow, εμφανίζει ακέραιους αριθμούς μεγέθους 16 bit (από 0 έως 65535) στην οθόνη υγρών κρυστάλλων (LCD) του RCX. Πρακτικά, επειδή η οθόνη έχει τη δυνατότητα απεικόνισης τετραψήφιων αριθμών, οι αριθμοί που μπορούν να εμφανιστούν είναι από 0 έως 9999. Οι παράμετροι αυτής της συνάρτησης συστήματος (εικόνα 3.16) είναι ο χρόνος δειγματοληψίας (Sample Time) και ο τύπος δεδομένων που θα εμφανιστούν στην οθόνη (Data Type). Σαν χρόνο δειγματοληψίας θέτουμε τον Ts, δηλαδή κάθε 10 msec, στην οθόνη του RCX θα εμφανίζεται ένας αριθμός και σαν τύπο δεδομένων βάζουμε ακέραιους (integer) αριθμούς 16 bit.



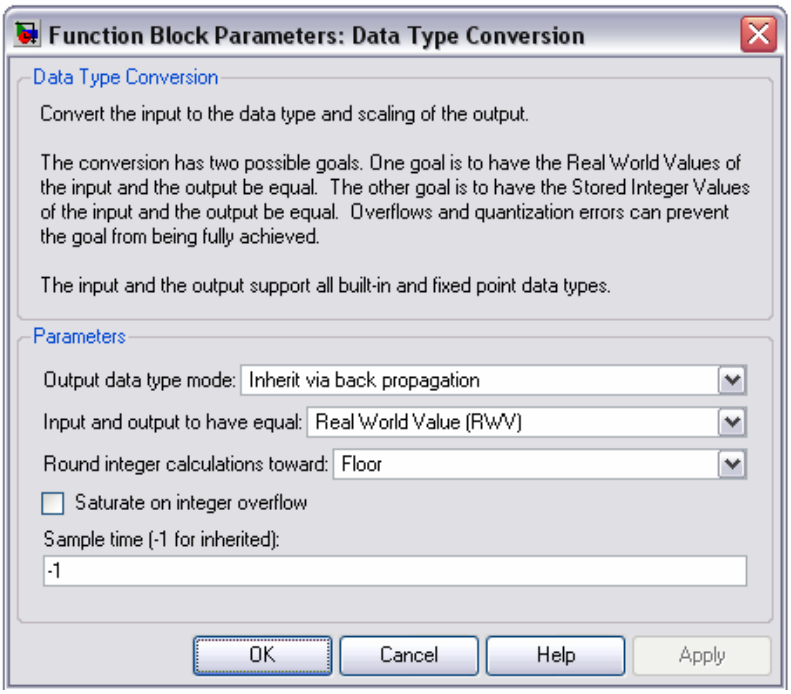
Εικόνα 3.16 Παράμετροι του δομικού στοιχείου της οθόνης LCD.

3. **Δομικό Στοιχείο Λογικού Τελεστή (Bitwise Logical Operator):** Το δομικό αυτό στοιχείο του Stateflow, εκτελεί αριθμητική ολίσθηση (arithmetic shift), δηλαδή μετατοπίζει τα bit ή την υποδιαστολή ενός δυαδικού αριθμού δεξιά ή αριστερά. Οι παράμετροι του δομικού αυτού στοιχείου (εικόνα 3.17) είναι ο αριθμός των δυαδικών ψηφίων που θα μετατοπιστούν προς τα δεξιά, για αριστερή ολίσθηση χρησιμοποιούμε αρνητικό αριθμό, και ο αριθμός των θέσεων που θα μετατοπιστεί η υποδιαστολή του δυαδικού αριθμού. Η ολίσθηση προς τα δεξιά κατά μία θέση αποτελεί ουσιαστικά διαίρεση με το δύο ενώ η ολίσθηση προς τα αριστερά κατά μία θέση αποτελεί πολλαπλασιασμό με το δύο. Εδώ χρησιμοποιούμε τον τελεστή αυτό για να μειώσουμε τις τιμές που μας δίνει το δομικό στοιχείο του αισθητήρα φωτός επειδή οι τιμές αυτές είναι πολύ μεγάλες (0-65535) και δεν μπορούν να εμφανιστούν στην οθόνη LCD του RCX (εμφανίζονται μέχρι 4 ψηφία). Για το λόγο αυτό, θέτουμε σαν παράμετρο του δομικού αυτού στοιχείου την ολίσθηση κατά 4 θέσεις προς τα δεξιά τις τιμής που θα δώσει το δομικό στοιχείο του αισθητήρα φωτός που σημαίνει ότι η τιμή που θα πάρει ο τελεστής θα μειωθεί κατά τον παράγοντα 2^4=16. Για παράδειγμα, εάν το δομικό στοιχείο του αισθητήρα δώσει την τιμή 50000, ο τελεστής θα μετατρέψει τον αριθμό αυτό σε δυαδικό και θα εκτελέσει αριθμητική ολίσθηση, μετατοπίζοντας τα bit του δυαδικού αριθμού κατά 4 θέσεις προς τα δεξιά, έχοντας σαν αποτέλεσμα την εμφάνιση στην οθόνη LCD του RCX της τιμής 3125 (50000/2^4).



Εικόνα 3.17 Παράμετροι του δομικού στοιχείου Bitwise Logical Operator.

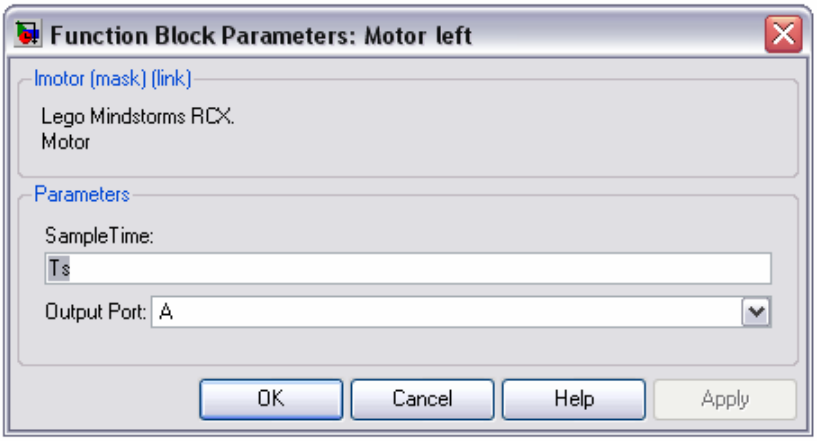
4. **Δομικό Στοιχείο Μετατροπής Τύπου Δεδομένων (Data Type Conversion):** Το δομικό στοιχείο αυτό χρησιμοποιείται για την μετατροπή του τύπου των δεδομένων της εισόδου (εδώ του λογικού τελεστή Bitwise Logical Operator) στον τύπο και την κλίμακα της εξόδου (εδώ της εξόδου LCD του RCX). Η έξοδος από τον λογικό τελεστή είναι μία μη προσημασμένη ακέραια μεταβλητή 16 bit (uint16). Ωστόσο, η είσοδος στην οθόνη LCD του RCX απαιτεί μία ακέραια μεταβλητή 16 bit (int 16). Για το λόγο αυτό μετατρέπουμε το τύπο δεδομένων από uint16 σε int16. Οι παράμετροι που δίνονται στο δομικό αυτό στοιχείο φαίνονται στην εικόνα 3.18.



Εικόνα 3.18 Παράμετροι του δομικού στοιχείου Data Type Conversion.

5. **Δομικά Στοιχεία Κινητήρων (Outputs):** Σαν έξοδοι του μοντέλου χρησιμοποιούνται δύο συναρτήσεις συστήματος, η Motor right και η Motor left. Και οι δύο αυτές συναρτήσεις συστήματος έχουν σαν παραμέτρους τον χρόνο δειγματοληψίας (Sample Time) και τον αριθμό της θύρας εξόδου (Output Port) όπως φαίνεται στην εικόνα 3.19. Ο χρόνος δειγματοληψίας καθορίζεται ως Ts και είναι ο χρόνος που το Simulink θα ελέγχει και θα δίνει τιμές στα στοιχεία αυτά. Ο αριστερός κινητήρας συνδέεται στην θύρα εξόδου με το γράμμα A στο RCX και ο δεξιός στη θύρα με το γράμμα C.

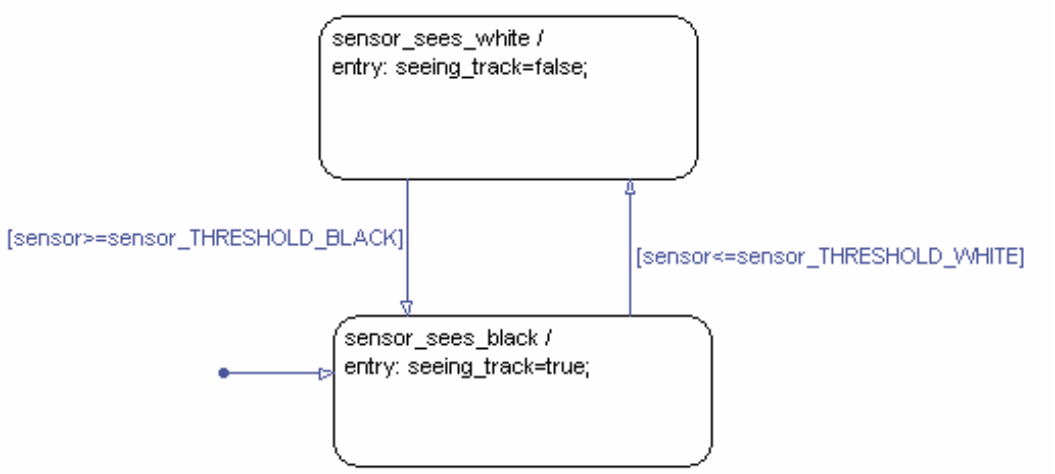
Οι συναρτήσεις συστήματος των κινητήρων περιγράφονται από δύο εισόδους που είναι η ταχύτητα (Speed) και η διεύθυνση περιστροφής τους (Direction). Η ταχύτητα είναι μία μη προσημασμένη (unsigned) σταθερά 8-bit, η οποία παίρνει τιμές από 0-255, και δηλώνεται στη μεταβλητή Speed στον Model Explorer (εικόνα 5.12). Η διεύθυνση περιστροφής είναι και αυτή μία μη προσημασμένη σταθερά 8-bit και παίρνει τιμές 1 (κίνηση προς τα εμπρός), 2 (κίνηση προς τα πίσω), 3 (φρένο) ή 0 (σταματημένη).



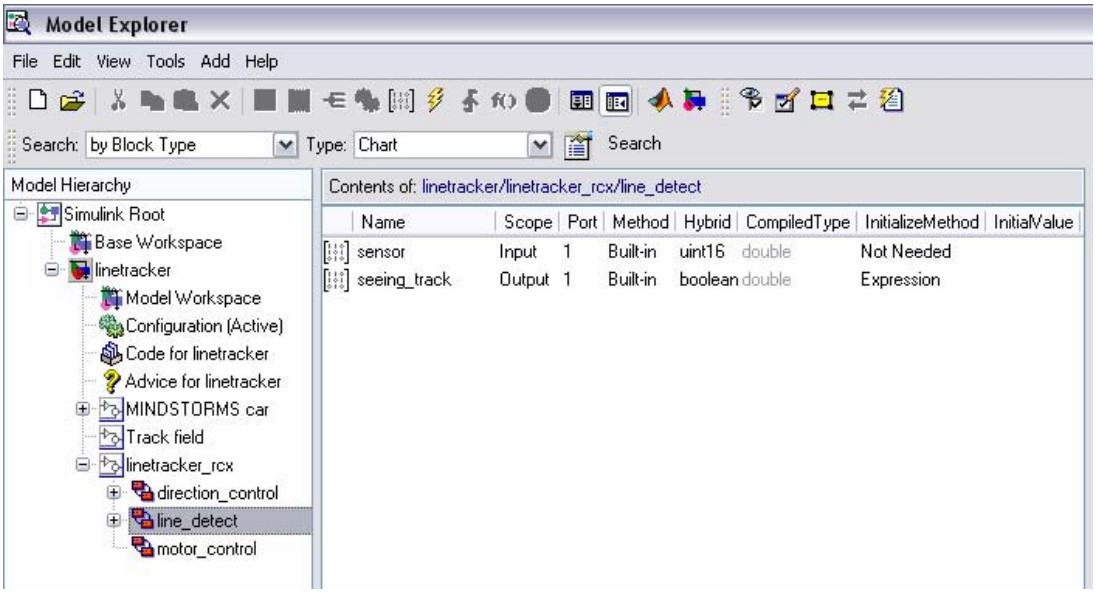
Εικόνα 3.19 Παράμετροι των δομικών στοιχείων των κινητήρων.

6. **Δομικά στοιχεία Stateflow:** Το μοντέλο Linetracker.mdl περιέχει 3 δομικά στοιχεία Stateflow, το line_detect, το direction_control και το motor_control. Τα διαγράμματα αυτά αναπαριστούν την στρατηγική ελέγχου του αυτοκινήτου RCX.

Το δομικό στοιχείο **line_detect** είναι υπεύθυνο για τον εντοπισμό της μαύρης γραμμής. Το διάγραμμα Stateflow του στοιχείου αυτού φαίνεται στην εικόνα 3.20:



Εικόνα 3.20 Διάγραμμα Stateflow για το δομικό στοιχείο line_detect.



Εικόνα 3.21 Ο Model Explorer για το δομικό στοιχείο line_detect.

Ανοίγοντας τον Model Explorer από το Simulink (κάνοντας κλικ στο View →Model Explorer), ανοίγει το παραπάνω παράθυρο (εικόνα 3.21).

Στο παράθυρο αυτό βλέπουμε τις χρησιμοποιούμενες μεταβλητές στο δομικό στοιχείο line_detect, του Stateflow. Αποτελείται από μία είσοδο (input) και μία έξοδο (output).

Η είσοδος του δομικού αυτού στοιχείου, η sensor, είναι μία μη προσημασμένη ακέραια μεταβλητή 16 bit (uint16), η οποία παίρνει τιμές από το δομικό στοιχείο του Simulink, LightSensor.

Η έξοδος, seeing_track, είναι μία Boolean μεταβλητή (παίρνει τιμές true ή false), και δείχνει εάν ο αισθητήρας φωτός του αυτοκινήτου ¨βλέπει¨ ή ¨δεν βλέπει¨ την μαύρη λωρίδα της πίστας.

Το διάγραμμα Stateflow αποτελείται από 2 καταστάσεις:

- **Ο αισθητήρας βλέπει μαύρο (sensor_sees_black):** Μόλις ενεργοποιηθεί η κατάσταση αυτή, εκτελούνται οι ενέργειες εισόδου (entry) δηλαδή η τιμή της μεταβλητής seeing_track γίνεται αληθής (true). Επίσης, η κατάσταση αυτή αποτελεί τον προορισμό μίας προκαθορισμένης μετάβασης (default transition) γεγονός που σημαίνει ότι η κατάσταση αυτή είναι η πρώτη που ενεργοποιείται. Με τον τρόπο αυτό, το αυτοκίνητο καταλαβαίνει ότι βρίσκεται πάνω από την μαύρη λωρίδα της πίστας.
- **Ο αισθητήρας βλέπει λευκό (sensor_sees_white):** Μόλις ενεργοποιηθεί η κατάσταση αυτή, εκτελούνται οι εντολές εισόδου, δηλαδή η τιμή της μεταβλητής seeing_track γίνεται ψευδής (false). Με τον τρόπο αυτό, το αυτοκίνητο καταλαβαίνει ότι βρίσκεται σε σημείο εκτός της πίστας.

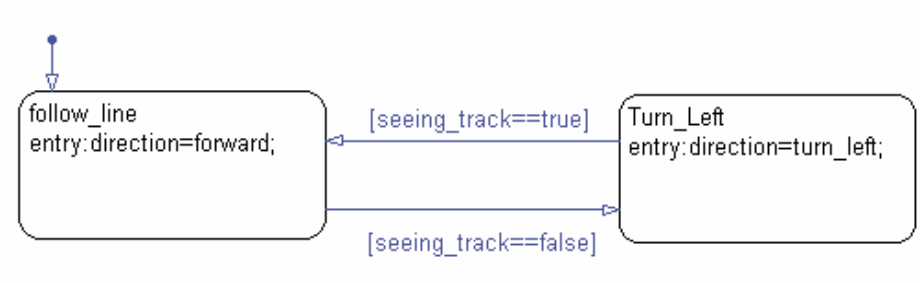
Όλες οι καταστάσεις ανήκουν στο ίδιο επίπεδο ιεραρχίας και έχουν αποκλειστική ανάλυση (exclusive decomposition). Αυτό σημαίνει ότι μόνο μία από τις καταστάσεις αυτές μπορεί να είναι ενεργή κάθε χρονική στιγμή.

Οι συνθήκες (conditions) που ενεργοποιούν τις μεταβάσεις από την μία κατάσταση στην άλλη είναι δύο:

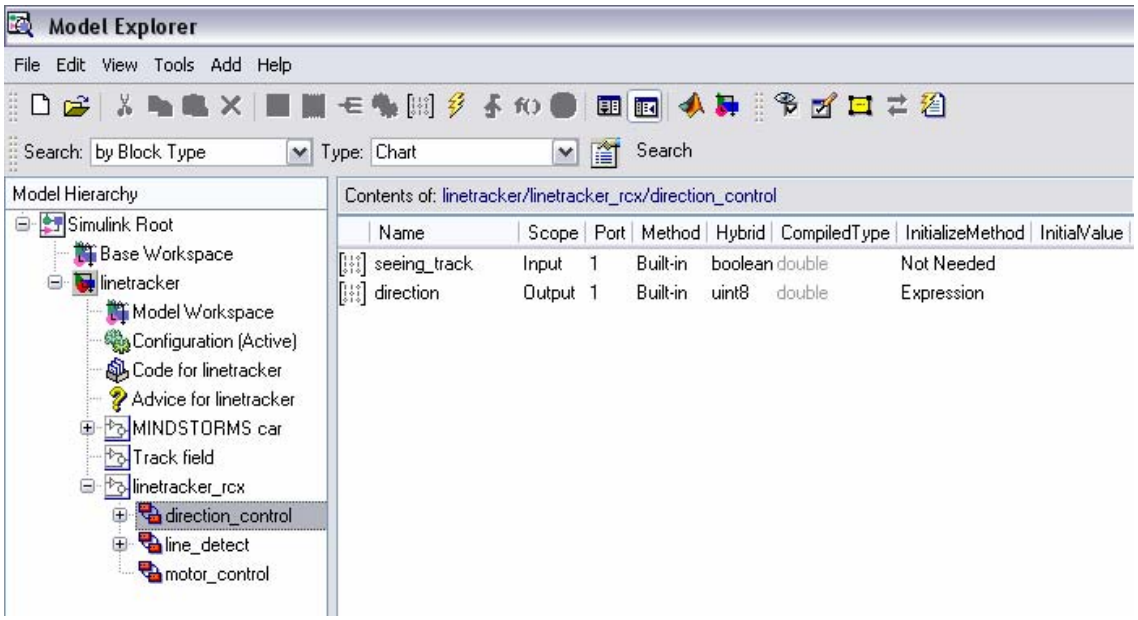
- **[sensor>=sensor_THRESHOLD_BLACK]:** Όταν η έξοδος του δομικού διαγράμματος του αισθητήρα φωτός δώσει τιμή που είναι μεγαλύτερη ή ίση από το όριο για το μαύρο (threshold_black) τότε το αυτοκίνητο θεωρεί ότι βρίσκεται σε μαύρο έδαφος. Η τιμή sensor_THRESHOLD_BLACK, είναι μία μη προσημασμένη σταθερά 16 bit (constant, uint16) η οποία δηλώνεται στον Model Explorer και η τιμή της είναι 3100.
- **[sensor<=sensor_THRESHOLD_WHITE]:** Όταν η έξοδος του δομικού διαγράμματος του αισθητήρα φωτός δώσει τιμή που είναι μικρότερη ή ίση από το όριο για το λευκό (threshold_white) τότε το αυτοκίνητο θεωρεί ότι βρίσκεται σε λευκό έδαφος. Η τιμή sensor_THRESHOLD_WHITE, είναι και

αυτή μία μη προσημασμένη σταθερά 16 bit (constant, uint16) η οποία δηλώνεται στον Model Explorer και η τιμή της είναι 3000.

Το δομικό στοιχείο **direction_control** είναι υπεύθυνο για τον έλεγχο της διεύθυνσης κίνησης του αυτοκινήτου. Το διάγραμμα Stateflow δίνεται στην εικόνα 3.22.



Εικόνα 3.22 Διάγραμμα Stateflow για το δομικό στοιχείο direction_control.



Εικόνα 3.23 Ο Model Explorer για το δομικό στοιχείο direction_control.

Ανοίγοντας τον Model Explorer από το Simulink (κάνοντας κλικ στο View →Model Explorer), ανοίγει το παραπάνω παράθυρο (εικόνα 3.23).

Στο παράθυρο αυτό βλέπουμε τις χρησιμοποιούμενες μεταβλητές στο δομικό στοιχείο detection_control, του Stateflow. Αποτελείται από μία είσοδο (input) και μία έξοδο (output).

Η είσοδος, seeing_track, είναι μία Boolean μεταβλητή (παίρνει τιμές true ή false) η οποία παίρνει τιμές από την έξοδο seeing_track του προηγούμενου δομικού στοιχείου.

Η έξοδος, direction, είναι μία μη προσημασμένη μεταβλητή 8 bit (uint8) η οποία καθορίζει την διεύθυνση μετακίνησης του αυτοκινήτου.

Το διάγραμμα Stateflow αποτελείται από 2 καταστάσεις:

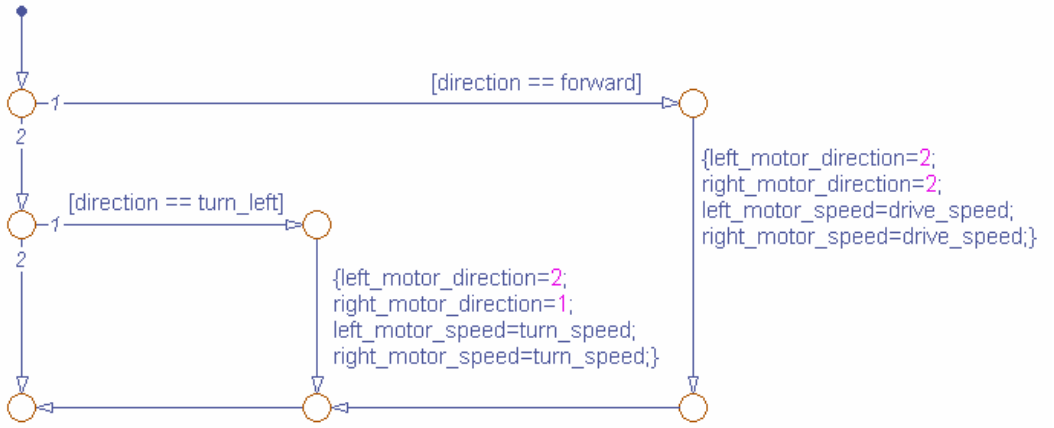
- **Follow_line:** Μόλις ενεργοποιηθεί η κατάσταση αυτή, εκτελούνται οι εντολές εισόδου, δηλαδή η μεταβλητή direction παίρνει την τιμή forward, και με τον τρόπο αυτό, το αυτοκίνητο κινείται προς τα εμπρός. Επιπλέον, η κατάσταση αυτή αποτελεί τον προορισμό μίας προκαθορισμένης μετάβασης και θα εκτελεστή πρώτη.
- **Turn_Left:** Μόλις ενεργοποιηθεί η κατάσταση αυτή, εκτελούνται οι εντολές εισόδου, δηλαδή η μεταβλητή direction παίρνει την τιμή turn_left, και με τον τρόπο αυτό, το αυτοκίνητο στρίβει προς τα αριστερά.

Όλες οι καταστάσεις ανήκουν στο ίδιο επίπεδο ιεραρχίας και έχουν αποκλειστική ανάλυση (exclusive decomposition). Αυτό σημαίνει ότι μόνο μία από τις καταστάσεις αυτές μπορεί να είναι ενεργή κάθε χρονική στιγμή.

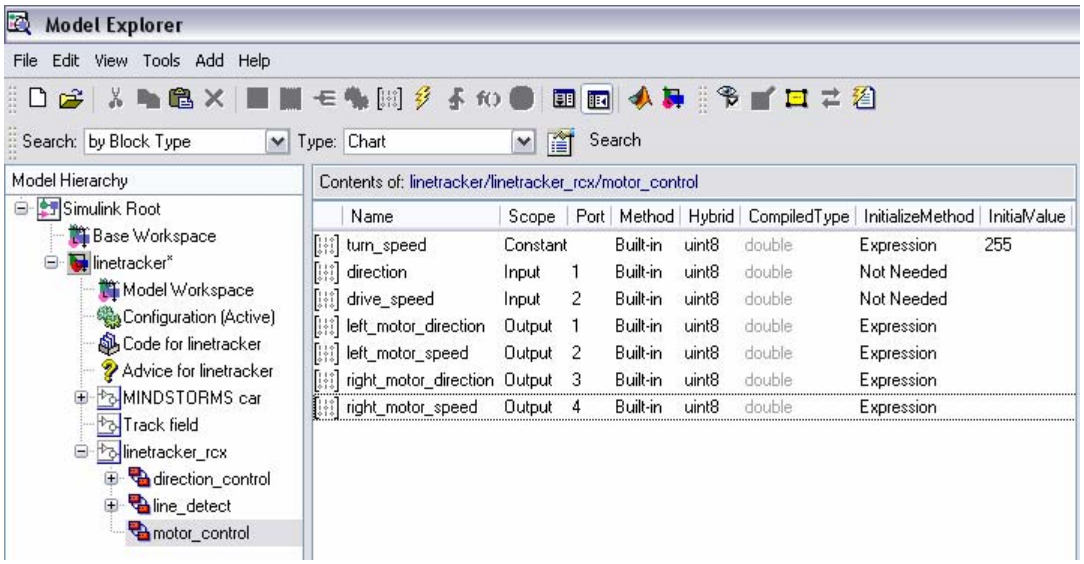
Οι συνθήκες (conditions) που ενεργοποιούν τις μεταβάσεις από την μία κατάσταση στην άλλη είναι δύο:

- **[seeing_track==true]:** Η συνθήκη αυτή είναι αληθής στην περίπτωση που το προηγούμενο δομικό στοιχείο δώσει ότι η μεταβλητή seeing_track είναι αληθής.
- **[seeing_track==false]:** Η συνθήκη αυτή είναι αληθής στην περίπτωση που το προηγούμενο δομικό στοιχείο δώσει ότι η μεταβλητή seeing_track είναι ψευδής.

Τέλος, το δομικό στοιχείο **motor_control** είναι υπεύθυνο για τον έλεγχο της κίνησης του αυτοκινήτου. Το διάγραμμα Stateflow δίνεται στην εικόνα 3.24.



Εικόνα 3.24 Διάγραμμα Stateflow για το δομικό στοιχείο motor_control.



Εικόνα 3.25 Ο Model Explorer για το δομικό στοιχείο motor_control.

Ανοίγοντας τον Model Explorer από το Simulink (κάνοντας κλικ στο View →Model Explorer), ανοίγει το παραπάνω παράθυρο (εικόνα 3.25).

Στο παράθυρο αυτό βλέπουμε τις χρησιμοποιούμενες μεταβλητές στο δομικό στοιχείο motor_control, του Stateflow. Αποτελείται από δύο εισόδους (inputs), τέσσερις εξόδους (outputs) και μία σταθερά (constant).

Οι είσοδοι αυτού του δομικού στοιχείου είναι δύο μη προσημασμένες μεταβλητές 8 bit (uint8), η direction και η drive_speed. Η direction παίρνει τιμές από την έξοδο, direction, του προηγούμενου δομικού διαγράμματος και καθορίζει την διεύθυνση της κίνησης του αυτοκινήτου. Η drive_speed παίρνει τιμές από 0-255, από το δομικό στοιχείο Speed, του διαγράμματος Simulink, και καθορίζει την ταχύτητα του αυτοκινήτου.

Η σταθερά turn_speed παίρνει την τιμή 255 και καθορίζει την ταχύτητα με την οποία θα στρίβει το αυτοκίνητο.

Οι έξοδοι αυτού του δομικού στοιχείου καθορίζουν την διεύθυνση περιστροφής και την ταχύτητα του δεξιού (right_motor_direction και right_motor_speed) και του αριστερού (left_motor_direction και left_motor_speed) κινητήρα και οι τιμές που δίνουν αποτελούν εισόδους για τα δομικά στοιχεία του Simulink, Motor right και Motor left αντίστοιχα.

Το διάγραμμα Stateflow αποτελείται από συνδετικούς συνδέσμους (connective junctions), κάθε ένας από τους οποίους οδηγεί και σε διαφορετική κατάσταση ανάλογα με το ποια συνθήκη είναι αληθής.

Εάν είναι αληθής η συνθήκη **[direction==forward]** τότε εκτελούνται οι ενέργειες συνθήκης:

```
{left_motor_direction=2;
 right_motor_direction = 2;
 left_motor_speed=drive_speed;
 right_motor_speed=drive_speed;}
```

Με τον τρόπο αυτό, οι κινητήρες αρχίζουν να κινούνται προς τα πίσω και έτσι το αυτοκίνητο, όπως και στο προηγούμενο μοντέλο (Explorer.mdl), θα αρχίσει να κινείται προς τα εμπρός. Οι ταχύτητες των δύο κινητήρων έχουν καθοριστεί στο δομικό στοιχείο Speed του διαγράμματος Simulink (250).

Διαφορετικά, εάν είναι αληθής η συνθήκη **[direction==turn_left]**, τότε εκτελούνται οι παρακάτω ενέργειες συνθήκης:

```
{left_motor_direction=2;
 right_motor_direction =1;
 left_motor_speed=turn_speed;
 right_motor_speed=turn_speed;}
```

Με τον τρόπο αυτό, ο αριστερός κινητήρας αρχίζει να κινείται προς τα πίσω ενώ ο δεξιός να κινείται προς τα εμπρός, κάνοντας το αυτοκίνητο να στρίψει προς τα αριστερά. Οι ταχύτητες των δύο κινητήρων έχουν καθοριστεί από τη σταθερά turn_speed (με τιμή 255) του δομικού στοιχείου Stateflow, όπως φαίνεται στο Model Explorer (εικόνα 3.25).

Στο σημείο αυτό θα αναλύσουμε την λογική ελέγχου (control logic) του μοντέλου.

Εκκινώντας το RCX, πατώντας το κόκκινο πλήκτρο (On - Off) και μετά το πλήκτρο εκκίνησης (Run), μετά από χρόνο 10 msec (δηλαδή το χρόνο δειγματοληψίας), ο αισθητήρας φωτός (LightSensor) θα μετρήσει το περιβάλλον φως και στη συνέχεια θα δώσει μία τιμή από 0 έως 65535 (όσο μικρότερη η τιμή τόσο περισσότερο φως). Η τιμή αυτή θα περάσει από τον λογικό τελεστή (Bitwise Logical Operator), θα μετατραπεί σε δυαδικό αριθμό και θα υποστεί αριθμητική ολίσθηση (arithmetic shift) προς τα δεξιά κατά 4 θέσεις. Αυτό σημαίνει ότι η τιμή που θα δώσει ο αισθητήρας θα διαιρεθεί με τον παράγοντα 2^4, ώστε να προκύψει ένας τετραψήφιος ακέραιος αριθμός (που θα κυμαίνεται από 0 έως $65535/2^4=4096$) που να μπορεί να εμφανιστεί στη οθόνη LCD του RCX. Στη συνέχεια αυτός ο αριθμός περνάει από τον μετατροπέα τύπου δεδομένων (Data Type Converter), μετατρέπεται από μη προσημασμένος ακέραιος (unsigned integer) σε ακέραιο (integer) και εμφανίζεται στην οθόνη LCD. Ταυτόχρονα, η τιμή αυτή χρησιμοποιείται και σαν είσοδος στο δομικό στοιχείο του Stateflow, line_detect.

Στο σημείο αυτό, δύο είναι οι πιθανές περιπτώσεις:

1. Η τιμή που θα δώσει ο αισθητήρας φωτός είναι μεγαλύτερη από το όριο sensor_THRESHOLD_BLACK (που ισούται με 3100). Στην περίπτωση αυτή, η συνθήκη [sensor>=sensor_THRESHOLD_BLACK] είναι αληθής και επομένως παραμένουμε στην κατάσταση sensor_sees_black. Εκτελείται η ενέργεια εισόδου (entry: seeing_track=true) και η Boolean μεταβλητή seeing_track γίνεται αληθής. Στην κατάσταση αυτή το αυτοκίνητο καταλαβαίνει ότι βρίσκεται σε μαύρο έδαφος. Η μεταβλητή seeing_track στη συνέχεια αποτελεί είσοδο στο επόμενο δομικό στοιχείο του Stateflow, direction_control. Στο στοιχείο αυτό, αφού η μεταβλητή seeing_track είναι αληθής και επίσης η κατάσταση follow_line αποτελεί προορισμό μιας προκαθορισμένης μετάβασης, ενεργοποιείται η κατάσταση follow_line. Εκτελείται η ενέργεια εισόδου (entry: direction=forward) και η μεταβλητή direction, παίρνει την τιμή forward (κίνηση προς τα εμπρός). Τέλος, η μεταβλητή αυτή, χρησιμοποιείται σαν είσοδος στο τελευταίο δομικό στοιχείο του Stateflow, motor_control. Αφού η συνθήκη [direction==forward] είναι αληθής, εκτελούνται οι ενέργειες συνθήκης:

   {left_motor_direction=2
   right_motor_direction=2
   left_motor_speed=drive_speed
   right_motor_speed=drive_speed}

   Οι ενέργειες αυτές καθορίζουν την διεύθυνση περιστροφής και των δύο κινητήρων (προς τα πίσω) και έτσι το αυτοκίνητο κινείται προς τα εμπρός (όπως στην περίπτωση του μοντέλου Explorer.mdl) με ταχύτητα περιστροφής την drive_speed που καθορίζεται από την σταθερά Speed (με τιμή 250) στον Model Explorer.

2. Η τιμή που θα δώσει ο αισθητήρας φωτός είναι μικρότερη από το όριο sensor_THRESHOLD_WHITE (που ισούται με 3000). Στην περίπτωση αυτή, η συνθήκη [sensor<=sensor_THRESHOLD_ WHITE] είναι αληθής με αποτέλεσμα να γίνει μετάβαση προς την κατάσταση sensor_sees_white. Εκτελείται η ενέργεια εισόδου (entry: seeing_track=false) και η Boolean μεταβλητή seeing_track γίνεται ψευδής. Στην κατάσταση αυτή το αυτοκίνητο καταλαβαίνει ότι βρίσκεται σε λευκό έδαφος. Η μεταβλητή seeing_track στη συνέχεια αποτελεί είσοδο στο επόμενο δομικό στοιχείο του Stateflow, direction_control. Στο στοιχείο αυτό, αφού η μεταβλητή seeing_track είναι ψευδής, ενεργοποιείται η κατάσταση Turn_left αφού ισχύει η συνθήκη [seeing_track==false]. Εκτελείται η ενέργεια εισόδου (entry: direction=turn_left) και η μεταβλητή direction, παίρνει την τιμή turn_left (στροφή προς τα αριστερά). Τέλος, η μεταβλητή αυτή, χρησιμοποιείται σαν είσοδος στο τελευταίο δομικό στοιχείο του Stateflow, motor_control. Αφού η συνθήκη [direction== turn_left] είναι αληθής, εκτελούνται οι ενέργειες συνθήκης:

   {left_motor_direction=2
   right_motor_direction=1
   left_motor_speed=drive_speed
   right_motor_speed=drive_speed}

   Οι ενέργειες αυτές καθορίζουν την διεύθυνση περιστροφής του αριστερού κινητήρα προς τα εμπρός και του δεξιού προς τα πίσω και έτσι το αυτοκίνητο στρίβει προς τα αριστερά (όπως στην περίπτωση του μοντέλου Explorer.mdl) με ταχύτητα περιστροφής την drive_speed (με τιμή 250) που καθορίζεται από την σταθερά speed στον Model Explorer.

Οι ενέργειες αυτές επαναλαμβάνονται κάθε 10 msec, δηλαδή σε κάθε χρόνο δειγματοληψίας, και το αποτέλεσμα τους είναι ότι όταν το αυτοκίνητο βρεθεί πάνω από μαύρο έδαφος, κινείται προς τα εμπρός, ακολουθώντας το, ενώ όταν βρεθεί πάνω από λευκό, στρίβει προς τα αριστερά, μέχρι να ξαναβρεθεί πάνω από μαύρο έδαφος. Με τον τρόπο αυτό, το αυτοκίνητο ακολουθεί την μαύρη λωρίδα της πίστας.

3.2 Κατασκευή και αποστολή μοντέλων στο RCX

Μετά την περιγραφή και την κατανόηση των μοντέλων Simulink, είναι αναγκαία η περιγραφή της διαδικασίας κατασκευής και αποστολής (download) τους, από τον υπολογιστή (με τη χρήση του πύργου υπερύθρων) προς τον στόχο (το RCX).

3.2.1 Η διαδικασία κατασκευής (Build Procedure)

Η διαδικασία κατασκευής του Real-Time Workshop ξεκινάει με την δημιουργία του αρχείου *model*.rtw (για παράδειγμα το *explorer*.rtw), το οποίο είναι μία ενδιάμεση αναπαράσταση ενός διαγράμματος Simulink και περιέχει πληροφορίες όπως τις τιμές διάφορων παραμέτρων, τους χρόνους δειγματοληψίας και την σειρά εκτέλεσης των δομικών στοιχείων του μοντέλου. Οι πληροφορίες αυτές αποθηκεύονται σε μορφή ASCII.

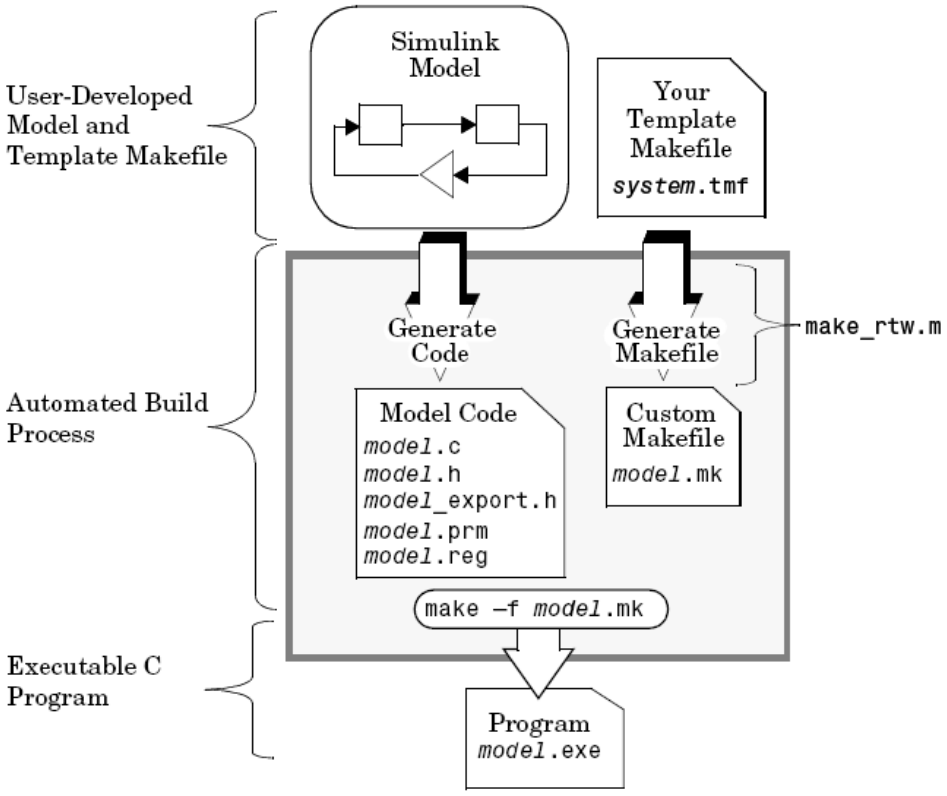
Μετά την δημιουργία του αρχείου *model*.rtw, η διαδικασία κατασκευής επικαλείται τον Target Language Compiler για την μετατροπή του σε κώδικα σχετικό με τον στόχο (target-specific code). Ο Target Language Compiler ξεκινά διαβάζοντας το αρχείο *model*.rtw. Στη συνέχεια, μεταγλωττίζει και εκτελεί τις εντολές στα αρχεία στόχου. Τα αρχεία στόχου, που αναφέρονται ως .tlc ή TLC files, καθορίζουν πως θα μετατραπεί το αρχείο *model*.rtw σε κώδικα σχετικό με τον στόχο. Ο Target Language Compiler ξεκινάει την εκτέλεση με το αρχείο .tlc. και μετατρέπει τις πληροφορίες των δομικών στοιχείων στο αρχείο *model*.rtw σε κώδικα σχετικό με τον στόχο. Η έξοδος του Target Language Compiler είναι ο πηγαίος κώδικας του δομικού διαγράμματος Simulink.

Το επόμενο βήμα της διαδικασίας κατασκευής είναι η δημιουργία ενός system makefile (εδώ του *rcx*.mk) από μία template makefile, την *rcx*.tmf. Το *rcx*.tmf είναι ένα template makefile για το περιβάλλον του χρησιμοποιούμενου στόχου. Σε αυτό γίνεται ο καθορισμός του μεταγλωττιστή (compiler), των επιλογών του μεταγλωττιστή (compiler options) και των πρόσθετων πληροφοριών του προορισμού (στόχου) του παραγόμενου εκτελέσιμου προγράμματος.

Στη συνέχεια, δημιουργείται το αρχείο *model*.mk. Μετά τη δημιουργία του, επικαλείται η εντολή make (make command) για την δημιουργία του εκτελέσιμου

προγράμματος. Η make command μπορεί προαιρετικά να αποστείλει το εκτελέσιμο πρόγραμμα στη συσκευή του στόχου (εδώ στο RCX).

Η παραπάνω διαδικασία φαίνεται γραφικά στην εικόνα που ακολουθεί:

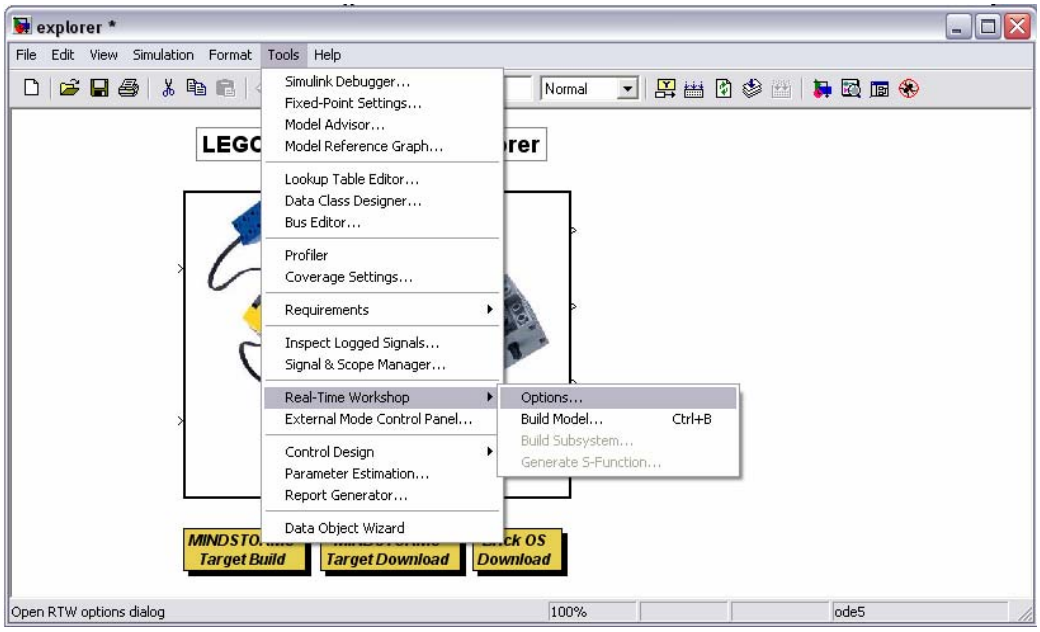


Εικόνα 3.26 Η διαδικασία κατασκευής**.**

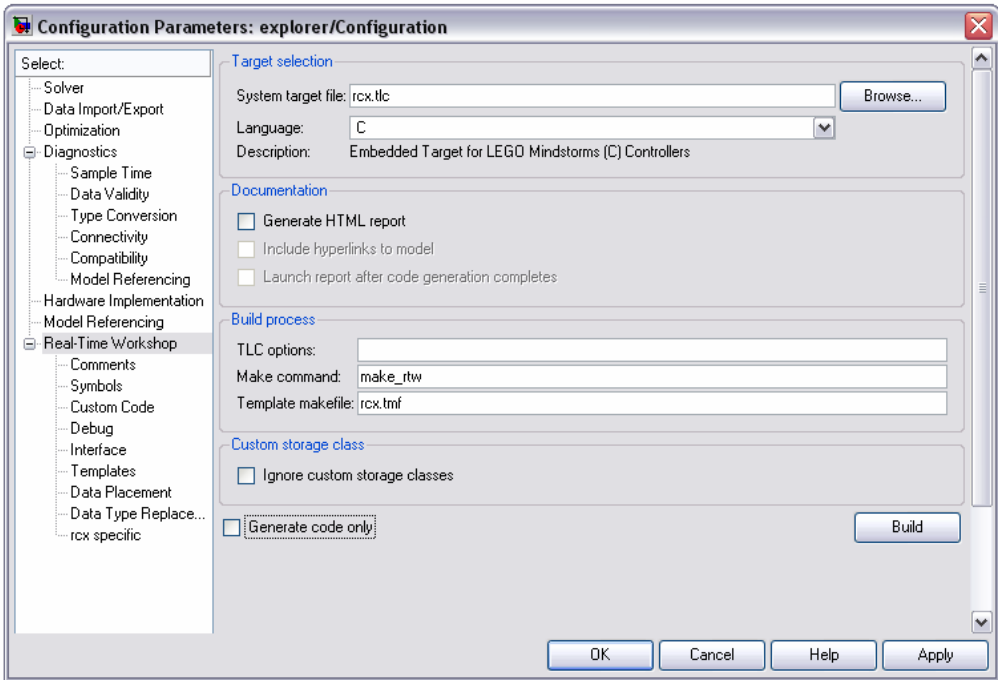
Στη συνέχεια θα δούμε τις ρυθμίσεις που πρέπει να γίνουν στο λογισμικό MATLAB ώστε τα μοντέλα explorer.mdl και linetracker.mdl να κατασκευαστούν από το Real-Time Workshop και να αποσταλούν στο RCX όπου θα τρέξουν σε πραγματικό χρόνο.

Στο κεντρικό παράθυρο της MATLAB, θέτουμε σαν κατάλογο εργασίας (Current directory) το κατάλογο που περιέχει τα αρχεία των μοντέλων explorer.mdl και linetracker.mdl. Στη συνέχεια, στη γραμμή εντολών της MATLAB πληκτρολογούμε την εντολή explorer ή linetracker, ανάλογα με το ποιο μοντέλο επιθυμούμε να ανοίξουμε και εμφανίζεται το κεντρικό παράθυρο του μοντέλου. Κάνοντας κλικ στο Tools→ Real-Time Workshop→ Options… (εικόνα 3.27) εμφανίζεται το παράθυρο

ρυθμίσεων των παραμέτρων (εικόνα 3.28) που πρέπει να κάνουμε στο Real-Time Workshop ώστε να γίνει η κατασκευή (build) και η αποστολή (download) του μοντέλου στο RCX.



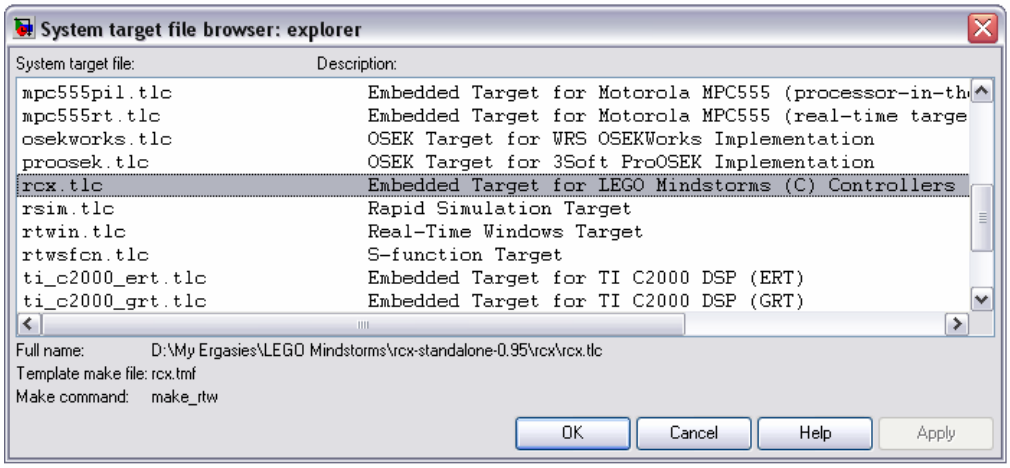
Εικόνα 3.27 Οι επιλογές του Real-Time Workshop.



Εικόνα 3.28 Παράθυρο ρυθμίσεων των παραμέτρων του Real-Time Workshop.

Στο παράθυρο ρυθμίσεων των παραμέτρων (εικόνα 3.28) εμφανίζονται οι παρακάτω παράμετροι:

- ✓ **Επιλογή Στόχου (Target Selection):** Στο τμήμα αυτό γίνεται η επιλογή του στόχου, στον οποίο θα αποσταλούν τα μοντέλα και θα τρέξουν σε πραγματικό χρόνο. Μετά την εγκατάσταση του στόχου rcx-0.95-standalone, το αρχείο στόχου rcx.tlc προστίθεται στο κατάλογο System target file browser (εικόνα 3.29).



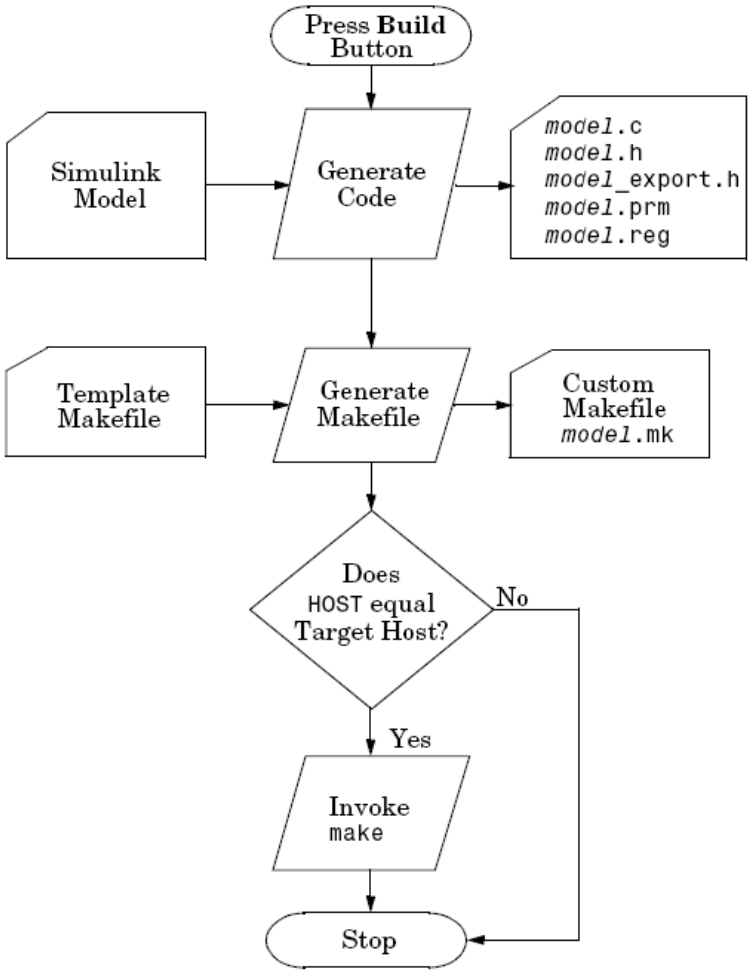
Εικόνα 3.29 Επιλογή του στόχου rcx.tlc.

- ✓ **Τεκμηρίωση (Documentation):** Διαλέγοντας την επιλογή **Generate HTML report**, το Real-Time Workshop παράγει μια αναφορά παραγωγής κώδικα σε μορφή HTML και την ανοίγει αυτόματα σε ένα ξεχωριστό παράθυρο. Με την επιλογή **Launch report after code generation completes** έχουμε την δυνατότητα εμφάνισης της αναφοράς αμέσως μετά την παραγωγή του κώδικα.
- ✓ **Παραγωγή μόνο του κώδικα (Generate Code Only):** Η επιλογή αυτή έχει ως αποτέλεσμα την παραγωγή κώδικα χωρίς την υποστήριξη της Make command. Ο κώδικας δεν μεταγλωττίζεται και δεν κατασκευάζεται εκτελέσιμο. Όταν η επιλογή **Generate code only** δεν είναι ενεργοποιημένη, χρησιμοποιείται το **template makefile**. Το Real-Time Workshop χρησιμοποιεί τα template makefiles για την κατασκευή ενός εκτελέσιμου από τον παραγόμενο κώδικα. Από σύμβαση, ένα template makefile έχει επέκταση .tmf και όνομα αντίστοιχο με τον στόχο.

Κάνοντας κλικ στο πλήκτρο **Build,** το Real-Time Workshop επικαλείται την εντολή make (make command) η οποία:

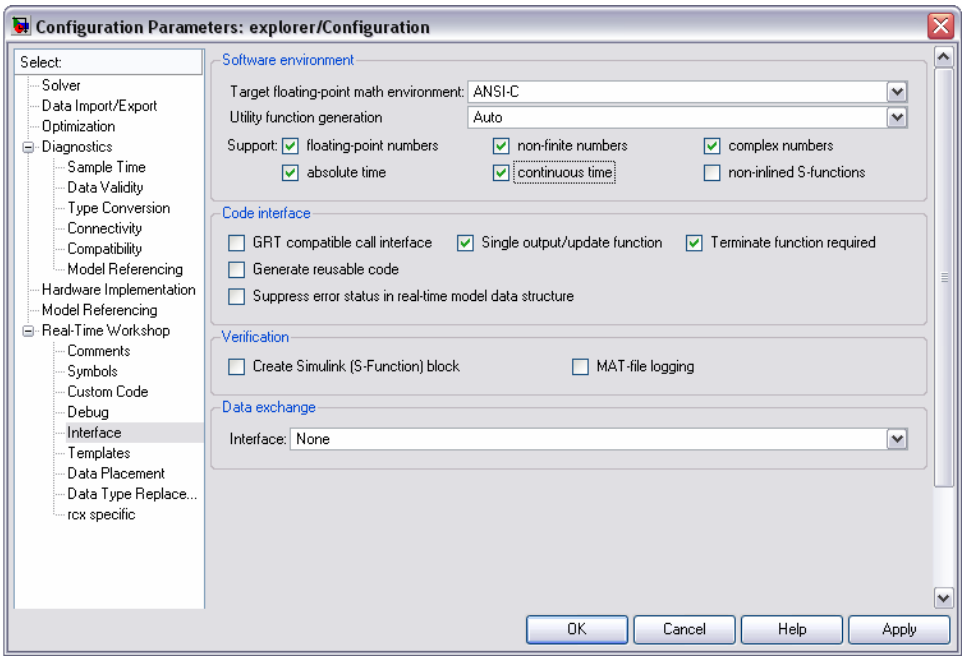
- Μεταγλωττίζει το δομικό διάγραμμα για την παραγωγή του αρχείου *model*.rtw (για παράδειγμα το *explorer*.rtw)
- Επικαλείται τον Target Language Compiler ο οποίος με τη σειρά του μεταγλωττίζει το πρόγραμμα TLC και λειτουργεί πάνω στο *explorer*.rtw για να δημιουργήσει τον παραγόμενο κώδικα
- Δημιουργεί ένα makefile με το όνομα *explorer*.mk από το template makefile (*rcx*.tmf)

Τα βήματα που εκτελούνται παρουσιάζονται με τη μορφή ενός διαγράμματος ροής (εικόνα 3.30):



Εικόνα 3.30 Το διάγραμμα ροής της αυτόματης διαδικασίας κατασκευής

Στη συνέχεια, κάνοντας κλικ στην επιλογή Interface, μεταφερόμαστε στις επιλογές αλληλεπίδρασης (εικόνα 3.31). Στις επιλογές αυτές το μόνο που θα προσθέσουμε είναι η υποστήριξη συνεχούς χρόνου (continuous time) για την αποφυγή μηνυμάτων σφάλματος, επειδή το μοντέλο χρησιμοποιεί μεταβλητές συνεχούς χρόνου.



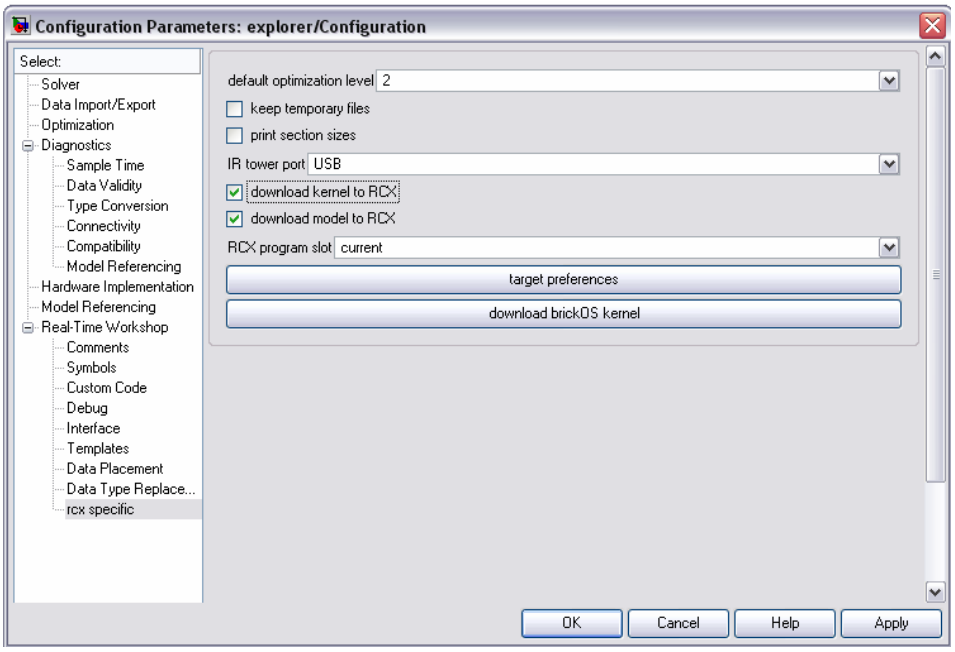
Εικόνα 3.31 Επιλογές αλληλεπίδρασης (Interface).

Τέλος, κάνοντας κλικ στην επιλογή rcx specific (εικόνα 3.32) μεταφερόμαστε στις επιλογές που είναι σχετικές με το στόχο rcx.tlc. Οι ρυθμίσεις που πρέπει να γίνουν εμφανίζονται παρακάτω:

- ✓ **IR tower port:** Με την επιλογή αυτή διαλέγουμε την θύρα επικοινωνίας του υπολογιστή με τον πύργο υπερύθρων. Επιλέγουμε την θύρα USB.
- ✓ **download kernel to RCX:** Η επιλογή αυτή επιτρέπει την αποστολή του λειτουργικού συστήματος BrickOS και την αποθήκευση του στη μνήμη του RCX κατά τη διάρκεια της διαδικασίας κατασκευής (built process). Η επιλογή αυτή δεν είναι αναγκαία. Για να στείλουμε το λειτουργικό σύστημα, χρησιμοποιούμε την παρακάτω επιλογή **download brickOS kernel.** Μετά την

ολοκλήρωση της αποστολής του BrickOS, από-επιλέγουμε την εντολή **download kernel to RCX** για την αποφυγή μηνυμάτων σφάλματος[2].

- ✓ **download model to RCX:** Η επιλογή αυτή επιτρέπει την αποστολή των μοντέλων στο RCX.
- ✓ **RCX program slot:** Με την επιλογή αυτή διαλέγουμε σε ποια θέση (slot 1-5) να αποθηκευτεί το μοντέλο στο RCX. Διαλέγουμε την επιλογή current που σημαίνει ότι το μοντέλο θα αποθηκευτεί στην θέση που εμφανίζεται εκείνη τη στιγμή στην οθόνη LCD του RCX.



Εικόνα 3.32 Επιλογές σχετικές με το RCX (rcx specific).

Μετά από την ρύθμιση των παραπάνω παραμέτρων, είναι δυνατή η κατασκευή και η αποστολή των μοντέλων στοRCX. Κάνοντας κλικ στο Tools→ Real-Time Workshop→ Build Model… (εικόνα 5.25) ή πατώντας ταυτόχρονα τα πλήκτρα Ctrl+B, το μοντέλο θα κατασκευαστεί. Ακολουθώντας την παραπάνω διαδικασία κατασκευής (build procedure), το Real-Time Workshop θα παραγάγει τον κώδικα του

[2]Τα προγράμματα που θα κατασκευαστούν με το στόχο RCX – Embedded target for Lego Mindstorms, απαιτούν την εγκατάσταση του λειτουργικού συστήματος BrickOS (LegOS), αντικαθιστώντας το κανονικό λειτουργικό σύστημα του RCX (RCX firmware). Η εγκατάσταση του λειτουργικού συστήματος BrickOS (LegOS) γίνεται μόνο την πρώτη φόρα που γίνεται χρήση του στόχου. Μετά, το BrickOS παραμένει στην μνήμη του RCX, που σημαίνει ότι εάν η επιλογή αυτή είναι ενεργή και υπάρχει ήδη εγκατεστημένο το BrickOS, η MATLAB θα εμφανίσει μήνυμα σφάλματος.

εκάστοτε μοντέλου και θα τον μεταγλωττίσει χρησιμοποιώντας τον δια-μεταγλωττιστή GCC (GCC cross compiler) για τον μικροελεγκτή της Hitachi, h8300-hms-gcc. Όταν ολοκληρωθεί η διαδικασία κατασκευής θα εμφανιστεί το παρακάτω μήνυμα στη γραμμή εντολών της MATLAB:

Successful completion of Real-Time Workshop built procedure for model: Explorer

Ταυτόχρονα, το μοντέλο αποστέλλεται στο RCX, μέσω του πύργου υπερύθρων και μπορεί να εκτελεστεί.

ΚΕΦΑΛΑΙΟ 4: ΣΥΜΠΕΡΑΣΜΑΤΑ

Η προσέγγιση της διδασκαλίας των Συστημάτων Αυτομάτου Ελέγχου με τα Lego Mindstorms, σαν εργαστηριακή διάταξη, μπορεί να συμβάλει στην εξάλειψη των δυσκολιών που συνεπάγεται η παραδοσιακή μέθοδος και να δημιουργήσει κατάλληλες συνθήκες μάθησης ώστε να γίνει αποτελεσματικότερη η διδασκαλία στους φοιτητές.

Η παρατήρηση ότι πολλοί φοιτητές έχουν μικρή εμπειρία στον προγραμματισμό κάνει τη χρήση μιας τέτοιας πλατφόρμας ικανή για την αντιμετώπιση των δυσκολιών που συναντούν και που συνήθως είναι ο αδύναμος λογικός συλλογισμός και η έλλειψη προσοχής στις λεπτομέρειες. Η άμεση εμπειρία που αποκτούν από τα ρομπότ αυτά, ο πειραματισμός και η ενεργή συμμετοχή τους, ευνοούν την ανάπτυξη προβληματισμού και την καλλιέργεια χαρακτηριστικών όπως την κριτική συμπεριφορά, την πρωτοβουλία και την δημιουργική σκέψη για την επίλυση των διαφόρων εργαστηριακών ασκήσεων. Τα εξαρτήματα Lego που χρησιμοποιούν, μετατρέπονται σε εργαλεία για την μοντελοποίηση μίας ευρείας ποικιλίας φυσικών και τεχνητών συστημάτων.

Η χρησιμότητα των Lego Mindstorms σαν εκπαιδευτικά εργαλεία έγκειται σε διάφορους παράγοντες. Ένας από αυτούς είναι η μεγάλη ποικιλία ρομπότ που μπορούν να κατασκευαστούν από αυτήν την πλατφόρμα. Η διαμορφώσιμη φύση των Lego Mindstorms, με την χρήση πολλών εξαρτημάτων, καθιστά εφικτή την κατασκευή μιας τεράστιας ποικιλίας ρομπότ. Το γεγονός αυτό, δίνει ένα σημαντικό πλεονέκτημα στα Lego Mindstorms έναντι στις άλλες πλατφόρμες, οι οποίες έχουν γενικά αμετάβλητα σώματα.

Ένας δεύτερος παράγοντας για τη χρησιμότητα της πλατφόρμας αυτής, είναι η μεγάλη ποικιλία διαφορετικών προγραμματιστικών περιβαλλόντων που είναι άμεσα διαθέσιμα στον χρήστη. Υπάρχουν πολλές επιλογές για την ανάπτυξη προγραμμάτων στο RCX, από γραφικά περιβάλλοντα για αρχάριους (ROBOLAB, RIS code) έως προχωρημένα περιβάλλοντα που βασίζονται σε γλώσσες προγραμματισμού όπως η C, η C++, η Forth, η Java και η Visual Basic (BrickOS, NQC, pbFORTH, lejOS είναι μερικά μόνο παραδείγματα). Αρχάριοι, οι οποίοι δεν έχουν ασχοληθεί με τον προγραμματισμό, μπορούν να ξεκινήσουν από τα, λιγότερο πολύπλοκα και

κατανοητά από αυτούς, γραφικά περιβάλλοντα, που προσφέρονται από τη Lego (ROBOLAB). Τα περιβάλλοντα αυτά, παρόλο που επιτρέπουν στους ενδιαφερόμενους να κατασκευάσουν προγράμματα με απλές λειτουργίες του ποντικιού, έχουν και σημαντικούς περιορισμούς στην χρήση του RCX. Οι περιορισμοί αυτοί μπορούν να ξεπεραστούν με τη χρήση ενός προχωρημένου περιβάλλοντος από αυτά που αναφέρθηκαν παραπάνω.

Στην παρούσα εργασία χρησιμοποιήσαμε το λειτουργικό σύστημα BrickOS το οποίο χρησιμοποιείται για τον προγραμματισμό του RCX σε γλώσσα C++. Σε συνεργασία με το γραφικό περιβάλλον του Simulink, μπορέσαμε να προγραμματίσουμε το ρομπότ με τη χρήση διαφόρων γραφικών στοιχείων (blocks). Ακολουθώντας απλά βήματα, είναι δυνατή η κατασκευή και ο προγραμματισμός των ρομπότ. Η μέθοδος αυτή είναι κατάλληλη, σε αρχικό στάδιο τουλάχιστον, για την κατανόηση από τους φοιτητές των βασικών εννοιών του προγραμματισμού και την εισαγωγή τους στο πνεύμα των Συστημάτων Αυτομάτου Ελέγχου.

Επιπλέον, τα Lego Mindstorms αποτελούν μία πολύ οικονομική πλατφόρμα, σε σχέση με τις διαθέσιμες επιλογές, η οποία παρέχει μεγάλη αποτελεσματικότητα και ευελιξία. Για το λόγο αυτό, ένα εργαστήριο μπορεί να εφοδιαστεί με αρκετά πακέτα, έτσι ώστε οι φοιτητές να μπορούν να δουλεύουν σε ομάδες και να αποκτούν πρακτικές εμπειρίες και στην κατασκευή ρομπότ αλλά και στο προγραμματισμό τους. Με τον τρόπο αυτό θα διδάσκονται στην πράξη, και όχι μόνο θεωρητικά, τις έννοιες των Συστημάτων Αυτομάτου Ελέγχου και θα έχουν την ευκαιρία να αλληλεπιδρούν με τα μοντέλα που κατασκευάζουν σε πραγματικό χρόνο.

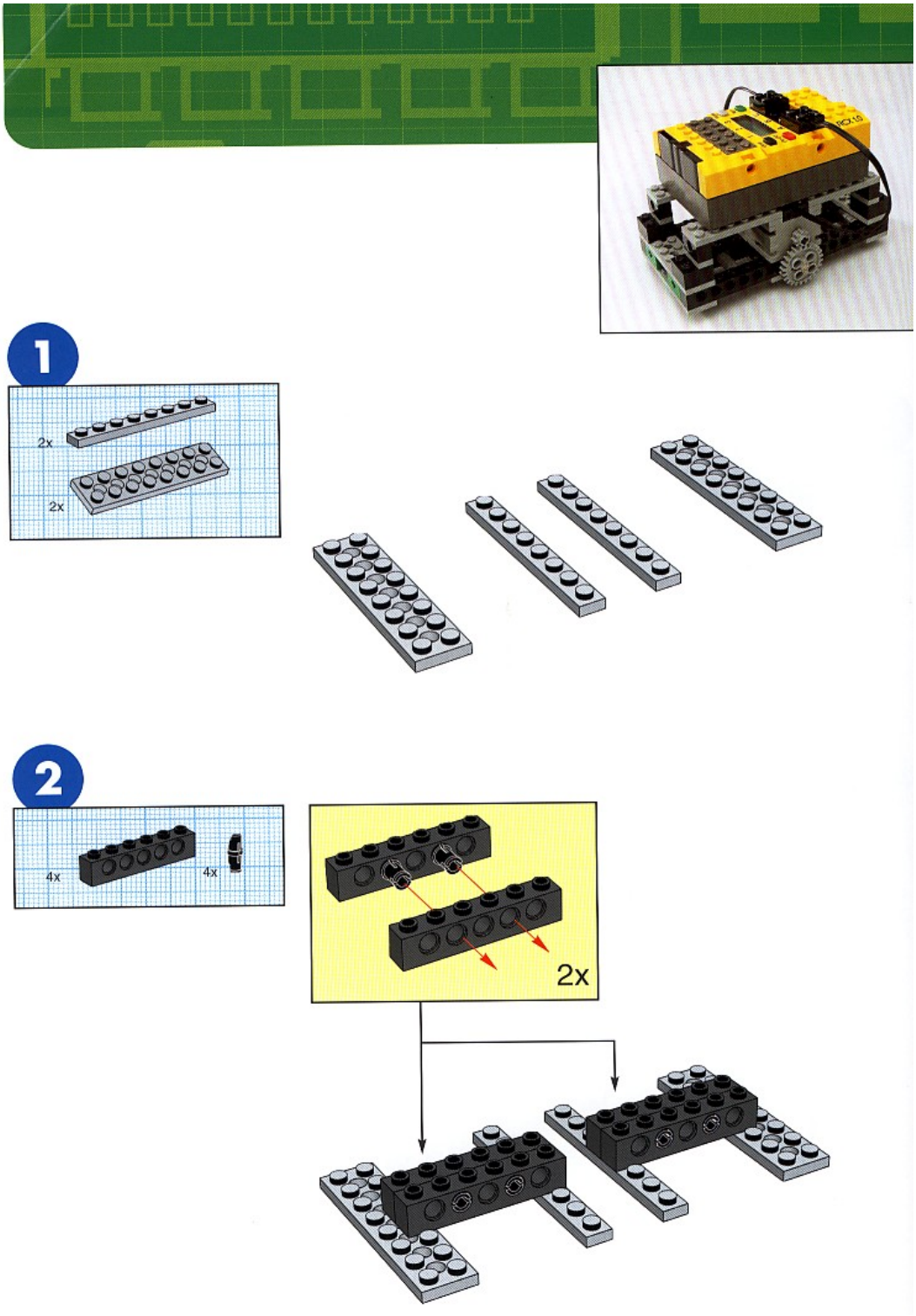
Ένα μειονέκτημα, ωστόσο, της πλατφόρμας των Lego Mindstorms έγκειται στο γεγονός ότι για να ελεγχθεί η ορθότητα ενός προγράμματος θα πρέπει να κατασκευαστεί το επιθυμητό μοντέλο και στη συνέχεια να σταλεί και να εκτελεστεί στο RCX αρκετές φορές (μέθοδος δοκιμής και σφάλματος). Το γεγονός αυτό επιμηκύνει την διαδικασία κατασκευής προγράμματος σημαντικά. Το πιο σημαντικό είναι ότι στα περισσότερα εργαστήρια δεν είναι δυνατό για τους φοιτητές να πάρουν τα ρομπότ και να εργαστούν με αυτά στο σπίτι τους. Αυτό έχει ως αποτέλεσμα, τη δραστική μείωση του χρόνου που έχουν οι φοιτητές για να επιτελέσουν τις εργασίες τους. Το πρόβλημα αυτό λύνεται με την προσομοίωση που γίνεται στο περιβάλλον της MATLAB. Με την προσομοίωση, οι φοιτητές μπορούν να εργαστούν σε ένα

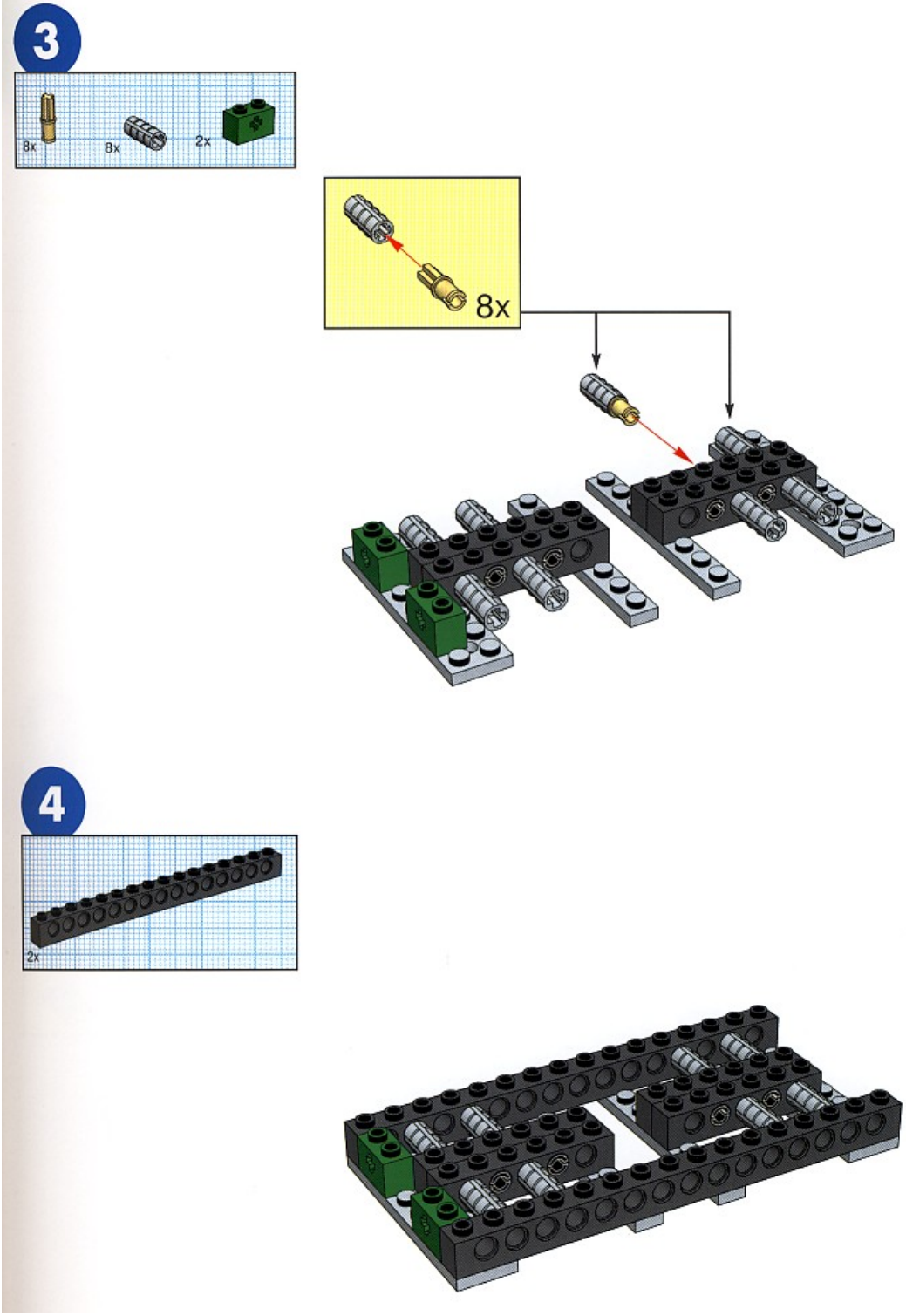
περιβάλλον εικονικής πραγματικότητας και από το σπίτι τους με τη χρήση του ηλεκτρονικού τους υπολογιστή. Επιπλέον, αποφεύγεται η συνεχής αποστολή και η εκτέλεση των προγραμμάτων για την διαπίστωση της ορθότητας τους. Αυτό γίνεται στον υπολογιστή με συνεπακόλουθο την μεγάλη εξοικονόμηση χρόνου.

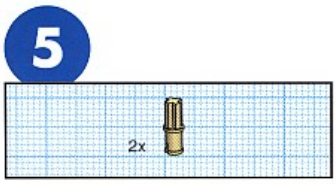
Τα Lego Mindstorms χρησιμοποιούνται παγκοσμίως σε πολλές τάξεις, από δημοτικά μέχρι πανεπιστήμια. Είναι ένα ισχυρό εργαλείο για την διδασκαλία μίας μεγάλης ποικιλίας θεμάτων όπως η κατασκευή ρομπότ, ο προγραμματισμός, η τεχνητή νοημοσύνη, τα συστήματα αυτομάτου ελέγχου και πολλά άλλα. Ο συνδυασμός της μάθησης και της διασκέδασης, η ποικιλία των διαφορετικών ρομποτικών κατασκευών, η ευελιξία του λογισμικού και το χαμηλό κόστος, θα έκαναν την πλατφόρμα των Lego Mindstorms πολύ ελκυστική για την χρήση της ως εργαστηριακή άσκηση στην διδασκαλία των συστημάτων αυτομάτου ελέγχου.

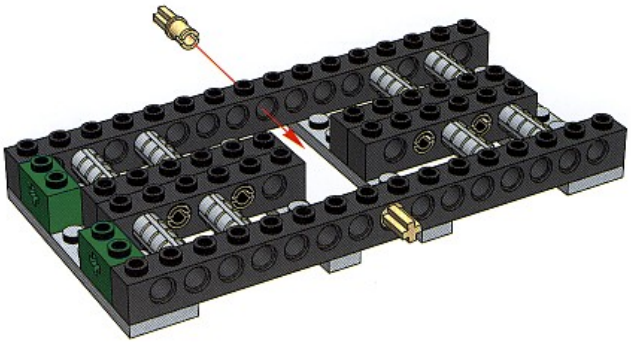
ΠΑΡΑΡΤΗΜΑ Α

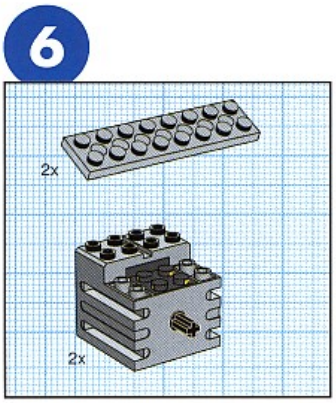
Στο παράρτημα αυτό δίνεται η σχηματική απεικόνιση, βήμα προς βήμα, της κατασκευής του αυτοκινήτου που χρησιμοποιείται στην παρούσα εργασία. Για την κατασκευή αυτή χρησιμοποιούνται τα διάφορα εξαρτήματα που περιέχονται στο πακέτο των Lego Mindstorms.

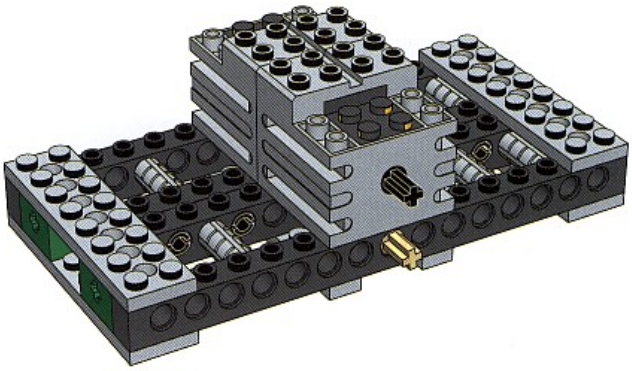


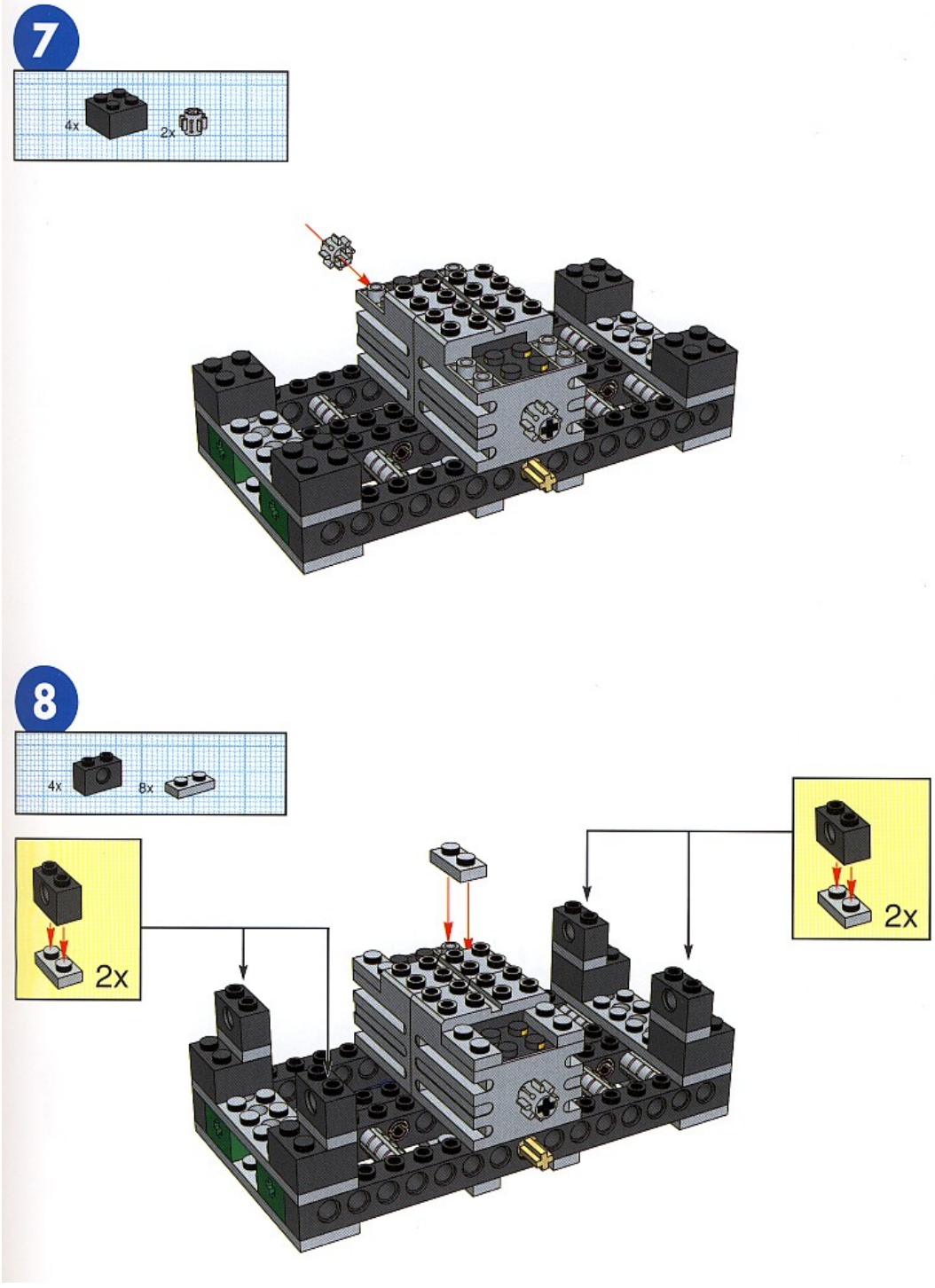

3
8x
8x
2x
8x
4
2x

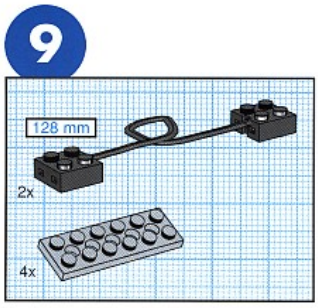

5
2x

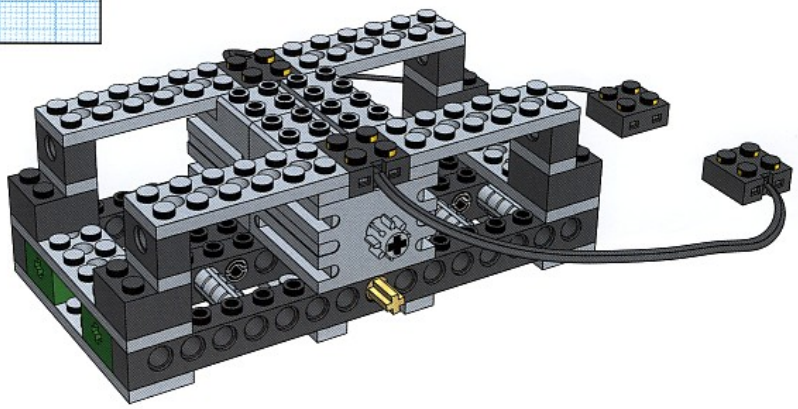


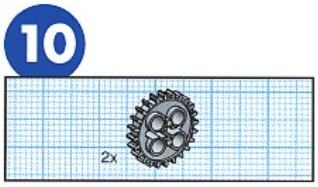

6
2x
2x

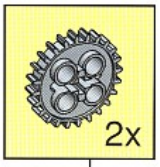


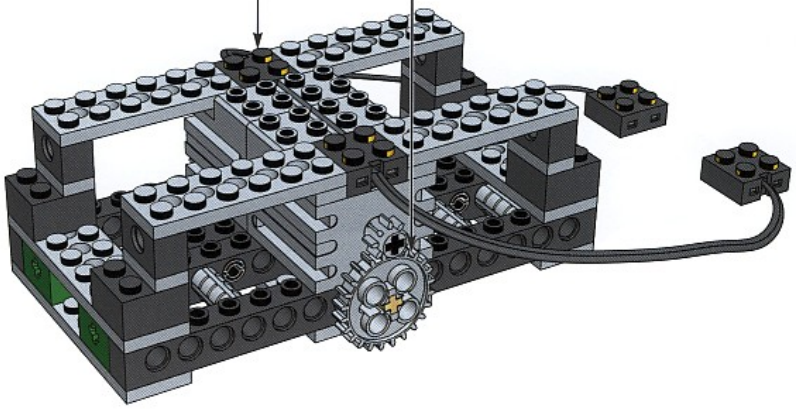

7
4x
2x
8
4x
8x
2x
2x

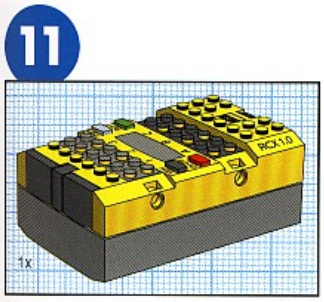

9
128 mm
2x
4x

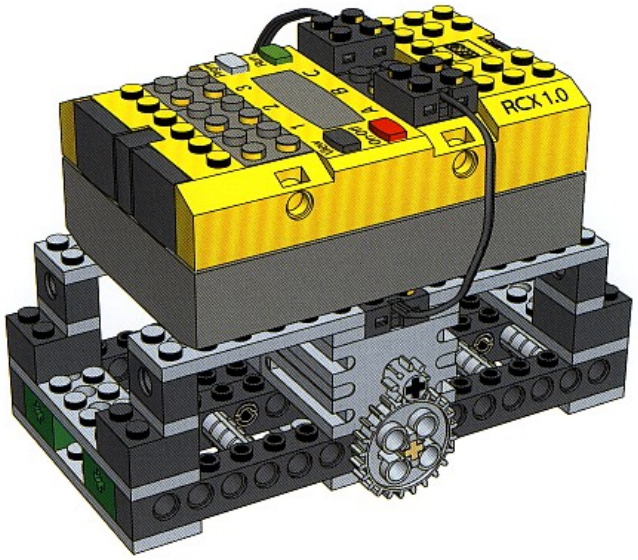


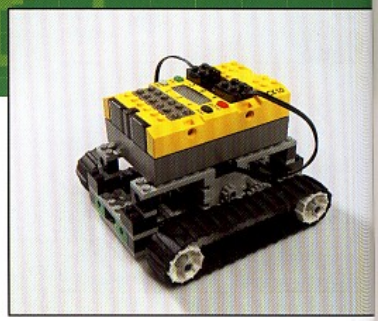

10
2x


2x

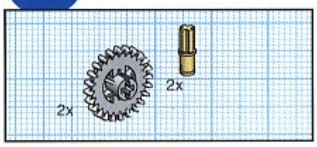


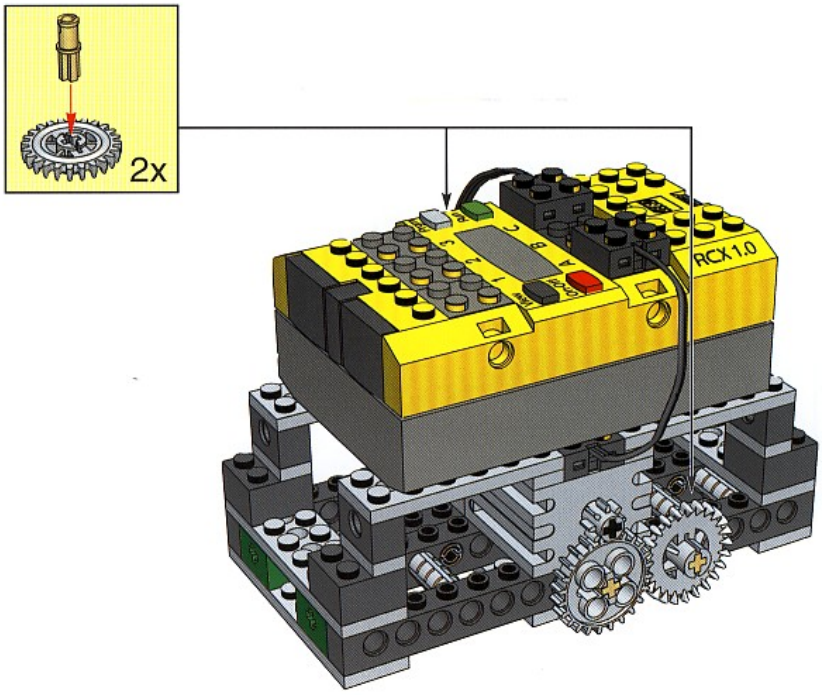

RCX 1.0
1X

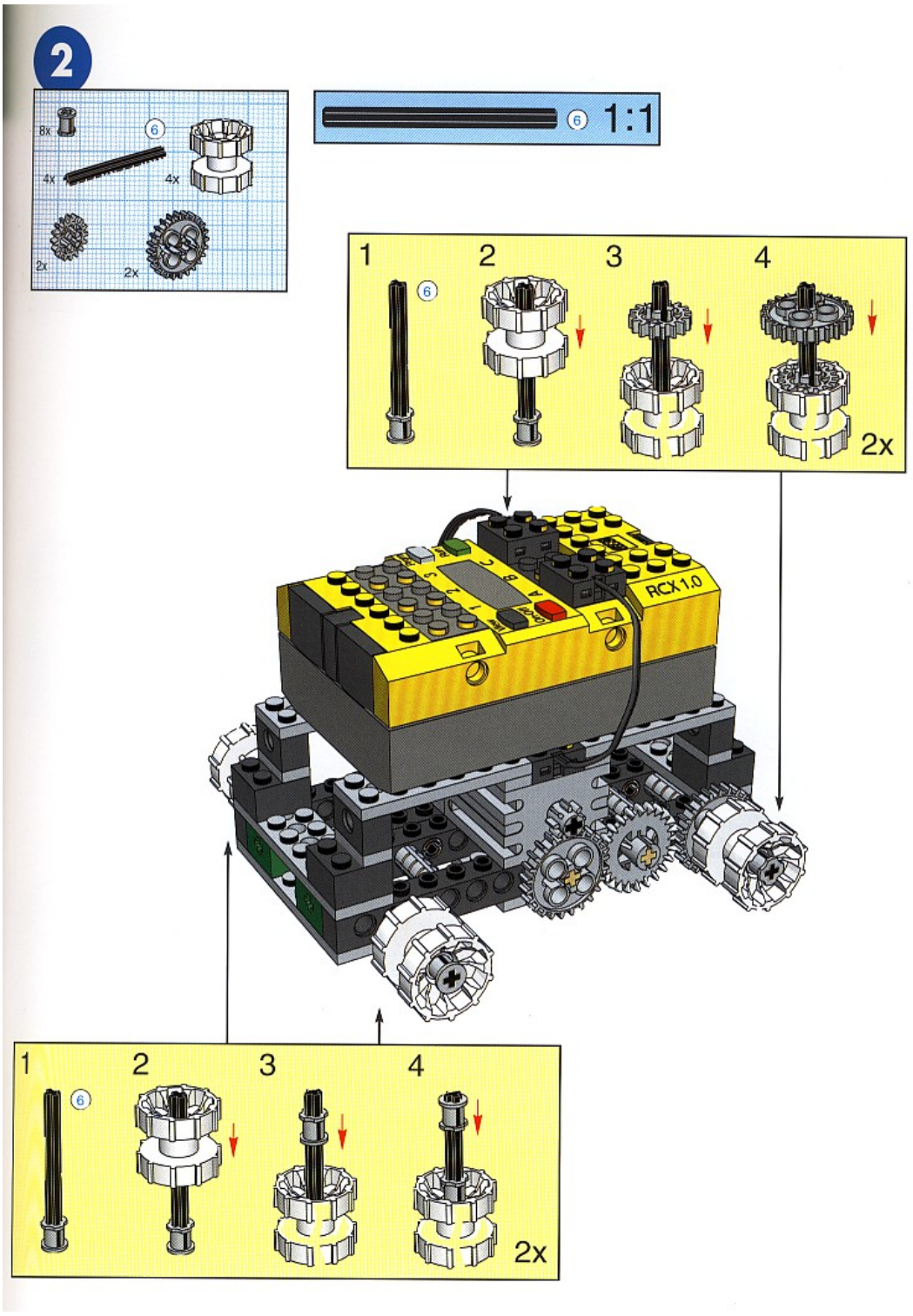

RCX 1.0



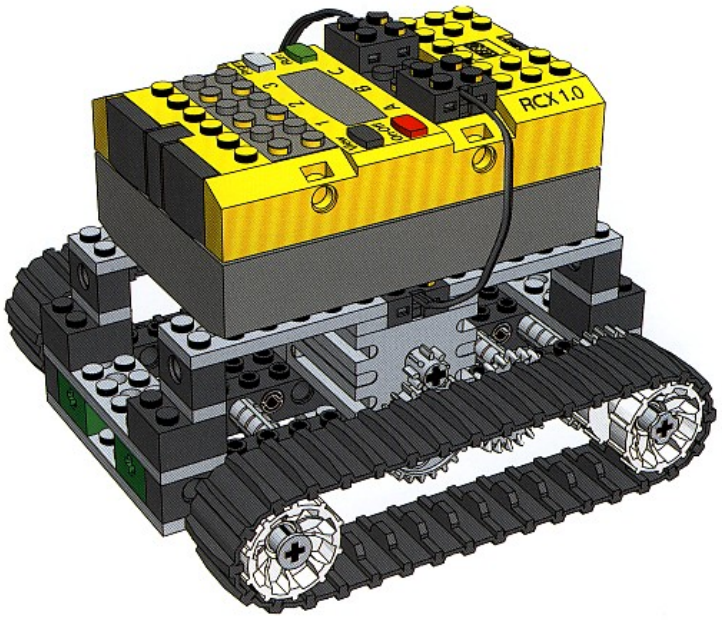
1

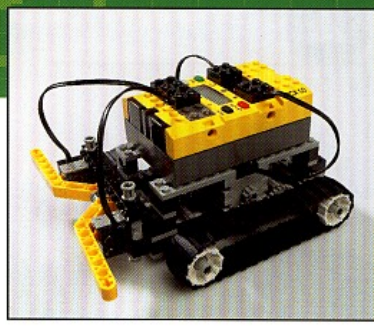


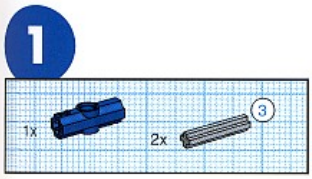


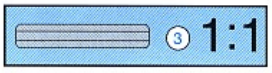

2
8x
6
4x
4x
2x
2x
6 1:1
1
2
3
4
6
2x
RCX 1.0
1
2
3
4
6
2x

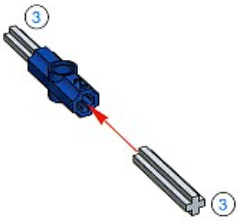

3
2x

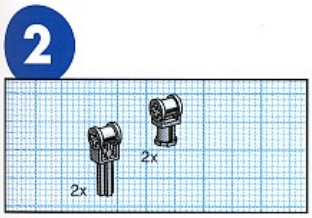

RCX 1.0

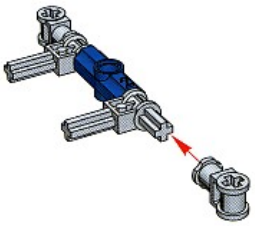


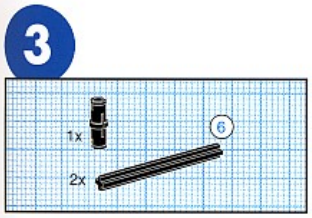

1
1x
2x
3


3
1:1

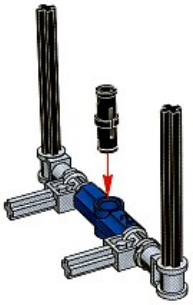

3
3

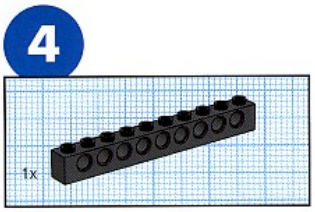

2
2x
2x

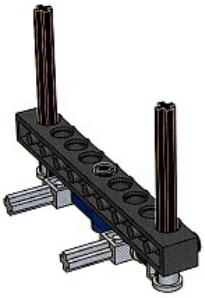


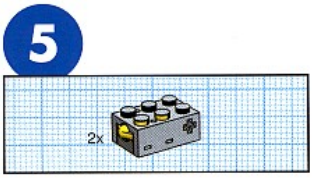

3
1x
6
2x

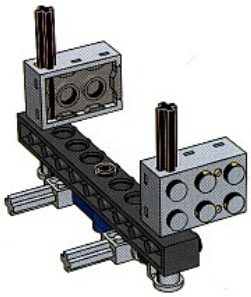

6
1:1

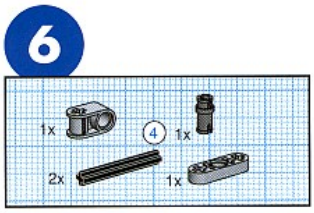


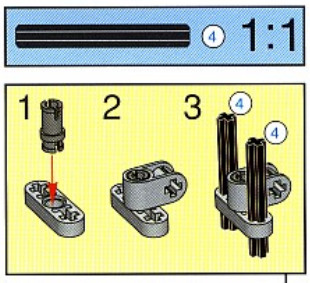

4
1x

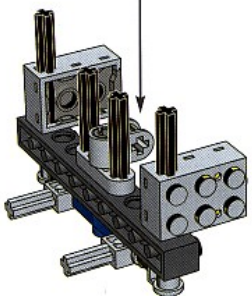


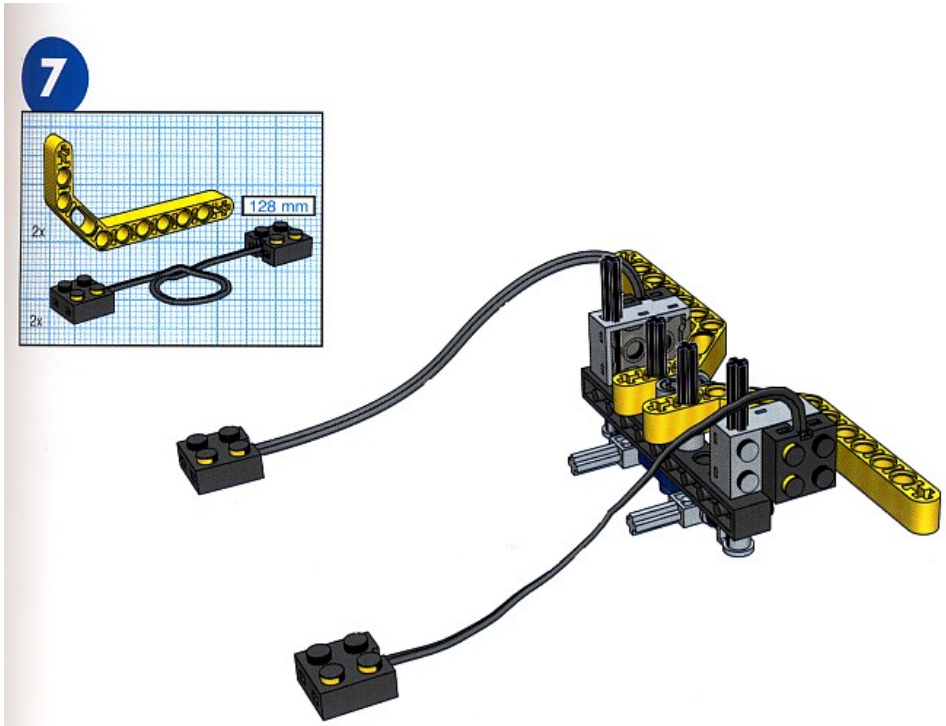

5
2x

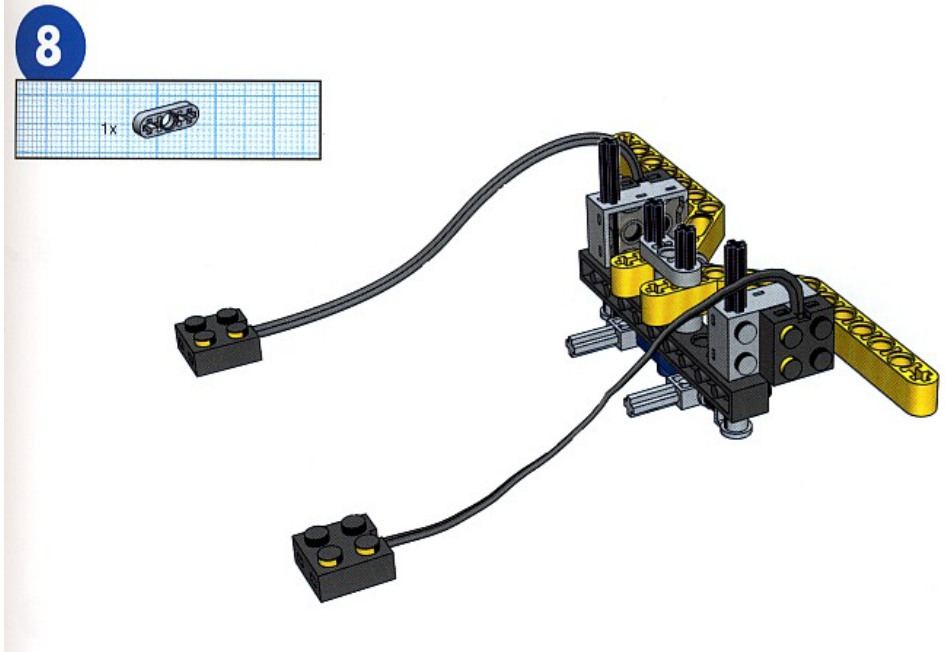


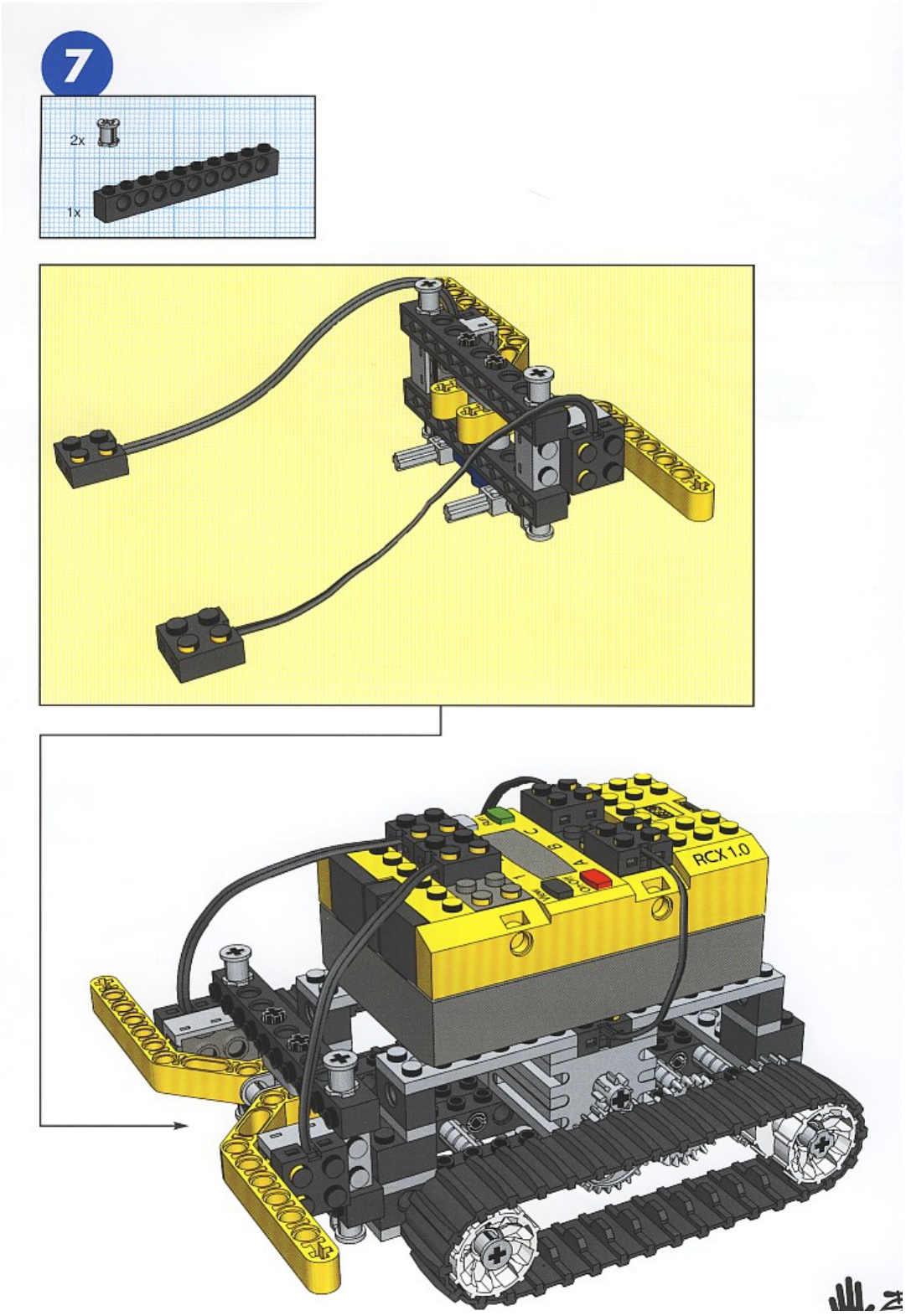

6
1x
4
1x
2x
1x


4 1:1
1
2
3
4
4



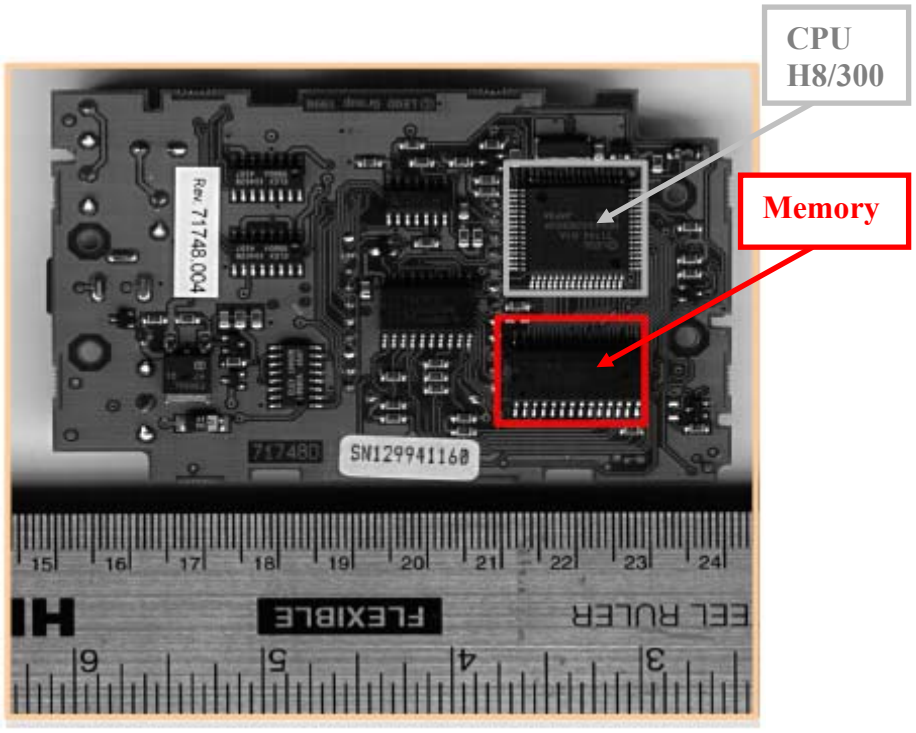

7
128 mm
2x
2x


8
1x

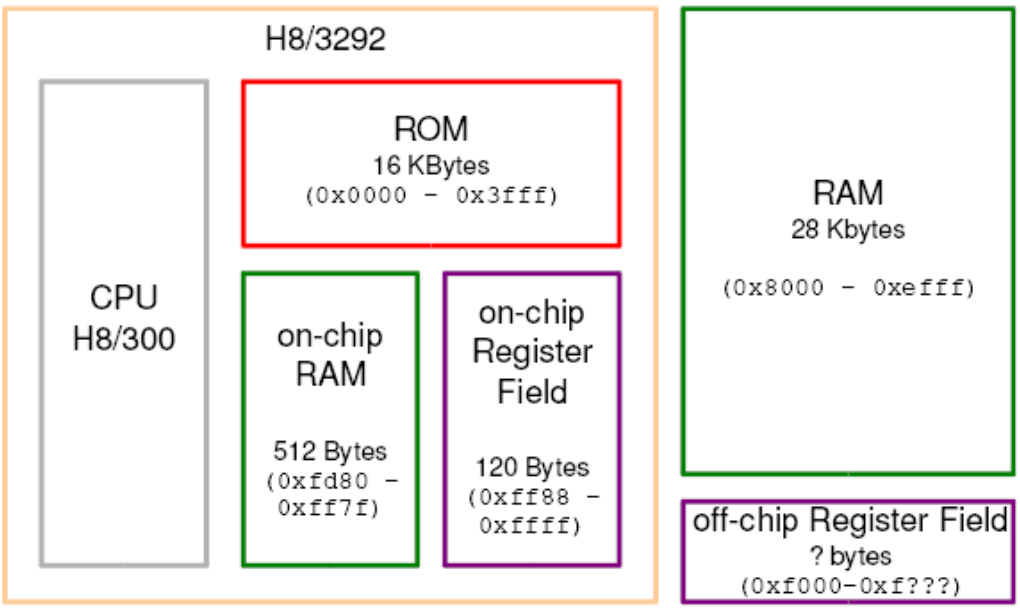

7
2x
1x
RCX 1.0

ΠΑΡΑΡΤΗΜΑ Β

Στο παράρτημα αυτό δίνονται τα βασικά χαρακτηριστικά του μικροελεγκτή της Hitachi H8/3292.



Εικόνα Β1 Η πλακέτα του RCX.



Εικόνα Β2 Το διάγραμμα του μικροελεγκτή Hitachi H8/3292.

Πίνακας 1 Τα βασικά χαρακτηριστικά του μικροελεγκτή Hitachi H8/3292.

Item	Specification
CPU	Two-way general register configuration • Eight 16-bit registers, or • Sixteen 8-bit registers High-speed operation • Maximum clock rate (ø clock): 16 MHz at 5 V, 12 MHz at 4 V or 10 MHz at 3 V • 8- or 16-bit register-register add/subtract: 125 ns (16 MHz), 167 ns (12 MHz), 200 ns (10 MHz) • 8 × 8-bit multiply: 875 ns (16 MHz), 1167 ns (12 MHz), 1400 ns (10 MHz) • 16 ÷ 8-bit divide: 875 ns (16 MHz), 1167 ns (12 MHz), 1400 ns (10 MHz) Streamlined, concise instruction set • Instruction length: 2 or 4 bytes • Register-register arithmetic and logic operations • MOV instruction for data transfer between registers and memory Instruction set features • Multiply instruction (8 bits × 8 bits) • Divide instruction (16 bits ÷ 8 bits) • Bit-accumulator instructions • Register-indirect specification of bit positions
Memory	• H8/3297: 60k-byte ROM; 2k-byte RAM • H8/3296: 48k-byte ROM; 2k-byte RAM • H8/3294: 32k-byte ROM; 1k-byte RAM • H8/3292: 16k-byte ROM; 512-byte RAM
16-bit free-running timer (1 channel)	• One 16-bit free-running counter (can also count external events) • Two output-compare lines • Four input capture lines (can be buffered)
8-bit timer (2 channels)	Each channel has • One 8-bit up-counter (can also count external events) • Two time constant registers
Watchdog timer (WDT) (1 channel)	• Overflow can generate a reset or NMI interrupt • Also usable as interval timer
Serial communication interface (SCI) (1 channel)	• Asynchronous or synchronous mode (selectable) • Full duplex: can transmit and receive simultaneously • On-chip baud rate generator
A/D converter	• 10-bit resolution • Eight channels: single or scan mode (selectable) • Start of A/D conversion can be externally triggered • Sample-and-hold function
I/O ports	• 43 input/output lines (16 of which can drive LEDs) • 8 input-only lines

Πίνακας 1 Τα βασικά χαρακτηριστικά του μικροελεγκτή Hitachi H8/3292 (συνέχεια).

Item	Specification
Interrupts	• Four external interrupt lines: $\overline{NMI}$, IRQ_0 to IRQ_2 • 19 on-chip interrupt sources
Wait control	• Three selectable wait modes
Operating modes	• Expanded mode with on-chip ROM disabled (mode 1) • Expanded mode with on-chip ROM enabled (mode 2) • Single-chip mode (mode 3)
Power-down modes	• Sleep mode • Software standby mode • Hardware standby mode
Other features	• On-chip oscillator

Item	Specification			
Series lineup	Part Number			
	Product Name	5-V Version (16 MHz) 4-V Version (12 MHz)	3-V Version (10 MHz)	Package	ROM
	H8/3292	HD6433292P16 HD6433292P12	HD6433292VP10	64-pin shrink DIP (DP-64S)	Mask ROM
		HD6433292F16 HD6433292F12	HD6433292VF10	64-pin QFP (FP-64A)	
		HD6433292TF16 HD6433292TF12	HD6433292VTF10	80-pin TQFP (TFP-80C)	

Πίνακας 2 Τα βασικά χαρακτηριστικά του κεντρικού επεξεργαστή H8/300.

The main features of the H8/300 CPU are listed below.

- Two-way register configuration
  - — Sixteen 8-bit general registers, or
  - — Eight 16-bit general registers
- Instruction set with 57 basic instructions, including:
  - — Multiply and divide instructions
  - — Powerful bit-manipulation instructions
- Eight addressing modes
  - — Register direct (Rn)
  - — Register indirect (@Rn)
  - — Register indirect with displacement (@(d:16, Rn))
  - — Register indirect with post-increment or pre-decrement (@Rn+ or @–Rn)
  - — Absolute address (@aa:8 or @aa:16)
  - — Immediate (#xx:8 or #xx:16)
  - — PC-relative (@(d:8, PC))
  - — Memory indirect (@@aa:8)
- Maximum 64-kbyte address space
- High-speed operation
  - — All frequently-used instructions are executed in two to four states
- Maximum clock rate (ø clock): 16 MHz at 5 V, 12 MHz at 4 V or 10 MHz at 3 V
  - — 8- or 16-bit register-register add or subtract: 125 ns (16 MHz), 167 ns (12 MHz), 200 ns (10 MHz)
  - — 8 × 8-bit multiply: 875 ns (16 MHz), 1167 ns (12 MHz), 1400 ns (10 MHz)
  - — 16 ÷ 8-bit divide: 875 ns (16 MHz), 1167 ns (12 MHz), 1400 ns (10 MHz)
- Power-down mode
  - — SLEEP instruction

ΠΑΡΑΡΤΗΜΑ Γ

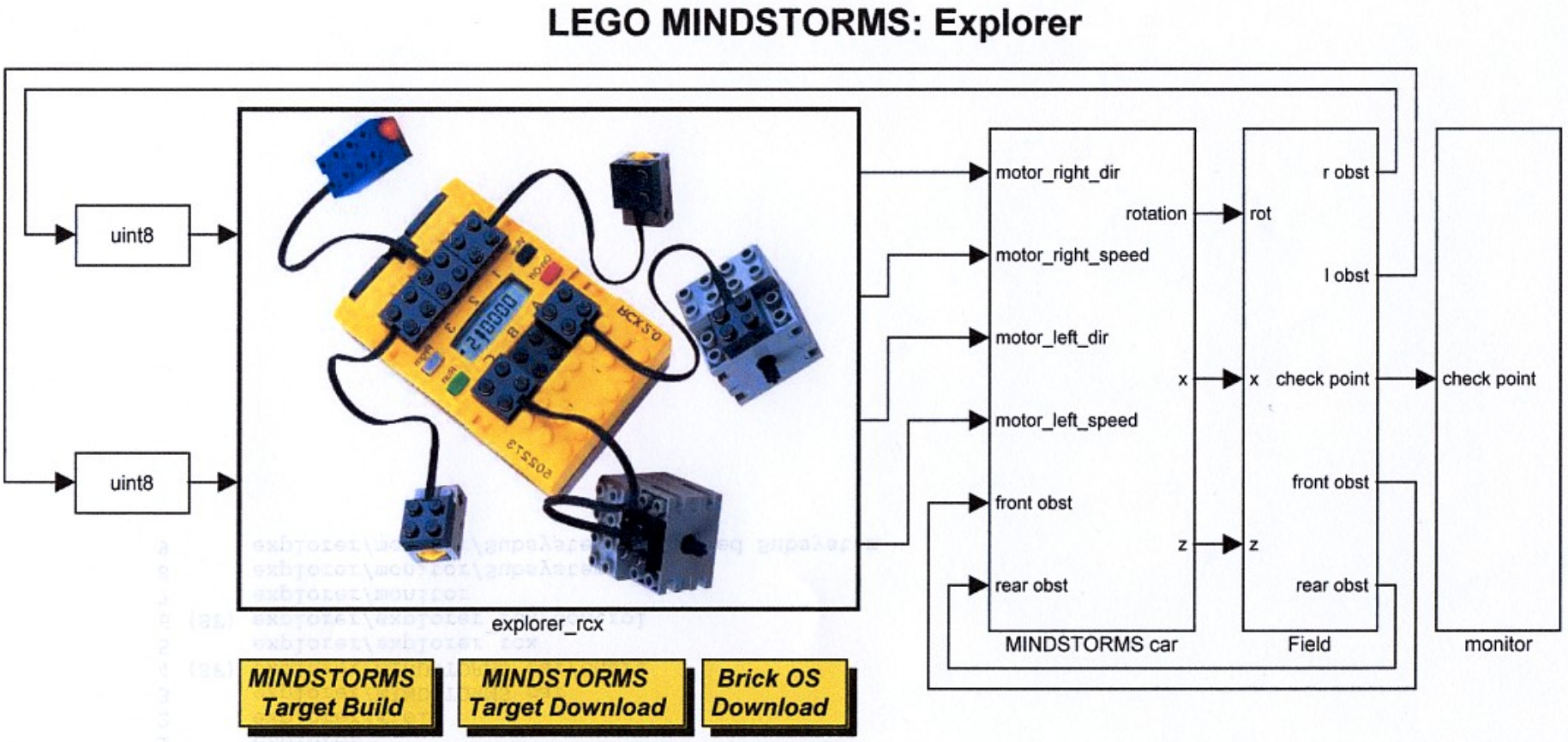
Στο παράρτημα αυτό δίνονται αναλυτικά τα πλήρη διαγράμματα Simulink και Stateflow για τα δύο μοντέλα Explorer.mdl και Linetracker.mdl. Τα μοντέλα αυτά περιλαμβάνουν και τον μηχανισμό της προσομοίωσης του μοντέλου στην οθόνη του υπολογιστή.

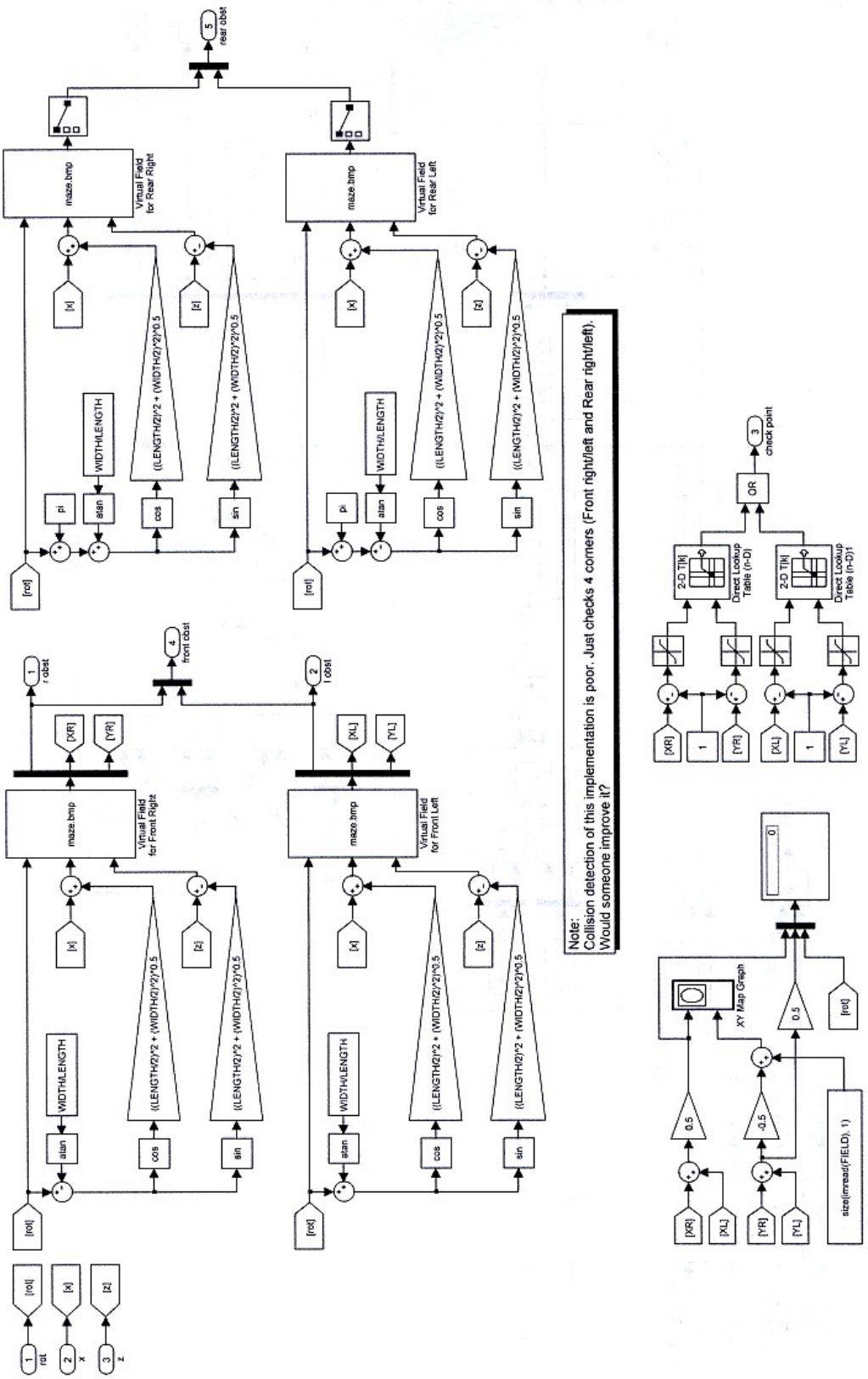
Explorer.mdl

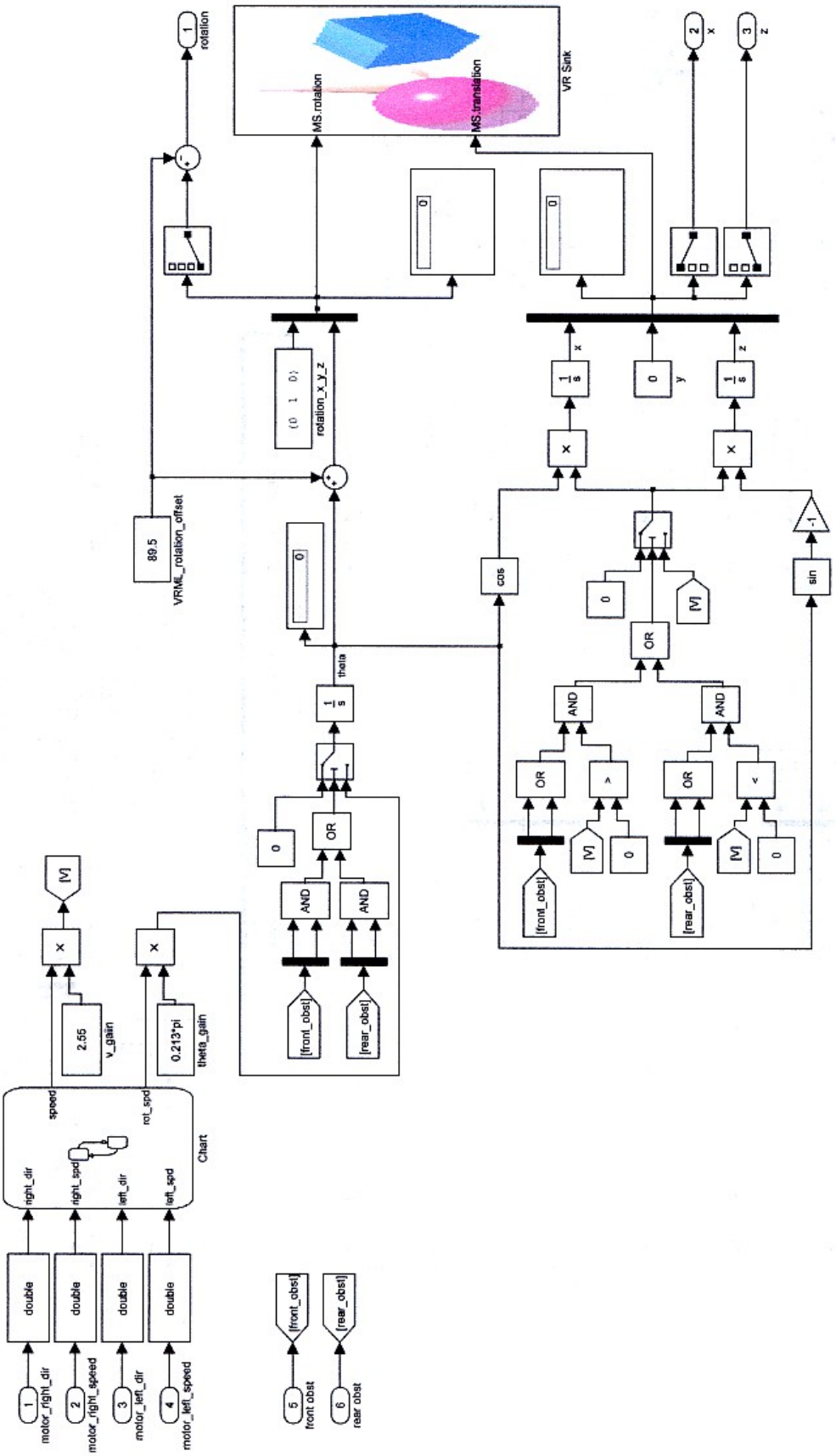
Σελίδα		Όνομα υποσυστήματος
101		explorer
102		explorer/Field
103		explorer/MINDSTORMS car
104	(Stateflow)	explorer/MINDSTORMS car/Chart
105		explorer/explorer_rcx
106	(Stateflow)	explorer/explorer_rcx/Control
107		explorer/monitor
107		explorer/monitor/Subsystem

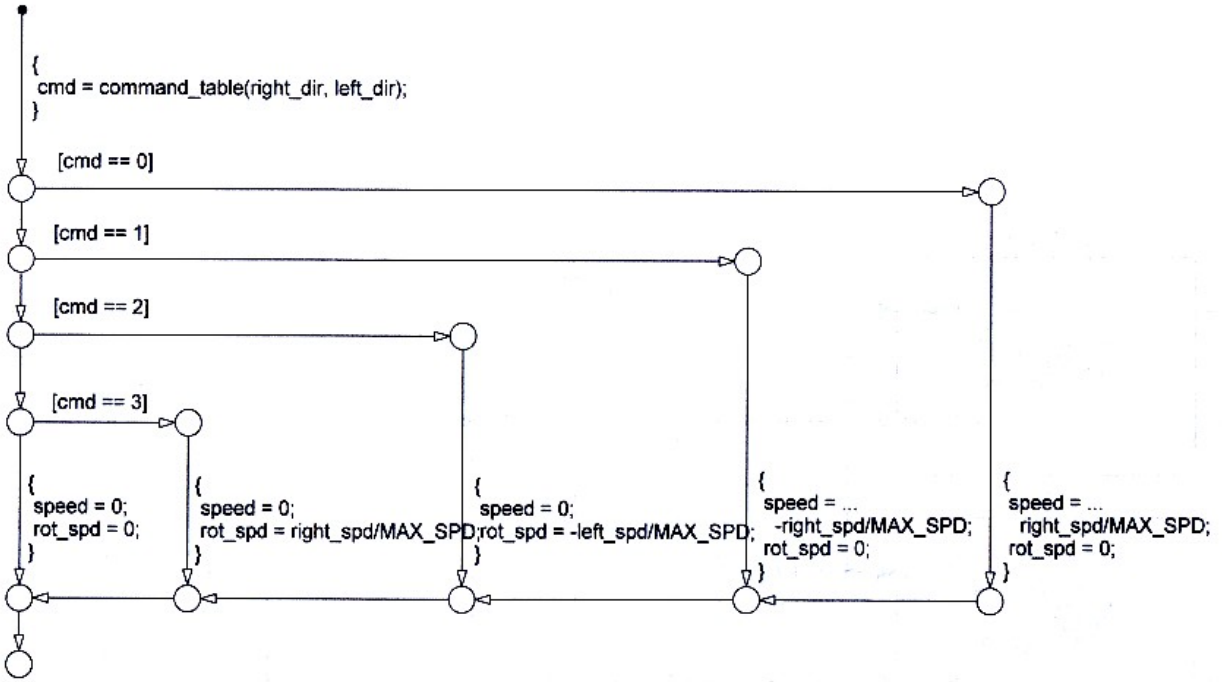
Linetracker.mdl

Σελίδα		Όνομα υποσυστήματος
108		linetracker
109		linetracker/MINDSTORMS car
110	(Stateflow)	linetracker/MINDSTORMS car/Chart
111		linetracker/Track field
112		linetracker/linetracker_rcx
113	(Stateflow)	linetracker/linetracker_rcx/direction_control
113	(Stateflow)	linetracker/linetracker_rcx/line_detect
113	(Stateflow)	linetracker/linetracker_rcx/motor_control


LEGO MINDSTORMS: Explorer
uint8
uint8
explorer_rcx
motor_right_dir
motor_right_speed
motor_left_dir
motor_left_speed
front obst
rear obst
rotation
x
z
MINDSTORMS car
rot
x
z
r obst
l obst
check point
front obst
rear obst
Field
check point
monitor
MINDSTORMS Target Build
MINDSTORMS Target Download
Brick OS Download

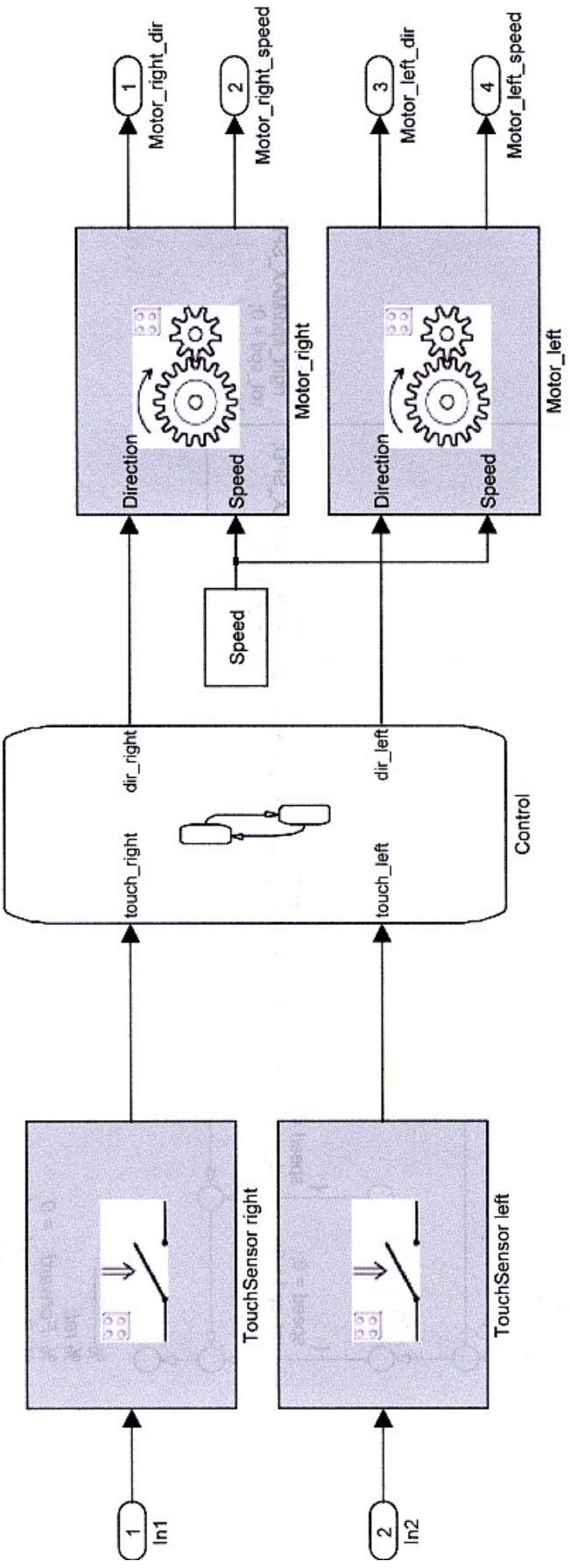

rear obst
maze.bmp
Virtual Field for Rear Right
Virtual Field for Rear Left
front obst
r obst
l obst
Virtual Field for Front Right
Virtual Field for Front Left
WIDTH/LENGTH
atan
cos
sin
pi
((LENGTH/2)^2 + (WIDTH/2)^2)^0.5
Note:
Collision detection of this implementation is poor. Just checks 4 corners (Front right/left and Rear right/left).
Would someone improve it?
2-D T[k]
Direct Lookup Table (n-D)
OR
check point
XY Map Graph
size(imread('FIELD', 1))

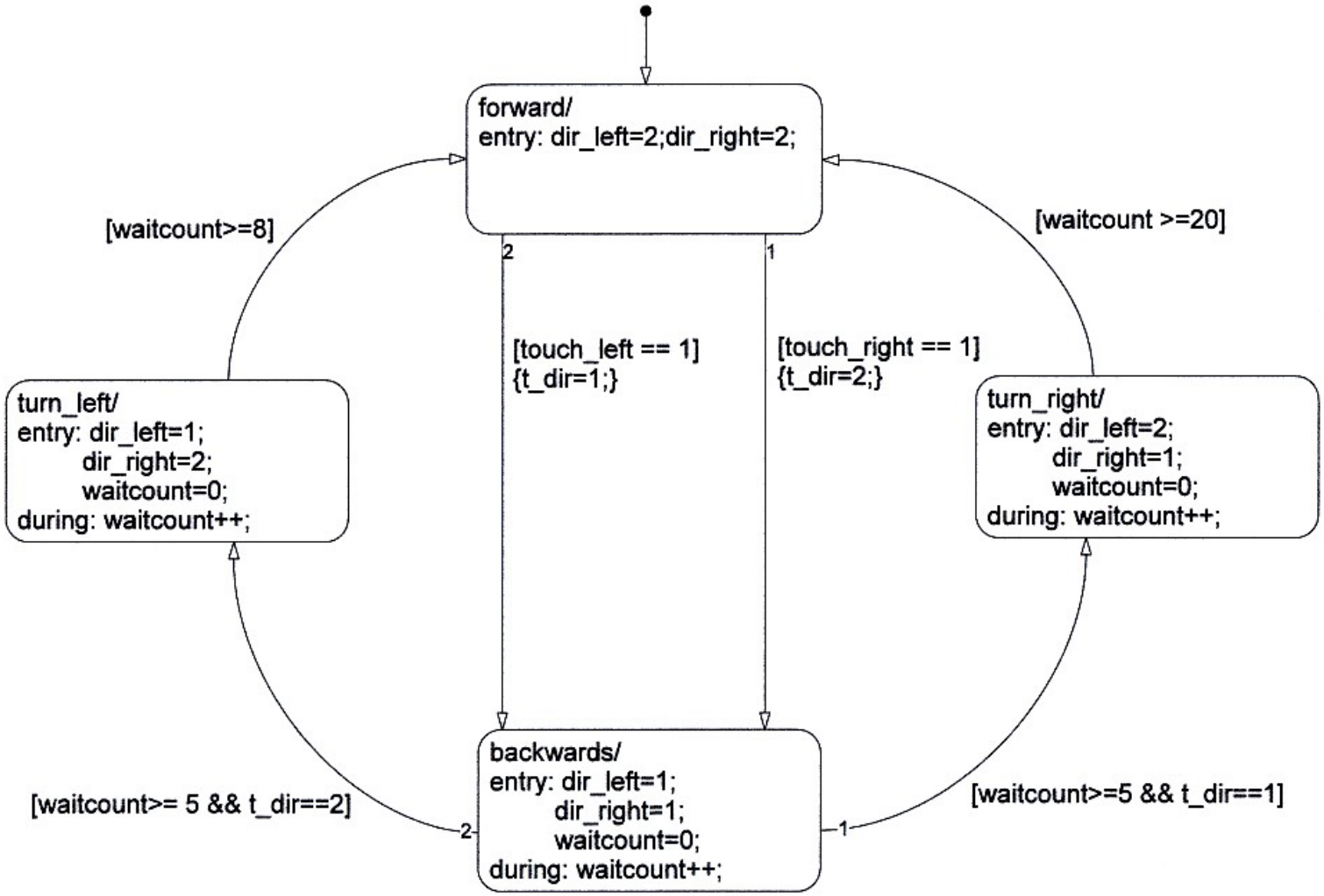

1
rotation
MS.rotation
MS.translation
VR Sink
2
x
3
z
0
0
rotation_x_y_z
(0 1 0)
1/s
x
0
y
1/s
z
x
x
89.5
VRML_rotation_offset
cos
sin
-1
0
[V]
OR
AND
AND
OR
OR
theta
1/s
>
<
[V]
[V]
0
0
0
OR
AND
AND
[front_obst]
[rear_obst]
[front_obst]
[rear_obst]
[V]
x
x
2.55
v_gain
0.213*pi
theta_gain
speed
rot_spd
Chart
right_dir
right_spd
left_dir
left_spd
double
double
double
double
1
motor_right_dir
2
motor_right_speed
3
motor_left_dir
4
motor_left_speed
5
front obst
6
rear obst

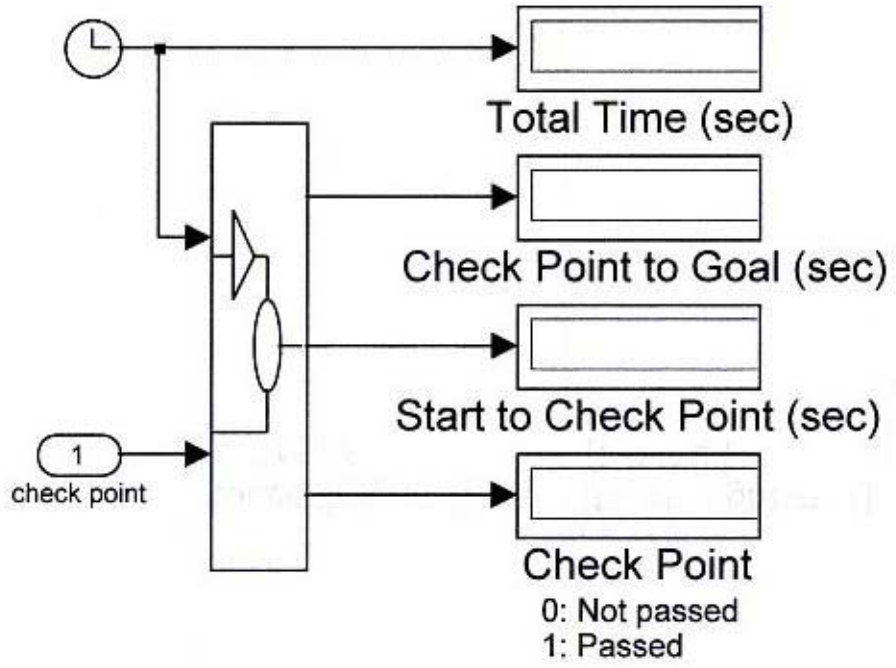



```
%----------
% ret:
%  Forward   = 0
%  Bachward  = 1
%  Turn right  = 2
%  Turn left   = 3
```

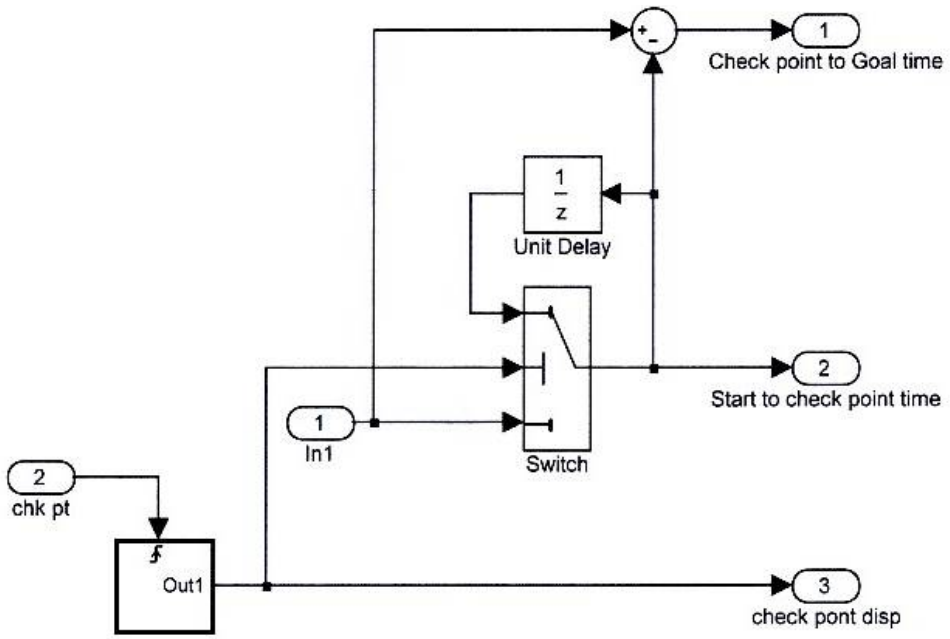
ret = command_table(r_dir, l_dir)


1
In1
2
In2
TouchSensor right
TouchSensor left
touch_right
dir_right
touch_left
dir_left
Control
Speed
Direction
Speed
Motor_right
Direction
Speed
Motor_left
1
Motor_right_dir
2
Motor_right_speed
3
Motor_left_dir
4
Motor_left_speed

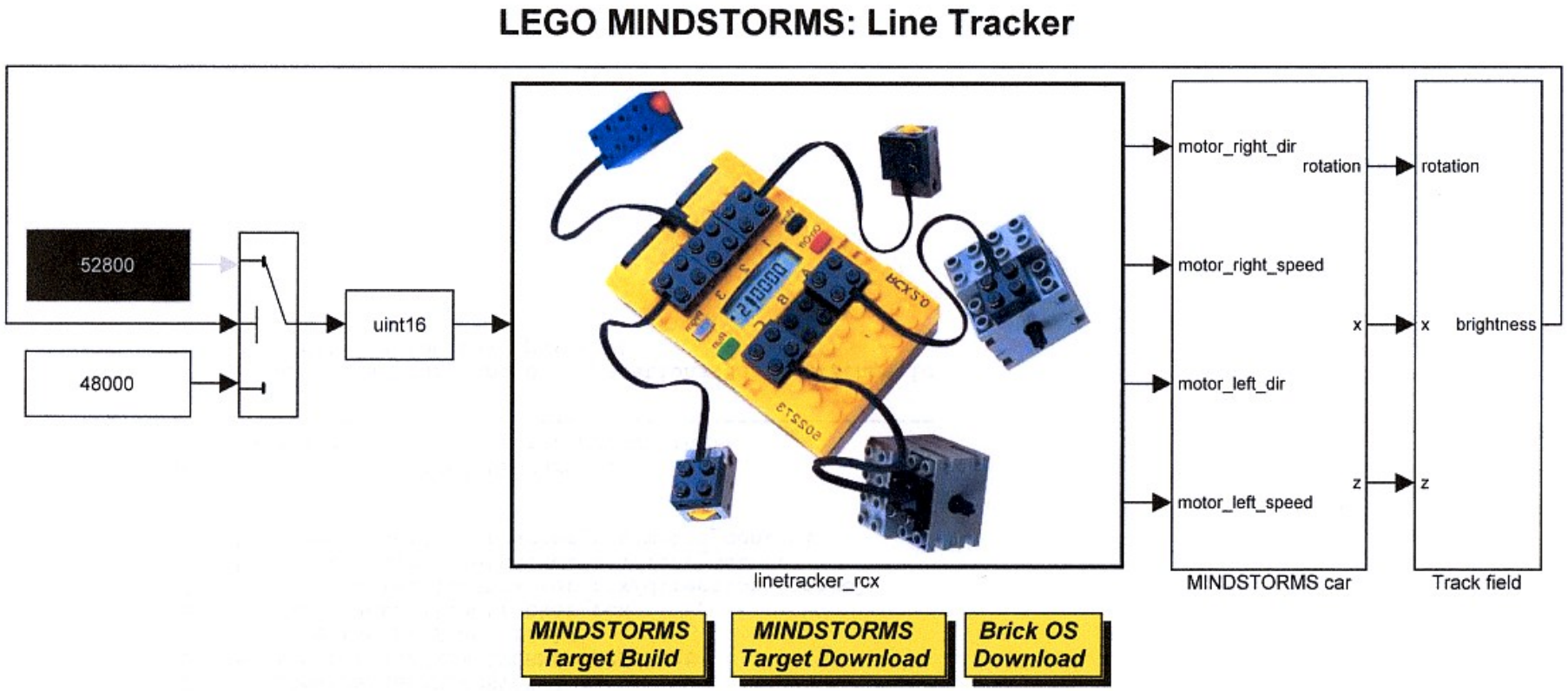

forward/
entry: dir_left=2;dir_right=2;
[waitcount>=8]
[waitcount >=20]
2
1
[touch_left == 1]
{t_dir=1;}
[touch_right == 1]
{t_dir=2;}
turn_left/
entry: dir_left=1;
dir_right=2;
waitcount=0;
during: waitcount++;
turn_right/
entry: dir_left=2;
dir_right=1;
waitcount=0;
during: waitcount++;
backwards/
entry: dir_left=1;
dir_right=1;
waitcount=0;
during: waitcount++;
[waitcount>= 5 && t_dir==2]
[waitcount>=5 && t_dir==1]
2
1

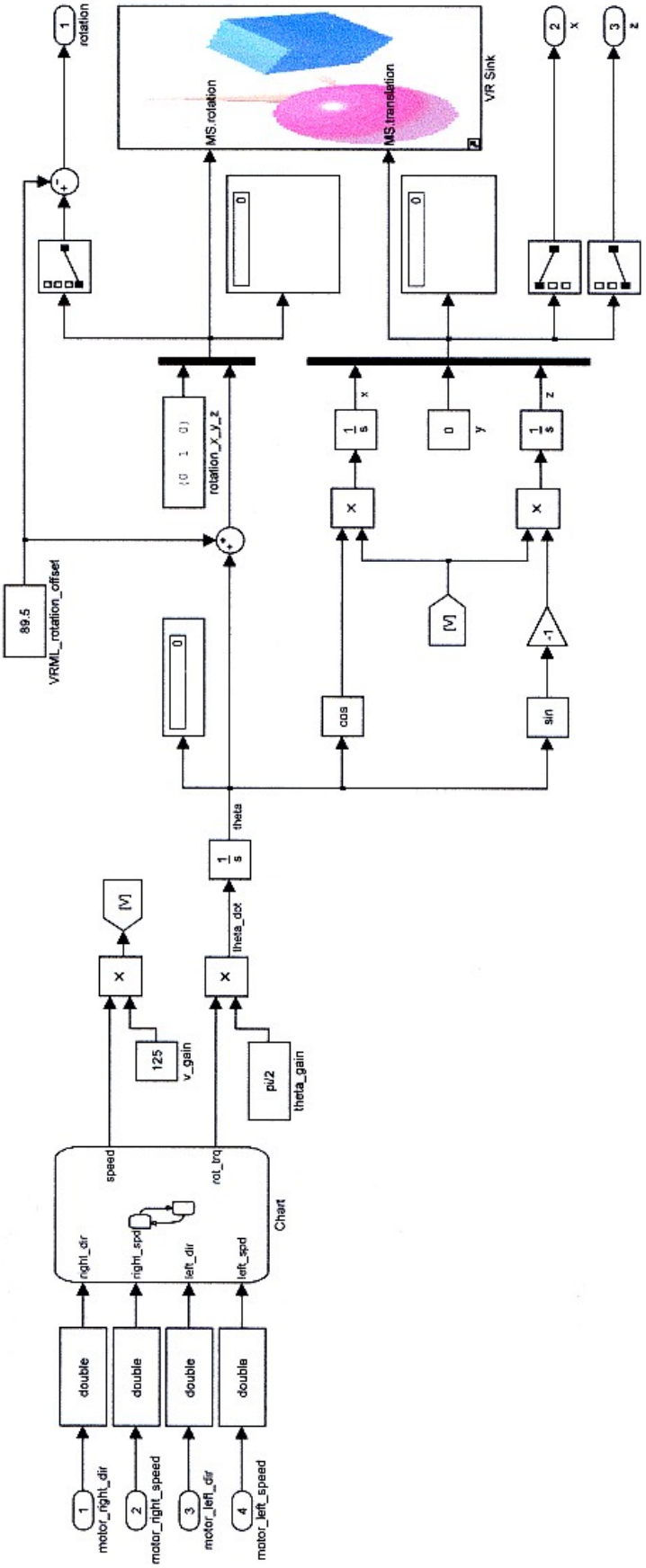


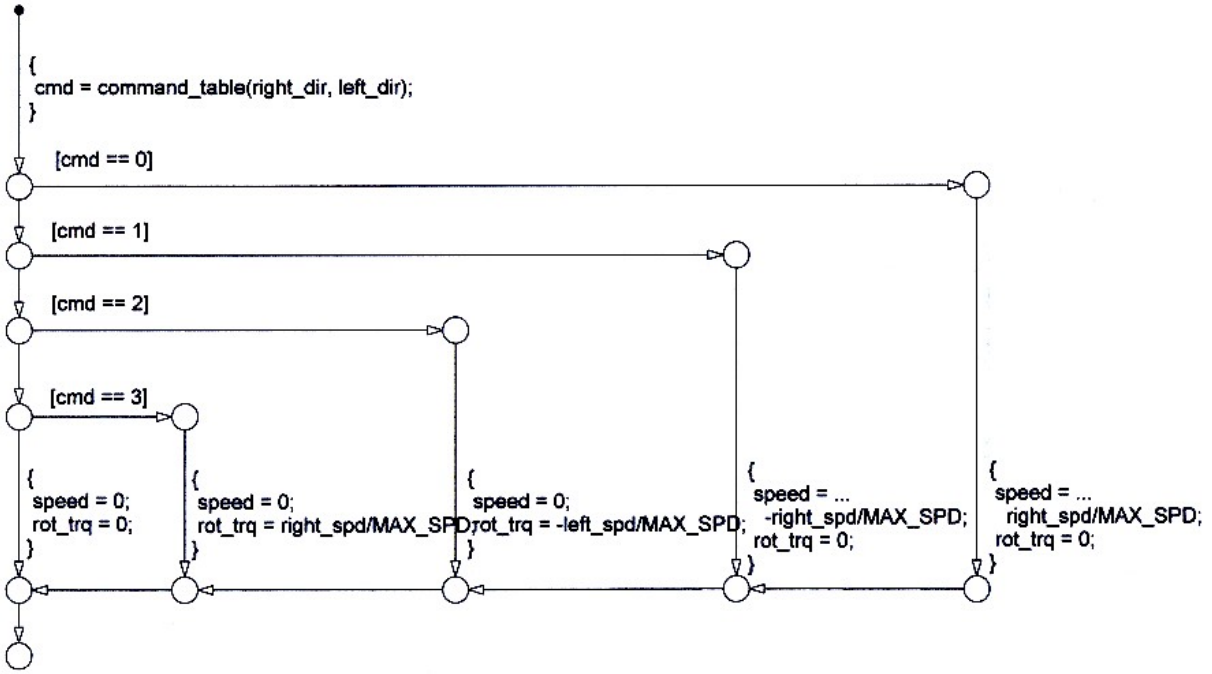
Εικόνα Γ1 explorer/monitor



Εικόνα Γ2 explorer/monitor/Subsystem

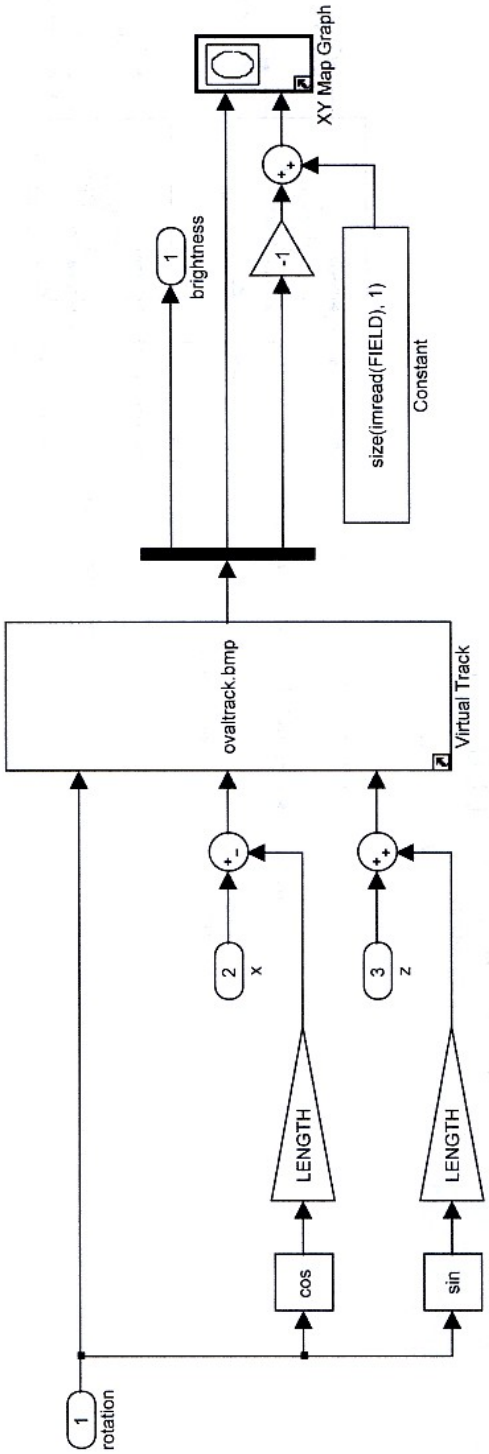

LEGO MINDSTORMS: Line Tracker
52800
48000
uint16
linetracker_rcx
motor_right_dir
rotation
motor_right_speed
motor_left_dir
motor_left_speed
MINDSTORMS car
rotation
x
brightness
z
Track field
MINDSTORMS Target Build
MINDSTORMS Target Download
Brick OS Download

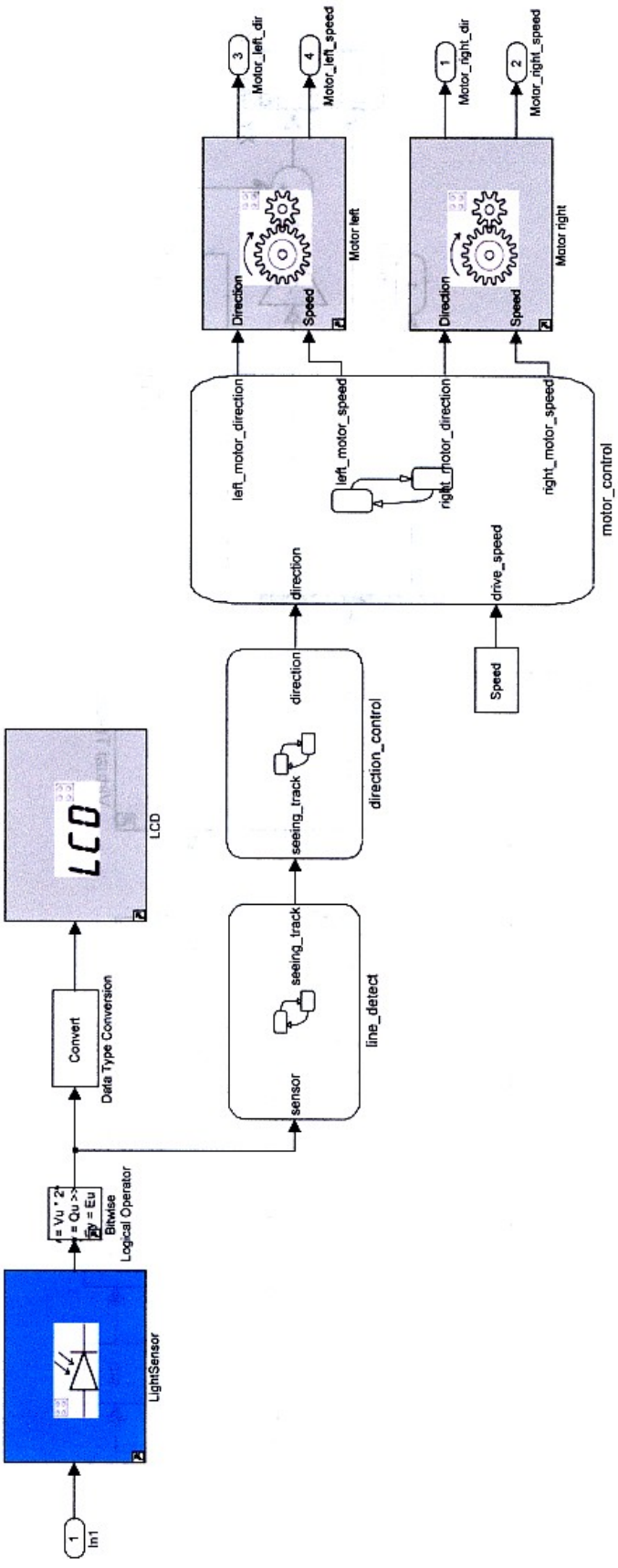

1
motor_right_dir
2
motor_right_speed
3
motor_left_dir
4
motor_left_speed
double
double
double
double
right_dir
right_spd
left_dir
left_spd
Chart
speed
rot_trq
125
v_gain
pi/2
theta_gain
theta_dot
1
s
theta
89.5
VRML_rotation_offset
rotation_x_y_z
cos
sin
-1
1
s
x
0
y
1
s
z
MS.rotation
MS.translation
VR Sink
1
rotation
2
x
3
z

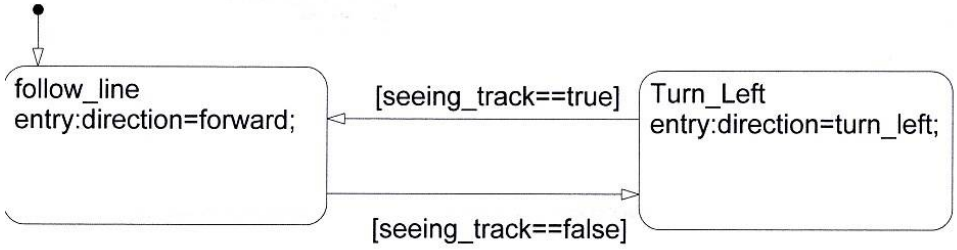



```
%--------
% ret:
%  Forward    = 0
%  Bachward   = 1
%  Turn right = 2
%  Turn left  = 3
```

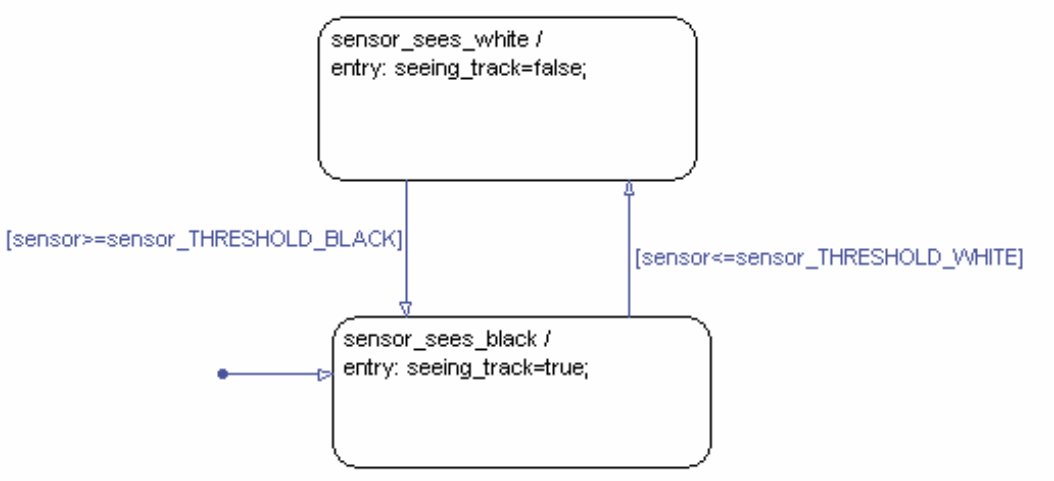
ret = command_table(r_dir, l_dir)


XY Map Graph
1
brightness
-1
size(imread(FIELD), 1)
Constant
ovaltrack.bmp
Virtual Track
2
x
3
z
LENGTH
LENGTH
cos
sin
1
rotation

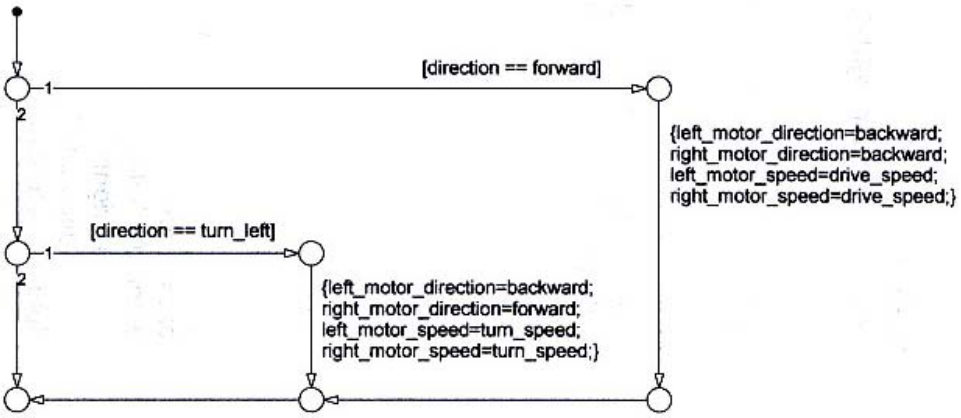

In1
LightSensor
Bitwise Logical Operator
Convert
Data Type Conversion
LCD
sensor
seeing_track
line_detect
seeing_track
direction
direction_control
direction
drive_speed
Speed
left_motor_direction
left_motor_speed
right_motor_direction
right_motor_speed
motor_control
Direction
Speed
Motor left
Direction
Speed
Motor right
3
Motor_left_dir
4
Motor_left_speed
1
Motor_right_dir
2
Motor_right_speed



Εικόνα Γ3 linetracker/linetracker_rcx/direction_control



Εικόνα Γ4 linetracker/linetracker_rcx/line_detect



Εικόνα Γ5 linetracker/linetracker_rcx/motor_control

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Αρναουτάκης Νεκτάριος (2005), ¨Εισαγωγή στο MATLAB Simulink¨.
2. Δερτούζου Μ. (2006), ¨Έκαναν το μάθημα παιχνίδι¨.
3. Καγκάνη Κατερίνα, Δαγδιλέλης Βασίλειος, Σατρατζέμη Μάγια, Ευαγγελίδης Γιώργος (2005), ¨Μια Μελέτη Περίπτωσης της Διδασκαλίας του Προγραμματισμού στη Δευτεροβάθμια Εκπαίδευση με τα Lego Mindstorms¨.
4. Christopher Beland, Wesley Chan, Dwaine Clarke, Richard Park, Michael Trupiano (2000), ¨Lego Mindstorms The Structure of an Engineering (R)evolution¨.
5. Dylan Evans (2006), ¨Teaching robotics with Lego Mindstorms¨.
6. Fauze Valério Polpeta (2003), ¨LEGO RCX Hitachi H8/3292¨.
7. Frank Klassner (2002), ¨A Case Study of Lego Mindstorms' Suitability for Artificial Intelligence and Robotics Courses at the College Level¨.
8. Jonathan Knudsen (2000), ¨Mindstorms in Education¨.
9. Jonathan Knudsen (2000), ¨Lego Mindstorms: An Introduction¨.
10. Kine Kvernstad Hansen, Siv Oma Rogdar, Karine Sorby (2002), ¨Risk assessment of Safety Critical Systems: An approach using LEGO Mindstorms for prototyping¨
11. Madeleine Schep, Nieves McNulty (2002), ¨Use of Lego Mindstorms Kits in Introductory Programming Classes: A Tutorial¨.
12. Max Abilgaard, Niels Holmgard Andersen, Christer Utzen, Kim Sune Hansen (2001), ¨Lego Mindstorms in Control System Education¨.
13. Myles McNally et al. (2006), ¨Do Lego Mindstorms Robots have a Future in CS Education? ¨.
14. Ole Caprani (2006), ¨RCX Manual¨.
15. Stig Nielsson (2000), ¨Instruction to the LegOS kernel¨.
16. The Mathworks, Real Time Workshop For Use with Simulink, Users Guide Version 3
17. The Mathworks, Real Time Workshop Embedded Coder For Use with Real Time Workshop, Users Guide Version 3
18. The Mathworks, Simulink Using, Version 6
19. The Mathworks, Stateflow and Stateflow Coder, Users Guide Version 5
20. The Mathworks, Writing S-Functions, Version 6

ΔΙΕΥΘΥΝΣΕΙΣ ΣΤΟ ΔΙΑΔΙΚΤΥΟ

21. www.e-paideia.net
22. http://fuzzy.iau.dtu.dk/download/lego2.pdf
23. http://legolab.daimi.au.dk/CSaEA/RCX/Manual.dir/RCXManual.html
24. http://www.lego.com/eng/education/mindstorms/default.asp
25. http://www.firstlegoleague.org
26. http://www.from.okay.pl/mgalczer/#
27. http://www.from.okay.pl/mgalczer/legosintro.html