Α.Τ.Ε.Ι. Θεσσαλονίκης
Σχολή Τεχνολογικών
Εφαρμογών
Τμήμα Ηλεκτρονικής

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΑΥΤΟΝΟΜΟΥ ΡΟΜΠΟΤ LEGO

Πτυχιακή Εργασία
των
Καλαϊτζίδη Φώτη
Ταρσένη Δημήτρη

Επιβλέπουσα: Παπαβραμίδου Παναγιώτα,
Εργαστηριακή Συνεργάτης

Θεσσαλονίκη, Φεβρουάριος 2010

Σε όλα εκείνα τα πρόσωπα που μας στήριξαν και πίστεψαν σε μας …

Πτυχιακή Εργασία των:
Καλαϊτζίδη Φώτη ΚΑΣ: 502103
Ταρσένη Δημήτρη ΚΑΣ: 502073

Επιβλέπουσα:
Παπαβραμίδου Παναγιώτα
Εργαστηριακή Συνεργάτης

Ημερομηνία ανάληψης:
19/09/2008

Ημερομηνία περάτωσης:
26/01/2010

Πρόλογος

Η παρούσα πτυχιακή εργασία πραγματεύεται την ανάπτυξη ενός αυτονόμου ρομπότ Lego και την προσαρμογή ενός αισθητήρα θερμοκρασίας με την χρήση του πρωτοκόλλου I^2C.

Με την ολοκλήρωση της εργασίας, αισθανόμαστε την ανάγκη να ευχαριστήσουμε όλους τους καθηγητές του εργαστηρίου Ψηφιακά Κυκλώματα Ι και ΙΙ, του ΑΤΕΙ Θεσσαλονίκης, του Τμήματος Ηλεκτρονικής, οι οποίοι μας εμπιστεύτηκαν και μας βοήθησαν για την εκπόνηση της.

Περίληψη

Σκοπός της Πτυχιακής Εργασίας είναι ο προγραμματισμός ενός αυτονόμου ρομπότ Lego NXT για την αποφυγή φυσικών εμποδίων και η προσαρμογή ενός αισθητήρα θερμοκρασίας με την χρήση του πρωτοκόλλου I^2C.

Ο αισθητήρας θερμοκρασίας που παρέχεται από την Lego δεν μπορεί να μετρήσει την θερμοκρασία περιβάλλοντος, αλλά μόνο την θερμοκρασία του υλικού με το οποίο έρχεται σε επαφή, όπως για παράδειγμα ζεστό νερό. Ο αισθητήρας θερμοκρασίας που κατασκευάσαμε και προγραμματίσαμε εμείς, δίνει την δυνατότητα στο χρήστη να πειραματιστεί με το ρομπότ σε σχέση με την θερμοκρασία που αντιλαμβάνεται στο περιβάλλον.

Το τελικό αποτέλεσμα της εργασίας είναι ένα ρομπότ που ακολουθεί μια μαύρη γραμμή και αλλάζει πορεία όταν αντιληφθεί κάποιο εμπόδιο. Επίσης καταγράφει την θερμοκρασία του περιβάλλοντος χώρου στην οθόνη LCD του NXT Brick. Αν η θερμοκρασία αυξηθεί πέρα από το όριο που του έχουμε θέσει, το ρομπότ κινείται προς αντίθετη κατεύθυνση.

Η πτυχιακή χωρίζεται σε τέσσερα κεφάλαια. Στην πρώτη ενότητα περιγράφονται αναλυτικά τα τμήματα από τα οποία αποτελείται το ρομπότ ενώ στο δεύτερο, παρουσιάζεται το λογισμικό που χρησιμοποιήθηκε για τον προγραμματισμό του και με ποιους τρόπους επιλύθηκαν τα προβλήματα που παρουσιάστηκαν. Στο τρίτο κεφάλαιο αναλύεται λεπτομερώς η λειτουργία της I^2C επικοινωνίας και στο τέταρτο ο τρόπος με το οποίο προσαρμόσαμε τον αισθητήρα AD7416 στο ρομπότ.

Abstract

Purpose of this thesis is the programming of an independent Lego NXT robot, the construction of a temperature sensor using the I^2C protocol and the familiarity of the I2C Bus for a novice user.

The existing temperature sensor of Lego cannot measure the temperature, but only the temperature of material in contact, such as hot water. The temperature sensor that we have constructed and programmed, allows the user to experiment with the robot in relation to the perceived temperature in the environment.

The final result of the work is a robot that follows a black line and changes course when it sees an obstacle. It also records the temperature of the room on the LCD screen of the NXT Brick. If the temperature rises beyond the limit of our set, the robot moves in the opposite direction.

The project is divided in four chapters. In the first chapter, the separate parts of robot are described in detail, whereas in the second chapter there is a description of the software of Robotc that was used for the programming of the robot. There are also reported the basic functions and points of the program. In the third chapter, I^2C protocol is introduced. Finally, the fourth chapter describes the digital temperature sensor AD7416 and how it was adapted to the robot.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1:

Η ΣΥΝΘΕΣΗ ΤΟΥ LEGO NXT

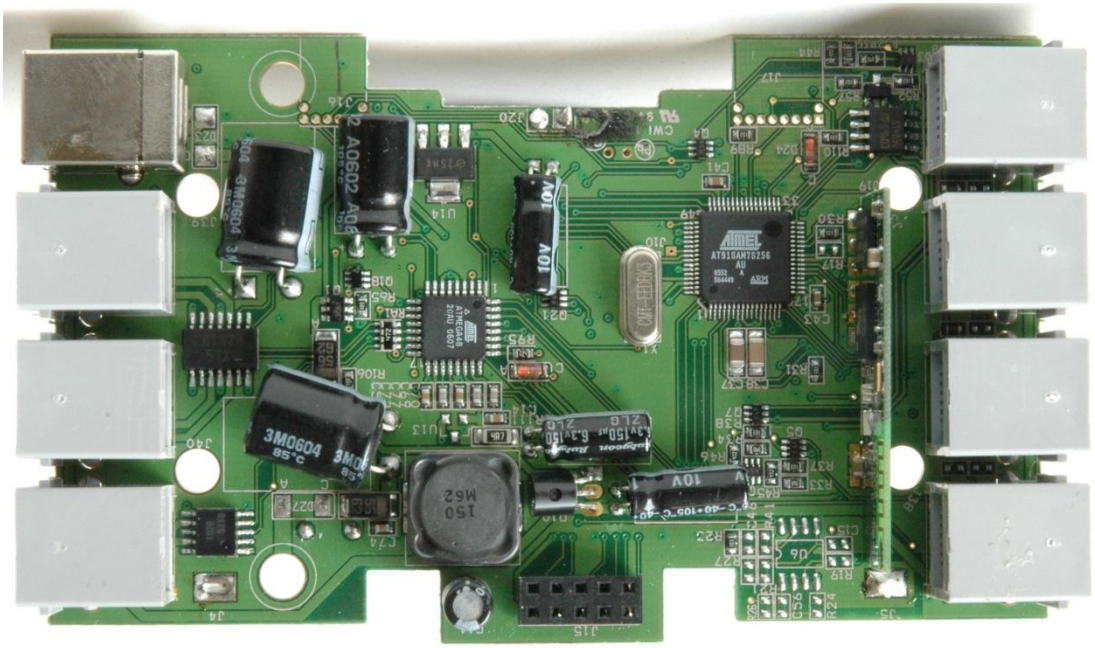
1.1. ΑΠΟ ΤΙ ΑΠΟΤΕΛΕΙΤΑΙ ΤΟ ΡΟΜΠΟΤ LEGO NXT

Το ρομπότ Lego Mindstorms NXT αποτελεί ένα ρομποτικό σετ, το οποίο κατασκευάζεται από την εταιρία Lego. Συγκεκριμένα το NXT αποτελεί την δεύτερη γενιά ρομποτικού σετ, καθώς το πρώτο που δημιουργήθηκε είναι το επιτυχημένο RCX. Το Lego mindstorms NXT αποτελείται από διάφορα «τουβλάκια» της Lego και από κάποια άλλα κομμάτια, τα οποία συμβάλουν στη δημιουργία του ρομπότ. Συγκεκριμένα τα κομμάτια αυτά είναι τα εξής**:** 1)Το κεντρικό τούβλο (NXT Brick), 2)Τα NXT motors, 3)Ο αισθητήρας υπερήχων, 4)Ο αισθητήρας φωτός, 5)Ο αισθητήρας ήχου και 6)Ο αισθητήρας αφής.

Εικόνα 1.1 Το ρομπότ Lego Mindstorms NXT

1.2. ΚΕΝΤΡΙΚΟ ΤΟΥΒΛΟ ΚΑΙ ΤΟ ΕΣΩΤΕΡΙΚΟ ΤΟΥ

Η κεντρική μονάδα του ρομπότ, το αποκαλούμενο κεντρικό τούβλο NXT, είναι ο εγκέφαλος του ρομπότ Lego Mindstorms. Είναι ένα έξυπνο ηλεκτρονικά ελεγχόμενο τούβλο LEGO που επιτρέπει στο ρομπότ να λειτουργεί και να εκτελεί τις διάφορες εργασίες. Είναι η κεντρική μονάδα επεξεργασίας όπου δίνει την λογική στο ρομπότ ώστε να λειτουργεί και αυτόνομα. Πάνω στην K.M.E. συνδέονται όλοι οι αισθητήρες και τα motor.



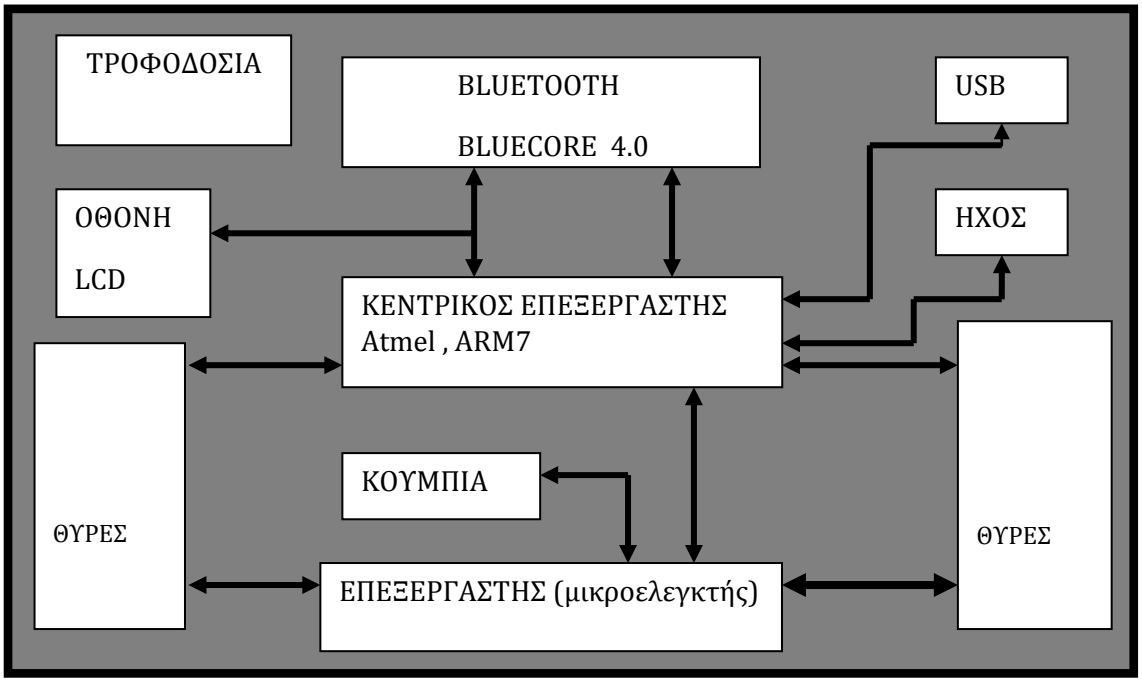
Εικόνα 1.2 Το εσωτερικό του κεντρικού τούβλου (NXT brick).

Το κεντρικό τούβλο NXT αποτελείται από τις εξής λειτουργικές μονάδες:

1) Κεντρικός Επεξεργαστής
2) Βοηθητικός Επεξεργαστής
3) Θύρες αισθητήρων
4) Θύρες κινητήρων
5) Θύρα επικοινωνίας USB
6) Bluetooth ασύρματη επικοινωνία
7) NXT κουμπιά
8) Μεγάφωνο

9) Οθόνη NXT
10) Τροφοδοσία NXT

Το παρακάτω διάγραμμα παρέχει την γραφική απεικόνιση του εσωτερικού του NXT τούβλου και συγκεκριμένα για το πώς όλες αυτές οι λειτουργικές μονάδες που αναφέρθηκαν, συνδέονται και ελέγχονται μέσα στο κεντρικό τούβλο NXT.

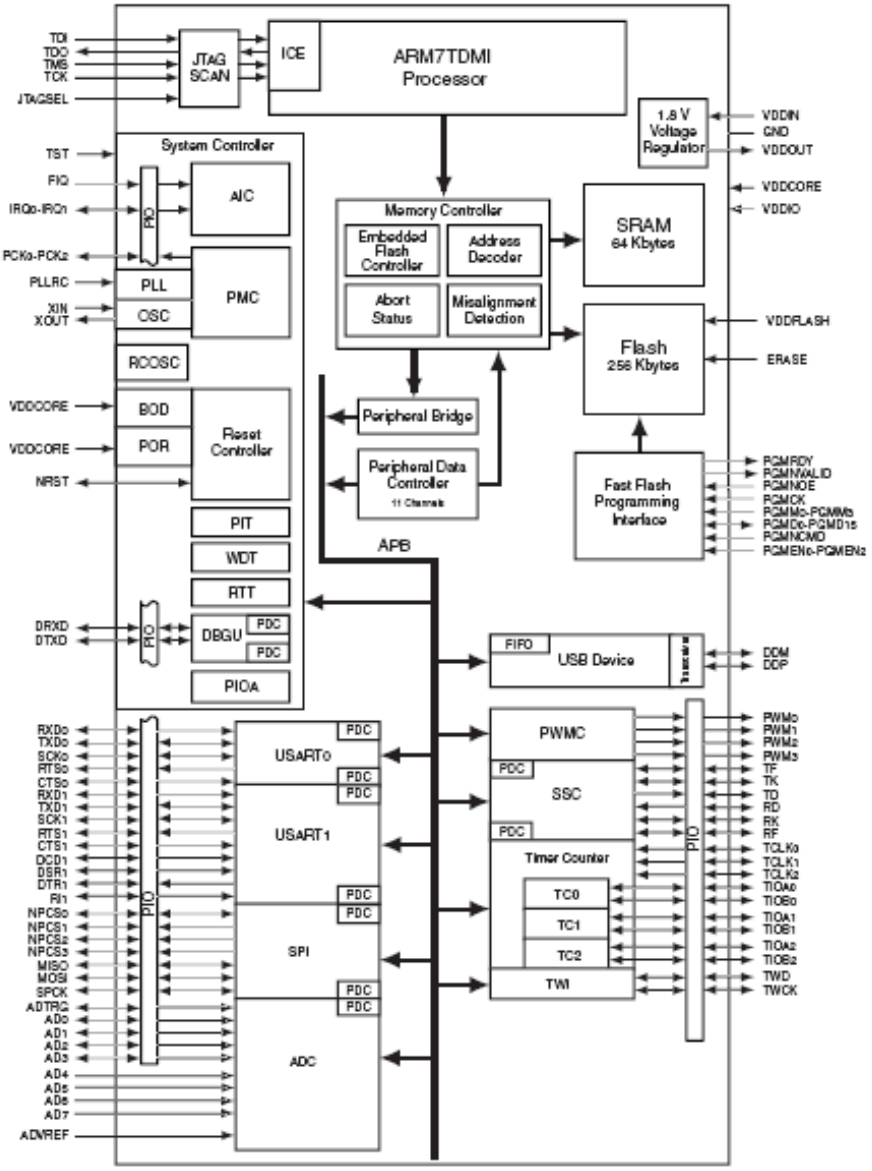


Εικόνα 1.1 Εσωτερικό του κεντρικού τούβλου NXT

Παρακάτω παρουσιάζονται αναλυτικά, οι λειτουργικές μονάδες του κεντρικού τούβλου του NXT.

1.2.1. ΚΕΝΤΡΙΚΟΣ ΕΠΕΞΕΡΓΑΣΤΗΣ

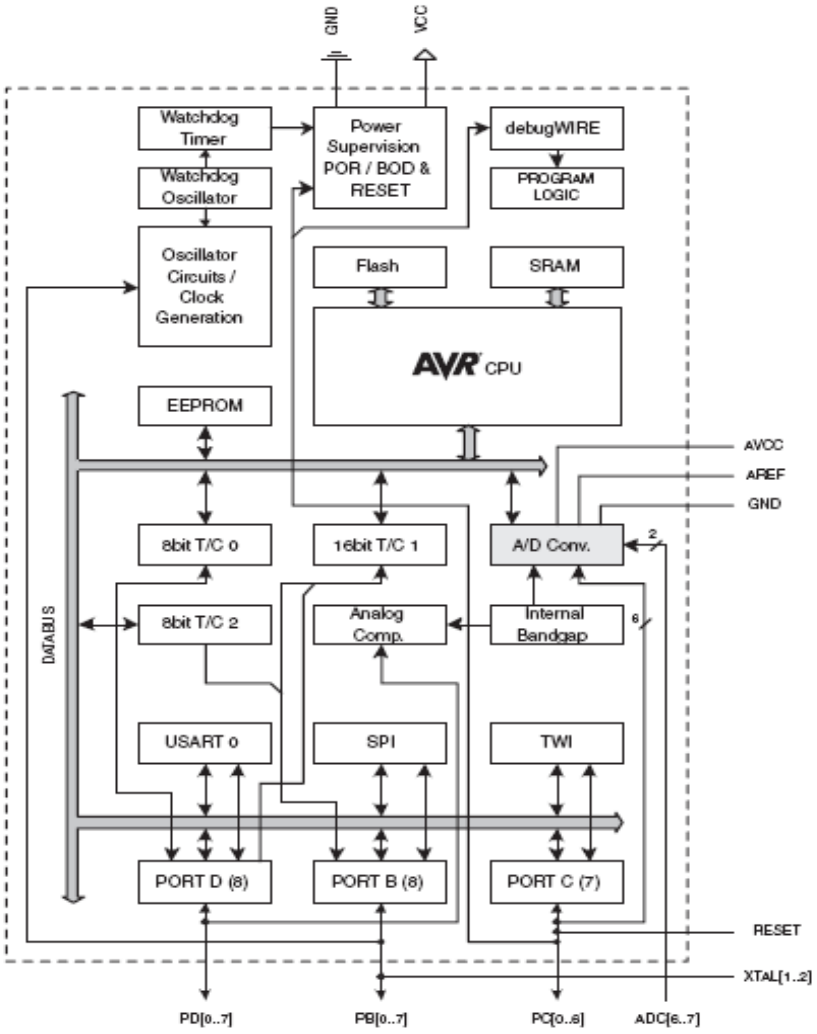
Πρόκειται για τον επεξεργαστή της Atmel 32-bit ARM AT91SAM7S256, με μνήμη προγράμματος ταχείας αποθήκευσης 256 KB FLASH ,με μνήμη δεδομένων στατικής μνήμης 64 KB RAM και ταχύτητα στα 48MHz.



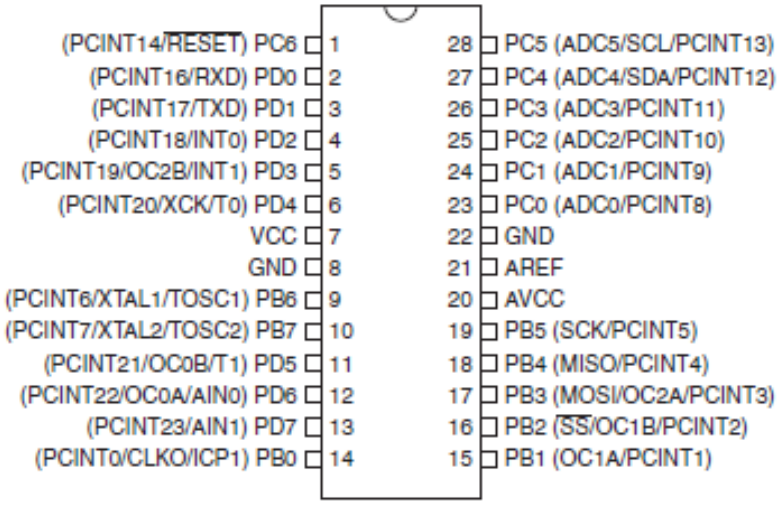
Εικόνα 1.2 Διάγραμμα της αρχιτεκτονικής του AT91SAM7S256

1.2.2. ΒΟΗΘΗΤΙΚΟΣ ΕΠΕΞΕΡΓΑΣΤΗΣ

Είναι ένας επεξεργαστής της Atmel 8-bit AVR, ATmega48, με μνήμη προγράμματος ταχείας αποθήκευσης 4 KB FLASH, μνήμη δεδομένων στατικής μνήμης 512 Bytes RAM, επαναπρογραμματιζόμενη μνήμη μόνο για ανάγνωση 256 Bytes EEPROM και ταχύτητα στα 8MHz.



Εικόνα 1.3 Διάγραμμα της αρχιτεκτονικής του AVR



Εικόνα 1.4 Οι ακροδέκτες του Atmega48

1.2.3. ΘΥΡΕΣ ΑΙΣΘΗΤΗΡΩΝ

Το NXT έχει 4 θύρες εισόδου που βρίσκονται στο κάτω μέρος του κεντρικού τούβλου NXT για τη σύνδεση των αισθητήρων , αναλογικών και ψηφιακών (analog & digital sensors). Αντιστοιχεί στις θύρες 1, 2, 3 και 4.

1.2.4. ΘΥΡΕΣ ΚΙΝΗΤΗΡΩΝ

Το NXT έχει 3 θύρες εξόδου για την τοποθέτηση των κινητήρων (motors) και βρίσκονται στο πάνω μέρος του κεντρικού τούβλου NXT. Αντιστοιχεί στις θύρες A, B και C.

1.2.5. ΘΥΡΑ ΕΠΙΚΟΙΝΩΝΙΑΣ USB

Σύνδεση καλωδίου USB 2.0 στη συγκεκριμένη θύρα πλήρης ταχύτητας (12Mbit/sec), ώστε να υπάρχει η επικοινωνία του ρομπότ με τον υπολογιστή. Χάρη στην θύρα USB υπάρχει η δυνατότητα αποστολής του κώδικα που έχουμε γράψει στην μνήμη του ρομπότ απ' όπου και θα τρέξει το πρόγραμμα μας. Επίσης μπορούμε να ανακτήσουμε ένα αρχείο προγράμματος που έχουμε στείλει στο ρομπότ, αν αυτό για κάποιο λόγο έχει χαθεί από τον Η/Υ μας.

1.2.6. BLUETOOTH ΑΣΥΡΜΑΤΗ ΕΠΙΚΟΙΝΩΝΙΑ

Ένα από τα πιο προηγμένα χαρακτηριστικά για το νέο Mindstorms NXT, είναι η ικανότητά να ελέγχεται και να επικοινωνεί το ρομπότ με τον υπολογιστή, αλλα και των ρομπότ μεταξύ τους, μέσω της τεχνολογίας Bluetooth. Το NXT brick υποστηρίζει την ασύρματη επικοινωνία Bluetooth συμπεριλαμβάνοντας στο εσωτερικό του, το τσιπ CSR bluecore 4, με εξωτερική μνήμη προγράμματος ταχείας αποθήκευσης 8Mbit FLASH και εσωτερική μνήμη δεδομένων 47Kbyte RAM. Αυτό το τσιπ δίνει τη δυνατότητα στο NXT να συνδεθεί ασύρματα με 3 συσκευές ταυτόχρονα, αλλά να μπορεί να επικοινωνήσει μόνο με μια συσκευή κάθε φορά. Αυτή η λειτουργία έχει εφαρμοστεί χρησιμοποιώντας το Serial Port Profile (SPP), το οποίο μπορεί να θεωρηθεί ασύρματος τμηματικός λιμένας. Το τούβλο μπορεί να επικοινωνήσει με τις συσκευές bluetooth, που μπορούν να προγραμματιστούν για να επικοινωνήσουν με τη χρησιμοποίηση του NXT πρωτοκόλλου επικοινωνίας, το οποίο υποστηρίζει το Serial Port Profile (SSP).

1.2.7. NXT ΚΟΥΜΠΙΑ

Στο μπροστινό μέρος του NXT, υπάρχουν τέσσερα κουμπιά**.**

Πορτοκαλί κουμπί: Άνοιγμα On/ Εισαγωγή Enter/ Εκτέλεση Run.

Σκούρο γκρι κουμπί: Μηδενισμός clear/ Επιστροφή Go back.

Φωτεινά γκρι δεξιά και αριστερά κουμπιά: Χρησιμοποιούνται για την μετακίνηση δεξιά και αριστερά στο μενού του NXT.

1.2.8. ΜΕΓΑΦΩΝΟ (LOUDSPEAKER)

Το NXT brick περιέχει τσιπ ενισχυτή ήχου για την βελτίωση της ποιότητας και επιπέδου του ήχου. Η έξοδος του ήχου ελέγχεται από τον μικροελεγκτή ARM7. Τα φίλτρα εισάγονται πριν ο ενισχυτής μειώσει τον θόρυβο στο σήμα. Το μεγάφωνο του NXT είναι 16 ohm μεγάφωνο με διάμετρο 21mm. Το μεγάφωνο του NXT παραδίδει ποιότητα ήχου στα 8KHZ και κανάλια ήχου με 8-bit ανάλυση και 2 – 16 KHZ δειγματοληψία. Ο παρακάτω πίνακας δείχνει την ροή του ρεύματος και την κατανάλωση ισχύος όταν ο ήχος παίζεται σε δυο διαφορετικές συχνότητες.

Συχνότητα	Ροή ρεύματος (mA)	Κατανάλωση ισχύος (mW)
440Hz	102	169
4KHz	78	97

Πίνακας 1.1 Ροή ρεύματος και ισχύς που καταναλώνεται.

1.2.9. ΟΘΟΝΗ NXT

Ένας πίνακας απεικόνισης έχει προστεθεί στο κεντρικό τούβλο NXT για να διευκολύνεται η χρήση του ρομπότ. Η απεικόνιση αυτή, είναι μια ασπρόμαυρη LCD γραφική απεικόνιση με ανάλυση 100x64 pixels. Οι διαστάσεις της LCD οθόνης είναι 26x40.6mm και ο ελεγκτής της οθόνης είναι ο ultrachip 1601. Τα δεδομένα της οθόνης περιέχονται στην μνήμη ως μια περιοχή δυο τιμών (2 byte). Συγκεκριμένα τα δεδομένα στέλνονται στον ελεγκτή της LCD οθόνης με τον εξής τρόπο: Το

πρώτο byte ελέγχει τα πρώτα 8 pixels κάθετα και το δεύτερο byte ελέγχει τα επόμενα 8 pixels οριζοντίως.

Τεχνικές προδιαγραφές οθόνης:

1. Διάταξη οθόνης:	100x64pixels
2. Τάση τροφοδοσίας (VDD):	3.0Volt
3. Διαστάσεις οθόνης:	26x40.6mm
4. Οδηγός γραφικού περιγράμματος:	1/65 κύκλοι εργασίας, 1/9 διαγώνια γραμμή.
5. LCD Οδηγός τάσης(VLCD):	9.0 Volt
6. Οπτική κατεύθυνση:	6 o'clock

1.2.10. ΤΡΟΦΟΔΟΣΙΑ NXT

Η τροφοδοσία του NXT ρομπότ παρέχεται με δύο εναλλακτικούς τρόπους:

1) Με 6 αλκαλικές μπαταρίες AA (1.5V η κάθε μια).
2) Με επαναφορτιζόμενη ιονίζουσα λιθίου μπαταρία 7.4 V/1400 mAh.Για την φόρτιση αυτής της μπαταρίας χρησιμοποιείται μετασχηματιστής, με τάση εισόδου 230V AC στα 50Hz και εναλλασόμενη έξοδο AC στα 10V/700mA.

Εικόνα 1.3 Η επαναφορτιζόμενη ιόντων λιθίου μπαταρία 7.4 V/1400 mAh

Η χρήση της συγκεκριμένης μπαταριάς, πλεονεκτεί σε σχέση με την χρήση των 6AA μπαταριών, καθώς αυτές τελειώνουν πιο γρήγορα και γενικά όσον αφορά από

οικονομικής άποψης δεν συμφέρουν συγκριτικά με την επαναφορτιζόμενη μπαταρία λιθίου. Επιπλέον δεν απαιτείται πολύ ώρα για την επαναφόρτιση της. Συγκεκριμένα σε 4 ώρες φορτίζει πλήρως και δεδομένου ότι είναι μπαταρία λιθίου μπορεί να φορτίσει μερικώς ή και να φορτίζεται για πολλές ώρες χωρίς να υπάρχουν αρνητικά αποτελέσματα. Η μπαταρία αυτή εμπεριέχεται στην βασική έκδοση LEGO MINDSTORMS EDUCATION NXT ενώ ο φορτιστής της έρχεται κατόπιν παραγγελίας από τον αντιπρόσωπο της LEGO στην Ελλάδα.

1.3. ΟΙ ΑΙΣΘΗΤΗΡΕΣ

Το ρομπότ NXT lego mindstorms, αποτελείται από τέσσερις αισθητήρες (αναλογικούς και ψηφιακούς), οι οποίοι είναι οι εξής: α) αισθητήρας φωτός, β) αισθητήρας ήχου, γ)αισθητήρας αφής και δ) αισθητήρας υπερήχων και συνδέονται στις τέσσερις θύρες εισόδου του κεντρικού τούβλου NXT. Επιπλέον οι αναλογικοί αισθητήρες του NXT χωρίζονται σε 2 υποκατηγορίες: i) τους ενεργητικούς, αυτός ο τύπος αισθητήρα απαιτεί συνεχή παροχή ενέργειας ώστε να μετρηθεί το επίπεδο του αισθητήρα. Υπάρχει επίσης η δυνατότητα μέτρησης χρόνου. ii) τους παθητικούς, στους οποίους, δεν απαιτείται καμία μέτρηση του χρόνου. Στο NXT ρομπότ δεν υπάρχουν ενεργητικοί αισθητήρες σε σύγκριση με το προηγούμενο ρομπότ πρώτης γενιάς, όπου είχε και ήταν ο αισθητήρας φωτός και ο αισθητήρας περιστροφής. Αντίθετα στο NXT υπάρχουν 3 παθητικοί αισθητήρες: ο αισθητήρας φωτός, ο αισθητήρας αφής και ο αισθητήρας ήχου. Ο αισθητήρας υπερήχων ανήκει στην κατηγορία των ψηφιακών αισθητήρων. Παρακάτω παρουσιάζεται αναλυτικά η περιγραφή όλων των αισθητήρων που χρησιμοποιούνται στο NXT ρομπότ.

1.3.1. ΑΙΣΘΗΤΗΡΑΣ ΦΩΤΟΣ

Ο αισθητήρας φωτός είναι ένας από τους δύο αισθητήρες που δίνουν όραση στο ρομπότ ενώ ο άλλος είναι ο αισθητήρας υπερήχων. Ο αισθητήρας φωτός επιτρέπει στο ρομπότ να διακρίνει μεταξύ φωτός και σκοταδιού. Μπορεί να αναγνωρίσει την ένταση του φωτός σε ένα δωμάτιο ή να μετρήσει την ένταση φωτός των χρωματισμένων επιφανειών και να αναγνωρίσει το χρώμα της επιφάνειας. Είναι ένας πιο ευαίσθητος αισθητήρας, επιτρέποντας περισσότερη ακρίβεια στην

μέτρηση φωτός κατά μήκος μιας κλίμακας, από 0 (κανένα φως) έως 100 (πολύ φωτεινός). Είναι επίσης δυνατό να κληθεί η πηγή υπέρυθρου φωτός που βρίσκεται κάτω από τον αισθητήρα, έτσι ώστε ο αισθητήρας να μετρά μόνο το περιβαλλοντικό φως από τα περίχωρά του.

Εικόνα 1.4 Αισθητήρας φωτός.

1.3.2. ΑΙΣΘΗΤΗΡΑΣ ΉΧΟΥ

Ο αισθητήρας ήχου δίνει στο ρομπότ την δυνατότητα να αντιλαμβάνεται ήχους και να ανταποκρίνεται σε αυτούς. Ο αισθητήρας ήχου έχει ένα μικρόφωνο και μπορεί να χρησιμοποιηθεί για να ανιχνεύσει το εύρος ενός ηχητικού σήματος (ένταση του ήχου). Ο αισθητήρας ήχου μπορεί να ανιχνεύσει τόσο ντεσιμπέλ[db] όσο και προσαρμοσμένα ντεσιμπέλ [dBA]. Ένα ντεσιμπέλ είναι μια μονάδα μέτρησης της ηχητικής πίεσης. Αναλυτικά για την ανίχνευση των δυο συγκεκριμένων μεγεθών έχουμε τα εξής: α)dBA για ανίχνευση προσαρμοσμένων ντεσιμπέλ, η ευαισθησία του αισθητήρα είναι προσαρμοσμένη στην ευαισθησία του ανθρώπινου αυτιού. Με άλλα λόγια, αυτοί είναι οι ήχοι που το ανθρώπινο αυτί μπορεί να ακούσει. β) dB, για ανίχνευση πρότυπων (αδιόρθωτων) ντεσιμπέλ, όλοι οι ήχοι μετρούνται με την ίδια ευαισθησία. Έτσι, οι ήχοι μπορεί να περιλαμβάνουν ορισμένα κομμάτια που είναι πολύ υψηλά ή πολύ χαμηλά για να τα ακούσει το ανθρώπινο αυτί. Ο αισθητήρας ήχου μπορεί να μέτρα στάθμη ηχητικής πίεσης έως 90db. Οι στάθμες ηχητικής πίεσης είναι εξαιρετικά περίπλοκες, έτσι η τιμή του αισθητήρα ήχου στο NXT εμφανίζεται σε ποσοστό επί τοις εκατό. Όσο χαμηλότερο είναι το ποσοστό, τόσο πιο χαμηλής έντασης είναι ο ήχος.

Για παράδειγμα:

- 4-5% είναι όπως ένα σιωπηλό δωμάτιο.
- 5-10% κάποιος που μιλάει από κάποια απόσταση.
- 10-30% είναι μια φυσιολογική συνομιλία κοντά στον αισθητήρα ή μουσική σε κανονική έντασης.
- 30-100% είναι άτομα που φωνάζουν ή μουσική που παίζεται σε μεγάλη ένταση.

Έτσι με την κατάλληλη χρήση του αισθητήρα ήχου μπορεί ο χρήστης να προγραμματίσει το ρομπότ να κάνει διάφορες κινήσεις, εφόσον ο αισθητήρας αντιληφθεί κάποιο ηχητικό σήμα.

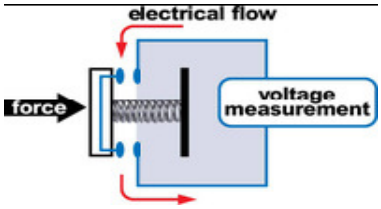
Εικόνα 1.5 Αισθητήρας Ήχου.

1.3.3. ΑΙΣΘΗΤΗΡΑΣ ΑΦΗΣ

Ο αισθητήρας αφής δίνει στο ρομπότ την αίσθηση της αφής. Ο αισθητήρας αφής ανιχνεύει όταν αυτός πιέζεται από κάτι άλλο και όταν απελευθερωθεί. Ο νέος αισθητήρας αφής έχει μια τρύπα πάνω στο πιεζόμενο κουμπί του αισθητήρα, που δίνει στους χρήστες περισσότερες επιλογές για τον αισθητήρα στα σχέδιά τους.

Εικόνα 1.6 Αισθητήρας Αφής

Οι νέοι και οι υπάρχοντες αισθητήρες αφής είναι πολύ βασικοί αισθητήρες. Έτσι μπορούν να είναι σε λειτουργία (επιτρέποντας στην ηλεκτρική ροή για να κινηθεί ανεμπόδιστο το ρομπότ) ή εκτός λειτουργίας (διακόπτοντας την ηλεκτρική ροή). Μπορούμε να δούμε πώς η ηλεκτρική ροή εισέρχεται ή διακόπτεται στο παρακάτω σχήμα. Ένα ρομπότ μπορεί να αρχίσει μια δράση ή μια αντίδραση όταν αυτό αντιληφθεί μια αλλαγή στην τάση λόγω της ηλεκτρικής ροής που είτε εισέρχεται στο σύστημα, είτε που δεν εισέρχεται.



Εικόνα 1.5 Ηλεκτρική εισροή στο κύκλωμα του αισθητήρα**.**

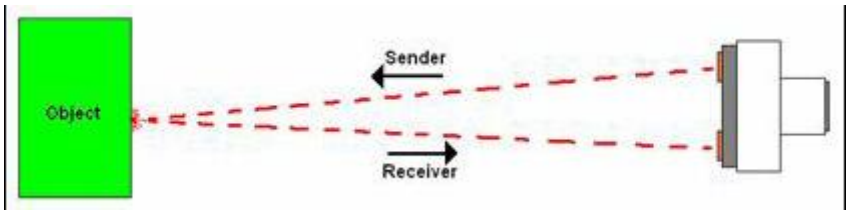
Έτσι ο χρήστης μπορεί να χρησιμοποιήσει τον αισθητήρας αφής για να κάνει το ρομπότ να συλλέγει πράγματα: ένας ρομποτικός βραχίονας εφοδιασμένος με τον αισθητήρας αφής κάνει το ρομπότ να γνωρίζει αν υπάρχει ή δεν υπάρχει κάτι στο χέρι για να το αρπάξει. Επιπλέον μπορεί να χρησιμοποιηθεί ο αισθητήρα αφής στο ρομπότ για να αποφασίσει με μια εντολή. Για παράδειγμα, πιέζοντας τον αισθητήρα αφής μπορεί το ρομπότ να περπατά ή και να μιλά με κατάλληλο προγραμματισμό.

1.3.4. ΑΙΣΘΗΤΗΡΑΣ ΥΠΕΡΗΧΩΝ

Όπως αναφέραμε παραπάνω ο αισθητήρας υπερήχων είναι ο δεύτερος κατά σειρά αισθητήρας που δίνει όραση στο ρομπότ Έτσι επιτρέπει στο ρομπότ να βλέπει και να εντοπίζει αντικείμενα χρησιμοποιώντας τις υπερηχητικές μετρήσεις. Μπορεί επίσης να χρησιμοποιηθεί από το χρήστη για να κάνει το ρομπότ να αποφεύγει τα εμπόδια, να καταλαβαίνει και να μετρά απόσταση και να ανιχνεύει κίνηση. Επίσης μέτρα απόσταση σε εκατοστά και σε ίντσες. Είναι σε θέση να μετρά αποστάσεις από 0 έως 255 εκατοστά με ακρίβεια +/- 3 εκατοστά.

Εικόνα 1.7 Αισθητήρας Υπερήχων

Επιπλέον χρησιμοποιεί την εξής επιστημονική αρχή: μετρά την απόσταση από τον υπολογισμό του χρόνου που χρειάζεται ένα ηχητικό κύμα να χτυπήσει ένα αντικείμενο και να επιστρέψει πίσω (όπως για παράδειγμα η ηχώ). Η καλύτερη ανίχνευση του σήματος, θα προέλθει από τα αντικείμενα που είναι άμεσα μπροστά από τον αισθητήρα. Αντικείμενα μεγάλου μεγέθους με σκληρές επιφάνειες επιστρέφουν τις καλύτερες αναγνώσεις. Αντιθέτως αντικείμενα που αποτελούνται από μαλακό ύφασμα, που έχουν κάποιου είδους κυρτότητα όπως μια μπάλα, που είναι πολύ λεπτά ή μικρά, δεν ανιχνεύονται εύκολα από τον αισθητήρα. ***Σημειώνουμε ότι δυο ή και περισσότεροι αισθητήρες υπερήχων που λειτουργούν στον ίδιο χώρο, μπορεί να διακόπτουν ο ένας τις αναγνώσεις του άλλου.***



Εικόνα 1.8 Ανίχνευση σήματος από τον αισθητήρας υπερήχων

1.4. ΟΙ ΣΕΡΒΟΚΙΝΗΤΗΡΕΣ

Το ρομποτικό σετ αποτελείται από τρεις σερβοκινητήρες(motors), οι οποίοι συνδέονται στις τρείς θύρες εξόδου (A, B και C) του ρομπότ και οι οποίοι δίνουν στο ρομπότ την ικανότητα να κινηθεί. Οι τρεις σερβοκινητήρες έχουν την

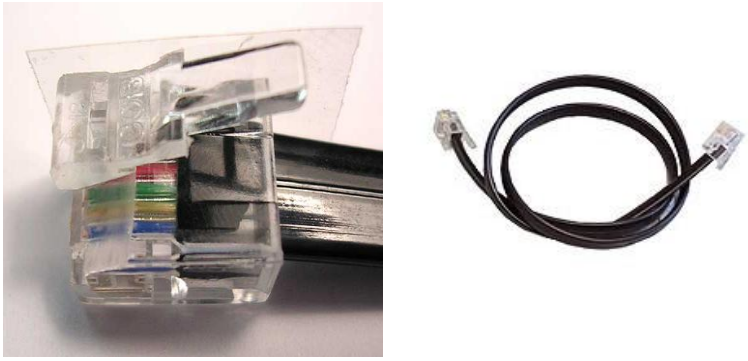
δυνατότητα να μεταβάλλουν την ταχύτητα και τη δύναμη περιστροφής τους όπως και άλλες επιλογές, οι οποίες αφήνουν το χρήστη να τα χρησιμοποιήσει όπως επιθυμεί. Κάθε σερβοκινητήρας έχει έναν ενσωματωμένο αισθητήρα περιστροφής. Αυτός επιτρέπει τον έλεγχο των κινήσεων του ρομπότ με ακρίβεια. Ο αισθητήρας περιστροφής μετρά την περιστροφή του κινητήρα σε μοίρες ή πλήρης περιστροφές (ακρίβεια +/- μία μοίρα). Μία περιστροφή είναι ίση με 360 μοίρες, οπότε αν ορίσουμε την περιστροφή του κινητήρα σε 180 μοίρες, ο άξονας εξόδου θα κάνει μισή στροφή. Επίσης ο ενσωματωμένος αισθητήρας περιστροφής σε κάθε κινητήρα, επιτρέπει να ορίσουμε διαφορετικές ταχύτητες για τους κινητήρες θέτοντας διαφορετικές παραμέτρους ισχύος στο λογισμικό.

Εικόνα 1.9 Οι σερβοκινητήρες(motors) του ρομπότ

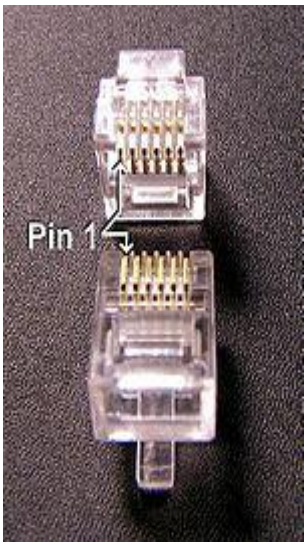
Έναντι των παλαιών 9ν τετραγωνικών μηχανών που περιλαμβάνονται στο αρχικό σύστημα εφευρέσεων ρομποτικής, οι μηχανές των NXT είναι πολύ πιο αργές, αλλά αποζημιώνουν την ταχύτητά τους στο ποσοστό της δύναμης που κρατούν. Ο κάθε σερβοκινητήρας, κάτω από καμία επιβάρυνση τρέχει σε 170 περιστροφές/λεπτό έναντι της παλαιάς μηχανής που έτρεχε σε 360 περιστροφές/λεπτό. Η διαφορά εντούτοις είναι στην ένταση ρεύματος που συσσωρεύεται από τις μηχανές. Ο σερβοκινητήρας του NXT συσσωρεύει 60μA της δύναμης, ενώ η παλαιά μηχανή συσσωρεύει 3.5μA. Αυτή η αυξανόμενη δύναμη συσσωρεύεται στη συνέχεια και παρέχει περισσότερη ροπή. Οι σερβοκινητήρες χάνουν ταχύτητα όταν επιβαρυνθούν με φορτίο 50 Ν/εκατοστό. Στο χρονοτριβώντας σημείο αυτό, ο σερβοκινητήρας συσσωρεύει 2Α του ρεύματος. Οι πιο αργές ταχύτητες, είναι το αποτέλεσμα ενός εσωτερικού εξαρτήματος που παρέχει την πρόσθετη ροπή.

1.5. ΚΑΛΩΔΙΟ NXT

Οι αισθητήρες και τα motors του NXT συνδέονται στο κεντρικό τούβλο μέσω ενός καλωδίου με 6-pin, με αναλογικές και ψηφιακές διεπαφές. Ο τύπος του συγκεκριμένου καλωδίου είναι ο RJ12 (Registered jack12). Η λειτουργία των καλωδίων εξαρτάται από το αν χρησιμοποιούνται για την εισαγωγή των αισθητήρων ή σερβοκινητήρων. Επίσης η λειτουργία τους εξαρτάται και από το ποιός τύπος αισθητήρα συνδέεται στο NXT (αναλογικός ή ψηφιακός). Η αναλογική διεπαφή είναι συμβατή (με την χρήση ενός προσαρμογέα) με την παλαιότερη έκδοση RCX. Η ψηφιακή διεπαφή υποστηρίζει τόσο την I^2C επικοινωνία όσο και την επικοινωνία RS-485.



Εικόνα 1.10 Καλώδιο RJ12

Pin	Name	Function	Color	Pin Numbering
1	ANA	Analog interface, +9V Supply	white	
2	GND	Ground	black	
3	GND	Ground	red	
4	IPOWERA	+4.3V Supply	green	
5	DIGIAI0	I^2C Clock (SCL), RS-485 A	yellow	
6	DIGIAI1	I^2C Data (SDA), RS-485 B	blue	

Πίνακας 1.2 Αντιστοίχιση χρωμάτων, ονομάτων λειτουργιών και θέσεων του NXT καλωδίου.

ΚΕΦΑΛΑΙΟ 2:

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΟΥ ΡΟΜΠΟΤ

2.1. ΤΑ ΛΟΓΙΣΜΙΚΑ ΚΑΙ ΟΙ ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Το ρομπότ της Lego NXT μπορεί να προγραμματιστεί με πολλές από τις διαθέσιμες γλώσσες προγραμματισμού αλλά και από μια μεγάλη γκάμα λογισμικών που το περιβάλλουν. Το κάθε λογισμικό που κυκλοφορεί χρησιμοποιεί την δική του γλώσσα προγραμματισμού και έχει το δικό του γραφικό περιβάλλον. Αρκετά από τα προγράμματα που κυκλοφορούν διατίθενται σε ελεύθερη έκδοση, ενώ κάποια άλλα χρειάζεται να προβείτε στην αγορά τους για να μπορέσετε να προγραμματίσετε το ρομπότ. Η πιο εύκολη από αυτές τις γλώσσες είναι παραγωγής της ίδιας της Lego σε συνεργασία με την NI[1] που έχει στην παραγωγή της το LabVIEW, αποτελείται μόνο από γραφικό περιβάλλον και απευθύνεται κυρίως σε παιδιά από 12 ετών και άνω. Η NI και η Lego ξεκίνησαν να συνεργάζονται το 1998 αναπτύσσοντας το ROBOLAB, το λογισμικό αυτό δημιουργήθηκε για το πρώτο LEGO MINDSTORMS RCX, με περιβάλλον για παιδιά του γυμνασίου και του λυκείου. Το ROBOLAB που βασίζεται στο NI LabVIEW, έχει βοηθήσει την LEGO MINDSTORMS να γίνει το κύριο εργαλείο εκμάθησης και εφευρέσεων ρομποτικής για όλους τους εκπαιδευτικούς παγκοσμίως. Στους πίνακες 2.1.1. και 2.1.2. φαίνονται τα λογισμικά και οι γλώσσες προγραμματισμού που είναι διαθέσιμα στο εμπόριο. Με κόκκινα γράμματα διακρίνεται το λογισμικό της RobotC που επιλέξαμε εμείς για τον προγραμματισμό του ρομπότ μας.

[1] National Instruments

Program	NXT-G retail	NXT-G Educational	RoboLab 2.9	NBC	NXC
Language	Graphic	Graphic	Graphic	Assembly	(Not exactly) C

Πίνακας 2.1 Λογισμικά και γλώσσες προγραμματισμού του Lego NXT.

Program	RobotC	NI LabVIEW Toolkit	leJOS NXJ	pbLua	nxtOSEK
Language	C	Graphic	Java	Lua	ANSI-C/ C++

Πίνακας 2.2 Λογισμικά και γλώσσες προγραμματισμού του Lego NXT.

Όπως φαίνεται υπάρχουν αρκετά λογισμικά που δημιουργήθηκαν στηριζόμενα σε υπάρχουσες γλώσσες προγραμματισμού. Κάποιος μπορεί να επιλέξει συμφώνα με τις γνώσεις του, το αντίστοιχο λογισμικό. Επίσης διαρκώς εμφανίζονται νέα λογισμικά που παρέχουν περισσότερες δυνατότητες προγραμματισμού. Μέχρι στιγμής έχουμε αναφέρει μόνο το RoboLab που ήταν το πρώτο στάδιο για την εξέλιξη της εκπαιδευτικής μάθησης. Παρακάτω θα αναφερθούμε περιληπτικά στα περισσότερα λογισμικά εκτός της RobotC που θα αναλυθεί μόνη της λεπτομερώς.

1) NXT-G

Το λογισμικό NXT-G (retail και Educational) βασίζεται στη χρήση εικονιδίων και είναι μια εκπαιδευτική έκδοση του επαγγελματικού λογισμικού LabVIEW της NI, λογισμικό που χρησιμοποιούν παγκοσμίως επιστήμονες και μηχανικοί, προκειμένου να σχεδιάσουν, να ελέγξουν και να δοκιμάσουν προϊόντα και συστήματα. Το λογισμικό έχει μια διαισθητική διεπαφή “σύρε και άφησε” (drag and drop) και ένα γραφικό προγραμματιστικό περιβάλλον, το οποίο καθιστά την εφαρμογή προσιτή για έναν αρχάριο, αλλά και εξίσου δυναμική για έναν εξειδικευμένο χρήστη.

2) RoboLab 2.9

Το RoboLab ήταν η βασική έκδοση λογισμικού για το RCX μοντέλο, βασιζόταν σε γραφική διεπαφή που αναπτύχτηκε στο Tufts University χρησιμοποιώντας ως μηχανή το NI LabVIEW.

3) NBC

Το Next Byte Codes (NBC) είναι ένα απλό λογισμικό που βασίζεται στην γλώσσα προγραμματισμού Assembly. Η Assembly είναι μια εύχρηστη για τους ανθρώπους μορφή γλώσσας προγραμματισμού σε σχέση με τη γλώσσα μηχανής που χρησιμοποιεί μια συγκεκριμένη αρχιτεκτονική υπολογιστή. Ένα πρόγραμμα σε γλώσσα μηχανής είναι ένα μοτίβο από bits που κωδικοποιούνται σε εντολές του επεξεργαστή. Αυτό γίνεται πιο ευανάγνωστο αντικαθιστώντας τις ακολουθίες των bits με μνημονικά σύμβολα.

4) NXC

Το Not eXactly C (NXC) είναι ένα λογισμικό που απευθύνετε σε ανθρώπους που έχουν πολύ καλές γνώσεις στον προγραμματισμό. Το NXC βασίζεται σε παρόμοια γλώσσα προγραμματισμού με την C και λειτουργεί με την βοήθεια του μεταγλωττιστή της NBC. Όπως και οι υπόλοιπες μπορεί να χρησιμοποιηθεί για να προγραμματίσουμε το NXT Brick. Στην ουσία το NXC είναι ο ανάδοχος του NQC για το NXT.

5) leJOS

Το LEGO Java Operating System (LeJos) είναι μια γλώσσα προγραμματισμού βασισμένη σε JAVA που χρησιμοποιείται για να προγραμματίζουμε το NXT και το παλαιότερο RCX. Από τη στιγμή που η JAVA είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού, έτσι και η LeJos μας προσφέρει όλα τα πλεονεκτήματα του αντικειμενοστραφούς προγραμματισμού. Το LeJOS αναπτύχθηκε εξ' ολοκλήρου από τον Jose Solorzano, αλλά τώρα συντηρείται από τους Paul Andrews και Jürgen Stuber. Το γεγονός ότι το leJOS είναι ένα καινούργιο σχετικά λογισμικό για το NXT, σημαίνει ότι μπορούμε να κάνουμε τα προγράμματα μας πιο εξελιγμένα

από ότι ήταν δυνατό χρησιμοποιώντας είτε το προγραμματιστικό περιβάλλον της LEGO (NXT-G) είτε το NQC (Not Quite C).

6) nxtOSEK

Το nxtOSEK είναι ένα λογισμικό που λειτουργεί σε πραγματικό χρόνο (Real-Time Operating System) για το προγραμματισμό του NXT. Το nxtOSEK αποτελείται από το I/O οδηγό που είναι μέρος του leJOS NXJ, TOPPERS OSEK RTOS και έναν κώδικα που ενώνει τα προηγούμενα μεταξύ τους. Επίσης το nxtOSEK επιτρέπει τον προγραμματισμό του NXT με ANSI-C/C++.

7) pbLua

Η pbLua είναι μια εντελώς διαφορετική γλώσσα προγραμματισμού που διαθέτει όμως ένα πολύ καλό προτέρημα. Στην Lua μπορούμε να συμπεριλάβουμε στην βιβλιοθήκη της και άλλες βιβλιοθήκες που είναι γραμμένες σε άλλες γλώσσες και αυτή να της μεταγλωττίσει ώστε να μπορέσουμε να τις χρησιμοποιήσουμε.

2.2. ΤΟ ΛΟΓΙΣΜΙΚΟ ROBOTC

Όπως αναφέραμε σε προηγούμενη ενότητα, το λογισμικό που επιλέξαμε για να προγραμματίσουμε το ρομπότ μας είναι η RobotC. Η RobotC είναι ένα λογισμικό για την εκπαιδευτική ρομποτική. Χρησιμοποιεί την C γλώσσα προγραμματισμού και έχει ένα εύκολο στην χρήση του περιβάλλον. Υποστηρίζει πολλές διαφορετικές πλατφόρμες όπως είναι η LEGO MINDSTORMS NXT και η FIRST VEX. Δίνει την δυνατότητα γρήγορου Download του High Performance Firmware για το ρομπότ. Όπως επίσης διαθέτει έναν Interactive Run-Time Debugger, με τον οποίο εντοπίζονται εύκολα και γρήγορα τα λάθη του προγράμματος. Αλλά παράλληλα είναι ένα πολύ ισχυρό λογισμικό για γνώστες της C γλώσσας προγραμματισμού, που θέλουν να την χρησιμοποιήσουν σε πιο πολύπλοκες κατασκευές.

2.2.1. ΟΙ ΛΟΓΟΙ ΓΙΑ ΤΟΥΣ ΟΠΟΙΟΥΣ ΕΠΙΛΕΞΑΜΕ ΤΗΝ ROBOTC.

1. Ο βασικός λόγος ήταν ότι η RobotC χρησιμοποιεί την C ως γλώσσα προγραμματισμού.
2. Με την RobotC ο χρήστης έχει την δυνατότητα να δημιουργήσει τις δικές του συναρτήσεις ανάλογα με τις ανάγκες του. Στην παρούσα περίπτωση δημιουργήσαμε την συνάρτηση με την οποία λειτουργεί ο αισθητήρας θερμοκρασίας AD7416.
3. Διαθέτει μια μεγάλη γκάμα έτοιμων εντολών για τους υπάρχοντες αισθητήρες και σερβοκινητήρες που περιέχονται σε μια Education συσκευασία του NXT και
4. Περιέχει στο Οδηγό Βοήθειας μια πληθώρα παραδειγμάτων για να ξεκινήσει κανείς να προγραμματίζει το ρομπότ και πολλά sample programs τα οποία διευκολύνουν στην εξοικείωση με την RobotC.

2.3 ΤΟ FIRMWARE ΤΟΥ ΡΟΜΠΟΤ

Το NXT τούβλο περιέχει το firmware του (ένα κομμάτι του λογισμικού που ενσωματώνεται στο hardware), το οποίο μπορεί να θεωρηθεί ως λειτουργικό σύστημα του NXT. Δεδομένου ότι αποθηκεύεται στη flash μνήμη, δεν θα διαγραφεί εάν σβήσουμε το NXT ή αφαιρέσουμε τις μπαταρίες. Το πρωταρχικό firmware συνοδεύεται από ένα CD και πρέπει να κατεβεί από τον Η/Υ στο NXT τουλάχιστον μια φορά. Εντούτοις μπορούμε να το σβήσουμε ή να το ξαναφορτώσουμε όσο συχνά επιθυμούμε, στην περίπτωση που υπάρχουν updated εκδόσεις firmware διαθέσιμες. Οι νέες εκδόσεις firmware κυκλοφορούν συχνά. Η επίσημη έκδοση βρίσκεται για δωρεάν κατέβασμα στο official site της Lego. Μπορούμε ακόμη και να το αντικαταστήσουμε με άλλα πιο κατάλληλα firmware, ανάλογα με τις ανάγκες μας. Τα firmware αυτά προσφέρουν: α) καλύτερη απόδοση, β) ιδιαίτερα χαρακτηριστικά γνωρίσματα, γ) υποστήριξη επιπλέων γλωσσών προγραμματισμού. Η RobotC για παράδειγμα που είναι ένα από τα πιο πρόσφατα λογισμικά προγραμματισμού για το NXT, παράγει το δικό της firmware το οποίο εγκαθιστούμε στο ρομπότ με την πρώτη εκκίνηση του λογισμικού για να μπορεί να αναγνωρίζει τις συναρτήσεις της RobotC. Αυτό το firmware πρέπει να κατεβεί από τον υπολογιστή στο NXT προτού προβούμε στην δημιουργία ενός κώδικα για να

προγραμματίσουμε το ρομπότ. Επίσης κρίνουμε σκόπιμο να αναφέρουμε πως η Lego έχει απελευθερώσει το firmware για το NXT Brick κατατάσσοντας το στα software ως “Open Source”. Έχοντας λοιπόν το firmware με την δυνατότητα του “Λογισμικού Ανοικτού Κώδικα”, ο καθένας μπορεί ελεύθερα να το χρησιμοποιεί, να το αντιγράφει, να το διανέμει και να το τροποποιεί ανάλογα με τις ανάγκες του. Έτσι η Lego με αυτόν τον τρόπο μας δίνει την ελευθερία να πειραματιστούμε ακόμα και με το firmware του ρομπότ, χωρίς να μας περιορίζει στο ελάχιστο.

2.4. Ο ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΟΥ ΡΟΜΠΟΤ

2.4.1. ΤΟ ΠΡΟΓΡΑΜΜΑ

Σε αυτό το σημείο θα αναλύσουμε λεπτομερώς τον τρόπο με τον οποίο συντάξαμε το κυρίως πρόγραμμα. Για να γίνει το κυρίως πρόγραμμα όσον τον δυνατών μικρότερο χρησιμοποιήθηκαν συναρτήσεις που καλούνται κατά την διάρκεια λειτουργιάς του προγράμματος.

Οι συναρτήσεις:

Για να εκτελεστεί μια από τις παρακάτω συναρτήσεις θα πρέπει για παράδειγμα να γράψουμε turnLeft(); σε κάποιο σημείο του προγράμματος. Παρακάτω θα αναλυθούν λεπτομερώς οι κυριότερες συναρτήσεις που δημιουργήσαμε.

1. **Συνάρτηση turnLeft()**

Καλώντας την **turnLeft()** εκτελούνται οι εντολές μέσα στις αγκύλες. Πιο συγκεκριμένα με την εντολή **nMotorEncoder[motorB]=0;** μηδενίζεται αρχικά ο μετρητής που ελέγχει την περιστροφή του σερβοκινητήρα. Αμέσως μετά το

πρόγραμμα μπαίνει μέσα στην while και το ρομπότ στρίβει αριστερά μέχρι ο μετρητής του σερβοκινητήρα να γίνει μεγαλύτερος από 180 (while(nMotorEncoder[motorB] < 180). Αυτό σημαίνει πως το ρομπότ θα στρίψει ακριβώς 90^{o} αριστερά. Οι εντολές μετά την εκτέλεση της while χρησιμοποιούνται για να σταματήσει το ρομπότ μέχρι να ελέγξει την επόμενη if από το κυρίως πρόγραμμα.

```
void turnLeft()
{
	nMotorEncoder[motorB] = 0;
	while(nMotorEncoder[motorB] < 180)
	{
		motor[motorC] = -20;
		motor[motorB] = 20;
	}

	motor[motorC] = 0;
	motor[motorB] = 0;

}
```

2. Συνάρτηση turnRight()

Η συνάρτηση αυτή λειτουργεί με την ίδια λογική της turnLeft(), με την διαφορά ότι το ρομπότ στρίβει δεξιά 180^{o} γιατί έχουμε επιλέξει την τιμή του MotorEncoder να γίνει μεγαλύτερη από 360.

```
void turnRight()
{
	nMotorEncoder[motorB] = 0;
	while(nMotorEncoder[motorB] < 360)
	{
		motor[motorC] = 20;
		motor[motorB] = -20;
	}

	motor[motorC] = 0;
	motor[motorB] = 0;

}
```

3. Συνάρτηση gothedark()

Για να καταφέρουμε να κάνουμε το ρομπότ να κινείται όσο το δυνατόν πιο ευθύγραμμα, δημιουργήσαμε την παρακάτω συνάρτηση. Με την συνάρτηση αυτή το ρομπότ ακολουθεί μια μαύρη γραμμή. Όταν εκτελείται η if το ρομπότ κινείται με μικρή κλίση προς τα δεξιά (εκτελώντας τις εντολές μέσα στην else) για όσο διάστημα βλέπει την μαύρη ταινία. Όταν βγει από αυτήν, το ρομπότ κινείται με μικρή κλίση προς τα αριστερά (εκτελώντας τις εντολές κάτω από την if) για να ξανά επιστρέψει στην μαύρη ταινία. Με αυτόν τον τρόπο, καταφέρνουμε το ρομπότ να κινείται ευθεία ακλουθώντας την μαύρη ταινία.

```
void gothedark()
{
  if(SensorValue(lightSensor) >45)
  {
    motor[motorB]= 30;
    motor[motorC]= 25;
  }
  else
  {
    motor[motorB]= 25;
    motor[motorC]= 30;
  }
}
```

Το κυρίως πρόγραμμα:

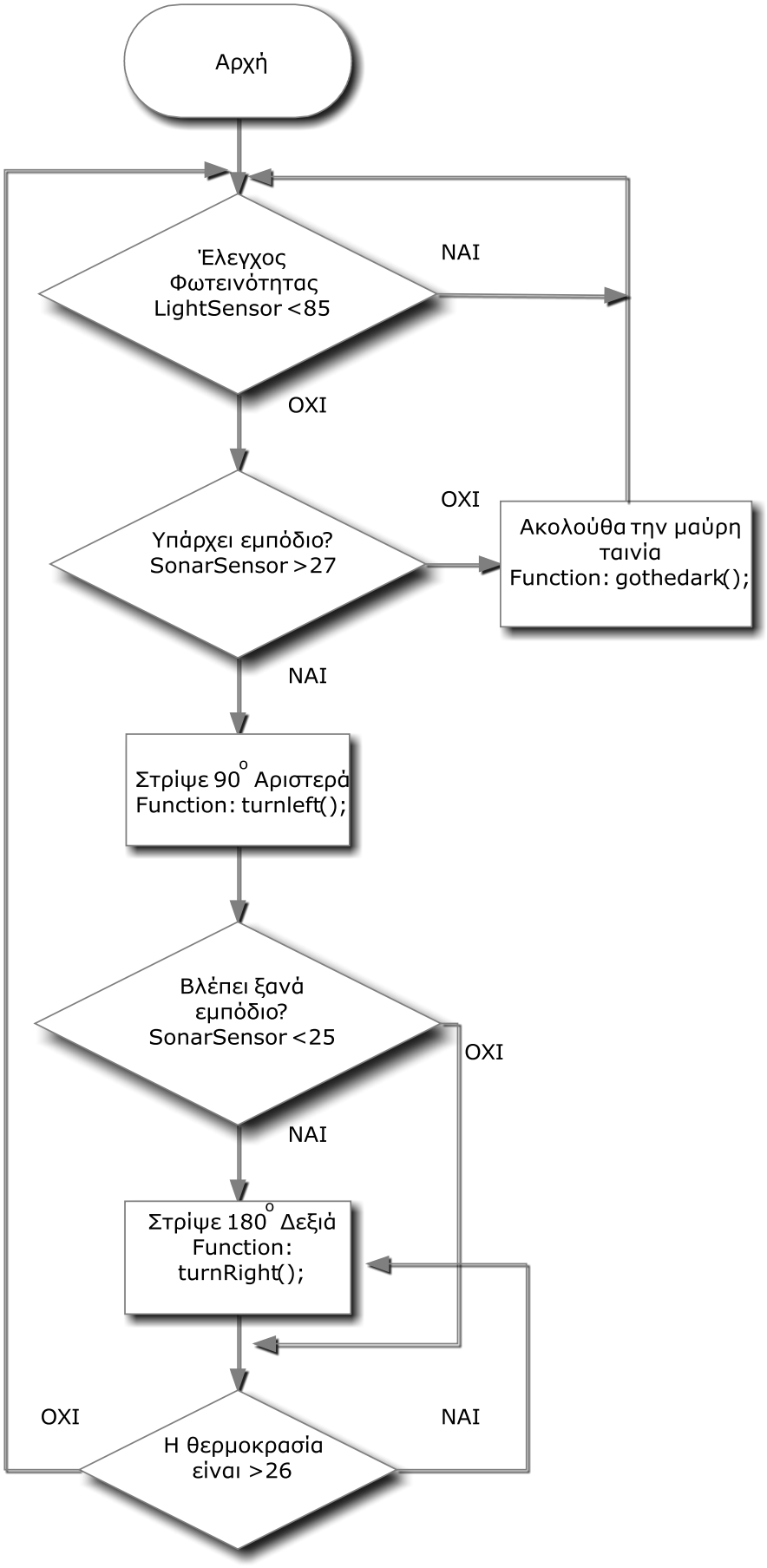
Ξεκινώντας το πρόγραμμα ελέγχεται η συνθήκη της while. Αν ο αισθητήρας φωτός (light sensor) δεν βλέπει έντονο φως, το πρόγραμμα μπαίνει μέσα στην while και ξεκινάει να ελέγχει τις συνθήκες if. Η πρώτη if κάνει έλεγχο για το αν βλέπει ο αισθητήρας υπερήχων κάποιο φυσικό εμπόδιο. Αν δεν βλέπει κανένα φυσικό εμπόδιο, που σημαίνει ότι ισχύει η συνθήκη (SensorValue(SonarSensor) >27) το πρόγραμμα μπαίνει μέσα στην if και καλεί την συνάρτηση gothedark() που αναφέραμε παραπάνω. Μόλις ο αισθητήρας δει εμπόδιο με την συνθήκη της δεύτερης if μπαίνει μέσα και καλεί την συνάρτηση turnLeft(). Σε περίπτωση που ξαναδεί εμπόδιο μετά την πρώτη στροφή που θα κάνει αριστερά, μπαίνει μέσα

στην τρίτη if και καλεί την συνάρτηση turnRight(). Στην τελευταία if το ρομπότ ελέγχει αν η θερμοκρασία δωματίου είναι μικρότερη από 26. Στην περίπτωση που η θερμοκρασία μας ανέβει πάνω από 26 βαθμούς, καλείται η συνάρτηση turnRight() για να αποφύγει το σημείο που ανίχνευσε την μεγάλη θερμοκρασία.

```
task main()
{
  while(SensorValue(lightSensor) < 75 )
  {
    if(SensorValue(SonarSensor) >27)
    {
      gothedark();
    }
    if(SensorValue(SonarSensor) <26)
    {
      turnLeft();
      wait1Msec(500);
    }

          if(SensorValue(SonarSensor) <25)
          {
            turnRight();
            wait1Msec(500);
          }

          if (TempSensor() > 26)
          {
            turnRight();

            motor[motorB]= 0;
            motor[motorC]= 0;
            wait1Msec(15000);
          }
    }
}
```

2.4.2. ΤΟ ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ

Για να γίνει όσο τον δυνατόν πιο κατανοητός ο τρόπος με τον οποίο λειτουργεί το πρόγραμμα μας, το παρακάτω γράφημα παρουσιάζει το διάγραμμα ροής του.


Αρχή
Έλεγχος Φωτεινότητας LightSensor <85
ΝΑΙ
ΟΧΙ
Υπάρχει εμπόδιο? SonarSensor >27
ΟΧΙ
Ακολούθα την μαύρη ταινία Function: gothedark();
ΝΑΙ
Στρίψε 90° Αριστερά Function: turnleft();
Βλέπει ξανά εμπόδιο? SonarSensor <25
ΟΧΙ
ΝΑΙ
Στρίψε 180° Δεξιά Function: turnRight();
ΟΧΙ
Η θερμοκρασία είναι >26
ΝΑΙ

2.4.3. ΤΑ ΠΡΟΒΛΗΜΑΤΑ ΠΟΥ ΠΑΡΟΥΣΙΑΣΤΗΚΑΝ ΚΑΤΑ ΤΗΝ ΔΙΑΡΚΕΙΑ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΚΑΙ ΜΕ ΠΟΙΟΥΣ ΤΡΟΠΟΥΣ ΕΠΙΛΥΘΗΚΑΝ

Κατά την διάρκεια του προγραμματισμού αντιμετωπίσαμε δυο προβλήματα. Το πρώτο αφορούσε τις εντολές του I^2C που χρησιμοποιήσαμε για την δημιουργία της συνάρτησης του αισθητήρα θερμοκρασίας. Το δεύτερο σχετιζόταν με την σωστή κίνηση του ρομπότ μας. Αμέσως παρακάτω θα αναφερθούμε λεπτομερέστερα στα προβλήματα αυτά.

1. Οι εντολές του I^2C:

Η RobotC αν και περιέχει πολλά παραδείγματα ωστόσο τα περισσότερα από αυτά αφορούν τις απλές εντολές. Το λογισμικό αυτό δεν μας παρείχε πολλές δυνατότητες για τον τρόπο σύνταξης σύνθετων εντολών του I^2C, όπως εκείνες που χρησιμοποιήσαμε στην δημιουργία της δικής μας συνάρτησης, για την λειτουργία του αισθητήρα θερμοκρασίας.

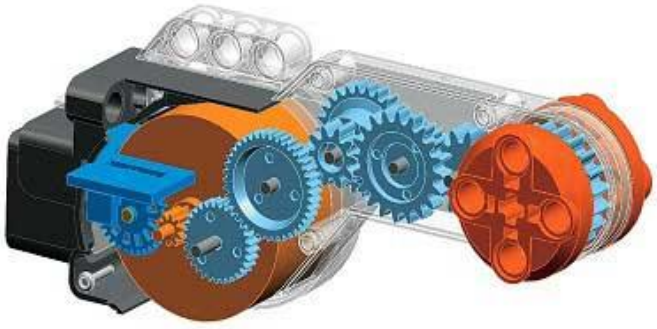
Για να καταφέρουμε να κατανοήσουμε την λειτουργία σύνθετων εντολών της I^2C χρειάστηκε να αναζητήσουμε πληροφορίες σε ξενόγλωσσα φόρουμ και κυρίως σε εκείνο της ίδιας της RobotC. Ωστόσο ακόμα και εκεί δεν υπήρχε κάτι συγκεκριμένο παρά μόνο προγράμματα κώδικα που είχαν αναρτήσει χρήστες του φόρουμ. Τα προγράμματα αυτά τα αναλύσαμε για να καταλάβουμε τον τρόπο με τον οποίο λειτουργούν οι παρακάτω εντολές:

a. *SensorI2CCustom*
b. *SendI2CMsg*
c. *nI2CStatus και*
d. *readI2CReplay*

Οι παραπάνω εντολές θα αναλυθούν στο κεφάλαιο 3 που αναφέρεται στο πρωτόκολλο I^2C.

2. Η κίνηση του ρομπότ:

Ένα μεγάλο πρόβλημα που είχαμε να αντιμετωπίσουμε ήταν αυτό της σωστής πορείας του ρομπότ μέσα στον λαβύρινθο. Στην αρχή δεν μπορούσαμε να βρούμε κάποια εντολή ή συνάρτηση που να κάνει το ρομπότ να στρίβει σε συγκεκριμένες μοίρες. Επίσης το ρομπότ δεν μπορούσε να διανύσει μια μεγάλη ευθεία χωρίς να παρεκκλίνει έστω και λίγο διότι τα γρανάζια που υπάρχουν στο εσωτερικό των σερβοκινητήρων είναι πλαστικά (βλ. Εικόνα 2.1). Ως εκ τούτου, υπάρχει μεγάλη πιθανότητα να υποστούν φθορές μετά από πολλές χρήσεις. Οι φθορές αυτές προκαλούν αργότερα την δυσλειτουργία των σερβοκινητήρων.



Εικόνα 2.1 Το εσωτερικό ενός motor

Προκειμένου να επιλύσουμε το πρώτο πρόβλημα δηλαδή το ρομπότ να στρίβει 90 ή 180 μοίρες, χρησιμοποιήσαμε την παρακάτω συνάρτηση, έτσι επιτύχαμε να μικρύνουμε κατά πολύ το κυρίως πρόγραμμα.

```
void turnLeft()
{
   nMotorEncoder[motorB] = 0;   // Κάνε Reset το Motor Encoder του motor B.
   while(nMotorEncoder[motorB] < 180)  // Μέχρι το Motor Encoder του motor B να
γίνει 180 στρίψε αριστερά.
    {
       motor[motorC] = -20;
       motor[motorB] = 20;
    }
   motor[motorC] = 0;
   motor[motorB] = 0;
}
```

Στην ουσία όταν ο μετρητής περιστροφής του σερβοκινητήρα γίνει μεγαλύτερος από 180, το ρομπότ θα έχει στρίψει ακριβώς 90 μοίρες αριστερά. Για τις 180 μοίρες

επιλέχτηκε η τιμή 360. Οι σερβοκινητήρες περιέχουν αισθητήρες περιστροφής τους οποίους χρησιμοποιήσαμε με βάση τις παραπάνω εντολές.

Όσον αφορά το δεύτερο πρόβλημα επιλύθηκε χρησιμοποιώντας τον αισθητήρα φωτός, βάζοντας το ρομπότ να ακολουθεί μια μαύρη γραμμή.

ΚΕΦΑΛΑΙΟ 3:

ΤΟ I^2C ΠΡΩΤΟΚΟΛΛΟ

3.1. ΕΙΣΑΓΩΓΗ ΣΤΟ I^2C

Το I^2C Bus αποτελεί ένα πρωτόκολλο που χρησιμοποιείται για την διασύνδεση μεταξύ διαφορετικών ολοκληρωμένων, όπως των μικροελεγκτών και των υπόλοιπων ολοκληρωμένων περιφερειακών όπως EEPROMs, A/D μετατροπείς, LCD drivers, διάφοροι αισθητήρες αλλά και με άλλους μικροελεγκτές. Είναι ένα πρωτόκολλο που μεταδίδει δεδομένα σειριακά και είναι μια από τις λειτουργίες της μονάδας του PICmicro που ονομάζεται σύγχρονη σειριακή θύρα (SSP). Ο άλλος τρόπος λειτουργίας της είναι ως Σειριακό Περιφερειακό Interface (SPI). Με το I^2C αποφεύγεται η χρησιμοποίηση ενός παράλληλου διαύλου δεδομένων που εισάγει μεγάλη πολυπλοκότητα στη σχεδίαση αλλά και μεγαλύτερο κόστος. Βρίσκει πολλές εφαρμογές στα σύγχρονα ηλεκτρονικά συστήματα όπως σε συσκευές εικόνας και ήχου, σε τηλεφωνικές συσκευές, modems, dip switches, embedded microprocessor boards αλλά και στην επικοινωνία αισθητήρων θερμοκρασίας με τις οθόνες όπου παρουσιάζονται τα αποτελέσματα των μετρήσεων.

3.2. ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

Το πρωτόκολλο I^2C (inter-integrated circuit) αναπτύχθηκε την δεκαετία του 1980 από την Philips, αρχικά με σκοπό την εύκολη επικοινωνία μεταξύ μιας Κεντρικής Μονάδας Επεξεργασίας (CPU) με τα περιφερειακά κυκλώματα μιας τηλεόρασης. Συνδέοντας πολλαπλές συσκευές σ' ένα μικροεπεξεργαστή, οι γραμμές μεταφοράς δεδομένων και διευθύνσεων έπρεπε παραδοσιακά να συνδεθούν για κάθε μία

συσκευή. Αυτό απαιτούσε πολλούς ακροδέκτες από τον μικροεπεξεργαστή, πολλές γραμμές πάνω σε μια τυπωμένη πλακέτα και πολλά εξαρτήματα που έπρεπε να συναρμολογηθούν μαζί. Σαν αποτέλεσμα, αυτά τα συστήματα ήταν ακριβής παραγωγής και επίφοβα σε παρεμβολές και θόρυβο. Η έρευνα για την αντιμετώπιση των παραπάνω προβλημάτων έγινε από την Philips στην Ολλανδία και κατέληξε σε ένα δισύρματο καλώδιο επικοινωνίας που ονομάστηκε I^2C bus. Το I^2C είναι η συντομογραφία του Inter-IC bus και όπως φαίνεται και από την ονομασία του έχει ως σκοπό να παρέχει επικοινωνιακή ζεύξη μεταξύ ολοκληρωμένων κυκλωμάτων. Σήμερα, το I^2C bus χρησιμοποιείται σε πολύ περισσότερες εφαρμογές και όχι μόνο σε εξαρτήματα ήχου και βίντεο όπως για παράδειγμα το ρομπότ της Lego NXT. Το I^2C bus είναι γενικά αποδεκτό στην βιομηχανία και έχει υιοθετηθεί από πολλούς κατασκευαστές chip όπως Xicor, ST Microelectronics, Infineon Technologies, Intel, Texas Instruments, Maxim, **Atmel**, Analog Devices και άλλες εταιρείες.

3.3. ΠΕΡΙΓΡΑΦΗ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΤΟΥ I^2C BUS

3.3.1. ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΜΟΝΑΔΑΣ

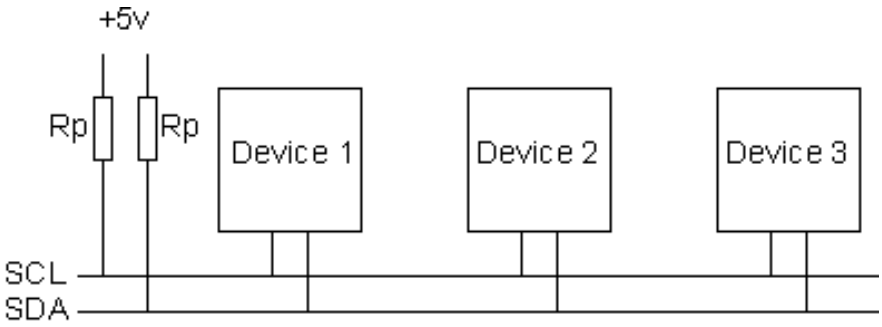
Μερικά από τα σημαντικότερα χαρακτηριστικά της μονάδας αυτής είναι:

- Επιλογή χρησιμοποίησης του ως master ή slave
- Υποστήριξη multi-master λειτουργίας, χωρίς να χάνονται μηνύματα κατά την διάρκεια της ανάκτησης ελέγχου από οποιονδήποτε master (arbitration)
- Αυτόματη επαναποστολή αποτυχημένων μηνυμάτων
- Δυνατότητα διευθυνσιοδότησης των 7-bits και των 10-bits

- Κάθε συσκευή που συνδέεται πάνω στο bus έχει τη δικιά της μοναδική διεύθυνση
- Ο slave μπορεί να είναι είτε συσκευή λήψης μόνο είτε συσκευή μετάδοσης με δυνατότητα να λαμβάνει και να στέλνει δεδομένα
- Υποστήριξη γενικής κλήσης (general call addresses) από τον master
- Υποστήριξη επιλογής ταχύτητας μετάδοσης των 8-bit πακέτων μεταξύ του standard clock (SCL) mode που είναι 100Kbps, του fast mode που είναι 400Kbps και του High Speed mode που είναι 3.4Mbps.
- Δεν επηρεάζεται από απότομες μεταβολές της τάσης αλλά και τον θόρυβο
- Επιτρέπει να συνδεθούν 112 συσκευές σε 7-bit διευθυνσιοδότηση και 1024 σε 10-bit διευθυνσιοδότηση.

3.3.2. ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΟΥ I^2C BUS

Το I^2C υλοποιείται με την χρήση δύο καλωδίων διπλής κατεύθυνσης. Είναι δύο από τα έξι καλώδια που συνδέονται στις θύρες των αισθητήρων του NXT, τα αποκαλούμενα SCL (Serial Clock) και SDA (Serial Data). Το SCL είναι η γραμμή ρολογιών (clock) και χρησιμοποιείται για να συγχρονίσει τα δεδομένα κατά τη διάρκεια των μεταφορών από το I^2C bus. Το SDA είναι η γραμμή δεδομένων (data). Οι γραμμές SCL και SDA συνδέονται με όλες τις συσκευές στο I^2C bus. Επίσης πρέπει να υπάρχει ένα τρίτο καλώδιο που είναι η γείωση ή αλλιώς τα 0 volt. Όπως επίσης υπάρχει ένα καλώδιο των 5volt, που είναι η τροφοδοσία που διανέμεται στις συσκευές. Και οι δυο γραμμές, SCL και SDA, είναι οδηγοί “ανοικτών αγωγών”. Αυτό σημαίνει ότι το τσιπ μπορεί να οδηγήσει την έξοδο σε χαμηλά επίπεδα, αλλά δεν μπορεί να την οδηγήσει σε υψηλά επίπεδα. Για να είναι σε θέση η γραμμή να οδηγηθεί σε υψηλά επίπεδα, πρέπει να προστεθούν pull-up αντιστάσεις στην 5volt γραμμή τάση τροφοδοσίας. Πρέπει δηλαδή να υπάρχει μια αντίσταση από τη γραμμή SCL στη 5volt γραμμή και άλλη μια αντίσταση από τη γραμμή SDA στη 5volt γραμμή. Χρειάζεται μόνο ένα σύνολο pull-up αντιστάσεων για ολόκληρο το I^2C bus, όχι για κάθε συσκευή, όπως μπορείτε να διακρίνεται στην παρακάτω εικόνα **3.1.**:



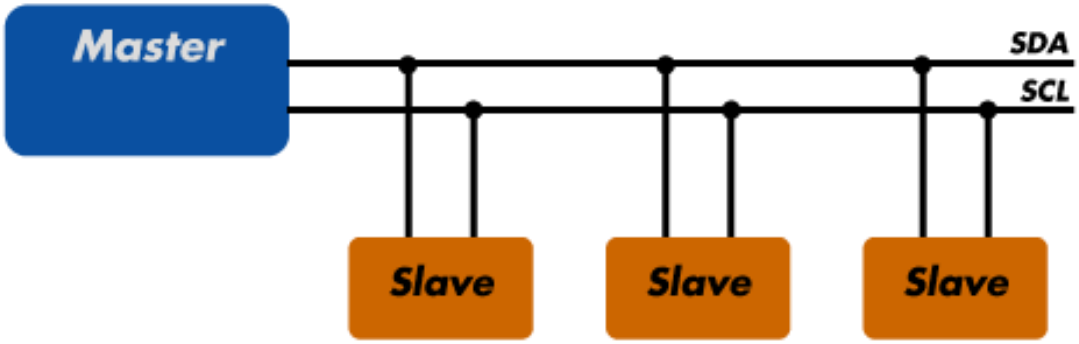
Εικόνα 3.1. Τρόπος σύνδεσης pull-up αντιστάσεων στο i2c bus

Όσον αφορά τις τιμές των αντιστάσεων που πρέπει να χρησιμοποιηθούν, αυτές μπορεί να κυμαίνονται από ένα όριο 1.8ΚΩ μέχρι και 82ΚΩ, αλλά τίποτα δεν είναι δεδομένο και σίγουρο για την επίτευξη του τέλειου αποτελέσματος. Η χρήση των pull-up αντιστάσεων είναι απαραίτητη γιατί αν λείπουν οι συγκεκριμένες αντιστάσεις, οι γραμμές SCL και SDA θα βρίσκονται συνεχώς σε χαμηλό επίπεδο (σχεδόν κοντά στα 0 volt) και το I^2C bus δεν θα λειτουργεί. Η τιμή των pull-up αντιστάσεων μαζί με την χωρητικότητα των γραμμών και των εισόδων των συσκευών καθορίζουν την καθυστέρηση στην άνοδο του παλμού και για αυτό θα πρέπει το γινόμενο τους να είναι αρκετά μικρό ώστε να είναι ευδιάκριτο το μέτωπο του παλμού. Για το λόγο αυτό η τιμή των pull-up αντιστάσεων θα πρέπει να είναι αρκετά μικρή. Παράλληλα όμως κάθε συσκευή που προσπαθεί να οδηγήσει μια γραμμή στο μηδέν θα πρέπει να απορροφά ρεύμα λίγο μεγαλύτερο από VDD/RP. Έτσι είναι κατανοητό ότι η τιμή της αντίστασης RP δεν μπορεί να είναι όσο μικρή επιθυμούμε.

3.3.3. Ο ΤΡΟΠΟΣ ΔΙΑΣΥΝΔΕΣΗΣ ΤΩΝ ΣΥΣΚΕΥΩΝ (MASTER-SLAVE)

Οι συσκευές στο I^2C bus είναι είτε masters είτε slaves. Η master συσκευή είναι πάντα η συσκευή που οδηγεί τη γραμμή ρολογιών SCL. Στην προκειμένη περίπτωση το NXT brick είναι η master συσκευή. Ενώ οι slaves συσκευές είναι αυτές που ανταποκρίνονται στην master. Στο NXT ρομπότ οι slaves συσκευές είναι οι περιφερικές συσκευές (π.χ. αισθητήρες) που συνδέονται στο NXT brick. Υπεύθυνος για την έναρξη μιας μεταφοράς δεδομένων στο I^2C bus είναι μόνο ο master. Συνήθως υπάρχουν πολλαπλάσιοι slaves στο I^2C bus, εντούτοις υπάρχει

κανονικά μόνο ένας master. Είναι πιθανό να υπάρξουν πολλαπλοί master, αλλά είναι κάπως ασυνήθιστο. Στο ρομπότ, ο master θα είναι ο ελεγκτής και οι slaves θα είναι οι υπομονάδες. Επίσης ο master και ο slave μπορούν να μεταφέρουν τα στοιχεία από το I^{2}C bus, αλλά εκείνη η μεταφορά ελέγχεται πάντα από τον master.



Εικόνα 3.2 Ένα τυπικό διάγραμμα σύνδεσης συσκευών με το I^{2}C

3.3.4. I^{2}C BUS ΕΠΙΚΟΙΝΩΝΙΑ

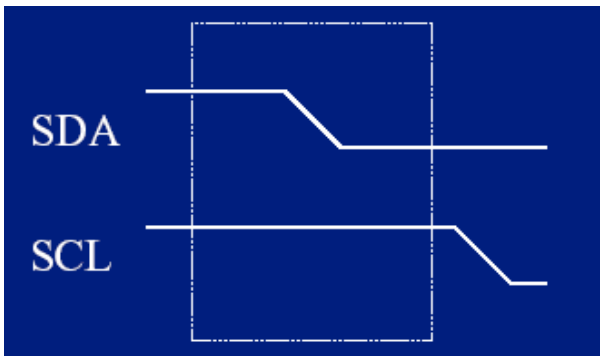
Σύμφωνα λοιπόν με όσα έχουμε αναφέρει για να επιτευχθεί η μετάδοση των επιθυμητών δεδομένων, είναι απαραίτητη η παρουσία ενός τουλάχιστον Master. Ο Master του bus είναι το ολοκληρωμένο που δίνει τις εντολές στα υπόλοιπα ολοκληρωμένα. Ελέγχει την γραμμή του ρολογιού (SCL) και «κηρύσσει» την έναρξη των επιθυμητών διαδικασιών ενώ κάθε άλλο ολοκληρωμένο θεωρείται ως slave. Οι πιθανές καταστάσεις που μπορούν να συμβαίνουν πάνω στο bus και δηλώνουν το σύνολο των σημείων που ανταλλάσσουν οι συσκευές μεταξύ τους ώστε να μεταφερθούν τα δεδομένα, είναι οι εξής:

- START Data Transfer (S)
- Acknowledged (Επιβεβαίωση) (ACK)
- Μη-Επιβεβαίωση(NOT-ACK) (NACK)
- Ανάγνωση / Εγγραφή Δεδομένων (READ/WRITE)
- RESTART (R)
- REPEATED START (Rs)
- STOP Data Transfer (P)

Αμέσως παρακάτω θα αναφερθούμε λεπτομερέστερα σε όλες τις καταστάσεις.

3.3.4.1. START DATA TRANSFER

Στην κατάσταση Start σηματοδοτείται η εκκίνηση της μεταφοράς δεδομένων που θέλουμε. Η κατάσταση αυτή σηματοδοτείται μόνο από τον master και σε περίπτωση multi-master περιβάλλοντος από εκείνον τον master που θα εκπέμψει πρώτος. Για να μπει το καλώδιο στην Start κατάσταση πρέπει αρχικά να είναι σε αδρανή κατάσταση και όταν τελικά μπει στην Start κατάσταση θεωρείται απασχολημένο και δεν μπορεί να το οικειοποιηθεί άλλος master ή οποιαδήποτε άλλη συσκευή που είναι συνδεδεμένη πάνω στο καλώδιο.
Επίσης την κατάσταση αυτή μπορούμε να την συναντήσουμε όχι μόνο στην αρχή αλλά και κατά την διάρκεια της μεταφοράς των δεδομένων. Η μορφή του σήματος αυτού φαίνεται στην παρακάτω εικόνα:



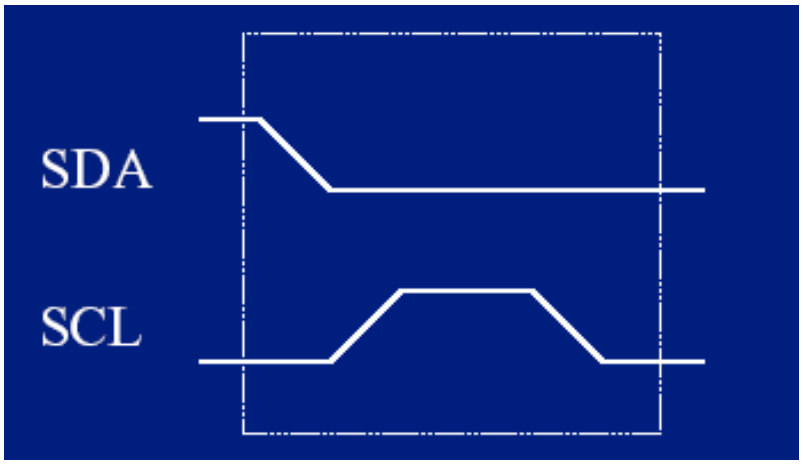
Εικόνα 3.3 Αρχικοποίηση μεταφοράς δεδομένων στο I2C bus

Όταν λοιπόν το bus δεν χρησιμοποιείται, οι εξωτερικές pull-up αντιστάσεις κρατούν τις εξωτερικές γραμμές σε υψηλή στάθμη σήματος. Όπως παρατηρούμε από το παραπάνω σχήμα ο master που θα εκπέμψει την start κατάσταση θα πρέπει να θέσει την SDA γραμμή από υψηλή σε χαμηλή στάθμη στη διάρκεια που η SCL γραμμή θα είναι σε υψηλή στάθμη.

3.3.4.2. ACKNOWLEDGED (ΕΠΙΒΕΒΑΙΩΣΗ)

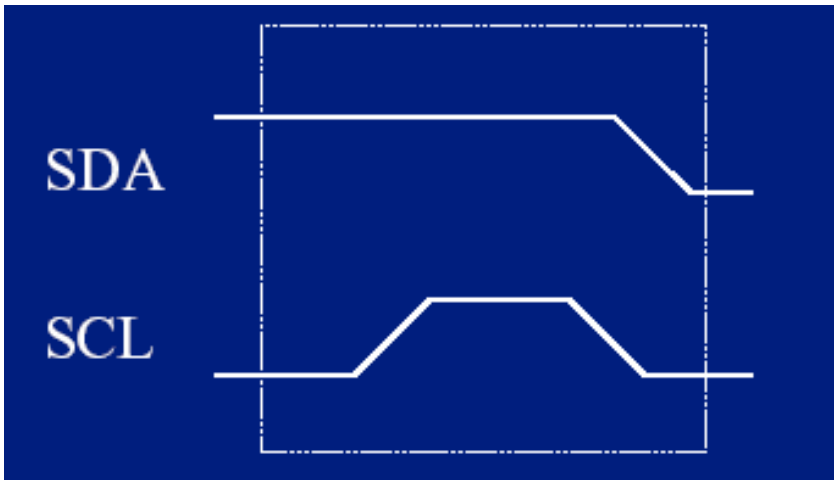
Στο τέλος της διαδικασίας αποστολής ενός byte δεδομένων στον δέκτη του σήματος, αποστέλλεται από αυτόν ένα σήμα επιβεβαίωσης(ACK) ότι έλαβε αυτό το byte. Στην περίπτωση που το byte δεδομένων που αποστέλλεται είναι το τελευταίο

της όλης μετάδοσης ή ο δέκτης δεν μπορεί να λάβει άλλα δεδομένα ή γενικά κάτι πήγε στραβά στην όλη μετάδοση, τότε αποστέλλεται από τον δέκτη ένα σήμα μη-επιβεβαίωσης (NACK).



Εικόνα 3.4 Σήμα επιβεβαίωσης μεταφοράς δεδομένων στο I2C bus

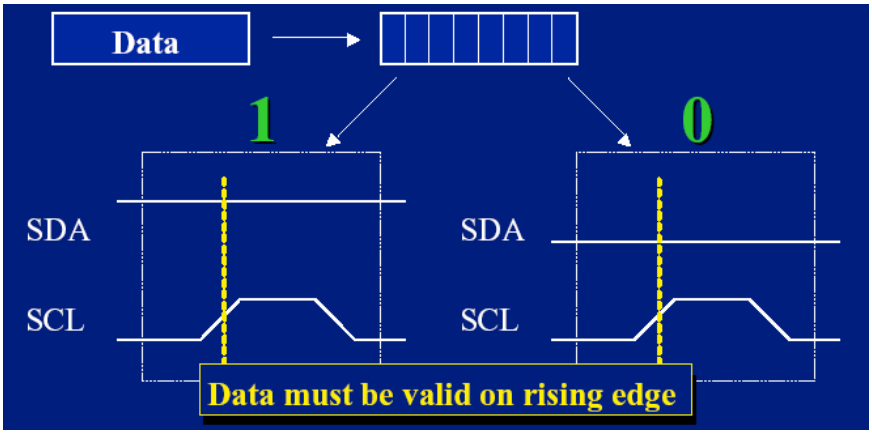
Όπως φαίνεται από το παραπάνω σχήμα τα δεδομένα επιβεβαιώνονται όταν ο δέκτης «σπρώχνει» την SDA γραμμή στην χαμηλή στάθμη στην διάρκεια ενός παλμού του ρολογιού (SCL). Στην αντίθετη περίπτωση της μη-επιβεβαίωσης (NACK), στην διάρκεια ενός παλμού του ρολογιού (SCL), ο δέκτης «κρατάει» την SDA γραμμή σε υψηλή στάθμη, κάτι που φαίνεται και από το παρακάτω σχήμα.



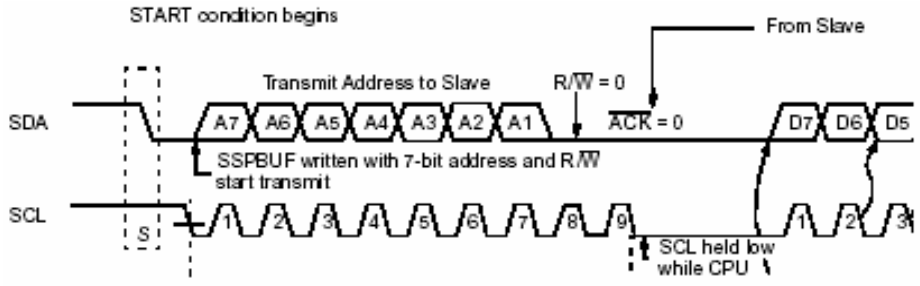
Εικόνα 3.5 Σήμα μη-επιβεβαίωσης μεταφοράς δεδομένων στο I2C bus

3.3.4.3. ΑΝΑΓΝΩΣΗ / ΕΓΓΡΑΦΗ ΔΕΔΟΜΕΝΩΝ

Τόσο η ανάγνωση όσο και η εγγραφή είναι γνωστά μετά την αρχικοποίηση μιας μεταφοράς δεδομένων. Η μεταφορά δεδομένων γίνεται σε πακέτα των 8 bits. Τα δεδομένα θεωρούνται έγκυρα όταν η SCL γραμμή είναι σε υψηλή στάθμη ενώ όταν η SCL δεν είναι σε υψηλή στάθμη δεν επιτρέπεται να αλλάξουν, όπως φαίνεται και από την εικόνα 3.6. Στην περίπτωση τώρα που ο master θέλει να κάνει εγγραφή (write) δεδομένων στον slave, τοποθετεί τα προς εγγραφή δεδομένα πάνω στην SDA γραμμή και παράγει κατάλληλους παλμούς ρολογιού. Μετά των 8ο παλμό εάν όλα πήγαν καλά, ο slave παίρνει την γραμμή SDA στην κατοχή του και την μηδενίζει, δηλαδή παράγει ένα σήμα ACK. Στην περίπτωση που ο slave δεν ανταποκρίνεται με το παλμό ACK καταλαβαίνουμε αμέσως ότι υπάρχει κάποιο σφάλμα και το λογισμικό θα πρέπει να προβλέπει την διαδικασία που θα ακολουθείται σε μια τέτοια περίπτωση.

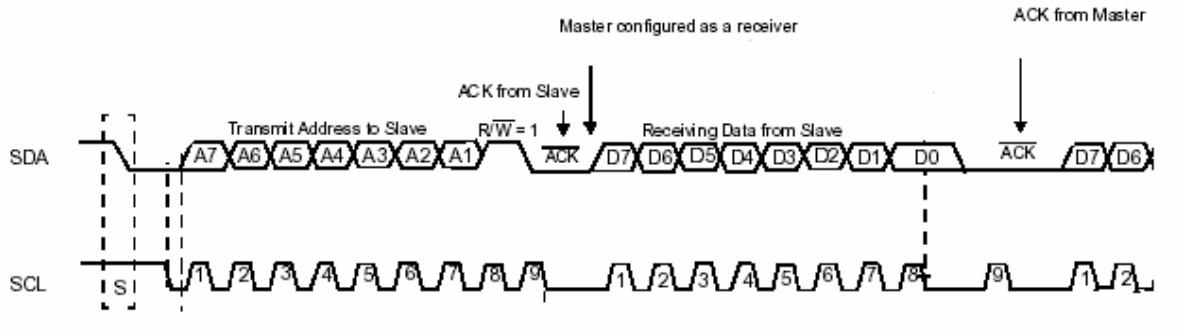


Εικόνα 3.6 Σήμα έγκυρης μεταφοράς δεδομένων στο I^2C bus



Εικόνα 3.7 Σήμα εγγραφής δεδομένων στο I^2C bus

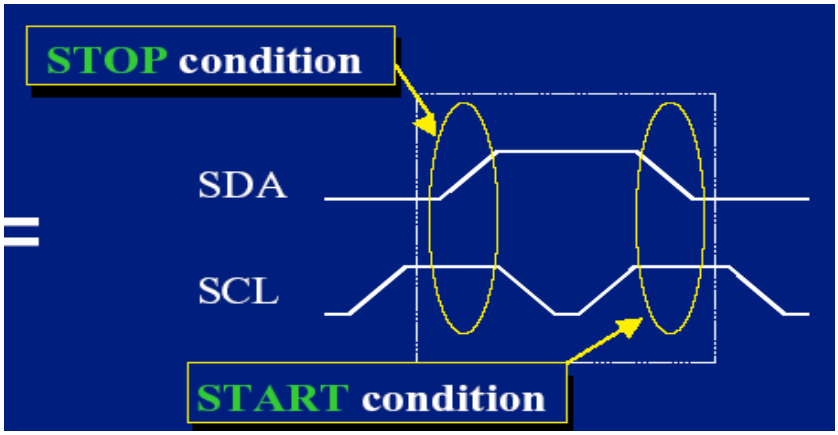
Στην περίπτωση που ο master θέλει να κάνει ανάγνωση (read) δεδομένων από τον slave, παράγει κατάλληλους παλμούς ρολογιού και διαβάζει τα δεδομένα στην SDA γραμμή.



Εικόνα 3.8 Σήμα ανάγνωσης δεδομένων στο I^2C bus

3.3.4.4. RESTART

Η restart κατάσταση δεν είναι τίποτε άλλο από μια stop κατάσταση που ακολουθείται αμέσως από μια start κατάσταση. Σε αυτήν την κατάσταση μπορούμε να έχουμε και αλλαγή της κατεύθυνσης των δεδομένων καθώς επίσης και αλλαγή του ολοκληρωμένου που μεταδίδει ή στέλνει τα δεδομένα (νέα διεύθυνση-νέα συσκευή).



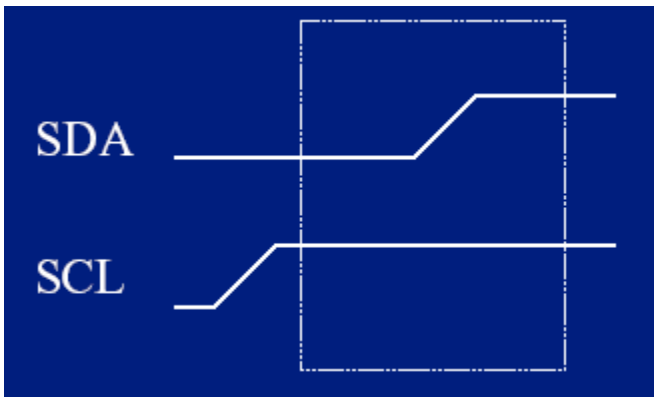
Εικόνα 3.9 Restart σήμα στο I^2C bus

3.3.4.5. REPEATED START

Η repeated start κατάσταση δεν είναι τίποτε άλλο από μια κατάσταση που επιτρέπει στον master να εκτελέσει μια start κατάσταση χωρίς να χρειάζεται η δήλωση κάποιας stop κατάστασης. Ο έλεγχος του bus από τον master δεν χάνεται ποτέ. Σε αυτήν την κατάσταση μπορούμε να έχουμε και αλλαγή της κατεύθυνσης των δεδομένων καθώς επίσης και αλλαγή του ολοκληρωμένου που μεταδίδει ή στέλνει τα δεδομένα(νέα διεύθυνση-νέα συσκευή).

3.3.4.6. STOP DATA TRANSFER

Η κατάσταση Stop σηματοδοτεί το τέλος μιας μεταφοράς δεδομένων. Μαζί με την κατάσταση Start έχει την μεγαλύτερη προτεραιότητα και μπορεί να εκπεμφθεί σε οποιαδήποτε χρονική στιγμή της μεταφοράς δεδομένων. Με το που σηματοδοτηθεί η stop κατάσταση το καλώδιο απελευθερώνεται και άλλες συσκευές μπορούν να το χρησιμοποιήσουν για να μεταφέρουν δεδομένα. Η μορφή του σήματος Stop φαίνεται στην παρακάτω εικόνα:



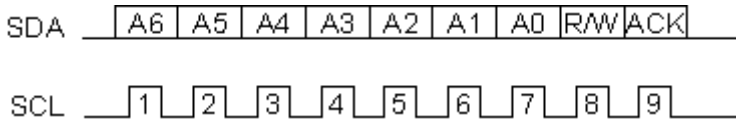
Εικόνα 3.10 STOP σήμα στο I^2C bus

Όπως παρατηρούμε από το παραπάνω σχήμα ο master που θα εκπέμψει την stop κατάσταση θα πρέπει να θέσει την SDA γραμμή από χαμηλή σε υψηλή στάθμη στη διάρκεια που η SCL γραμμή θα είναι σε υψηλή στάθμη. Έτσι όταν ολοκληρωθεί η stop κατάσταση οι SDA, SCL γραμμές θα είναι και οι δύο σε υψηλή στάθμη και το καλώδιο μπαίνει σε αδρανή κατάσταση.

3.3.5. Η ΔΙΕΥΘΥΝΣΗ ΤΗΣ I^2C ΕΠΙΚΟΙΝΩΝΙΑΣ

Όλες οι I^2C διευθύνσεις είναι είτε 7 bits είτε 10 bits. Η χρήση των διευθύνσεων 10 bits είναι σπάνια και δεν είναι απαραίτητη για το δικό μας αισθητήρα θερμοκρασίας που θα χρησιμοποιήσουμε στο ρομπότ. Αυτό σημαίνει ότι μπορεί να έχουμε μέχρι 128 συσκευές στο I^2C Bus, δεδομένου ότι ένας 7 bit αριθμός μπορεί να είναι από 0 έως 127. Εφόσον στο ρομπότ Lego NXT υπάρχουν 4 είσοδοι για την σύνδεση των αισθητήρων, δεν είναι απαραίτητη η 10bit διεύθυνση . Κατά την αποστολή της 7-bit διεύθυνσης, στέλνονται πάντα 8 bits. Το πρόσθετο bit χρησιμοποιείται για να ενημερώσει την slave συσκευή εάν ο master θα γράψει σε αυτό ή θα διαβάζει από

αυτό. Εάν το bit είναι 0, τότε ο master γράφει στον slave. Εάν το bit είναι 1 ο master διαβάζει από τον slave. Η 7-bit διεύθυνση τοποθετείται στα 7 ανώτερα bits του byte και το ανάγνωσης/γραφής κομμάτι (R/W) είναι στο LSB (λιγότερο σημαντικό bit).



Εικόνα 3.11 Διάταξη της 7-bit διεύθυνσης

3.4. ΤΟ I²C ΣΤΗΝ ROBOTC

Το ρομπότ της Lego NXT ενσωματώνει το I²C πρωτόκολλο. Αυτό σημαίνει πως μπορούμε να συνδέσουμε ψηφιακούς αισθητήρες που έχουν δυνατότητα επικοινωνίας με το I²C Bus και να τους κάνουμε να λειτουργούν στο ρομπότ μας. Παρακάτω αναλύεται ο τρόπος με τον οποίο συντάξαμε τις εντολές του I²C για να δημιουργήσουμε την δική μας συνάρτηση.

Αρχικά δηλώνουμε τις μεταβλητές του προγράμματος όπως φαίνονται παρακάτω.

```
byte Buffer_Temperature[1];
int temperature_1;
```

Με το const tSensors δηλώνουμε σε πια θύρα θα συνδέεται ο αισθητήρας μας. Στην προκειμένη περίπτωση επιλέξαμε την θύρα 1. Ενώ με το sensorI2CCustom επιτυγχάνεται η αποστολή ενός byte ανά ms.

```
const tSensors AD7416 = S1;
SensorType[AD7416] = sensorI2CCustom;
```

Για να ρυθμίσουμε τις διευθύνσεις με τις οποίες θα επικοινωνεί ο αισθητήρας θερμοκρασίας με το ρομπότ χρησιμοποιήσαμε τις εντολές που φαίνονται παρακάτω. Ορίζουμε ένα πίνακα με τέσσερα στοιχεία, τον I2C_Message[4], στον οποίο δηλώνουμε τις τέσσερις παραμέτρους που χρησιμοποιούμε στην συνέχεια για την λήψη των δεδομένων από τον αισθητήρα. Αυτό γίνεται πάντα σε δεκαεξαδική μορφή. Η 0x03 είναι το μήκος του μεταδιδόμενου μηνύματος. Η 0x90

είναι εντολή για να γράψει στον Address Pointer του αισθητήρα. Και τέλος, η 0x01 είναι ο Configuration Register και η 0x00 ο Temperature Value Register.

```
byte I2C_Message[4] = {0x03, 0x90, 0x01, 0x00};
```

Με το sendI2CMsg στέλνεται μήνυμα από το I2C στην πόρτα που έχουμε ορίσει για τον αισθητήρα. Με την while(nI2CStatus[AD7416]==STAT_COMM_PENDING) δημιουργείται μια καθυστέρηση μέχρι να ολοκληρωθεί η συναλλαγή με το I^2C.

```
sendI2CMsg(AD7416, I2C_Message[0], 0);
while(nI2CStatus[AD7416] == STAT_COMM_PENDING)
{
  ;
}
```

Με τις παρακάτω εντολές ξεκάνει η διαδικασία αποστολής της τιμής που έχει ο αισθητήρας στην οθόνη LCD του NXT. Αρχικά καθαρίζουμε την οθόνη με την εντολή eraseDisplay. Στον πίνακα I2C_Message[3] ορίζουμε την διεύθυνση με την οποία ο αισθητήρας θερμοκρασίας θα στείλει τα δεδομένα του στο ρομπότ. Πιο συγκεκριμένα η διεύθυνση πάνω στο Bus είναι η 0x02, η 0x90 είναι για να γίνει η εγγραφή στον Address Pointer και η 0x00 είναι η τιμή που θα πάρει από τον Temperature Value Register. Μετά στέλνονται τα δεδομένα με το sendI2CMsg και με την while δημιουργείται η καθυστέρηση μέχρι να ολοκληρωθεί η συναλλαγή με το I^2C.

```
eraseDisplay();
byte I2C_Message[3] = {0x02, 0x90, 0x00};
sendI2CMsg(AD7416, I2C_Message[0], 1);
while(nI2CStatus[AD7416] == STAT_COMM_PENDING)
{
  ;
}
```

Οι εντολές που ακολουθούν είναι για να γράψουμε την τιμή της θερμοκρασίας που έχει μετρήσει ο αισθητήρας πάνω στην οθόνη του NXT όπως φαίνεται στην εικόνα 3.12.

```
readI2CReply(AD7416, Buffer_Temperature[0], 1);
temperature_1 = Buffer_Temperature[0];
nxtDisplayCenteredBigTextLine(0,"PTYXIAKI");
```

```
  nxtDisplayTextLine(2,"Kalaitzidis Fotis");
  nxtDisplayTextLine(3,"Tarsenis Dimitris");
  nxtDisplayCenteredTextLine(4,"Temp is:");
  nxtDisplayCenteredBigTextLine(6,"%doC", temperature_1);
  wait1Msec(50);
}
```

Εικόνα 3.12 Αποτύπωση της τιμής του αισθητήρα στη οθόνη του NXT.

Τέλος με την εντολή return καταχωρείται η τιμή του αισθητήρα στην μεταβλητή temperature_1.

```
return (temperature_1);
```

ΚΕΦΑΛΑΙΟ 4:

ΠΡΟΣΑΡΜΟΓΗ ΤΟΥ ΑΙΣΘΗΤΗΡΑ AD7416

4.1. ΕΙΣΑΓΩΓΗ

Το ολοκληρωμένο AD7416 που έχουμε επιλέξει για την Π.Ε. είναι ένας ψηφιακός αισθητήρας που λειτουργεί στα 10-bit και αλληλεπιδρά με το πρωτόκολλο επικοινωνίας I^2C. Σκοπός της υπάρχουσας κατασκευής είναι η σύνδεση ενός αισθητήρα θερμοκρασίας που παρέχει την δυνατότητα στο Lego NXT, να ελέγχει την θερμοκρασία περιβάλλοντος. Η θερμοκρασία μπορεί να αποτυπωθεί στην οθόνη του NXT και να χρησιμοποιηθεί για τον έλεγχο θερμοκρασίας ενός δωματίου ή για οποιαδήποτε άλλη εφαρμογή, ανάλογα με τις παραμέτρους που θα τεθούν στο πρόγραμμα. Στο παρόν κεφάλαιο αναλύεται ο τρόπος με τον οποίο καταφέραμε να κάνουμε τον αισθητήρα να επικοινωνεί μέσω του πρωτοκόλλου I^2C με το ρομπότ.

4.2. ΠΕΡΙΦΡΑΦΗ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ AD7416

4.2.1. ΓΕΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

Μερικά από τα σημαντικότερα χαρακτηριστικά του αισθητήρα θερμοκρασίας AD7416 είναι:

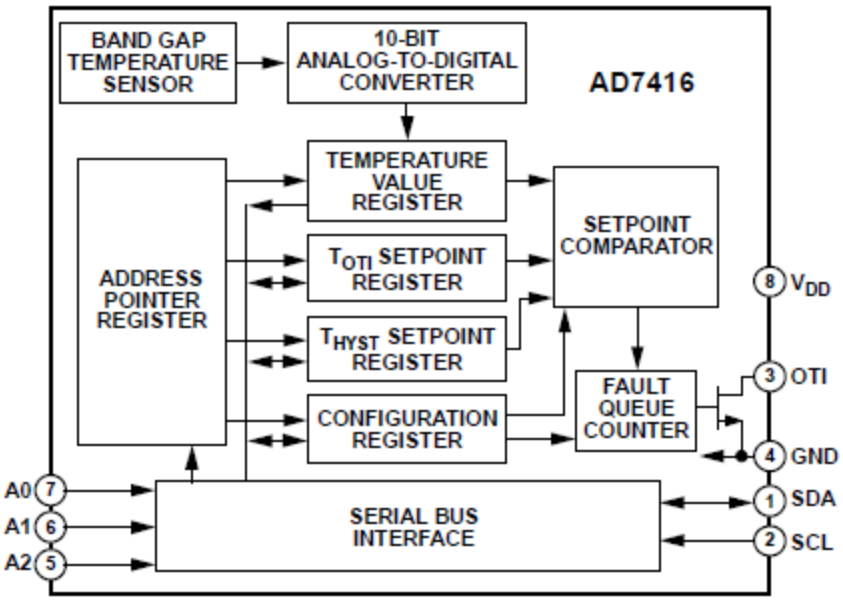
- 10-bit μετατροπέας αναλογικού σήματος σε ψηφιακό, με 15μs και 30μs χρόνους μετατροπής.
- Αισθητήρας θερμοκρασίας που μετρά θερμοκρασίες από -40^{o}C έως +125^{o}C.

- Δείκτης θερμοκρασίας πέρα του ορίου μέτρησης.
- Αυτόματη πτώση ισχύς στο τέλος της μετατροπής.
- Όριο παροχής τάσης: 2,7V – 5,5V.
- Συμβατός με την I^2C επικοινωνία.
- Η επιλέξιμη τμηματική διεύθυνση επιτρέπει τη σύνδεση μέχρι 8 συσκευών AD7416 σε ένα ενιαίο bus. και
- Ο AD7416 είναι μια ανώτερη αντικατάσταση για τους αισθητήρες LM75.

4.2.2. ΟΙ ΚΑΤΑΧΩΡΗΤΕΣ ΤΟΥ AD7416

Σε αντίθεση με τους AD7417 και AD7418 που έχουν επτά καταχωρητές, ο AD7416 έχει πέντε (Βλ. Εικονα 4.1). Αναλυτικότερα οι καταχωρητές είναι οι εξής:

- Temprerature Value Register (Read-Only Address 0x00)
- T_{OTI} Setpoint Register (Read/Write Address 0x03)
- T_{HYST} Setpoint Register (Read/Write Address 0x02)
- Configuration Register (Read/Write Address 0x01)
- Address Pointer Register (Selects data register for read/write)



Εικόνα 4.1 Το εσωτερικό μπλοκ διάγραμμα του AD7416

4.2.2.1. Ο ADDRESS POINTER REGISTER

Ο Address Pointer Register είναι καταχωρητής 8-bit που αποθηκεύει μια διεύθυνση που οδηγεί σε έναν από τους τέσσερις καταχωρητές. Το πρώτο byte δεδομένων εγγραφής του AD7416 είναι η διεύθυνση ενός από τους καταχωρητές δεδομένων, το οποίο είναι αποθηκευμένο στον address pointer register και επιλέγει τα δεδομένα του καταχωρητή στον οποίο είναι γραμμένα τα επόμενα bytes δεδομένων. Μόνο τα τρία LSBs του address pointer καθορίζουν τον καταχωρητή δεδομένων όπως φαίνεται στην εικόνα 4.2.

P7[1]	P6[1]	P5[1]	P4[1]	P3[1]	P2	P1	P0
0	0	0	0	0	Register select		

[1] P3 to P7 must be set to 0.

P2	P1	P0	Registers
0	0	0	Temperature value
0	0	1	Configuration register
0	1	0	T_{HYST} setpoint
0	1	1	T_{OTI} setpoint
1	0	0	ADC value (AD7417/AD7418 only)
1	0	1	Config2 (AD7417/AD7418 only)

Εικόνα 4.2 Address Pointer Register

4.2.2.2. TEMPERATURE VALUE REGISTER (ADDRESS 0X00)

Ο Temperature Value Register είναι καταχωρητής 16-bit που λειτουργεί μόνο για ανάγνωση της θερμοκρασίας. Τα 10 MSBs, όπως φαίνονται στην εικόνα 4.3, είναι για την ανάγνωση της θερμοκρασίας, ενώ τα bit από το D5 έως το D0 δεν χρησιμοποιούνται.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6
MSB	B8	B7	B6	B5	B4	B3	B2	B1	LSB

Εικόνα 4.3 Τα 10bit ανάγνωσης της θερμοκρασίας.

Ο πίνακας 4.1 παρουσιάζει την μορφή των δεδομένων της θερμοκρασίας. Οι τιμές από -128°C έως +127°C είναι οι θεωρητικές τιμές που μπορεί να πάρει ο αισθητήρας θερμοκρασίας. Στην πράξη, το εύρος μέτρησης θερμοκρασίας

περιορίζεται στην λειτουργία της συσκευής. Όπως ο AD7416 που μπορεί να μετρήσει από -40°C έως +125°C. Σε αυτό το σημείο να αναφέρουμε ότι ο AD7416 έχει μια απόκλιση ±2°C.

Temperature	Digital Output
–128°C	10 0000 0000
–125°C	10 0000 1100
–100°C	10 0111 0000
–75°C	10 1101 0100
–50°C	11 0011 1000
–25°C	11 1001 1100
–10°C	11 1101 1000
–0.25°C	11 1111 1111
0°C	00 0000 0000
+0.25°C	00 0000 0001
+10°C	00 0010 1000
+25°C	00 0110 0100
+50°C	00 1100 1000
+75°C	01 0010 1100
+100°C	01 1001 0000
+125°C	01 1111 0100
+127°C	01 1111 1100

Πίνακας 4.1 Θεωρητικές τιμές θερμοκρασίας.

4.2.2.3. CONFIGURATION REGISTER (ADDRESS 0X01)

Ο Configuration Register είναι ένας καταχωρητής των 8-bit που χρησιμοποιείται για να καθορίσει τον τρόπο λειτουργίας ανάγνωσης / εγγραφής του AD7416. Στην εικόνα 4.4 διακρίνονται τα 8bit και τι καθορίζει το καθένα.

D7	D6	D5	D4	D3	D2	D1	D0
Channel selection			Fault queue		OTI polarity	Cmp/Int	Shutdown

Εικόνα 4.4 Τα bit του Configuration Register

Αναλυτικά, τα τρία πρώτα bit D7 έως D5 είναι για τον έλεγχο των καναλιών όπως περιγράφεται στον πίνακα 4.2. Πιο συγκεκριμένα για τον AD7416 τα bit αυτά θα είναι πάντα 000. Τα αμέσως δυο επόμενα (D4, D3) χρησιμοποιούνται για να

οριστεί το μήκος της ουράς σφάλματος (Βλ. πίνακα 4.3). Το bit D2 ορίζει την έξοδο OTI (είναι μια λογική έξοδος που δεν χρειάστηκε να την χρησιμοποιήσουμε, γι' αυτό είναι ασύνδετη στο κύκλωμα μας). Ενώ το bit D1 επιλέγει τον τρόπο λειτουργίας να γίνεται σύγκριση ή διακοπή και τέλος το bit D0=1 για λειτουργία Shutdown (προεπιλογή: D0=0).

D7	D6	D5	Channel Selection
0	0	0	Temperature sensor (all parts), Channel 0
0	0	1	A_{IN1} (AD7417 only), Channel 1
0	1	0	A_{IN2} (AD7417 only), Channel 2
0	1	1	A_{IN3} (AD7417 only), Channel 3
1	0	0	A_{IN4} (AD7417) and A_{IN} (AD7418), Channel 4

Πίνακας 4.2 Επιλογή καναλιού.

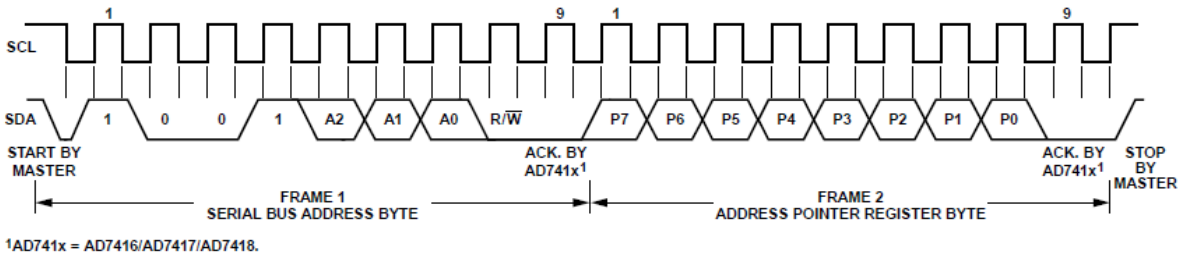
D4	D3	Number of Faults
0	0	1 (power-up default)
0	1	2
1	0	4
1	1	6

Πίνακας 4.3 Ορισμός του μήκους σφάλματος.

4.2.3. ΕΓΓΡΑΦΗ ΣΤΟΝ AD7416

Ανάλογα με τον καταχωρητή εγγραφής, υπάρχουν τρείς διαφορετικές μέθοδοι για την εγγραφή. Για τον AD7416 χρησιμοποιείται μόνο μια από τις τρείς.

Τα δεδομένα γράφονται στον address pointer register για να μπορούν να διαβαστούν εκ των υστέρων. Για να διαβάσει από έναν συγκεκριμένο καταχωρητή, ο address pointer θα πρέπει να περιέχει την διεύθυνση του εν λόγω καταχωρητή. Αν δεν το κάνει, η σωστή διεύθυνση πρέπει να συντάσσεται με την μορφή ενός byte από τον address pointer για να πραγματοποιηθεί η εγγραφή, όπως φαίνεται στην εικόνα 4.4. Η λειτουργία εγγραφής αποτελείται από μια διεύθυνση ακολουθίας στο Bus που ακολουθείται από ένα byte του address pointer. Δεν γράφονται δεδομένα σε οποιοδήποτε καταχωρητή δεδομένων.

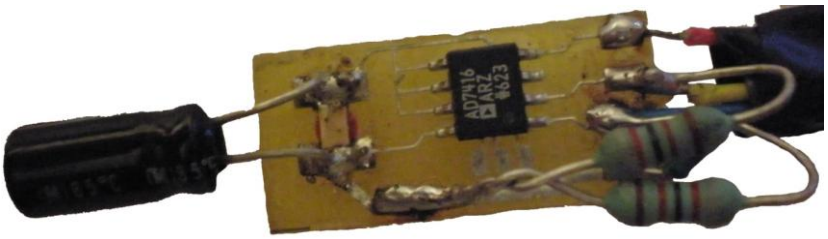


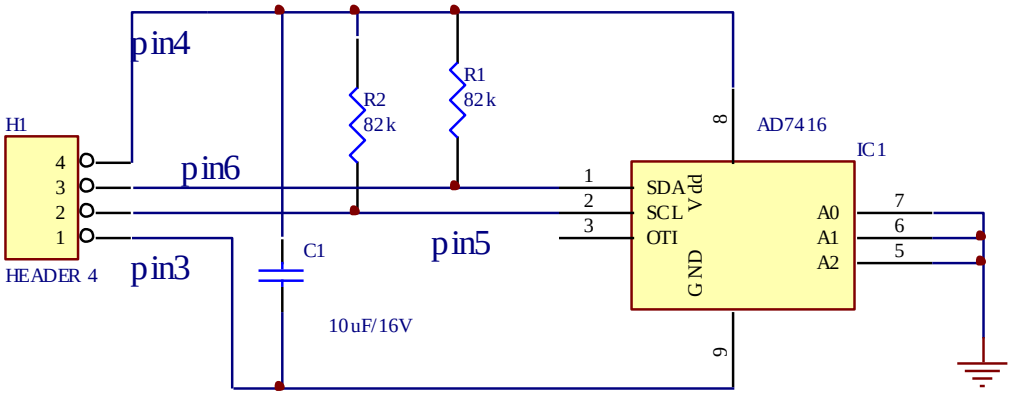
Εικόνα 4.4 Γράφοντας στον Address Pointer Register για να επιλέξει έναν καταχωρητή δεδομένων, για την μετέπειτα λειτουργία ανάγνωσης.

Στον AD7416 το byte ανάγνωσης αποτελείται από μια σταθερή ακολουθία bit (1001) συν τα bit που δημιουργούνται ανάλογα με τον τρόπο σύνδεσης στα ποδαράκια A2, A1 και A0. Όπως αναφέραμε παραπάνω για τον AD7416 επιλέγεται πάντα των καταχωρητή Temperature Value (000), δηλαδή τα ποδαράκια συνδέονται στην γείωση.

4.3. Η ΚΥΚΛΩΜΑΤΙΚΗ ΔΟΜΗ ΤΟΥ ΑΙΣΘΗΤΗΡΑ

Στην παρακάτω εικόνα φαίνεται ο αισθητήρας όπως έχει συνδεθεί με το καλώδιο του NXT. Επίσης στην εικόνα 4.5 φαίνεται η κυκλωματική δομή του αισθητήρα σύμφωνα με όλα όσα έχουμε αναφέρει στα κεφάλαια μας.





Εικόνα 4.5 Κυκλωματική δομή του AD7416 για το NXT.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1) C Για Μηχανικούς, Εκδόσεις Τζιόλα
   Συγγραφείς: Tan H.H., D' Orizio T.B.
   (Επιμέλεια μετάφρασης: Μανωλάκης Δημήτριος, Χαρίτων Πολάτογλου)
2) I2C Bus Specification
   http://www.nxp.com/acrobat_download2/literature/9398/39340011.pdf
3) Μετάδοση δεδομένων μέσω του πρωτοκόλλου I^2C
   Διπλωματική Εργασία του Νικόλαου Μουζή
4) Digital Temperature Sensor AD7416/AD7417/AD7418
   http://www.analog.com/static/imported-files/data_sheets/AD7416_7417_7418.pdf
5) ROBOTC I2C Howto (Part I) http://mightor.wordpress.com/2009/11/08/robotc-i2c-howto-part-i/
6) http://www.embedded.com/story/OEG20010718S0073
7) http://www.edc.uoc.gr/~sanagn/wordpress/?tag=nxt
8) http://en.wikipedia.org/wiki/Lego_Mindstorms_NXT#Programming
9) NXT Programming Software
   http://www.teamhassenplug.org/NXT/NXTSoftware.html

ΚΩΔΙΚΑΣ

```
#pragma config(Sensor, S3,    lightSensor,        sensorLightActive)
#pragma config(Sensor, S4,    SonarSensor,        sensorSONAR)
#pragma config(Motor,  motorB,          left,          tmotorNormal, PIDControl, encoder)
#pragma config(Motor,  motorC,          right,         tmotorNormal, PIDControl, encoder)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//


void turnLeft()
{

	nMotorEncoder[motorB] = 0;
	while(nMotorEncoder[motorB] < 180)
	{
		motor[motorC] = -20;
		motor[motorB] = 20;
	}

	motor[motorC] = 0;
	motor[motorB] = 0;

}

void turnRight()
{

	nMotorEncoder[motorC] = 0;
	while(nMotorEncoder[motorC] < 360)
	{
		motor[motorC] = 20;
		motor[motorB] = -20;
	}

	motor[motorC] = 0;
	motor[motorB] = 0;

}

void gothedark()
{
  if(SensorValue(lightSensor) >45)
  {
    motor[motorB]= 30;
    motor[motorC]= 25;
  }
  else
  {
    motor[motorB]= 25;
    motor[motorC]= 30;
  }
}
```

```
int TempSensor();
task main()
{
  while(SensorValue(lightSensor) < 75 )
  {
    if(SensorValue(SonarSensor) >27)
    {
     gothedark();
    }
    if(SensorValue(SonarSensor) <26)
            {
     turnLeft();
     wait1Msec(500);
    }

            if(SensorValue(SonarSensor) <25)
            {
             turnRight();
             wait1Msec(500);
            }

            if (TempSensor() > 26)
            {
             turnRight();

             motor[motorB]= 0;
             motor[motorC]= 0;
             wait1Msec(15000);
            }
  }
}

int TempSensor()
{

    byte Buffer_Temperature[1];
    int temperature_1;

    const tSensors AD7416 = S1;
    SensorType[AD7416] = sensorI2CCustom;

    byte I2C_Message[4] = {0x03, 0x90, 0x01, 0x00};
    sendI2CMsg(AD7416, I2C_Message[0], 0);
    while(nI2CStatus[AD7416] == STAT_COMM_PENDING)
    {
      ;
    }
      {
       eraseDisplay();
       byte I2C_Message[3] = {0x02, 0x90, 0x00};
       sendI2CMsg(AD7416, I2C_Message[0], 1);
       while(nI2CStatus[AD7416] == STAT_COMM_PENDING)
       {
         ;
       }
       readI2CReply(AD7416, Buffer_Temperature[0], 1);
       temperature_1 = Buffer_Temperature[0];
       nxtDisplayCenteredBigTextLine(0,"PTYXIAKI");
       nxtDisplayTextLine(2,"Kalaitzidis Fotis");
```

```
        nxtDisplayTextLine(3,"Tarsenis Dimitris");
                        nxtDisplayCenteredTextLine(4,"Temp is:");
        nxtDisplayCenteredBigTextLine(6,"%doC", temperature_1);
        wait1Msec(50);
       }

    return (temperature_1);
}
```