ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ (Α.Τ.Ε.Ι.Θ.)
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ (ΣΤΕΦ)
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΑΥΤΟΜΑΤΙΣΜΟΥ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

"Χρήση του ROS για συγκέντρωση τιμών από αυτό-οδηγούμενο όχημα με Arduino"

ΚΑΡΑΜΗΤΣΟΣ ΓΕΩΡΓΙΟΣ
ΑΜ: "102754"
ΣΙΑΦΑΚΑΣ ΘΕΟΦΑΝΗΣ
ΑΜ: "102696"

ΕΠΙΒΛΕΠΩΝΚΑΘΗΓΗΤΗΣ: ΔΗΜΗΤΡΙΟΣ ΜΠΕΧΤΣΗΣ, ΚΑΘΗΓΗΤΗΣ ΕΦΑΡΜΟΓΩΝ

ΘΕΣΣΑΛΟΝΙΚΗ, ΜΑΡΤΙΟΣ 2018

Περίληψη - Abstract

Περίληψη

Από τις αρχές του περασμένου αιώνα η ιστορία της ανθρωπότητας μετράει χιλιάδες επιτεύγματα στον τομέα της βιομηχανίας. Από την πρώτης βιομηχανική φάση όπου ξεκίνησε ο εκβιομηχανισμός των παραγωγικών διαδικασιών μέχρι και σήμερα, στην τέταρτη βιομηχανική φάση όπου η χρήση του Διαδικτύου των Πραγμάτων και των Ρομποτικών Συστημάτων οδηγούν στην Έξυπνη Βιομηχανία. Τα τελευταία χρόνια η χρήση Αυτοοδηγούμενων Οχημάτων (Autonomous Guided Vehicles) έχει αυξήσει την παραγωγή, έχει μειώσει τον περιβαλλοντικό αντίκτυπο και τη σωματική και ψυχική φθορά του εργατικού δυναμικού, καθώς ένα ρομπότ είναι σε θέση να διεκπεραιώσει εργασίες μονότονες αλλά και επικίνδυνες για τον άνθρωπο.

Η παρούσα πτυχιακή αποτελείται από δυο κομμάτια. Το πρώτο κομμάτι αποτελεί έναν οδηγό ο οποίος προσφέρει εκτενή ανάλυση της επικοινωνίας μεταξύ του περιβάλλοντος ανάπτυξης ρομποτικών εφαρμογών ROS και του μικροελεγκτή Arduino. Το δεύτερο κομμάτι αποτελεί την προσομοίωση μιας βιομηχανικής εφαρμογής όπου το όχημα ακολουθεί μια προκαθορισμένη διαδρομή και επιλέγει εναλλακτικές διαδρομές όταν λάβει τα κατάλληλα ερεθίσματα από βιομηχανικό χώρο.

Abstract

During the last decades the industrial revolution has changed the backbone of the factories. The industry's transformation started from the outbreak of the first industrial phase where the manufacturing automation procedures took over, to the today's fourth industrial phase where the use of Information and Communication Technologies, the Internet of Things and Robotic Technologies are paving the way to the factories of the future. Automated Guided Vehicles are used to increase the production rate and improve the Health Safety and Environmental conditions.

This paper consists of two distinct parts. The first part could be used as a guide that offers extensive analysis on the communication between the Robot Operating System and the Arduino microcontroller. The second part is the implementation of a prototype that could be used in a real world application which allows the vehicle to follow a predefined path and take real time decisions.

Ευχαριστίες

Στο σημείο αυτό θα θέλαμε να ευχαριστήσουμε όλους όσοι συνέβαλαν στην ολοκλήρωση της διπλωματικής αυτής εργασίας και συγκεκριμένα:

- τον καθηγητή κ. Δημήτριο Μπεχτσή του Τμήματος Μηχανικών Αυτοματισμού, για την ουσιαστική καθοδήγησή τους ως επιβλέποντα της πτυχιακής αυτής εργασίας,
- την οικογένειά μας και τους φίλους μας για τη συμπαράσταση και την υπομονή τους κατά τη διάρκεια των φοιτητικών μας χρόνων.

Θεσσαλονίκη, Μάρτιος 2018

Καραμήτσος Γεώργιος

Σιαφάκας Θεοφάνης

Περιεχόμενα

1. Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται ανάλυση του ορισμού της ρομποτικής και παρατίθενται αναφορές για την θεματική προέλευση των ρομπότ μέσα από την μυθολογία διαφορετικών λαών και την επιρροή που άσκησαν ανά τους αιώνες στην λογοτεχνία, τον κινηματογράφο και την επιστήμη.

1.1. Ρομποτική

Η ιστορία της ρομποτικής διακατέχεται από την επιθυμία του ανθρώπου να μετατρέψει την φαντασία σε πραγματικότητα. Είναι μια ιστορία πλούσια σε κινηματογραφική δημιουργικότητα, επιστημονική πρωτοπορία και καινοτόμο όραμα. Ο ορισμός του ρομπότ είναι αμφιλεγόμενος στην επιστημονική κοινότητα, καθώς υπάρχουν δύο όψεις στο ίδιο νόμισμα. Σε μια ενδιαφέρουσα προσέγγιση για την περιγραφή και τον ορισμό της ρομποτικής αναφέρεται ότι η ρομποτική «ασχολείται με την παραγωγή ελεγχόμενων κινήσεων φυσικών αντικειμένων από υπολογιστή σε μια ευρεία ποικιλία ρυθμίσεων»[1].

1

Halperin D. Kavraki L.E., Latombe, J.C. (1997). Robotics *CRC Handbook of Discrete and Computational Geometry*, J.E. Goodman and J. O'Rourke (eds.), CRC Press, Boca Raton, FL, Chapter 41, p. 755-778.

Η μία όψη αντιπροσωπεύει την εικόνα εκείνη που βλέπουμε στις ταινίες επιστημονικής φαντασίας, ένα ανδροειδές με χαρακτηριστικά ίδια με εκείνα του ανθρώπου δημιουργού του. Η άλλη όψη είναι το μηχανικό, επαναλαμβανόμενο μα αποτελεσματικό ρομπότ του βιομηχανικού αυτοματισμού. Πολλές είναι οι σκοπιές από τις οποίες μπορούμε να αντιληφθούμε την έννοια του ρομπότ, καθώς η επιστήμη της ρομποτικής παρομοιάζεται ως ένα σύμπαν με άπειρες δυνατότητες και προοπτικές. Ένα είναι το σίγουρο, πως τα ρομπότ τις τελευταίες δεκαετίες έχουν κερδίσει σταδιακά έδαφος στην καθημερινότητα μας και θα συνεχίσουν ως ένα αναπόσπαστο κομμάτι της εξέλιξης της ανθρωπότητας.

1.2. Αναφορές των Ρομπότ στην Μυθολογία

Πολλοί είναι οι ερευνητές που εντοπίζουν το υπόβαθρο της ρομποτικής στην αρχαιότητα, καθώς υπάρχουν πολλές ιστορίες σε όλο τον κόσμο που κάνουν αναφορές σε τεχνητά όντα.

Στην ελληνική μυθολογία εξιστορείται η ιστορία του Πυγμαλίωντα, ο οποίος παράφορα ερωτευμένος με μια γυναίκα, λαξεύει την μορφή της από ελεφαντόδοντο και την ονομάζει Γαλάτεια[2]. Αφού προσευχήθηκε στην θεά Αφροδίτη η επιθυμία του πραγματοποιήθηκε, η Γαλάτεια ήρθε στην ζωή και έγινε γυναίκα του.

Στην εβραϊκή μυθολογία γίνεται αναφορά στο Golem, ένα άγαλμα φτιαγμένο από πηλό, το οποίο με την χρήση ενός μαγικού παπύρου ερχόταν στην ζωή εκτελώντας επαναλαμβανόμενα καθήκοντα[3] . Τον 15^{ο} αιώνα ο Bombastus von Hohenheim, γνωστός ως Paracelsus, κάνει

[2] Gilbert W. S.. (1886). *Pygmalion and Galatea: An original mythological comedy in three acts*. Chicago Publishing Company.

[3] Idel M. (1990). *Golem: Jewish Magical and Mystical Traditions on the Artificial Anthropoid*. Albany, New York:

αναφορά στο "Homonculus" ή αλλιώς "ανθρωπάριο" στο De natura rerum[4] το 1537 . Πρόκειται για ένα μικρό πλάσμα το οποίο έχει την ικανότητα να εισέρχεται στο μυαλό ενός ανθρώπου και να επηρεάζει τις σκέψεις του.

Ο Frankenstein, η δημοφιλής νουβέλα της Mary Shelley[5], εξιστορεί την ιστορία του επιστήμονα Victor Frankenstein ο οποίος χρησιμοποιώντας γνώσεις ανατομίας και αλχημείας δημιουργεί ένα "τέρας" το οποίο αδυνατεί να ελέγξει, έχοντας ως συνέπεια τον χαμό της ζωής του και των αγαπημένων του .

Η έμπνευση για την δημιουργία αυτών των ιστοριών είναι η επιθυμία του ανθρώπου να αγγίξει την τελειότητα, εκφράζοντας με αυτό τον τρόπο την ανάγκη για φυγή από την υποτέλεια και το σφάλμα της ανθρώπινης φύσης.

1.3. Ρομπότ και Κινηματογραφικές Ταινίες

Ο ερχομός των κινηματογραφικών ταινιών έφερε στην ζωή πολλά από αυτά τα μυθικά πλάσματα, αλλά δημιούργησε και μια φαινομενικά αστείρευτη πηγή νέων τεχνητών πλασμάτων[6].

Το 1926, ο σκηνοθέτης Fritz Lang στην ταινία του "Metropolis" εισάγει την Maria, ένα ρομπότ το οποίο έχει ως σκοπό να ξεγελάσει τον "Feder" και τους κατοίκους της "Metropolis"[7].

State University of New York Press. p. 296.

4 Grafton, A. (1999). *Natural Particulars: Nature and the Disciplines in Renaissance Europe*. MIT Press.

5 Shelley, M. (1818) . *Frankenstein or, The Modern Prometheus* . Lackington, Hughes, Harding, Mavor & Jones.

6 Kurfess Th. R. (2004) p.2.

7 Lang F. (2003). *Interviews* ed. by B. K. Grant, University Press of Mississippi,

Το 1951, η ταινία "The day the earth stood still", εισάγει το ρομπότ Dort[8].

Το 1956, στην ταινία "Forbiden Planet" κάνει την εμφάνιση του ο Robby the Robot, ένα ρομπότ το οποίο έχει εμφανιστεί σε πολλές ταινίες[9]. Το 1977, στην ταινία "Star Wars", κάνουν την εμφάνιση τους δύο από τα πιο εμβληματικά ρομπότ στην ιστορία του κινηματογράφου, ο R2-D2 και ο C3PO[10].

Το 1982, ο Ridley Scott φέρνει στην μεγάλη οθόνη την ταινία "Blade Runner", βασισμένη στο βιβλίο "Do Androids dream of electric sheep" του Phillip K. Dick, όπου τα ανδροειδή ρομπότ ονόματι Replicants επαναστατούν εναντίον των ανθρώπων δημιουργών τους[11].

Η τηλεόραση και ο κινηματογράφος εμβαθύνουν στην σχέση ανθρώπου-μηχανής, απεικονίζοντας την ευφυΐα των ρομπότ και την κοινή τους πορεία με την ανθρωπότητα σε τεχνολογικές ουτοπίες, αλλά και την ενδεχόμενη απειλή της ανθρώπινης ματαιοδοξίας.

1.4. Εφευρέσεις που οδήγησαν στην Ρομποτική

Το πεδίο της ρομποτικής έχει εξελιχθεί τις τελευταίες χιλιετίες χωρίς καμία αναφορά στην λέξη ρομπότ μέχρι τις αρχές του 20ου αιώνα. Το 270 π.Χ., ο αρχαίος Έλληνας εφευρέτης Κτησίβιος με καταγωγή από την Αλεξάνδρεια, δημιούργησε ένα υδραυλικό ρολόι το οποίο ονόμασε κλεψύδρα[12]. Λειτουργώντας με νερό, η κλεψύδρα χρησιμοποιούσε μια χορδή προσκολλημένη σε

8 North E. H (1951). The Day the Earth stood still . The movie screenplay.

9 Schwartzman R. (1999). Engenderneered Machines in Science Fiction. *Studies in Popular Culture*. Vol. 22, No. 1 p. 75-87.

10 Breazeal C. L. (2002). *Designing Social Robots.* MIT Press, p. 240.

11 Kerman J. K. (1997). *Retrofitting Blade Runner: Issues in Ridley Scott's Blade Runner and Phillip K. Dick's Do Androids dream of electric sheep*? University of Wisconsin Press, p.10.

12 Kurfess Th. R. (2004). Robotics and Automation Handbook, CRC Press, p. 2.

Wadhwa A.S. (2012). Role of it in manufacturing sector. Proceedings of the National Conference on Trends and

ένα σωσίβιο, η οποία τεντωνόταν κατά μήκος μιας τροχαλίας ώστε να εντοπίζει την ώρα[13] . Ο μηχανισμός τραβούσε τα βλέμματα των περαστικών, οι οποίοι ενθουσιασμένοι τον παρακολουθούσαν καθώς απέδιδε την ώρα ή τους έκλεβε την ώρα, για αυτό και η εναλλακτική του ονομασία ήταν "water thief"[14] .

Το 1801 ο Joseph Marie Jacquard κατασκεύασε ένα αργαλειό ο οποίος χρησιμοποιούσε μια σειρά από διάτρητες καρτέλες, με τις οποίες μπορούσε να ελέγξει την επανάληψη των μοτίβων που χρησιμοποιούνταν για να υφάνουν χαλιά και κλωστές[15]. Τον 19ο αιώνα ο Charles Babbage, μαθηματικός και εφευρέτης ήθελε να απαλλαχθεί από τον χειρωνακτικό έλεγχο των μαθηματικών πινάκων. Εκείνη την εποχή μαθηματικοί πίνακες είχαν ένα μεγάλο φάσμα εφαρμογών στην φυσική, στην μηχανική, στον κατασκευαστικό τομέα, στον τραπεζικό τομέα. Έτσι κάνοντας χρήση των διάτρητων καρτελών του Joseph Jacquard, ο Babbage κατασκεύασε μια αυτόματη αριθμομηχανή, οι αρχές της οποίας οδήγησαν στην δημιουργία των υπολογιστών και του προγραμματισμού[16].

Το 1873 δίνεται στον Christopher Miner Spencer, τον εφευρέτη του αυτόματου τουφεκιού [17], η πατέντα για την κατασκευή ενός τόρνου, ο οποίος περιλάμβανε ένα κεντροφόρο άξονα και έναν

Advances in Mechanical Engineering, YMCA University of Science & Technology, p. 550-554.

[13] Wadhwa A.S. (2012). Role of it in manufacturing sector. Proceedings of the National Conference on Trends and

Advances in Mechanical Engineering, YMCA University of Science & Technology, p. 550-554.

[14] Landels J. G. (1979). Water-Clocks and Time Measurement in Classical Antiquity, *Endeavour*, Vol. 3, p. 32-37.

[15] Heath F. G. Origins of the binary code, Scientific American JSTOR, Vol. 227, p. 76-83.

[16] Hyman A. (1985). *Charles Babbage: Pioneer of the Computer*, Princeton University Press.

[17] URL: http://americansocietyofarmscollectors.org/wp-content/uploads/2013/05/B039_Lewis.pdf– 06.08.2017

πυργίσκο. Η εφεύρεση του έδωσε άλλη διάσταση στην κατασκευή των τόρνων και αυτοματοποίησε περαιτέρω την διαδικασία[18] .

Το 1892 ο Seward Babbitt κατασκευάζει έναν κινητήριο γερανό με δαγκάνα ο οποίος χρησιμοποιήθηκε για να μεταφέρει ράβδους από ένα κλίβανο. Η εφεύρεση αναφέρθηκε για πρώτη φορά στην ιστορία της ρομποτικής όχι σαν ρομπότ, αλλά σαν παροχή τεχνολογικής καινοτομίας [19] .

Ο Nikola Tesla το 1898 στο Madison Square Garden της Νέας Υόρκης παρουσίασε μια αυτοκινούμενη βάρκα την οποία μπορούσε να την ελέγξει μέσω ενός ραδιοπομπού με την χρήση ραδιοκυμάτων. Ο ίδιος είχε δηλώσει πως έβλεπε τον εαυτό του ως ένα αυτόματο αντιδρώντας σε ερεθίσματα και καταστάσεις και πως μια μέρα οι άνθρωποι θα προικίσουν μια μηχανή με το δικό της μυαλό. Όταν ο Nikola Tesla ρωτήθηκε εάν η βάρκα του ήταν ικανή για να χρησιμοποιηθεί ως σύστημα μεταφοράς εκρηκτικών, εκείνος απάντησε: "Δεν βλέπεις μια ασύρματη τορπίλη. Βλέπεις το πρώτο από την φυλή των ρομπότ, μηχανικοί άνθρωποι οι οποίοι θα διεκπεραιώνουν την κοπιαστική δουλειά της ανθρώπινης φυλής" [20].

1.5. Η Πρώτη Χρήση της Λέξης Ρομπότ

[18] Wadhwa A.S. (2012). Role of it in manufacturing sector. *Proceedings of the National Conference on Trends and*

Advances in Mechanical Engineering, YMCA University of Science & Technology, Faridabad, Haryana, Oct 19-

20, p. 550-554.

[19] Nocks L. (2007). *The Robot: The Life Story of a Technology*, p.45, Greenwood Press.

[20] Cheney M. (1981). *Tesla: Man Out of Time*, Touchstone, p. 162.

Η λέξη ρομπότ δεν υπήρχε στο λεξιλόγιο της βιομηχανίας και της επιστήμης, πόσο μάλλον σε εκείνο της λογοτεχνίας μέχρι το 1920. Η λέξη robot προέρχεται από την λέξη της Παλαιάς Εκκλησιαστικής σλαβικής γλώσσας «rabota»[21], που σημαίνει δουλεία. Η λέξη έχει επίσης ρίζες στα γερμανικά, ρώσικα, πολωνικά και τσέχικα και σε αρκετές άλλες σλαβικές γλώσσες. Η λέξη rabota είναι προϊόν της δουλοπαροικίας της Κεντρικής Ευρώπης, όπου το χρέος ενός οφειλέτη ξεπληρωνόταν είτε με καταναγκαστική εργασία είτε με παροχή υπηρεσιών στον δανειστή[22].

Το 1920, ο Karel Čapek έγραψε το θεατρικό έργο "Rossum's Universal Robots", γνωστό και ως R.U.R το οποίο έκανε πρεμιέρα στις 25 Ιανουαρίου του 1921 στην Πράγα και μέχρι το 1923 είχε εισάγει την λέξη ρομπότ σε 30 γλώσσες[23] .

Ο Karel Capek γεννήθηκε το 1890 στην πόλη Male Svatonovice της Αυστροουγγαρίας, πλέον σήμερα κομμάτι της Δημοκρατίας της Τσεχίας. Ακολουθώντας το τέλος του 1ου Παγκοσμίου Πολέμου, η πένα του απέκτησε ένα βαθύ πολιτικό χρώμα γράφοντας εκθέσεις για τον Ναζισμό, τον ρατσισμό και την δημοκρατία σε μια Ευρώπη υπό κρίση.

Το R.U.R λαμβάνει χώρα σε ένα μακρινό μέλλον όπου οι εργάτες δεν γεννιούνται, αλλά κατασκευάζονται και λέγονται ρομπότ. Τα ρομπότ δημιουργούνται σε ένα απομονωμένο νησί από την εταιρεία "Rossum's Universal Robots" η οποία ανήκει σε μια ομάδα αποτελούμενη από έναν πατέρα και γιο του. Ο πατέρας, Old Rossum είναι ο επιστήμονας ο οποίος έδωσε πνοή στα ρομπότ με απώτερο σκοπό να δημιουργήσει ένα ον δίχως ατέλειες, κυριευμένος από την ματαιοδοξία του να καταλάβει την θέση του Δημιουργού. Ο γιος του, Young Rossum έβλεπε τα

[21] URL: http://www.etymonline.com/index.php?term=robot – 09.08.2017

[22] URL: www.npr.org/2011/04/22/135634400/science-diction-the-origin-of-the-word-robot – 09.08.2017

[23] Čapek, K. (1920). *Rossum's Universal Robot (R.U.R)*. Translated by Paul Selver and Nigel Playfair Mineola, New York: Dover Publications.

ρομπότ ως μια κερδοφόρα επιχείρηση. Τα ρομπότ ήταν φτιαγμένα από οργανική ύλη ώστε να είναι οικονομικά και αποδοτικοί εργάτες, προικισμένοι με την ικανότητα να συγκρατούν οποιαδήποτε πληροφορία, αλλά χωρίς την ιδιοσυγκρασία και την δυνατότητα να δημιουργήσουν δικές τους σκέψεις, απαλλαγμένα από τον πόνο του να είναι άνθρωποι. Αναπόφευκτα ξεσπά πόλεμος ανάμεσα στα ρομπότ και στους ανθρώπους με τα ρομπότ να αναδεικνύονται οι νικητές, όμως η φόρμουλα και η τεχνογνωσία απαραίτητη για την δημιουργία τους έχει καταστραφεί. Τελικά τα ρομπότ απρόσμενα αναπτύσσουν την ικανότητα να αισθάνονται, να αγαπούν, να νιώθουν ανθρώπινα συναισθήματα μην έχοντας πια ανάγκη την χρήση της φόρμουλας[24] .

1.6. Η Πρώτη Χρήση της Λέξης Ρομποτική

Ο Isaac Asimov ήταν ένας ακόμη συγγραφέας επιστημονικής φαντασίας που είχε ένα βαθυστόχαστο αντίκτυπο στην ιστορία της ρομποτικής. Γεννήθηκε στις 2 Ιανουαρίου του 1920 στην πόλη Petrovichi της Ρωσίας, όπου στην πορεία η οικογένεια του μετανάστευσε στις Ηνωμένες Πολιτείες της Αμερικής το 1923. Αποφοίτησε από το Columbia University το 1938 και το τελείωσε το 1948 το διδακτορικό του[25]. Ο Isaac Asimov έγινε γνωστός για τη συγγραφή μικρών ιστοριών επιστημονικής φαντασίας.

Κατά την πρώτη περίοδο της συγγραφικής του καριέρας συνέβαλε σε μεγάλο βαθμό στον δημιουργικό κλάδο της ρομποτικής.

Το 1950 εκδόθηκε το "I, Robot", ένα βιβλίο που περιελάμβανε 9 μικρές ιστορίες η θεματολογία των οποίων αφορούσε τα ρομπότ και την τεχνολογία και στις οποίες έγιναν κατά

[24] Cork S. J. (2011). *From Prometheus to Pistorious: a genealogy of physical ability*. A thesis submitted to the Department of Sociology In conformity with the requirements for the degree of Masters of Arts, Queen's University Kingston, Ontario, Canada

[25] URL: https://www.biography.com/people/isaac-asimov-9190737 – 12.8.2017

καιρούς αναφορές για τους νόμους, τους οποίους πρέπει να εφαρμόζονται για την ομαλή και ασφαλή λειτουργία των ρομπότ[26]. Οι 3 νόμοι του Asimov εμφανίσθηκαν για πρώτη φορά σε συνοπτική μορφή στην μικρή ιστορία "Runaround"[27]. Πρόκειται για την ιστορία ενός ρομπότ, στο οποίο έχει ανατεθεί η εξόρυξη ορυκτών από την επιφάνεια του πλανήτη Ερμή. Καθώς εξελίσσεται η ιστορία τα ίχνη του ρομπότ χάνονται. Διεξάγεται έρευνα και προκύπτει πως το ρομπότ έχει παραβιάσει 2 νόμους και είναι σε κατάσταση ανυπακοής, έχοντας μπει σε μια ρουτίνα διανύοντας κύκλους στην επιφάνεια του πλανήτη, εξ ου και το όνομα "Runaround".

Το 1985 ο Isaac Asimov τροποποίησε τη λίστα για να συμπεριλάβει τον Νόμο 0 στους ήδη υπάρχοντες νόμους. Οι 4 Νόμοι της ρομποτικής είναι οι εξής[28] :

Μηδενικός Νόμος: Το ρομπότ δεν θα κάνει κακό στην ανθρωπότητα, ούτε με την αδράνεια του θα επιτρέψει να βλαφτεί η ανθρωπότητα[29] .

Πρώτος Νόμος: Το ρομπότ δεν θα κάνει κακό σε άνθρωπο, ούτε με την αδράνεια του θα επιτρέψει να βλαφτεί ανθρώπινο ον, εφόσον αυτό δεν αντιτίθεται στο μηδενικό νόμο.

Δεύτερος Νόμος: Το ρομπότ πρέπει να υπακούει τις διαταγές που του δίνουν οι άνθρωποι, εκτός αν αυτές αντιτίθενται στον πρώτο νόμο.

Τρίτος Νόμος: Το ρομπότ οφείλει να προστατεύει την ύπαρξη του, εφόσον αυτό δεν αυτό δεν έρχεται σε αντίθεση με τον πρώτο και δεύτερο νόμο.

Ο Isaac Asimov έφυγε από την ζωή τον Απρίλιο του 1992 σε ηλικία 72. Κατά την διάρκεια της ζωής του έλαβε περίπου 100.000 γράμματα, από τα οποία απάντησε προσωπικά περίπου 90.000. [30]

[26] Asimov I. (1950). *I, Robot.* Gnome Press
[27] Asimov, I. (1942). *Runaround*. Robot series, Astounding Science Fiction, Street & Smith.
[28] Etzioni O & Weld D. (1994). The First Law of Robotics: (a call to arms), AAAI p.17.
[29] Anderson M & S. L. Anderson (ed.) (2011*). Machine Ethics*. Cambridge University Press, p. 268.

Η δημιουργία συσκευών και η αυτοματοποίηση ορισμένων διαδικασιών στην βιομηχανία επηρέασε την λογοτεχνία και την ευρύτερη τέχνη, ιδιαίτερα μετά το τέλος του 1ου Παγκοσμίου Πολέμου. Η ρομποτική από τις σελίδες των θεατρικών έργων και φουτουριστικών νουβελών γίνεται αντικείμενο έρευνας σε εργαστήρια πανεπιστημίων μετά το τέλος του 2ου Παγκοσμίου Πολέμου. Οι επόμενες δεκαετίες φέρνουν τον άνθρωπο πιο κοντά στην εποχή της μηχανής.

[30] Kurfess Th. R. (2004), p. 2

2. Βιομηχανικά Ρομποτικά Συστηματα

Σε αυτή την ενότητα αναφέρονται οι πρώτες ρομποτικές εφαρμογές στον τομέα της βιομηχανίας και η επακόλουθη έρευνα και καινοτόμες κατασκευές που ακολούθησαν από το δεύτερο μισό του 20ου μέχρι και σήμερα.

2.1. Η Γέννηση του Βιομηχανικού Ρομπότ

Μετά το πέρας του δευτέρου παγκοσμίου πολέμου, η Αμερική βίωσε μια ισχυρή ώθηση στη βιομηχανία. Η ταχύτατη ανάπτυξη της τεχνολογίας είχε ως αντίκτυπο την δημιουργία των σερβοκινητήρων, τη ψηφιακή λογική, τα ηλεκτρικά στοιχεία στερεής κατάστασης κ.τ.λ.π . Ορόσημο αυτής της περιόδου αποτελεί η συνεργασία του George Devol και του Joseph Engelberger και η τυχαία συνάντησή τους το 1956.

Ο Joseph F.Engelberger γεννήθηκε στις 26 Ιουλίου 1925 στην Νέα Υόρκη. Μεγαλώνοντας ο Engelberger γοητεύεται από τον κόσμο της επιστημονικής φαντασίας και ιδιαίτερα από των όσων είχαν γραφτεί εκείνη την εποχή από τον Isaac Asimov. Ο Joseph F. Engelberger ,φυσικός,μηχανικός και επιχειρηματίας ήταν υπεύθυνος για την γέννηση μιας από τις σημαντικότερες βιομηχανίες,αποκτώντας έτσι την παγκόσμια αναγνώριση ως "πατέρα της ρομποτικής"[31].

[31] URL: https://www.robotics.org/joseph-engelberger/about.cfm

Το 1946 ένας δημιουργικός εφευρέτης ονόματι George Devol Jr πήρε το δίπλωμα ευρεσιτεχνίας για μια επαναλαμβανόμενη συσκευή η οποία είχε ως στόχο τον έλεγχο μηχανών. Η κίνηση του Devol για την αυτοματοποίηση αυτής της συσκευής τον οδήγησε σε μια άλλη εφεύρεση το 1954 για την οποία έλαβε το δίπλωμα ευρεσιτεχνίας και την ονόμασε Unimate. Η παρούσα εφεύρεση καθιστά διαθέσιμη για πρώτη φορά στα χρονικά μια μηχανή μεγαλύτερης ή μικρότερης σκοπιμότητας η οποία είχε καθολική εφαρμογή σε μια μεγάλη ποικιλία εφαρμογών στις οποίες επιθυμείτε κυκλικός έλεγχος[32].

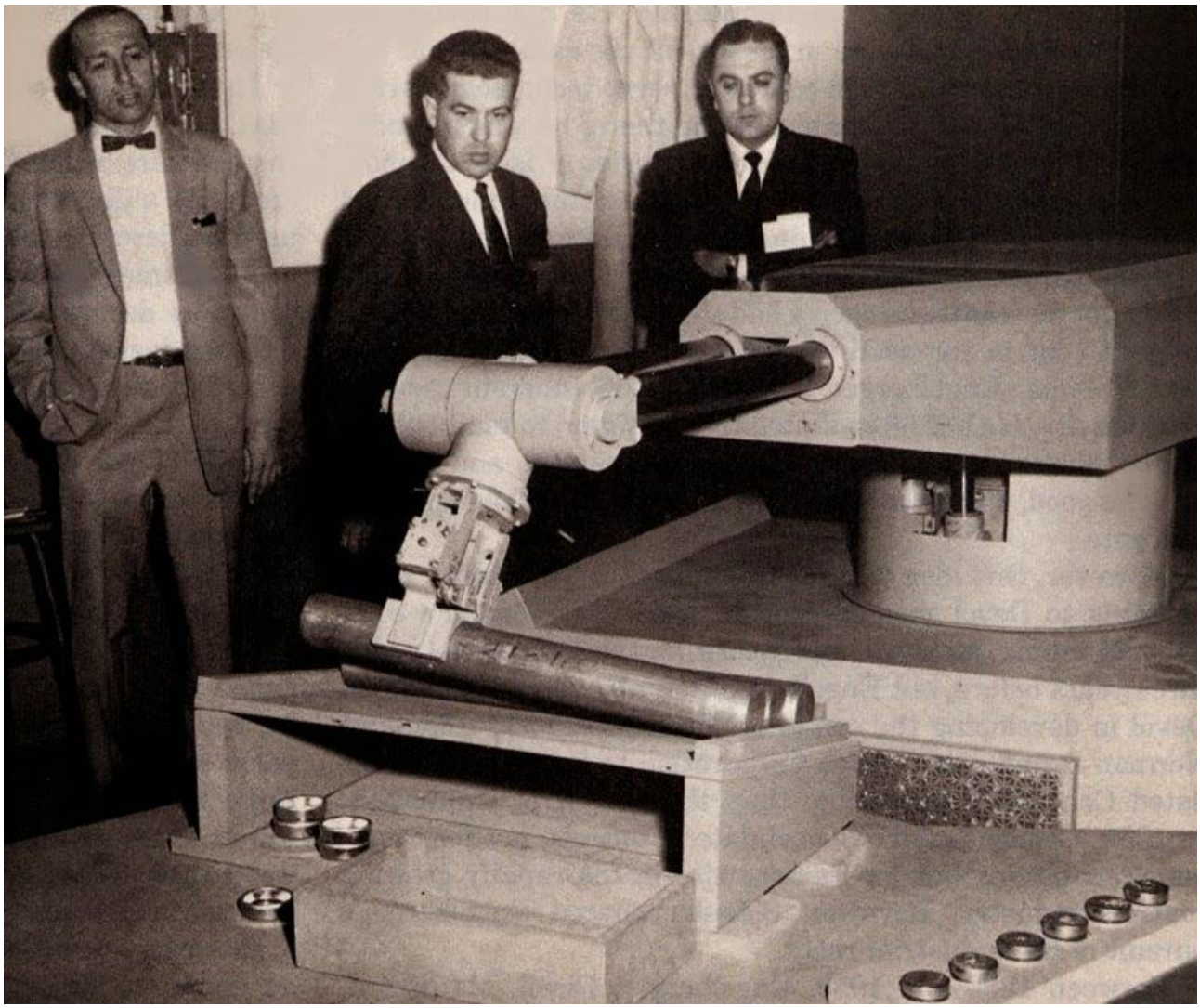
Το 1956 η συνάντηση των δυο εφευρετών έχει ως επακόλουθο την κατηγοριοποίηση της συσκευής του George Devol ως ρομπότ και το συμπέρασμα πως θα μπορούσε να χρησιμοποιηθεί στην βιομηχανική παραγωγή και πιο συγκεκριμένα στην εκτέλεση εργασιών επικινδύνων για τον άνθρωπο. Στην πορεία ο Engelberger και ο Devol δημιουργούν μια εταιρική σχέση ,η οποία οδήγησε στην εγκαθίδρυση της εταιρείας με την επωνυμία Unimation (Universal Animation) οδηγώντας στην "γέννηση" του βιομηχανικού ρομπότ.

32 /uploads/media/History_of_Industrial_Robots_online_brochure_by_IFR_2012.pdf

Εικόνα 1. Joseph Engelberger - George Devol (πηγή http://cyberneticzoo.com)

2.2. Εφαρμογές των Ρομπότ στην Βιομηχανία

Το 1959 κατασκευάστηκε το πρώτο βιομηχανικό ρομπότ της εταιρείας Unimation, ζυγίζοντας δύο τόνους και ελεγχόμενο από ένα πρόγραμμα το οποίο ήταν εφαρμοσμένο σε μαγνητικό τύμπανο (μαγνητική συσκευή αποθήκευσης δεδομένων που εφευρέθηκε απο τον Gustav Tauschek).Το 1961 εγκαταστάθηκε στο εργοστάσιο της General Motors στην πόλη Ewing, New Jersey το οποίο έφτιαχνε χερούλια παραθύρων και πορτών, κιβώτια ταχυτήτων, φωτιστικά και άλλα υλικά εσωτερικού χώρου για αυτοκινητοβιομηχανίες. Ακολουθώντας εντολές οι οποίες ήταν αποθηκευμένες σε μαγνητικό τύμπανο,ο βραχίονας του ρομπότ ανύψωνε και στοίβαζε ζεστά μεταλλικά κομμάτια (βλέπε Εικόνα 2).



Εικονα 2. Ο πρώτος ρομποτικός βραγχίονας της εταιρείας Unimate (πηγή methodofphysics.blogspot.gr)

Το 1962 στην Αμερική εγκαθίσταται από την εταιρεία AMF (American Machine and Foundry) το πρώτο κυλινδρικό ρομπότ με την επωνυμία Versatran στο εργοστάσιο της εταιρείας Ford στο Canton της Αμερικής. Η ονομασία προήλθε από τις λέξεις "versatile transfer"(ευέλικτη μεταφορά)[33].

33 URL:https://web.archive.org/web/20121224213437/http://www.ifr.org/uploads/media/History_of_Industrial_Robots_online_brochure_by_IFR_2012.pdf

Το 1969 η Νορβηγική εταιρεία Trallfa δημιουργεί το πρώτο εμπορικό ρομπότ ζωγραφικής. Τα συγκεκριμένα ρομπότ αναπτύχθηκαν για εσωτερική χρήση το 1967 με σκοπό τον ψεκασμό μπογιάς σε καροτσάκια και για το λόγο ότι υπήρχε μεγάλη έλλειψη εργατικού δυναμικού στη χώρα εκείνη την περίοδο.

Την ίδια χρονιά η Unimation υπογράφει συμφωνία με την Kawasaki Heavy Industries για την κατασκευή και αγορά Unimate ρομπότ με σκοπό την εγκατάσταση ρομποτικών βραχιόνων στις βιομηχανίες των Ασιατικών χωρών. Η Kawasaki ανάπτυξε την παραγωγή μηχανημάτων εξοικονόμησης εργασίας και η Ιαπωνία έγινε πρωτοπόρος στο πεδίο των βιομηχανικών ρομπότ. Η εταιρεία εκείνη την χρονιά δημιούργησε Kawasaki-Unimate 2000, το πρώτο βιομηχανικό ρομπότ που είχε κατασκευαστεί έως τότε στην Ιαπωνία.

Το 1973 η Γερμανική εταιρεία KUKA προχωρά από την χρήση των Unimate ρομπότ στην ανάπτυξη και κατασκευή των δικών τους ρομπότ. Το ρομπότ Famulus ήταν το πρώτο το οποίο είχε έξι ηλεκτρομηχανικούς καθοδηγουμένους άξονες και μεγάλο φάσμα κινήσεων[34].

Την ίδια χρονιά η Ιαπωνική εταιρεία Hitachi δημιουργεί το αυτόματο ρομποτικό μπουλόνι σε βιομηχανία σκυροδέματος. Πρόκειται για το πρώτο βιομηχανικό ρομπότ με δυναμικούς αισθητήρες όρασης για κινούμενα αντικείμενα. Αναγνωρίζοντας βίδες μέσα σε ένα κινούμενο καλούπι, στερέωνε/χαλάρωνε τους κοχλίες σε συγχρονισμό με το κινούμενο καλούπι.

Το Ichiro Kato πανεπιστήμιο στη Waseda της Ιαπωνίας το 1973 ανέπτυξε το παγκοσμίως πρώτο πλήρους κλίμακας ανθρωποειδές ρομπότ, με το όνομα Wabot-1. Το εν λόγω ρομπότ αποτελείται από ένα σύστημα ελέγχου των άκρων, ένα σύστημα οράσεως και ένα σύστημα

[34] URL: http://www.ijireeice.com/upload/2016/may-16/IJIREEICE%20120.pdf

συνομιλίας. Ήταν σε θέση να μετρήσει τις αποστάσεις και κατευθύνσεις προς τα αντικείμενα και να επικοινωνήσει με ένα άτομο στα Ιαπωνικά. Περπατώντας με τα κάτω άκρα του ήταν σε θέση να πιάσει και να μεταφέρει αντικείμενα με τα χέρια, τα οποία χρησιμοποιούσαν αισθητήρες αφής.

Μετά από μια δεκαετία μεγάλων ανακαλύψεων προς όφελος των βιομηχανιών η ανάπτυξη των βιομηχανικών ρομπότ συνεχώς εξελίσσεται. Ο πρώτος έλεγχος βιομηχανικού ρομπότ μέσω μικροεπεξεργαστή ήρθε στην αγορά το 1974 και αναπτύχθηκε από τον Richard Hohn για την εταιρεία Cincinnati Milacron με το όνομα T3.

Το 1974 η Ιαπωνική εταιρεία Hitachi ανέπτυξε μια ανανεωμένη έκδοση του Unimate Robot για την χρήση του ως συγκολλητή σημείων κατασκευάζοντας μοτοσυκλέτες Kawasaki. Στο ρομποτικό χέρι (Hi-T-Hand robot) προστέθηκαν αισθητήρες ανίχνευσης επαφής και δύναμης, δίνοντας έτσι την δυνατότητα στο ρομπότ να καθοδηγήσει καρφίτσες σε τρύπες σε ένα ρυθμό ενός δευτερολέπτου ανά ακίδα[35].

Το πρώτο πλήρως ηλεκτρικό βιομηχανικό ρομπότ με μικροεπεξεργαστή IRB 6 από την εταιρεία ASEA παραδόθηκε σε μια εταιρεία μηχανολογίας στη νότια Σουηδία. Με ανθρωπομορφικό σχεδιασμό η κίνηση του βραχίονα μιμούνταν αυτό του ανθρώπινου σκέλους, με ωφέλιμο φορτίο έξι κιλών και πέντε άξονες. Το πρώτο μοντέλο, IRB 6 αποκτήθηκε από την Magnussons στο Genarp με σκοπό να προσθέτει κερί και να γυαλίζει ανοξείδωτους χαλύβδινους σωλήνες.

35 URL:https://web.archive.org/web/20121224213437/http://www.ifr.org/uploads/media/History_of_Industrial_Robots_online_brochure_by_IFR_2012.pdf

Η Ιαπωνική εταιρεία Nachi αναπτύσσει το πρώτο ρομπότ το οποίο ελέγχονταν μέσω ενός ηλεκτροκινητήρα. Μέσω αυτής της καινοτομίας τα ρομπότ σημειακής συγκόλλησης εισήχθηκαν σε μια νέα εποχή ηλεκτρικών οδηγούμενων ρομπότ, αντικαθιστώντας την προηγούμενη εποχή της υδραυλικής κίνησης.

Ο Takeo Kanade σχεδίασε τον παγκοσμίως πρώτο βραχίονα άμεσης κίνησης καθώς εργαζόταν στο πανεπιστήμιο Carnegie Mellon στην Αμερική. Ένας βραχίονας άμεσης κίνησης είναι ένας μηχανικός βραχίονας στον οποίο οι άξονες των αρθρωτών αρμών συνδέονται άμεσα με τους ρότορες των κινητήρων με μεγάλη ροπή στρέψης. Επειδή ο βραχίονας δεν εμπεριέχει κάποιο μηχανισμό, όπως γρανάζι ή κιβώτιο ταχυτήτων μεταξύ των κινητήρων και των φορτίων τους, τα συστήματα κίνησης δεν έχουν οπισθοδρόμηση, μικρή τριβή και υψηλή ακαμψία αλλά είναι επιθυμητά για γρήγορα, ακριβή και προσαρμόσιμα ρομπότ. Το αποτέλεσμα αυτής της εφεύρεσης είναι μια κατασκευή που μπορεί να κινηθεί ελεύθερα και ομαλά,επιτρέποντας την χρήση ρομπότ υψηλής ακρίβειας.

Με το πέρασμα των χρόνων και το έτος 1985 η Γερμανική εταιρεία Kuka μετά τον σχεδιασμό και κατασκευή του ρομπότ Famulus συνεχίζει τις εφευρέσεις της στο πεδίο των βιομηχανικών ρομπότ και παρουσιάζει ένα νέο ρομποτικό βραχίονα σχήματος Z του οποίου το σχέδιο αγνοεί το παραδοσιακό παραλληλόγραμο. Επιτυγχάνει πλήρη ευελιξία τριών περιστροφικών κινήσεων για συνολικά έξι βαθμούς ελευθερίας. Η νέα κατασκευή αποθηκεύει χώρο στις βιομηχανίες[36].

Η Σουηδική εταιρεία ABB ανέπτυξε το Flex Picker το παγκοσμίως γρηγορότερο ρομπότ διαλογής αντικειμένων, βασισμένο στο ρομπότ Delta το οποίο αναπτύχθηκε από τον Raymond Clavel το 1998. Ήταν σε θέση να διαλέξει εκατόν είκοσι αντικείμενα το λεπτό ή να διαλέξει και

36 URL: http://www.ijireeice.com/upload/2016/may-16/IJIREEICE%20120.pdf

να απελευθερώσει με ταχύτητα δέκα μέτρων το δευτερόλεπτο χρησιμοποιώντας τεχνολογία εικόνας.

Συχνά τα ρομπότ πρέπει να εκτελούν και εργασίες συντονισμού. Και πάλι η εταιρεία ABB ανέπτυξε έναν ελεγκτή με την επωνυμία IRC5,ο οποίος ελέγχει ταυτόχρονα μέχρι και τέσσερα ρομπότ ή συσκευές εντοπισμού θέσης με μέγιστο αριθμό 36 σερβοαξονικών αξόνων. Έτσι τα ρομπότ φτάνουν σε κάθε απαιτούμενη θέση ραφής χωρίς να διακόπτεται η συγκόλληση. Για να βελτιωθεί η πορεία, η ακρίβεια και η επαναληψιμότητα συμβάλλουν σε ποιότητα υψηλότερου επιπέδου.

Το 2006 η Γερμανική εταιρεία Kuka προχωράει σε μια ακόμα ανακάλυψη παρουσιάζοντας το πρώτο "ρομπότ ελαφρού βάρους"("Light Weight Robot") σε συνεργασία με το DLR το "Ινστιτούτο Ρομποτικής και Μηχατρονικής " της Γερμανίας. Η εξωτερική δομή του ρομπότ αποτελούνταν από αλουμίνιο. Έχει ωφέλιμο φορτίο εφτά κιλών και εξαιτίας των ενσωματωμένων αισθητήρων ιδιαίτερη ευαισθησία. Αυτό το κάνει ιδανικό για εργασίες χειρισμού και συναρμολόγησης. Το ρομπότ είναι ενεργειακά αποδοτικό και φορητό και μπορεί να εκτελεί ένα μεγάλο φάσμα διαφορετικών καθηκόντων[37].

Το 2010 η Ιαπωνική εταιρεία Fanuc δημιουργεί το πρώτο αυτοδίδακτο ρομπότ. Το αυτοδίδακτο ρομπότ ελέγχου δόνησης της Fanuc του επιτρέπει να μάθει χαρακτηριστικά κραδασμών για υψηλότερες ταχύτητες και επιταχύνσεις. Ο έλεγχος στην εκμάθηση του συγκεκριμένου ρομπότ μειώνει το χρόνο κύκλου της κίνησης του ρομπότ καταστέλλοντας τις δονήσεις του ρομποτικού βραχίονα.

37 URL: http://www.ijireeice.com/upload/2016/may-16/IJIREEICE%20120.pdf

Το 2011 περισσότερα απο ένα εκατομμύριο βιομηχανικά ρομπότ εφαρμόστηκαν στη βιομηχανική διαδικασία σε όλο τον κόσμο. Σήμερα τα βιομηχανικά ρομπότ και γενικά τα ρομποτικά συστήματα αποτελούν το κλειδί της επιτυχίας σε μια αυτοματοποιημένη διαδικασία,έχοντας ως όφελος την βελτίωση της ποιότητας εργασίας, την άυξηση ποσοστού παραγωγής χωρίς να επηρεάζεται η ποιότητα των προϊόντων[38].

38

URL:https://web.archive.org/web/20121224213437/http://www.ifr.org/uploads/media/History_of_Industrial_Robots_online_brochure_by_IFR_2012.pdf

3. Arduino

Σε αυτή την ενότητα ο χρήστης εμβαθύνει στην λειτουργία του περιβάλλοντος του Arduino καθώς μαθαίνει τους κανόνες και τις απαραίτητες εντολές για την κατανόηση και συγγραφή κώδικα.

3.1. Εισαγωγή

Η λέξη Arduino αναφέρεται σε μια πλατφόρμα ανοιχτού λογισμικού και υλικού με στόχο την δημιουργία διαδραστικών εφαρμογών χρησιμοποιώντας μια απλοποιημένη γλώσσα προγραμματισμού για την συγγραφή εντολών και έναν δυνατό μα απλό επεξεργαστή ο οποίος είναι σε θέση να εκπληρώσει ένα μεγάλο φάσμα καθηκόντων ανεξαρτήτως δυσκολίας[39].

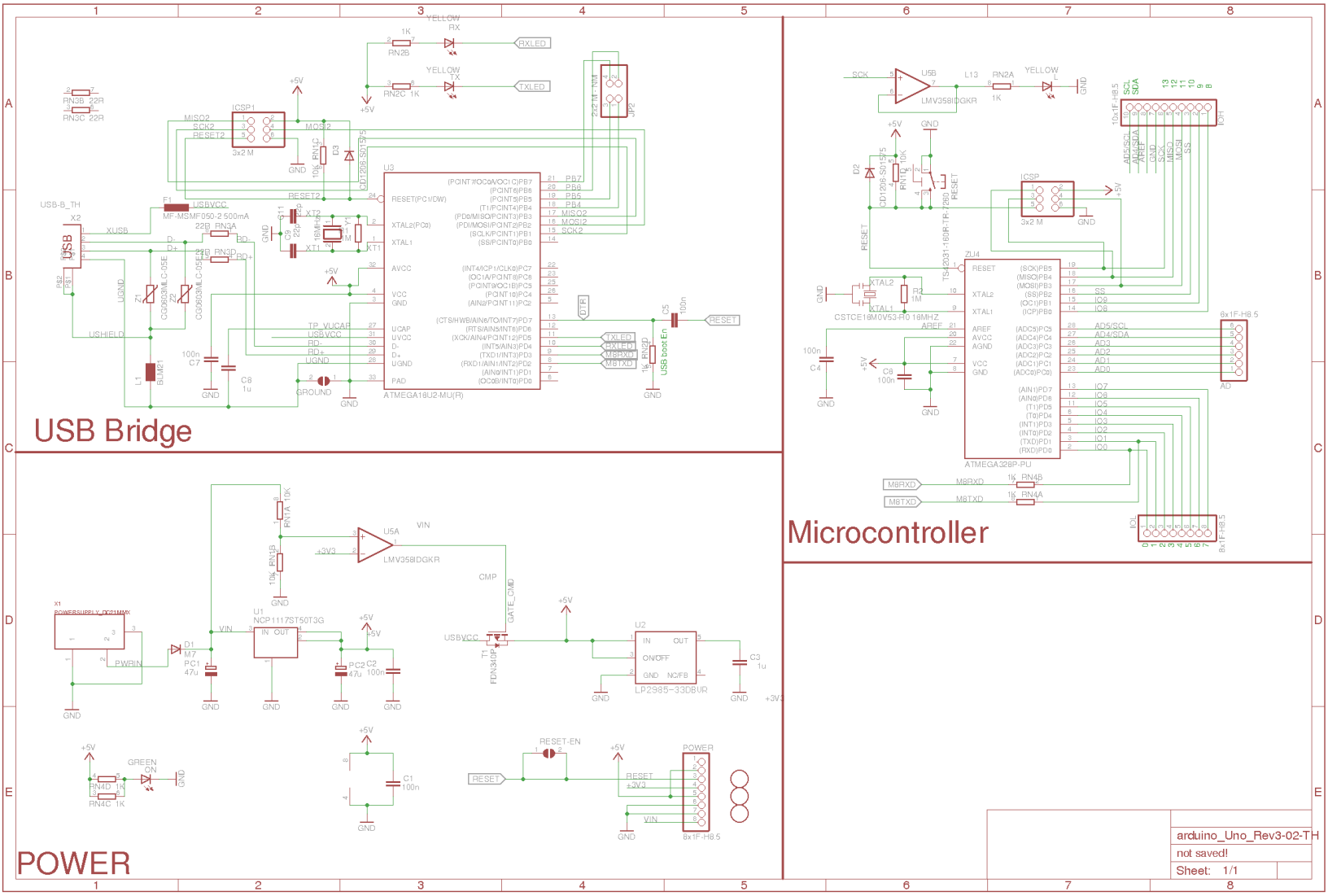
Ένας ελεγκτής Arduino έχει την δυνατότητα να διαβάσει μια είσοδο, για παράδειγμα, το φως μιας λυχνίας ή την ένδειξη ενός κουμπιού και αντίστοιχα να δώσει μια εντολή εξόδου, με την οποία να ανοίγει μια πόρτα ή να ενεργοποιείται ένας κινητήρας. Τα βασικά στοιχεία της εργαλειοθήκης του Arduino είναι ο μικροελεγκτής (microcontroller), η γλώσσα προγραμματισμού και το ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Enviroment, IDE).

39 Noble. J (2009). *Programming Interactivity: A Designer's Guide to Processing*, Arduino and Openframeworks. O'Reily Media, Inc, p. 92

3.2. Ελεγκτής Arduino (Arduino Board)

Ο μικροελεγκτής είναι ένας υπολογιστής μικρότερης κλίμακας ο οποίος χρησιμοποιεί έναν επεξεργαστή, ακροδέκτες Εισόδου/Εξόδου (I/O) και την υποδοχή USB για επικοινωνία με άλλες συσκευές και περιφερειακά καθώς και ένα μικρό ποσοστό Ram (Random-Access Memory). Σε αντίθεση με έναν κανονικό υπολογιστή η αρχιτεκτονική διαφέρει όσο αναφορά την τροφοδοσία του ελεγκτή και των συσκευών ή αισθητηρίων που συνδέονται με αυτόν[40].

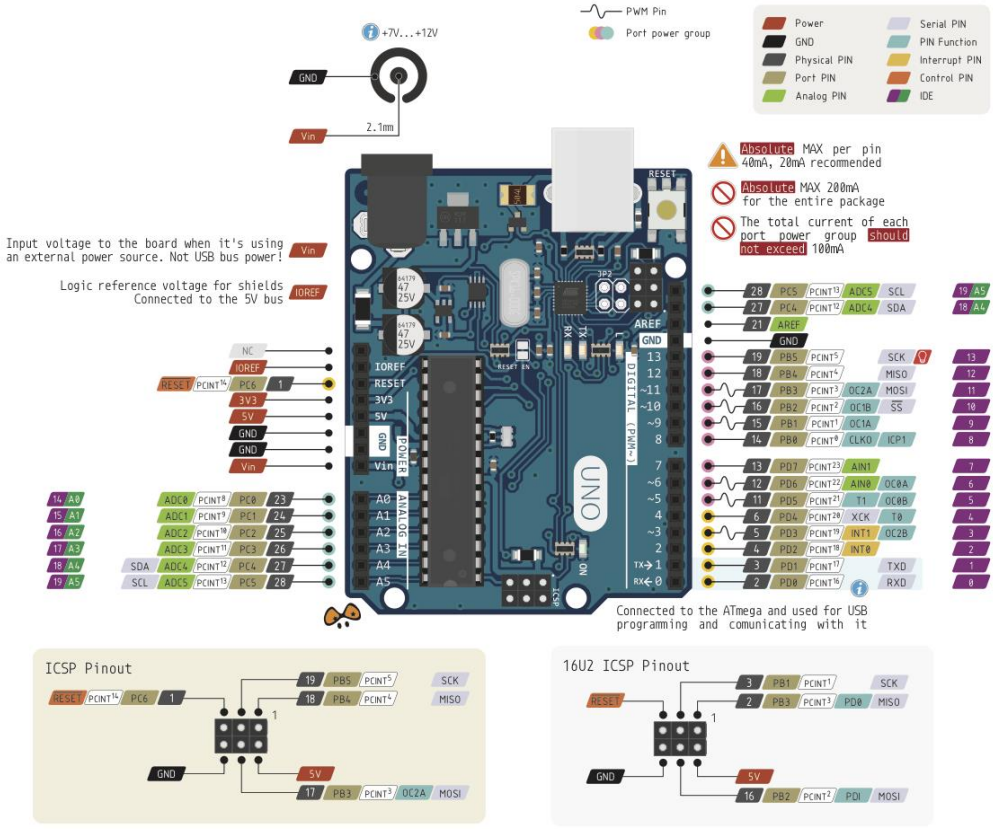
[40] Noble. J (2009) : 93



Σχεδιάγραμμα 1. Σχηματικές παραστάσεις του μικροελεγκτή, της γέφυρας usb και της τροφοδοσίας (Πηγή allaboutcircuit.com)

3.2.1. Περιγραφή των Στοιχείων της Πλακέτας Arduino

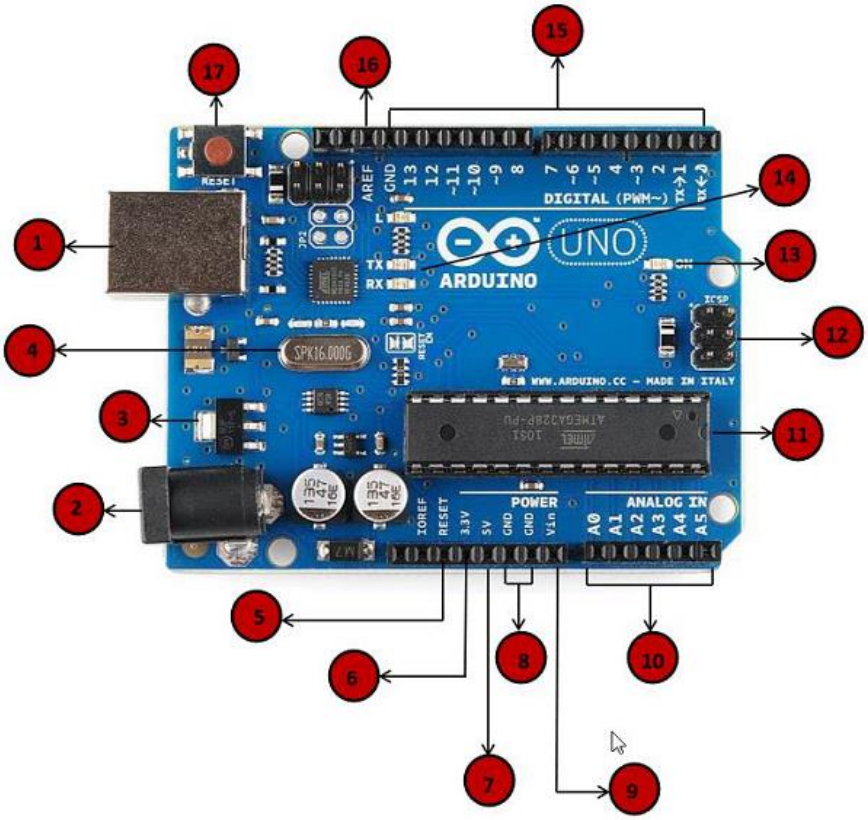
Στα παρακάτω δύο σχεδιαγράμματα διακρίνονται η σχεδίαση και οι ακροδέκτες Arduino. Στην συνέχεια γίνεται περιγραφή των δομικών στοιχείων του ελεγκτή.



Σχεδιάγραμμα 2. Ακροδέκτες του Arduino Uno (Πηγή google.gr)

3.3. Δομικά Στοιχεία του Arduino (Arduino Components)

Σε αυτό το κεφάλαιο δίνεται η περιγραφή των δομικών στοιχείων του Arduino Uno, λόγω της μεγάλης δημοτικότητας του ελεγκτή. Μερικοί ελεγκτές της οικογένειας του Arduino είναι διαφορετικοί από τον υποφαινόμενο όμως στην πλειοψηφία τους βρίσκει κανείς τα παρακάτω κοινά δομικά στοιχεία (βλ. Σχεδιάγραμμα 3).



Σχεδιάγραμμα 3. Arduino Uno (Πηγή tutorialspoint.com)

Τροφοδοσία – Usb (Power - USB)

Ο ελεγκτής μπορεί να τροφοδοτηθεί από οποιαδήποτε θύρα USB ενός υπολογιστή μέσω του καλωδίου USB (1).

Τροφοδοσία (Power)

Εναλλακτικά ένας ελεγκτής μπορεί να τροφοδοτηθεί απευθείας συνδέοντας ένα τροφοδοτικό με μια πηγή εναλλασσόμενου ρεύματος. Η προτεινόμενη τάση κυμαίνεται από 7 έως 12 Volts. Σε περίπτωση που ο ελεγκτής δεχτεί τάση της τάξης των 20 Volts κινδυνεύει να καταστραφεί (2).

Ρυθμιστής Τάσης (Voltage Regulator)

Ο ρυθμιστής τάσης ελέγχει την τάση που δέχεται ο Arduino, σταθεροποιώντας την DC τάση που χρησιμοποιείται από τον μικροελεγκτή και τα υπόλοιπα στοιχεία (3).

Κρυσταλλικός Ταλαντωτής (Crystal Oscillator)

Ο κρυσταλλικός ταλαντωτής χρησιμοποιείται για την αντιμετώπιση χρονικών προβλημάτων και ανάλογα με την διατομή του παράγει και αντίστοιχη συχνότητα. Ο ελεγκτής χρησιμοποιεί τον κρυσταλλικό ταλαντωτή για να υπολογίζει χρονικά διαστήματα που παρεμβάλλονται κατά την εκτέλεση ενός προγράμματος. Η τιμή που αναγράφεται επάνω στον κρυσταλλικό ταλαντωτή είναι η αντίστοιχη συχνότητα του σε Hertz (4).

Επαναφορά (Reset)

Ο Arduino έχει την δυνατότητα να επαναφέρει το πρόγραμμα που εκτελεί την παρούσα χρονική στιγμή στην αρχική του κατάσταση χρησιμοποιώντας το κουμπί Reset, είτε συνδέοντας ένα εξωτερικό κουμπί στον ακροδέκτη Reset (5).

Ακροδέκτες

Οι ακροδέκτες είναι υποδοχές οι οποίες έχουν την δυνατότητα να συνδεθούν με μικρά μεταλλικά ή χάλκινα καλώδια, παρέχοντας έναν δίαυλο επικοινωνίας μεταξύ του ελεγκτή και ενός αισθητηρίου είτε πρόκειται για είσοδο, είτε για έξοδο. Οι ακροδέκτες με τις ενδείξεις 3.3, 5, GND, VIN χρησιμοποιούνται κυρίως για την τροφοδότηση περιφερειακών συσκευών και αισθητηρίων, αλλά και για την τροφοδότηση του ελεγκτή.

3.3 Volts – παροχή 3.3 Volt (6)

5 Volts – παροχή 5 Volt (7)

GND, Ground – Γείωση (8)

VIN – Χρησιμοποιείται για την τροφοδότηση του Arduino από εξωτερική πηγή τροφοδοσίας, για παράδειγμα μια μπαταρία των 9 Volt (9).

Αναλογικοί ακροδέκτες (Analog Pins)

Οι αναλογικοί ακροδέκτες δίνουν την δυνατότητα να γραφτούν (write) ή διαβαστούν (read) τιμές από μία ή σε μία περιφερειακή συσκευή μέσα από ένα ευρύ φάσμα πληροφοριών. Συγκεκριμένα μπορούν να γραφτούν τιμές από 0 έως 255, δηλαδή 256 διαφορετικές καταστάσεις πληροφορίας αντιστοιχιζόμενες στο εύρος της τάσης του ελεγκτή από 0 Volt έως 5 Volt, ενώ μπορούν να διαβαστούν τιμές από 0 έως 1023, δηλαδή 1024 καταστάσεις πληροφορίας και αυτές αντιστοιχιζόμενες στο ίδιο εύρος των 5 Volt. Για παράδειγμα οι τιμές αυτές μπορούν να συλλεχτούν για την θέση ενός αντικειμένου από έναν υπέρυθρο αισθητήρα ή για την φωτεινότητα μια λυχνίας LED (10). (βλ. Σχεδιάγραμμα 3).

Μικροελεγκτής (Microcontroller)

Ο μικροελεγκτής είναι ο εγκέφαλος του Arduino, είναι υπεύθυνος για την εκτέλεση των εντολών του εκάστοτε προγράμματος που φορτώνεται στον ελεγκτή. Το ολοκληρωμένο κύκλωμα διαφέρει από ελεγκτή σε ελεγκτή συνήθως όμως προέρχονται από την Atmel Company. Στην περίπτωση του Arduino Uno χρησιμοποιείται ο Atmega 328. Πρόκειται για μια συσκευή των 8-bit η αρχιτεκτονική της οποίας της επιτρέπει να χειρίζεται 8 παράλληλα σήματα δεδομένων[41] (11).

Ακροδέκτης ICSP (ICSP Pin)

ICSP (In Circuit Serial Programming) ή SPI (Serial Peripheral Interface) είναι ένα πρωτόκολλο επικοινωνίας το οποίο χρησιμοποιείται για την σύνδεση με μία ή περισσότερες περιφερειακές συσκευές, συνήθως μικροελεγκτές. Σε μια σύνδεση SPI μία συσκευή είναι ο Κυρίαρχος (Master) και η άλλη είναι ο Σκλάβος (Slave). Κατά κανόνα υπάρχουν γραμμές επικοινωνίας μεταξύ των συσκευών:

MISO (Master In Slave Out) - Η γραμμή από την οποία ο Slave στέλνει δεδομένα στον Master

MOSI (Master Out Slave In) – Η γραμμή από την οποία ο Master στέλνει δεδομένα στις περιφερειακές συσκευές.

41 URL: http://smithsonianchips.si.edu/augarten/p38.htm

SCK (Serial Clock) – Η γραμμή από την οποία γίνεται η εκπομπή της συχνότητας που καθορίζει τον ρυθμό αποστολής δεδομένων από τον Master προς τις υπόλοιπες περιφερειακές συσκευές (12).

Σχεδιάγραμμα 4. ICSP (In Circuit Serial Programming) (Πηγή arduino.cc)

Η λυχνία LED ανάβει όταν το Arduino είναι συνδεδεμένος με μια πηγή τροφοδοσίας. Εάν η λυχνία είναι σβηστή ενδεχομένως να υπάρχει πρόβλημα με την τροφοδοσία της συσκευής (13).

Λυχνίες LED Tx και Rx (Tx and Rx LEDs)

Οι ακροδέκτες Tx (transmit) και Rx (receive) βρίσκονται στους ψηφιακούς ακροδέκτες 1 και 0 αντίστοιχα και χρησιμεύουν για σειριακή επικοινωνία με άλλες συσκευές. Οι αντίστοιχες λυχνίες ανάλογα με την ταχύτητα που αναβοσβήνουν, δείχνουν τον ρυθμό μετάδοσης κατά την διαδικασία αποστολής ή παραλαβής δεδομένων (14).

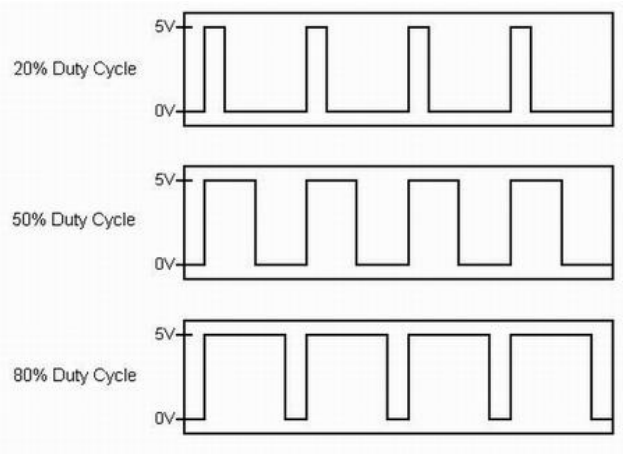
Ψηφιακοί Ακροδέκτες (Digital Pins)

Οι ψηφιακοί ακροδέκτες έχουν την δυνατότητα να μεταβούν ενδιάμεσα σε 2 καταστάσεις: Λογικό 1 (Digital High or High) και Λογικό 0 (Digital Low or Low). Η κατάσταση High σημαίνει πως 5V στέλνονται από τον ελεγκτή στον ακροδέκτη ή διαβάζονται από μια συσκευή ή αισθητήριο, η κατάσταση Low δηλώνει πως η τάση στον ακροδέκτη είναι 0 V. Η χρήση των

ψηφιακών ακροδεκτών μπορεί να φανεί ιδιαίτερα χρήσιμη στις περιπτώσεις όπου θέλει κανείς να διαχωρίσει μια λειτουργία σε 2 καταστάσεις όπως open/closed ή να ενημερώσει για την κατάσταση μιας διαδικασίας με κάποιο μήνυμα όπως για παράδειγμα "έτοιμος/όχι έτοιμος"[42] (15).

PWM (Pulse-Width Modulation)

Το Arduino χρησιμοποιεί την τεχνική PWM για να ελέγξει αναλογικά κυκλώματα με την χρήση ψηφιακής εξόδου. Ένας ψηφιακός ακροδέκτης χρησιμοποιεί την κατάσταση High (5V) και την κατάσταση Low (0V) παράγοντας έναν τετραγωνικό παλμό. Με την τεχνική PWM μπορεί κανείς να ελέγξει τον κύκλο λειτουργίας χρησιμοποιώντας εάν ποσοστό του παλμού δίνοντας την δυνατότητα να προσομοιώσει κανείς τάσεις μεταξύ 0 και 5V. [43]



42 URL: https://www.tutorialspoint.com/arduino/arduino_board_description.htm

43 URL: http://www.arduino-tutorials.com/arduino-pwm/

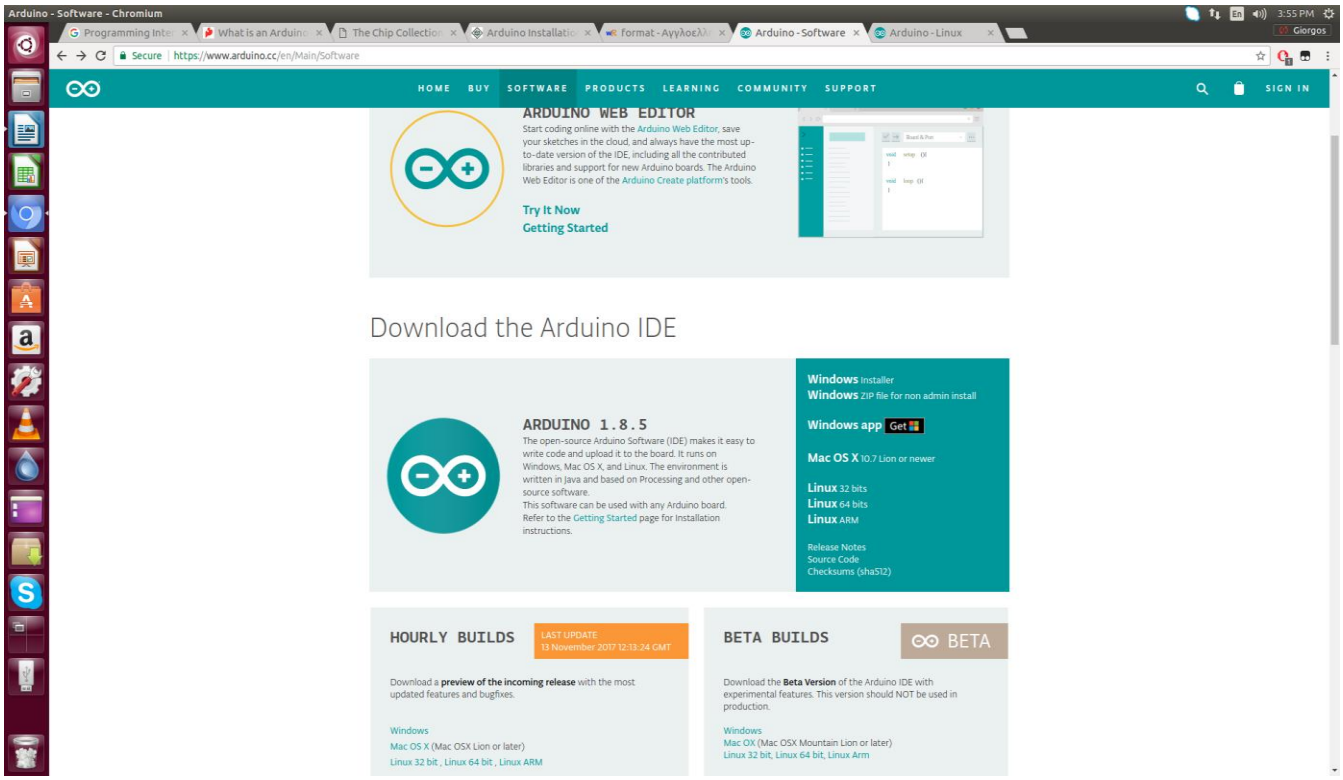
Σχεδιάγραμμα 5. PWM – Duty Cycle (Πηγή arduino.cc)

Τα περισσότερα εξαρτήματα που χρησιμοποιούνται από το Arduino δουλεύουν ικανοποιητικά στις τάσεις 3.3 Volt και 5 Volt, υπάρχουν όμως και εξαιρέσεις για αυτό τον λόγο προτείνεται η ανάγνωση του φύλλου δεδομένων (data sheet) πριν την χρήση ενός εξαρτήματος. Για την παρούσα πτυχιακή χρησιμοποιήθηκε ο ελεγκτής Arduino Mega 2560 Rev 3.

3.4. Εγκατάσταση του Περιβάλλοντος Ανάπτυξης (Arduino IDE)

Το επόμενο βήμα είναι η εγκατάσταση του περιβάλλοντος ανάπτυξης (Arduino IDE), το οποίο μπορεί κανείς να το κατεβάσει από την επίσημη ιστοσελίδα του Arduino[44] (βλ. Εικόνα 1). Ανάλογα με το λειτουργικό σύστημα που τρέχει κανείς στον υπολογιστή πρέπει να επιλέξει και την αντίστοιχη έκδοση.

[44] https://www.arduino.cc/en/Main/Software

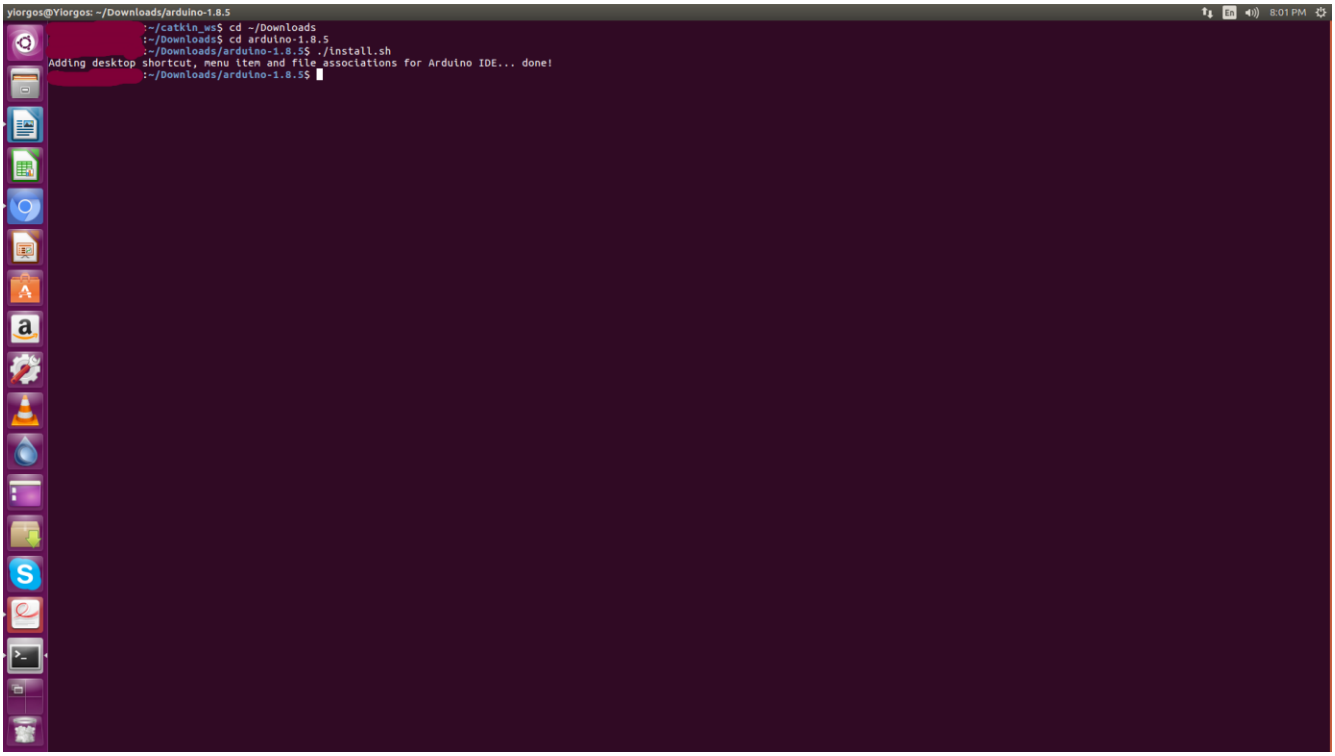


Εικόνα 3. Εγκατάσταση του περιβάλλοντος ανάπτυξης

3.4.1. Εγκατάσταση σε Ubuntu Linux

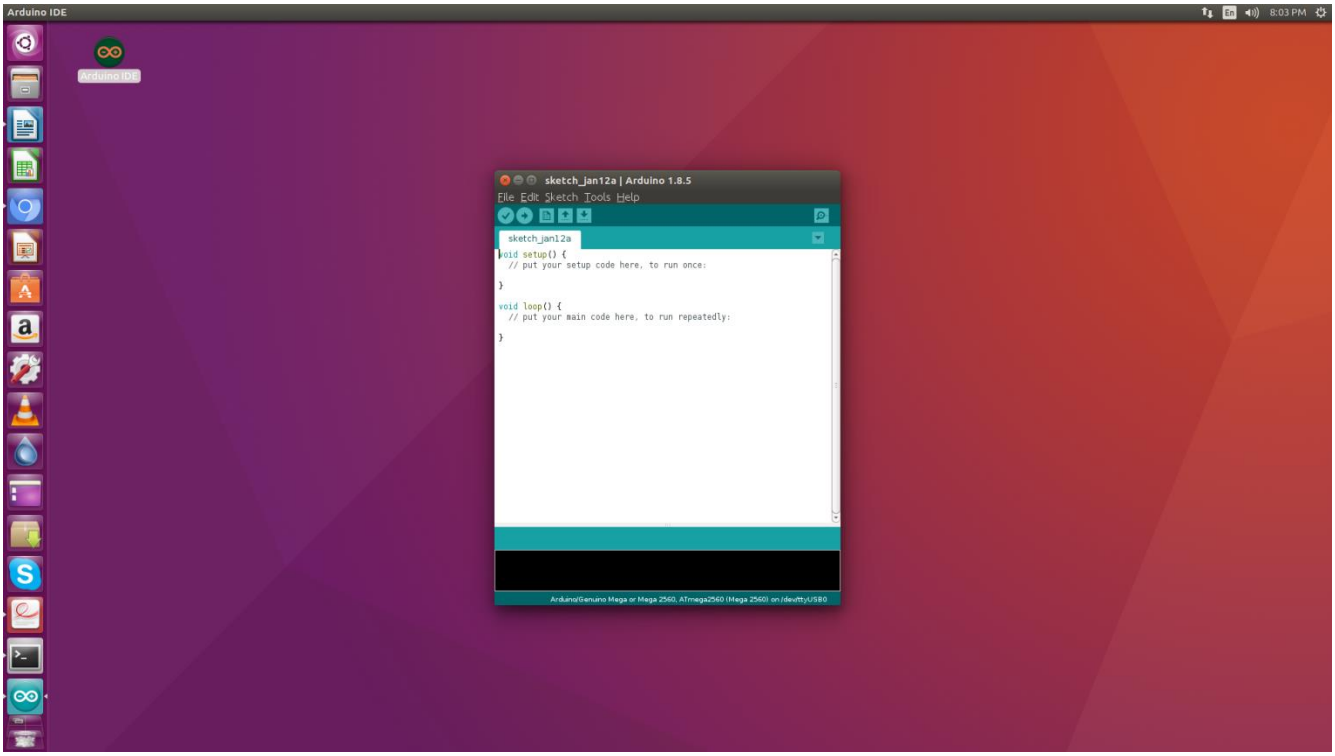
Επιλέγεται η έκδοση που ταιριάζει στο λειτουργικό σύστημα του χρήστη. Αφού κατέβει ο φάκελος με τα απαραίτητα αρχεία ανοίγεται ο συμπιεσμένος φάκελος και εξάγεται το αρχείο.

Εφόσον το αρχείο έχει αποσυμπιεστεί ανοίγεται ένα τερματικό και μεταφέρεται ο χρήστης στον φάκελο arduino πληκτρολογώντας την εντολή cd ~/ ”όνομα φακέλου”. Για παράδειγμα στην παρούσα πτυχιακή η σύνταξη της εντολής είναι cd ~/arduino-1.8.5. Έπειτα πληκτρολογείται η εντολή ./instal.sh ώστε να ολοκληρωθεί η διαδικασία (βλ. Εικόνα 4).



Εικόνα 4. Εγκατάσταση Arduino σε Ubuntu Linux

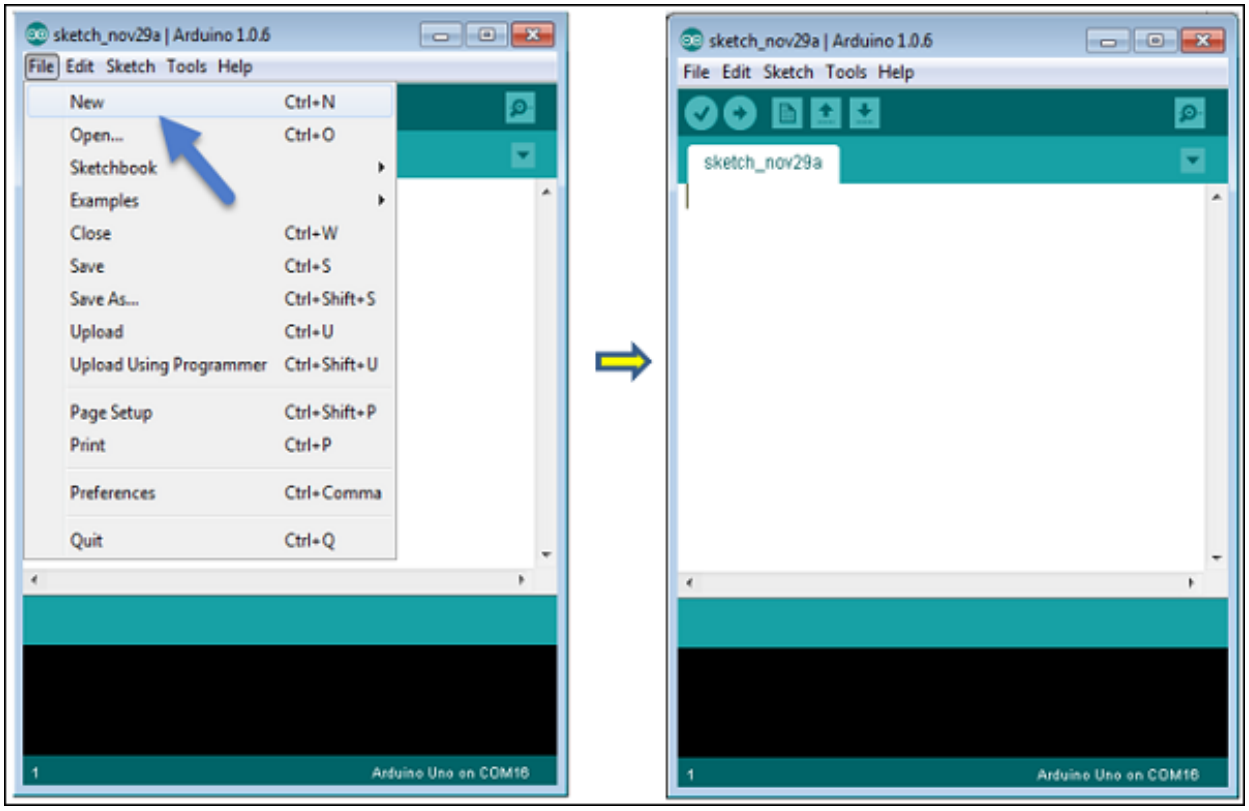
Μετά το τέλος της εγκατάστασης η εφαρμογή είναι έτοιμη για χρήση (βλ. Εικόνα 5 στην επόμενη σελίδα).



Εικόνα 5. Περιβάλλον Arduino σε Ubuntu Linux

3.5. Περιήγηση στο Περιβάλλον του Arduino (Arduino IDE), δημιουργία και άνοιγμα αρχείων

Μόλις ανοίξει η εφαρμογή μπορεί να διακρίνει κανείς την επιλογή File. Πιέζοντας την επιλογή File διακρίνονται περισσότερες οι επιλογές New και Open. Για την δημιουργία ενός νέου Project επιλέγεται η επιλογή File → New. Για την επιλογή ενός υπάρχοντος Project επιλέγεται File→ Open (βλ. Εικόνα 6).



Εικόνα 6. File – open

Για την επιλογή ενός ήδη υπάρχοντος παραδείγματος η διαδικασία είναι η εξής: File → Examples → Basics → Blink. Στην προκειμένη περίπτωση επιλέχτηκε το αρχείο Blink το οποίο αναβοσβήνει μια λυχνία LED. Δίνεται η δυνατότητα επιλογής οποιουδήποτε άλλου παραδείγματος.

3.5.1. Βιβλιοθήκες του Arduino (Arduino Libraries)

Το επόμενο βήμα είναι η συμπερίληψη βιβλιοθήκης στο περιβάλλον του Arduino. Η βιβλιοθήκη είναι ένα αρχείο γραμμένο σε γλώσσα C ή C++ το οποίο δίνει σε ένα πρόγραμμα μια επιπλέον λειτουργία, για παράδειγμα τον έλεγχο ενός πίνακα LED ή τον έλεγχο στροφών ενός κινητήρα.

Υπάρχουν ήδη προεγκατεστημένες βιβλιοθήκες στο περιβάλλον του Arduino, αλλά δίνεται η δυνατότητα στον χρήστη να εισάγει δικές του βιβλιοθήκες.[45]

Υπάρχουν 2 τρόποι για να προσθέσει κανείς βιβλιοθήκες. Ο πρώτος τρόπος είναι να χρησιμοποιηθεί ο Διαχειριστής Βιβλιοθήκης (Library Manager) με τον εξής τρόπο:

Sketch Include Library → Manage Libraries

Εφόσον ακολουθήθηκε η παραπάνω διαδικασία, θα εμφανιστεί ο Διαχειριστής Βιβλιοθήκης (Library Manager).

Επιλέγεται η βιβλιοθήκη που επιθυμεί ο χρήστης. Εφόσον γίνει έλεγχος της πιο πρόσφατης έκδοσης της βιβλιοθήκης που πρόκειται να εγκατασταθεί, επιλέγεται το κουμπί εγκατάσταση. Η βιβλιοθήκη πλέον έχει συμπεριληφθεί στο περιβάλλον και κατά συνέπεια και στον κώδικα.

Ο δεύτερος τρόπος είναι να γίνει λήψη της βιβλιοθήκης σε μορφή φακέλου. Ο φάκελος έχει το όνομα της βιβλιοθήκης και περιέχει ένα αρχείο με κατάληξη .cpp, ένα αρχείο με κατάληξη .h και συχνά αρχεία με κατάληξη .txt, παραδείγματα και άλλα αρχεία που χρειάζεται η βιβλιοθήκη.

Επιλέγεται Sketch → Include Library Add .ZIP Library

Εφόσον βρεθεί ο φάκελος της βιβλιοθήκης, δεν έχει σημασία αν είναι αποσυμπιεσμένος ή όχι απλά πατά κανείς Open.

Εφόσον η διαδικασία είναι επιτυχημένη θα εμφανιστεί στην κάτω αριστερή γωνία του προγράμματος ένα μήνυμα το οποίο θα επιβεβαιώνει την συμπερίληψη της βιβλιοθήκης στο περιβάλλον του Arduino.

[45] URL: https://www.arduino.cc/en/hacking/libraries

3.5.2. Επικοινωνία με τον Ελεγκτή

Στο Linux και σε άλλα λειτουργικά συστήματα τύπου Unix υπάρχουν κάποιοι σχετικοί κανόνες οι οποίοι ορίζουν την πρόσβαση ενός χρήστη ή μιας ομάδας χρηστών σε ένα φάκελο. Οι κανόνες αυτοί ονομάζονται άδειες (permissions) ή λειτουργίες φακέλων (file modes). Η εντολή chmod (change mode) χρησιμοποιείται για να ορίσει τον τρόπο με τον οποίο μπορεί κανείς να διαχειριστεί ένα φάκελο. Πληκτρολογείται η εντολή:

```
cd /dev

ls -a
```

Ο φάκελος /dev είναι ένας αποθηκευτικός χώρος για αρχεία τα οποία περιέχουν πληροφορίες για όλες τις συσκευές που χρησιμοποιούνται από το λειτουργικό σύστημα. Σκοπός είναι η αναζήτηση της σειριακής θύρας η οποία είναι συνδεδεμένη με τον μικροελεγκτή η οποία όπως προαναφέρθηκε είναι ορατή και μέσα από το περιβάλλον του Arduino. Η μορφή της εντολής που θα εκτελεστεί στο τερματικό είναι :

```
sudo chmod 777 <όνομα θύρας>
```

Ο αριθμός 777 με λίγα λόγια δίνει δικαιώματα ανάγνωσης, επεξεργασίας και εκτέλεσης ενός φακέλου στον χρήστη ή σε μια ομάδα χρηστών. Περισσότερες πληροφορίες στους παρακάτω συνδέσμους:

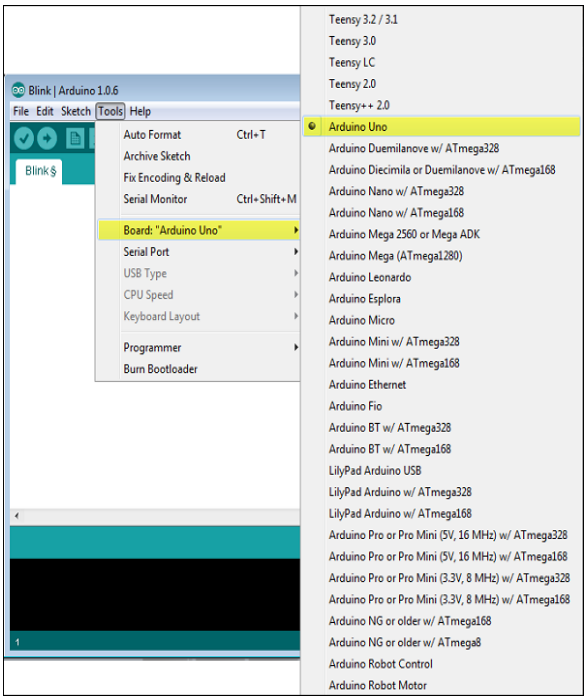
https://www.computerhope.com/unix/uchmod.htm

http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/dev.html

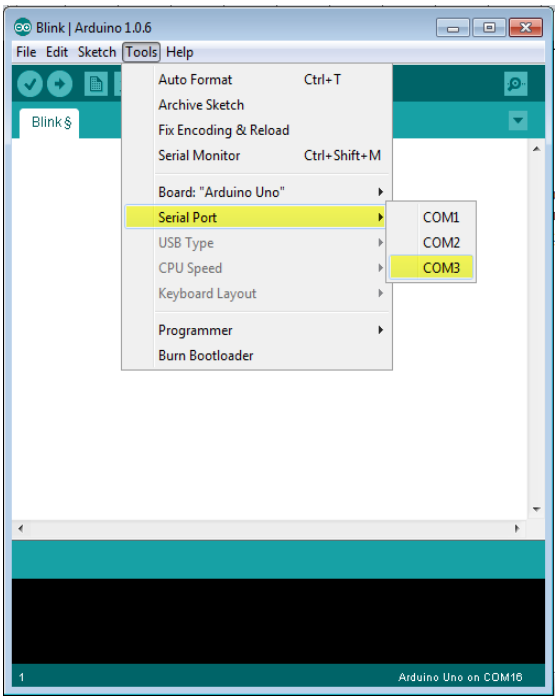
Το επόμενο βήμα είναι ο ορισμός του ελεγκτή Arduino που πρόκειται να συνδεθεί με τον υπολογιστή. Η επιλογή της πλακέτας πρέπει να είναι σωστή ώστε η ανάρτηση του κώδικα να πραγματοποιηθεί χωρίς την παρουσία σφάλματος.

Ακολουθείται η παρακάτω διαδικασία: Tools → Board (βλ. Εικόνα 8).

Τέλος γίνεται η επιλογή της θύρας με την οποία ο ελεγκτής θα συνδεθεί σειριακά με τον υπολογιστή. Για την επιλογή της θύρας επιλέγω Tools → Port (βλ. Εικόνα 9).



Εικόνα 7. Board Selection



Εικόνα 8. Port Selection

Για το ανέβασμα του προγράμματος στον ελεγκτή θα χρειαστούμε ένα καλώδιο USB A-to-B.

Πριν ξεκινήσει το ανέβασμα του προγράμματος στον ελεγκτή πρέπει να αναλυθεί η λειτουργία κάθε εικονιδίου που βρίσκεται στην γραμμή εργαλείων του Arduino IDE.

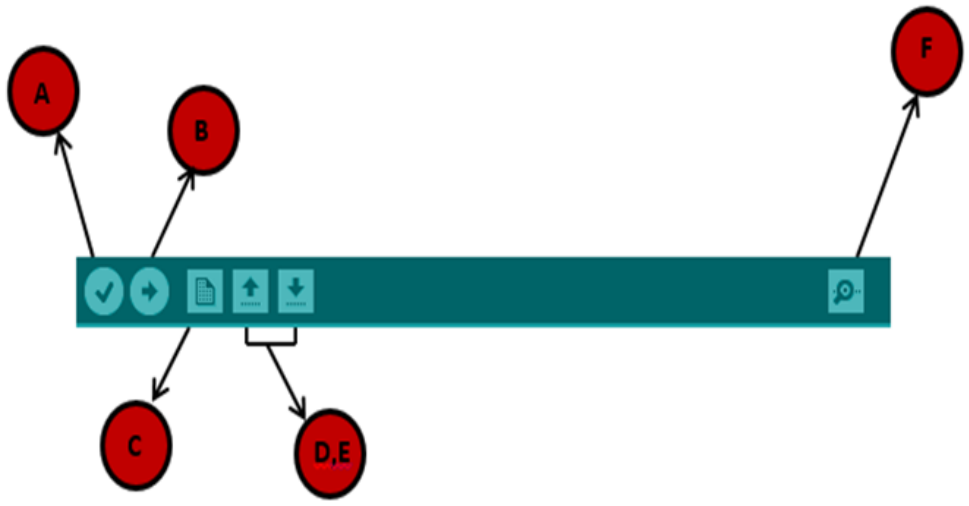
Verify (Επικύρωση)→ Γίνεται έλεγχος σφαλμάτων που μπορεί να προκύψουν στο πρόγραμμα.
(A) Upload (Ανέβασμα) → Γίνεται μεταγλώττιση και ανέβασμα του προγράμματος στον ελεγκτή Arduino(B).

New (Δημιουργία) →Δημιουργία νέου προγράμματος (C)

Open (Άνοιγμα) → Άνοιγμα ενός υπάρχοντος προγράμματος (D)

Save (Αποθήκευση)→Αποθήκευση ενός προγράμματος (E)

Serial Monitor (Παρακολούθηση Σειριακής) → Εμφανίζει τα δεδομένα που στέλνονται από τον ελεγκτή στην σειριακή θύρα και τα δεδομένα που στέλνονται από την σειριακή θύρα στον ελεγκτή (F) (βλ. Εικόνα 7).



Εικόνα 7. Γραμμή εντολών της εφαρμογής Arduino (Πηγή tutorialspoint.com)

Με την επιλογή “Upload” το πρόγραμμα μεταφράζεται και ανεβαίνει στον ελεγκτή. Στον ελεγκτή οι λυχνίες Tx και Rx αρχίζουν να αναβοσβήνουν. Εάν το ανέβασμα ήταν επιτυχημένο στην κάτω δεξιά γωνία του πρόγραμμα θα εμφανιστεί η ένδειξη “Done uploading”.

Σε περίπτωση που το ανέβασμα του προγράμματος στον ελεγκτή δεν είναι εφικτό είναι πολύ πιθανό η σειριακή θύρα να μην λειτουργεί σωστά . Σε αυτή την περίπτωση ο χρήστης επιλέγει Tools -> Bootloader. Σε περίπτωση που δεν δουλέψει πρέπει να γίνει είναι η αναζήτηση των “Arduino Drivers” ή η απεγκατάσταση της ήδη υπάρχουσας εφαρμογής και εγκατάσταση της πιο πρόσφατης έκδοσης από την ιστοσελίδα του Arduino που αναγράφεται παραπάνω. Εάν και αυτή η προσέγγιση αποτύχει καλό είναι να γίνει έλεγχος του καλωδίου USB.

3.6. Η Γλώσσα του Arduino (Arduino Language)

Η γλώσσα που χρησιμοποιεί το Arduino έχει σχεδιαστεί ώστε να υποστηρίζει την επικοινωνία μεταξύ ηλεκτρονικών στοιχείων. Δουλεύει με παρόμοιο τρόπο όπως η γλώσσα Processing είναι όμως διαφορετικά δομημένη ώστε να αντιμετωπίζει διαφορετικά προβλήματα. Η Processing είναι μια γλώσσα ανοιχτού λογισμικού με ολοκληρωμένο περιβάλλον ανάπτυξης, σχεδιασμένη για ηλεκτρονικές τέχνες και visual design.[46]

Η ενασχόληση με τα ηλεκτρονικά στοιχεία έχει να κάνει συνήθως με την αποστολή πληροφοριών υπό μορφή τάσης ή στην περίπτωση της αρχιτεκτονικής υπολογιστών, με την αποστολή μηνυμάτων σε δυαδική μορφή. Παρόλα αυτά οι συγκεκριμένες μορφές επικοινωνίας βρίσκονται πολύ κοντά στην γλώσσα της μηχανής (Low Level). Όταν στέλνει κανείς μια εντολή σαν ένα μοτίβο ηλεκτρικών σημάτων, δουλεύει στο επίπεδο όπου τα ηλεκτρικά στοιχεία

[46] URL: https://processing.org/overview/ - 16.11.2017

επικοινωνούν μεταξύ τους, σε σύγκριση με την αποστολή εντολής με την χρήση μιας γλώσσας υψηλού επιπέδου (High Level).

Η γλώσσα που χρησιμοποιεί κανείς στο προγραμματιστικό περιβάλλον του Arduino είναι βασισμένη στην γλώσσα C. Η γλώσσα C είναι μια παλιά γλώσσα προγραμματισμού, κατάλληλη στην προκειμένη περίπτωση λόγω της φύσης της κατασκευής της. Η C δημιουργήθηκε το 1972 στην επιστημονική και ερευνητική εταιρεία Bell Telephone Laboratories από τον Dennis Ritchie, σε μια εποχή όπου η υπολογιστική ισχύς ήταν δυσεύρετη.[47]

Η γλώσσα C μπορεί να μην είναι τόσο φιλική με έναν νέο χρήστη, είναι όμως οικονομική με τους πόρους της κάνοντας την ιδανική για τον προγραμματισμό μικροελεγκτών εκεί όπου οι πόροι είναι σαφώς λιγότεροι με εκείνους ενός υπολογιστή.

Εάν γνωρίζει κανείς τη γλώσσα C, θα του είναι εύκολο να μάθει να προγραμματίζει μια πλατφόρμα Arduino. Εάν κάποιος γνωρίζει τη γλώσσα Arduino, μπορεί να αντιληφθεί κάλλιστα τη γλώσσα C. Τα βασικά της γλώσσας που χρησιμοποιεί το Arduino είναι παρόμοια με την γλώσσα C++ και την Processing.[48]

[47] Kernighan. W. B, Dennis M. R (1978). *The C Programming Language.* Prentice Hall Software Series.

[48] Noble. J (2009). *Programming Interactivity: A Designer's Guide to Processing, Arduino and Openframeworks*. O'Reily

Media, Inc, p. 108.

3.6.1. Η Δομή ενός Προγράμματος

Όπως αναφέρθηκε παραπάνω η γλώσσα Arduino έχει πολλές ομοιότητες με την γλώσσα Processing όσων αναφορά την δομή του προγράμματος. Υπάρχει η συνάρτηση setup() όπου ο κώδικας μέσα στην συγκεκριμένη δήλωση τρέχει μια φορά κατά την εκκίνηση του προγράμματος και υπάρχει και η συνάρτηση loop() η οποία εκτελείται συνεχώς. Οι περισσότερες εφαρμογές Arduino απαρτίζονται από την δήλωση μεταβλητών, οι οποίες θα χρησιμοποιηθούν κατά την διάρκεια του προγράμματος, μια αρχικοποίηση των παραμέτρων του προγράμματος ώστε να είναι έτοιμο για να λειτουργήσει και συναρτήσεις οι οποίες χρησιμοποιούνται στον κύριο βρόγχο.[49]

3.6.2. Η Συνάρτηση Setup (Setup Function)

Η συνάρτηση setup() είναι το πρώτο κομμάτι κώδικα που εκτελείται σε μια εφαρμογή Arduino. Έστω πως χρειάζεται η χρήση της σειριακής θύρας για την αποσφαλμάτωση ενός προγράμματος, για να εκκινήσει κανείς την σειριακή θύρα χρειάζεται να πληκτρολογήσει την συνάρτηση Serial.begin() μέσα στην συνάρτηση setup(), ώστε να επιτευχθεί η σύνδεση μεταξύ του ελεγκτή και του υπολογιστή. Η σειριακή θύρα είναι μια διεπαφή επικοινωνίας μέσω του υπολογιστή και μιας περιφερειακής συσκευής μέσα από την οποία μεταφέρονται δεδομένα. Μέσω του Serial Monitor στο περιβάλλον του Arduino ο χρήστης με το κατάλληλο μοτίβο εντολών έχει την δυνατότητα να παρακολουθεί τα δεδομένα που μεταφέρονται.

Κάποιες συσκευές πρέπει να αρχικοποιούνται κατά την εκκίνηση του μικροελεγκτή, σε άλλες συσκευές χρειάζεται η αποστολή σήματος να ξεκινήσει μετά την έναρξη τους αλλά πριν

[49] Noble. J (2009), p. 105.

ξεκινήσει η λειτουργία τους. Όλες οι εφαρμογές πρέπει να έχουν μια συνάρτηση setup() ακόμη και στην περίπτωση που δεν έχει γραφτεί τίποτα σε αυτή. Η συγκεκριμένη συνθήκη είναι απαραίτητη για τον μεταφραστή ο οποίος εμφανίζει σφάλμα σε περίπτωση απουσίας της συνάρτησης setup().[50]

3.6.3. Η Συνάρτηση Loop (Loop Function)

Η συνάρτηση loop περιέχει οτιδήποτε χρειάζεται να συμβαίνει συνεχώς στην εφαρμογή, για παράδειγμα, τον έλεγχο της τιμής μιας μεταβλητής, την αποστολή πληροφορίας σε έναν υπολογιστή, την αποστολή σήματος σε έναν ακροδέκτη ή τον έλεγχο της θέσης ενός ρομπότ μέσω της ανάγνωσης τιμών από έναν κωδικοποιητή. Οποιαδήποτε εντολή σε αυτή την συνάρτηση εκτελείται συνεχώς μέχρι την απενεργοποίηση της εφαρμογής.[51]

3.6.4. Τύποι Δεδομένων (Data Types)

Εν αντιθέσει με τον άνθρωπο, ο υπολογιστής δεν γνωρίζει την διαφορά μεταξύ “1234” και “abcd”. Ο τύπος δεδομένων (data type) είναι μια κατηγοριοποίηση που προσδιορίζει τον τύπο δεδομένων που χρησιμοποιεί μια μεταβλητή και ποιες σχετικές πράξεις μπορούν να εφαρμοστούν σε αυτή χωρίς την πρόκληση σφάλματος. Για παράδειγμα ο τύπος δεδομένων string χρησιμοποιείται για την κατηγοριοποίηση αλφαριθμητικών χαρακτήρων και ένας τύπος δεδομένων int (ακέραιος) για την κατηγοριοποίηση ολόκληρων αριθμών.

[50] Noble. J (2009), p. 106

[51] Noble. J (2009), p. 107

Ο τύπος δεδομένων καθορίζει ποιες πράξεις μπορούν να κάνουν χρήση μιας μεταβλητής σε έναν υπολογισμό. Όταν μια γλώσσα προγραμματισμού απαιτεί από μια μεταβλητή να χρησιμοποιείται σε εφαρμογές που λαμβάνουν υπόψιν τον τύπο δεδομένων, αυτή η γλώσσα αναφέρεται ως αυστηρά δακτυλογραφημένη (strongly typed). Με αυτόν τον τρόπο αποφεύγονται σφάλματα, επειδή ενώ είναι λογικό να ζητήσει κανείς από τον υπολογιστή ή μικροελεγκτή να πολλαπλασιάσει έναν int με έναν float είναι παράλογο να ζητήσει κανείς πολλαπλασιασμό μεταξύ 2 μεταβλητών float και string. Όταν μια γλώσσα επιτρέπει σε μια μεταβλητή ενός τύπου δεδομένων να χρησιμοποιηθεί ως τιμή ενός διαφορετικού τύπου δεδομένων, η γλώσσα αυτή ονομάζεται αδύναμα δακτυλογραφημένη (weakly typed). Παρακάτω γίνεται αναφορά στους τύπους δεδομένων της γλώσσας Arduino.[52]

3.6.5. Τύποι δεδομένων της γλώσσας Arduino

void

Ο τύπος void χρησιμοποιείται μονάχα στην δήλωση συναρτήσεων. Υποδεικνύει πως η συνάρτηση αναμένεται να μην επιστρέψει κάποια πληροφορία στην συνάρτηση από την οποία καλέστηκε.

Παράδειγμα:

```
void Loop () {
        //rest of the code
```

[52] URL: http://searchmicroservices.techtarget.com/definition/data-type

}

bool (1 byte)

Μια boolean κρατά 2 τιμές, true ή false.

Παράδειγμα:

bool v = false;

bool s = true;

char (1 byte)

Ο τύπος δεδομένων char κωδικοποιεί τιμές χαρακτήρων και γράφονται με τον εξής τρόπο:

‘A’ ή “ABC” όταν πρόκειται για πίνακα χαρακτήρων.

Παρόλα αυτά οι χαρακτήρες αποθηκεύονται ως αριθμοί. Αυτό σημαίνει πως είναι εφικτό να γίνουν μαθηματικές πράξεις με την χρήση χαρακτήρων στις οποίες χρησιμοποιείται η μορφή ASCII. Για παράδειγμα η πράξη B+1 θα δώσει αποτέλεσμα 67, επειδή η τιμή του B είναι 66.

Παράδειγμα:

char a = ’b’;

char c = 58;

unsigned char (1 byte)

Ο τύπος unsigned char κωδικοποιεί αριθμούς από 0 έως 255.

Παράδειγμα:

unsigned char y = 240;

byte (1 byte)

Ο τύπος unsigned char κωδικοποιεί αριθμούς από 0 έως 255.

Παράδειγμα:

byte a = 200;

int (2 bytes)

Οι ακέραιοι αριθμοί είναι ο βασικός τύπος δεδομένων για αποθήκευση αριθμών. Έχουν εμβέλεια από -32768 έως 32767.

Παράδειγμα:

int a = 10;

unsigned int (2 bytes)

Ο τύπος unsigned int αποθηκεύει θετικές τιμές από 0 έως 65535.

Παράδειγμα:

unsigned int b = 50;

long (4 bytes)

Ο τύπος δεδομένων long κωδικοποιεί τιμές από -2.147.483.648 έως 2.147.483.647.

Παράδειγμα:

long a = 50000;

unsigned long (4 bytes)

Ο τύπος δεδομένων unsigned long αποθηκεύει μόνο θετικές τιμές από 0 έως 4.294.967.295. Έχει το ίδιο εύρος με τον long απλά μετατοπίζεται από το μηδέν και μετά.

Παράδειγμα:

unsigned long a = 10000;

float (4 bytes)

Ο τύπος δεδομένων float απεικονίζει αριθμούς με δεκαδικά ψηφία. Οι αριθμοί τύπου float χρησιμοποιούνται για να προσεγγίσουν αναλογικές και συνεχής τιμές επειδή έχουν μεγαλύτερη ανάλυση από τους ακέραιους (int). Έχουν εύρος από -3.4028235E+38 (E+38 = 10^38) μέχρι 3.4028235E+38.

Παράδειγμα:

float a = 1352,5;

double (4 bytes)

Ο τύπος δεδομένων double έχει την ίδια εφαρμογή με τον τύπο float. Ενώ θα περίμενε κανείς μια μεταβλητή τύπου double να καταλαμβάνει 8 byte στο Arduino (εκτός του Due) καταλαμβάνει 4 byte. Ο λόγος είναι ότι το Arduino είναι ένας μικροελεγκτής της τάξης των 32 bit (εκτός του Due 64 bit). [53]

Παράδειγμα:

double a = 45.352;

3.6.6. Πεδίο Σταθερών και Μεταβλητών (Constant and Variable Scope)

Οι μεταβλητές στην γλώσσα προγραμματισμού C, την οποία χρησιμοποιεί ο Arduino έχουν μια ιδιότητα η οποία ονομάζεται πεδίο. Πεδίο είναι η περιοχή του προγράμματος όπου δηλώνονται οι μεταβλητές.

[53] URL: https://playground.arduino.cc/Code/DatatypePractices

Ο αριθμός δίπλα από την ονομασία του εκάστωτε τύπου δεδομένων δηλώνει το μέγεθος μνήμης που καταλαμβάνει μια

μεταβλητή του συγκεκριμένου τύπου δεδομένων.

Υπάρχουν 2 είδη μεταβλητών, οι καθολικές (global) και οι τοπικές (local). Καθολική ονομάζεται η μεταβλητή η οποία αναγνωρίζεται από κάθε λειτουργία του προγράμματος. Τοπική ονομάζεται η μεταβλητή η οποία είναι ορατή μόνο στην συνάρτηση στην οποία δηλώθηκε.

Στο περιβάλλον του Arduino οποιαδήποτε μεταβλητή έχει οριστεί έξω από μια συνάρτηση (setup(), loop(), κ.α) θεωρείται ως καθολική.[54]

3.6.7. Τελεστές

Τελεστής ονομάζεται το σύμβολο το οποίο χρησιμοποιεί ο μεταφραστής του προγράμματος για να εκτελέσει εντολές και υπολογισμούς. Για παράδειγμα μπορεί κανείς να θέσει μια τιμή με τον τελεστή =, να ελέγξει την ισότητα δυο τιμών ή τελεστών ==, να προσθέσει με τον τελεστή + και πολλά άλλα.

Υπάρχουν 3 κατηγορίες τελεστών. Οι πρώτοι είναι οι μαθηματικοί τελεστές που εκτελούν μαθηματικές πράξεις. Οι δεύτεροι είναι οι τελεστές εκχώρησης που αλλάζουν τιμή σε μια μεταβλητή. Τρίτοι είναι οι τελεστές σύγκρισης οι οποίοι καθορίζουν εάν 2 μεταβλητές είναι ίσες, διάφορες, μεγαλύτερες από ή μικρότερες από άλλες μεταβλητές.[55]

Τελεστής	Χρήση
+, -, *, /	Πρόσθεση, Αφαίρεση, Πολλαπλασιασμός, Διαίρεση.

[54] URL: https://www.arduino.cc/reference/en/language/variables/variable-scope--qualifiers/scope/

[55] Noble. J (2009), p. 36

%	Υπόλοιπο: Επιστρέφει το υπόλοιπο μιας διαίρεσης.
=	Εκχώρηση: εκχωρεί την τιμή στα δεξία στην μεταβλητή στα αριστερά.
+=, -=, *=, /=	Μαθηματική εκχώρηση: Προσθέτει, αφαιρεί, πολλ/ζει ή διαιρεί την τιμή/μεταβλητή στα δεξιά με την τιμή στα αριστερά και θέτει το αποτέλεσμα στην τιμή/μεταβλητή στα αριστερά.
++	Προσθέτει 1 από την τιμή/μεταβλητή στα αριστερά.
--	Αφαιρεί 1 από την τιμή/μεταβλητή στα αριστερά.
==	Συγκρίνει την τιμή/μεταβλητή στα αριστερά με την τιμή/μεταβλητή στα δεξιά. Εάν είναι ίσες η συνθήκη είναι αληθής (true).
!=	Συγκρίνει την τιμή/μεταβλητή στα αριστερά με την τιμή μεταβλητή στα δεξιά. Εάν δεν είναι ίσες η συνθήκη είναι αληθής (true).
>, >=	Συγκρίνει την τιμή/μεταβλητή στα αριστερά με την τιμή/μεταβλητή στα δεξιά. Εάν η συνθήκη είναι μεγαλύτερη ή μεγαλύτερη/ίση από την τιμή στα δεξιά η συνθήκη είναι αληθής (true).
<, <=	Συγκρίνει την τιμή/μεταβλητή στα αριστερά με την τιμή/μεταβλητή στα δεξιά. Εάν η συνθήκη είναι μικρότερη ή μικρότερη/ίση από την τιμή στα δεξιά η συνθήκη είναι αληθής (true).

&&	Ελέγχει εάν η συνθήκη είναι αληθής, αριστερά και δεξιά του τελεστή. Εάν και οι 2 είναι αληθής ολόκληρη η συνθήκη είναι αληθής.
||	Ελέγχει εάν η συνθήκη είναι αληθής, αριστερά και δεξιά του τελεστή. Είτε η μία είτε η άλλη είναι αληθής ολόκληρη η συνθήκη είναι αληθής.

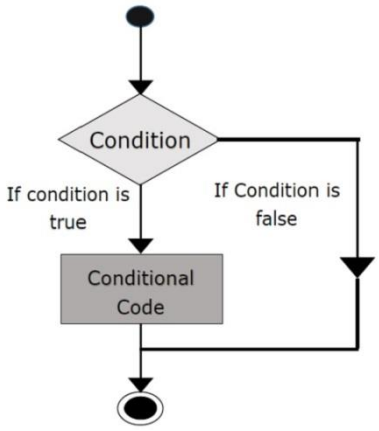
Πίνακας 1. Πίνακας Τελεστών.

3.6.8. Δομές Ελέγχου και Επανάληψης (Control and Loop Statements)

Πολύ συχνά προκύπτει η ανάγκη να τροποποιήσει κανείς τις εκτελούμενες εντολές ενός προγράμματος υπό συνθήκη. Τον συγκεκριμένο σκοπό εκπληρώνουν οι συνθήκες ελέγχου και επανάληψης. Η συνθήκη ελέγχου απαιτεί τον ορισμό μιας ή περισσότερων προϋποθέσεων ώστε να ελεγχθούν από το πρόγραμμα. Εάν η προϋπόθεση είναι αληθής (true) η λογική ροή του προγράμματος θα λάβει την κατεύθυνση που έχει προκαθορίσει ο χρήστης, σε σχέση με την αντίθετη περίπτωση όπου η συνθήκη είναι ψευδής (false). Ο βρόγχος επανάληψης εκτελεί μια λειτουργία έως ότου μια συνθήκη εκπληρωθεί ανεξάρτητα από τον αριθμό των επαναλήψεων.[56]

Δομή If/else

Η δομή if ελέγχει μια έκφραση και εκτελεί μια εντολή ή ένα πλήθος εντολών στην περίπτωση όπου η συνθήκη μέσα στην παρένθεση είναι αληθής[57].



[56] Noble. J (2009), p. 38

[57] URL: https://www.arduino.cc/reference/en/language/structure/control-structure/if/

Εικόνα 9. If Statement (Πηγή google.gr)

Παράδειγμα:

```
if (temperature >= 70)
{
  Serial.println("Danger! Shut Down the System");
}
else if (temperature >=60 && temperature < 70);
{
Serial.println("Warning! Attention required");
}
else
{
Serial.print("Safe! Continue task");
}
```

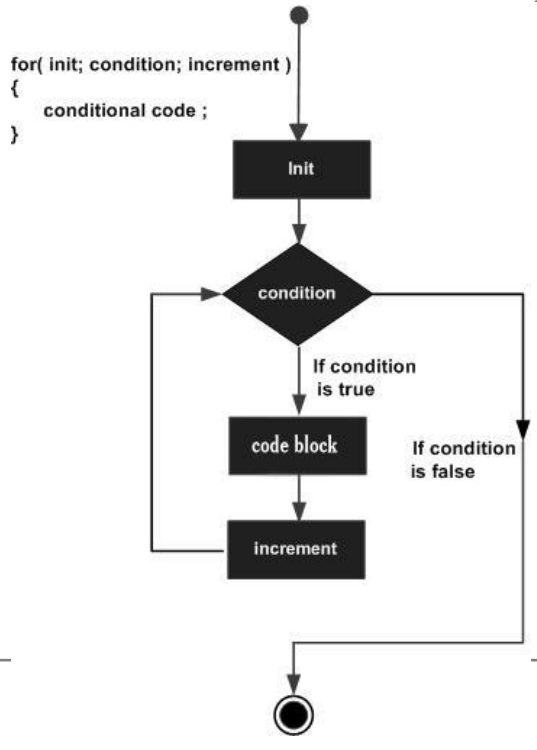
Βρόγχος Επανάληψης for (for Loop)

Ο βρόγχος for χρησιμοποιείται για την εκτέλεση ενός κομματιού κώδικα για συγκεκριμένο αριθμό επαναλήψεων[58].

[58] URL: https://www.arduino.cc/reference/en/language/structure/control-structure/for/

Παράδειγμα:

```
void loop()
{
   int x = 10;
   for (int I = 0; I < x; I++)
   {
        analogWrite(13 , I);
        if( i==9)
        {
                Serial.println(“Last call”);
        }
   }
                                                                    }
```



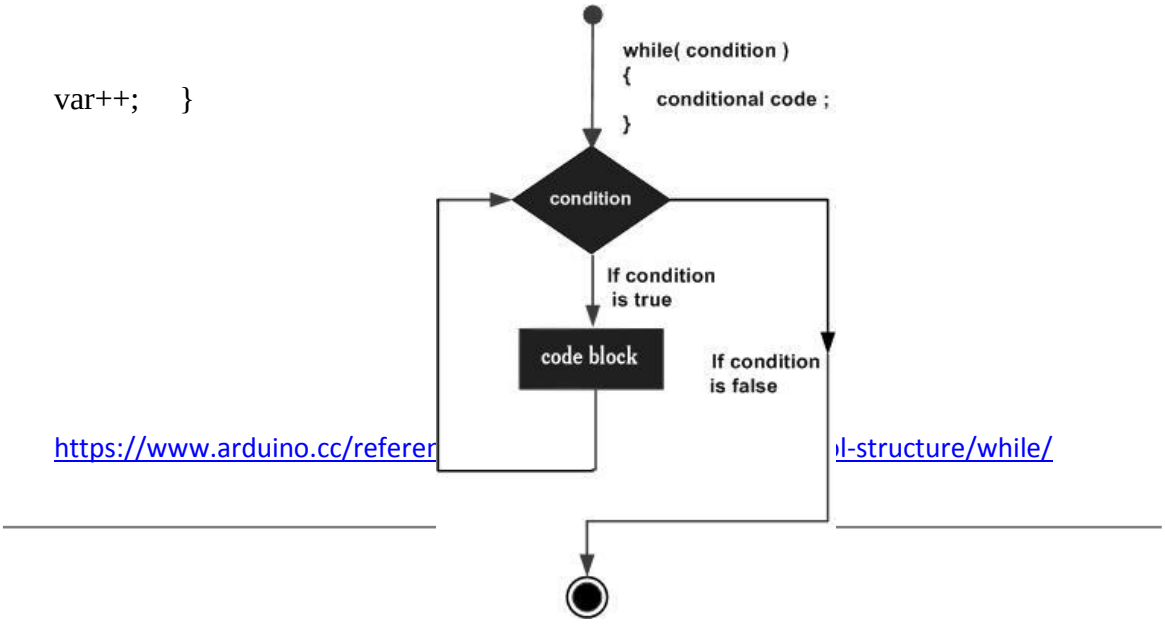
Εικόνα 10. for loop (πηγή Google.gr)

Βρόγχος Επανάληψης while (while Loop)

Η συνθήκη while επαναλαμβάνεται μέχρι η έκφραση μέσα στην παρένθεση να γινει ψευδής (false)[59].

Παράδειγμα:

```
void loop ()
{
   int var = 0;
   while (var < 200)
    {
        Serial.println(var);
        var++;     }
}
```



[59] https://www.arduino.cc/referer ol-structure/while/

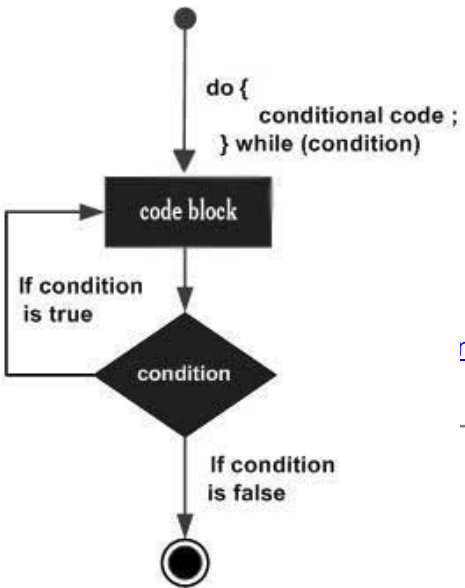
Εικόνα 11. While Loop (Πηγή google.gr)

Βρόγχος Επανάληψης do while (Do While Loop)

Ο βρόγχος do while δουλεύει με τον ίδιο ακριβώς τρόπο όπως η while με την διαφορά πως η συνθήκη του βρόγχου ελέγχεται στο τέλος παρά στην αρχή, άρα η διαδικασία θα τρέξει τουλάχιστον μια φορά[60].

Παράδειγμα:

```
do
{
delay(50);
x= readSensors();
} while(x < 100);
```



[60] URL: https://www.arduinc ntrol-structure/dowhile/

Εικόνα 12. Do While Loop (Πηγή google.gr)

continue

Η δήλωση continue παραλείπει την τρέχουσα επανάληψη ενός βρόγχου (For, While, Do While) και συνεχίζει προχωρώντας στην επόμενη επανάληψη[61].

Παράδειγμα:

```
for (x=0; x <= 255; x++)
{
        if (x > 40 && x < 120)
        {
                continue;
        }
        analogWrite(13 , x);
```

[61] URL: https://www.arduino.cc/reference/en/language/structure/control-structure/continue/

```
    delay(50);
}
```

break

Η δήλωση break χρησιμοποιείται για να σταματήσει έναν βρόγχο (For, While, Do While) παρακάμπτωντας την συνθήκη επανάληψης[62].

Παράδειγμα:

```
for (x=0; x < 255; x+=)
{
    analogWrite(sensor);
    if (sensor > threshold)
    {
        x = 0;
        break;
    }
    delay(50);
}
```

[62] URL: https://www.arduino.cc/en/Reference.Break

3.6.9. Πίνακες (Arrays)

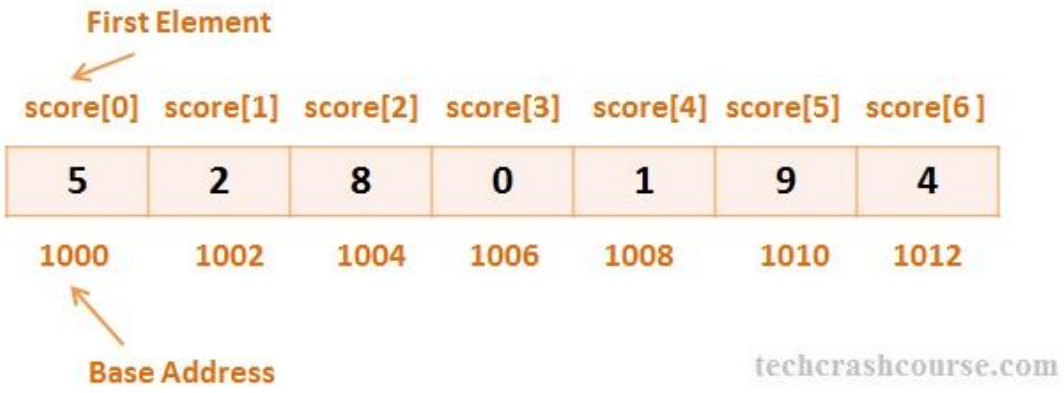
Πίνακας είναι μια διαδοχική, από περιοχές μνήμης, ομάδα δεδομένων του ίδιου τύπου. Πιο απλά ο πίνακας είναι μια λίστα από πολλαπλά στοιχεία του ίδιου τύπου δεδομένων, για παράδειγμα int ή char. Για να αναφερθεί κανείς σε ένα στοιχείο του πίνακα πρέπει να προσδιορίσει το όνομα του πίνακα και τον αριθμό της θέσης του συγκεκριμένου στοιχείου. Η θέση ενός πίνακα ονομάζεται θέση μνήμης (base address) και η αρίθμηση των στοιχείων ξεκινά παντοτε από το 0. Έτσι λαμβάνοντας υπόψιν τον πίνακα score τα στοιχεία του προσδιορίζονται ως score[0], score[1] κ.α. Στην παρακάτω εικόνα (βλ. Εικόνα 16) διακρίνεται η θέση που καταλαμβάνει ο κάθε δείκτης στην περιοχή μνήμης[63].

Παράδειγμα:

```
int n[10];

void setup() {}

void loop()

{

  for (int I = 0;i < 10;i++)

   {

        n[i] = 0;

   }

     for (int j = 0;j < 10;j++)

   {
```

[63] URL: https://www.arduino.cc/reference/en/language/variables/data-types/array/

```
    Serial.println(n[j]);
}
```



Εικόνα 13. Array (Η base address ισχύει για όλα τα Arduino εκτός του Due και SAMD) (πηγήGoogle.gr)

3.6.10. Συμβολοσειρές (Strings)

Οι συμβολοσειρές μπορούν να δημιουργηθούν με δύο τρόπους, είτε με την χρήση δεδομένων τύπου String, είτε με την χρήση πινάκων τύπου char. Με τον τύπο δεδομένων String οι συμβολοσειρές δηλώνονται ως αντικείμενα. Όποιος και να είναι ο τρόπος σύνταξης μια συμβολοσειρά είναι σε θέση να αναπαριστά ακολουθίες χαρακτήρων.

Στην περίπτωση όπου μια ακολουθία χαρακτήρων απεικονιστεί με την χρήση πίνακα char πρέπει να έχει κατά νου τον μηδενικό χαρακτήρα (κώδικας ASCII 0) ο οποίος μπαίνει στο τέλος της συμβολοσειράς καταλαμβάνοντας μια θέση. Αυτό σημαίνει πως στην περίπτωση που θέλει κανείς να γράψει την λέξη “arduino” σε μορφή συμβολοσειράς η οποία απαρτίζεται από 7 χαρακτήρες, πρέπει να υπολογίσει και τον μηδενικό χαρακτήρα οπότε ο πίνακας χρειάζεται 8

θέσεις για την αναπαράσταση ολόκληρης της λέξης. Η παρουσία του μηδενικού χαρακτήρα επιτρέπει στην συνάρτηση Serial.print() να γνωρίζει πότε τελειώνει μια συμβολοσειρά[64].

Παράδειγμα:

```
void setup() {}
  char str[8];
  Serial.begin(9600);
   str[0] = 'A';
  str[1] = 'R';
  str[2] = 'D';
  str[3] = 'U';
  str[4] = 'I';
  str[5] = 'N';
  str[6] = 'O';
  str[7] = 0;
}
void loop() {
  String one = "Hello";
```

[64] URL: https://www.arduino.cc/reference/en/language/variables/data-types/string/

```
  String one1 = String(a);

  String one2 = String("This is a String");

  String one3 = String(13);

  String one4 = String(45 , HEX);

  String one5 = String(255 , BIN);

}
```

3.6.11. Συναρτήσεις (Functions)

Η συνάρτηση είναι ένα μοτίβο εντολών το οποίο είναι σε θέση να εκτελέσει μια ορισμένη λειτουργία στο σημείο του προγράμματος που κλήθηκε. Χαρακτηριστικά γνωρίσματα μιας συνάρτησης είναι το όνομα, ο τύπος δεδομένων επιστροφής, οι παράμετροι οι οποίες αντικαθιστούν τις μεταβλητές του κυρίου προγράμματος μέσα στην συνάρτηση καθώς και μια μεταβλητή/τιμή η οποία επιστρέφεται στο κυρίως πρόγραμμα και συμπίπτει με τον τύπο δεδομένων της συνάρτησης. Όπως έχει αναφερθεί παραπάνω ένα πρόγραμμα Arduino χρειάζεται τις συναρτήσεις setup() και loop() για να λειτουργήσει, οποιαδήποτε άλλη συνάρτηση πρέπει να δημιουργηθεί έξω από τα όρια τους. Η τυπική περίπτωση για τη δημιουργία μιας συνάρτησης είναι η ανάγκη εκτέλεσης μιας συγκεκριμένης λειτουργίας αρκετές φορές σε ένα πρόγραμμα[65].

Παράδειγμα:

[65] URL: https://www.arduino.cc/en/Reference/FunctionDeclaration

```
int function( int x, int y){
  int result;
  result = x * y;
  return result;
}
void loop() {
  int I = 2;
  int j = 3;
  int k;
  k = function(i , j); // Το αποτέλεσμα θα είναι 6
}
```

3.5.11 Συναρτήσεις Εισόδου/Εξόδου (Input/Output Functions)

pinMode (Pin , Mode)

Ένας ακροδέκτης μπορεί να οριστεί είτε σαν Είσοδος είτε σαν Έξοδος σε ένα πρόγραμμα. Η συνάρτηση pinMode() ορίζει την κατεύθυνση που θα ακολουθήσει η ροή της πληροφορίας. Η δήλωση γίνεται στην συνάρτηση setup() όπου αρχικοποιούνται μια φορά οι πληροφορίες του προγράμματος.

digitalWrite (Pin, Value)

Η συνάρτηση digitalWrite θέτει την τιμή ενός ακροδέκτη σε λογικό 1 (HIGH) ή λογικό 0 (LOW), με απλά λόγια στέλνει 5V ή 0V.

digitalRead (Pin)

Η συνάρτηση digitalRead διαβάζει την τιμή ενός ακροδέκτη επιστρέφοντας HIGH ή LOW.

analogRead (Pin)

Η συνάρτηση analogRead διαβάζει την τιμή ενός ακροδέκτη επιστρέφοντας ένα εύρος τιμών από 0 έως 1023.

delay()

Αναβάλλει την λειτουργία του πρόγραμματος για ορισμένο χρόνο σε milliseconds.

millis()

Επιστρέφει τον χρόνο λειτουργίας του προγράμματος από την έναρξη του σε milliseconds.

Κλάση (Class)

Κλάση ονομάζεται ένα πρότυπο κώδικα για την δημιουργία αντικειμένων στα οποία παρέχεται αρχικοποίηση και εφαρμογή συγκεκριμένων λειτουργιών. Το σώμα μιας κλάσης αποτελείται από τα ιδιωτικά (private) και δημόσια (public) δεδομένα. Τα ιδιωτικά μέλη της κλάσης είναι προσπελάσιμα μόνο μέσα από την κλάση και όχι από κάποια μεταβλητή που έχει δηλωθεί στο κυρίως πρόγραμμα. Τα δημόσια μέλη είναι προσπελάσιμα μέσα και έξω από την κλάση. Κάποια δεδομένα της κλάσης είναι ιδιωτικά ώστε να μην μπορούν να προσπελαστούν καταλάθος από άλλες συναρτήσεις που βρίσκονται έξω από αυτή.

Τα στοιχεία δεδομένων που περιέχονται σε μια κλάση ονομάζονται μέλη δεδομένων (data members). Οι συναρτήσεις που περιέχονται σε μια κλάση ονομάζονται συναρτήσεις-μέλη. Τα αντικείμενα της κλάσης αποτελούν την παρουσία της κλάσης μέσα στο πρόγραμμα και χρησιμοποιούν τα μέλη της ως ιδιότητές τους. Στον Arduino οι κλάσεις συνήθως περιέχονται σε αρχεία βιβλιοθηκών με κατάληξη .h και περιλαμβάνονται στον κώδικα ενός πηγαίου προγράμματος με την εντολή #include[66].

[66] Lafore R. (2002). Object – Oriented Programming in C++ 4th Edition. Pearson Education Inc. p. 239.

4. ROS (Robot Operating System)

Σε αυτή την ενότητα ο χρήστης εμβαθύνει στην αρχιτεκτονική του ROS από την εγκατάσταση και την αρχικοποίηση του μέχρι την χρήση των λειτουργιών του.

4.1. Εισαγωγή

Το ROS (Robot Operating System) είναι ένα ευέλικτο περιβάλλον ανάπτυξης λογισμικού για εφαρμογές ρομποτικής. Παρέχει μια συλλογή από εργαλεία, βιβλιοθήκες και κανόνες με σκοπό την δημιουργίας πολύπλοκων και εξειδικευμένων ρομποτικών συστημάτων.

Το ROS ξεκίνησε το 2007 με το όνομα Switchyard από τον Morgan Quingley ως κομμάτι της ρομποτικής μελέτης του προγράμματος STAIR στο Stanford University[67]. Η κυρίως ανάπτυξη του ROS συνεχίστηκε το 2008 στην εταιρεία Willow Garage[68].

Η δημιουργία ενός λογισμικού γενικού σκοπού για εφαρμογές στην ρομποτική είναι ένα δύσκολο επίτευγμα. Τα προβλήματα που αντιμετωπίζει ένα ρομποτικό σύστημα μπορεί να μοιάζουν ασήμαντα σε έναν άνθρωπο υπάρχει όμως η πιθανότητα να εκδηλωθούν σε ακραίες διαστάσεις σε ένα περιβάλλον λογισμικού και η αντιμετώπισή τους είναι μια πρόκληση που δεν μπορεί να διεκπεραιωθεί μόνο από μια ερευνητική ομάδα ή ένα ερευνητικό κέντρο. Ως απορροή των παραπάνω, το ROS δημιουργήθηκε για να ενθαρρύνει την συλλογική ανάπτυξη ενός λογισμικού περιβάλλοντος με στόχο την χρήση στο ευρύτερο φάσμα των ρομποτικών εφαρμογών. Για παράδειγμα μια ομάδα μπορεί να έχει ειδίκευση στην χαρτογράφηση εσωτερικών χώρων και να συμβάλει στην ανάπτυξη ενός προγράμματος με στόχο την

67
68

δημιουργία χαρτών και μια άλλη ομάδα να κατέχει αλγόριθμους για την καλύτερη πλοήγηση ενός αυτοοδηγούμενου οχήματος σε ένα εσωτερικό χώρο. Το ROS μέσω της συλλογικής έρευνας εξελίσσεται συνεχώς χαράζοντας μια νέα πορεία στο μέλλον της ρομποτικής[69].

4.2. Πλεονεκτήματα του ROS (Advantages of ROS)

Το ROS συνοδεύεται από πακέτα εφαρμογών όπως για παράδειγμα τα πακέτα SLAM (Simultaneous Localization and Mapping) και AMCL (Adaptive Monte Carlo Localization) τα οποία χρησιμοποιούνται για την χαρτογράφηση ενός χώρου από αυτόνομα ρομπότ. Αυτές οι δυνατότητες ρυθμίζονται με την χρήση παραμέτρων και χρησιμοποιούνται άμεσα σε εφαρμογές.

Το ROS είναι εξοπλισμένο με μια πληθώρα εργαλείων όπως το Rviz και το Gazebo για την αποσφαλμάτωση, οπτικοποίηση και εκτέλεση προσομοιώσεων.

Είναι εξοπλισμένο από διάφορους οδηγούς αισθητήρων όπως ανιχνευτές Laser, Velodyne – LIDAR, Kinect και άλλους οι οποίοι συνδέονται με το ROS. Είναι ευέλικτο διότι επιτρέπει σε οποιοδήποτε ενδιάμεσο λογισμικό την επικοινωνία προγραμμάτων τα οποία είναι γραμμένα σε διαφορετικές γλώσσες από C++ και C έως Python και Java. Το λογισμικό του είναι χωρισμένο σε κόμβους (nodes) όπου ο καθένας εκτελεί ένα συγκεκριμένο φάσμα εργασιών. Σε περίπτωση κατάρρευσης ενός κόμβου οι υπόλοιποι συνεχίζουν να δουλεύουν διασφαλίζοντας με αυτόν τον τρόπο την ομαλή λειτουργία κατανεμημένων διεργασιών. Η κοινότητα του ROS παρέχει υποστήριξη σε όλους τους χρήστες του παγκοσμίως και αναπτύσσεται συνεχώς[70].

69 URL: http://www.ros.org/about-ros/

70 Lentin J. (2015). *Mastering ROS for Robotics Programming*. Packt Publishing Ltd. p.3.

4.3. Εγκατάσταση Ubuntu

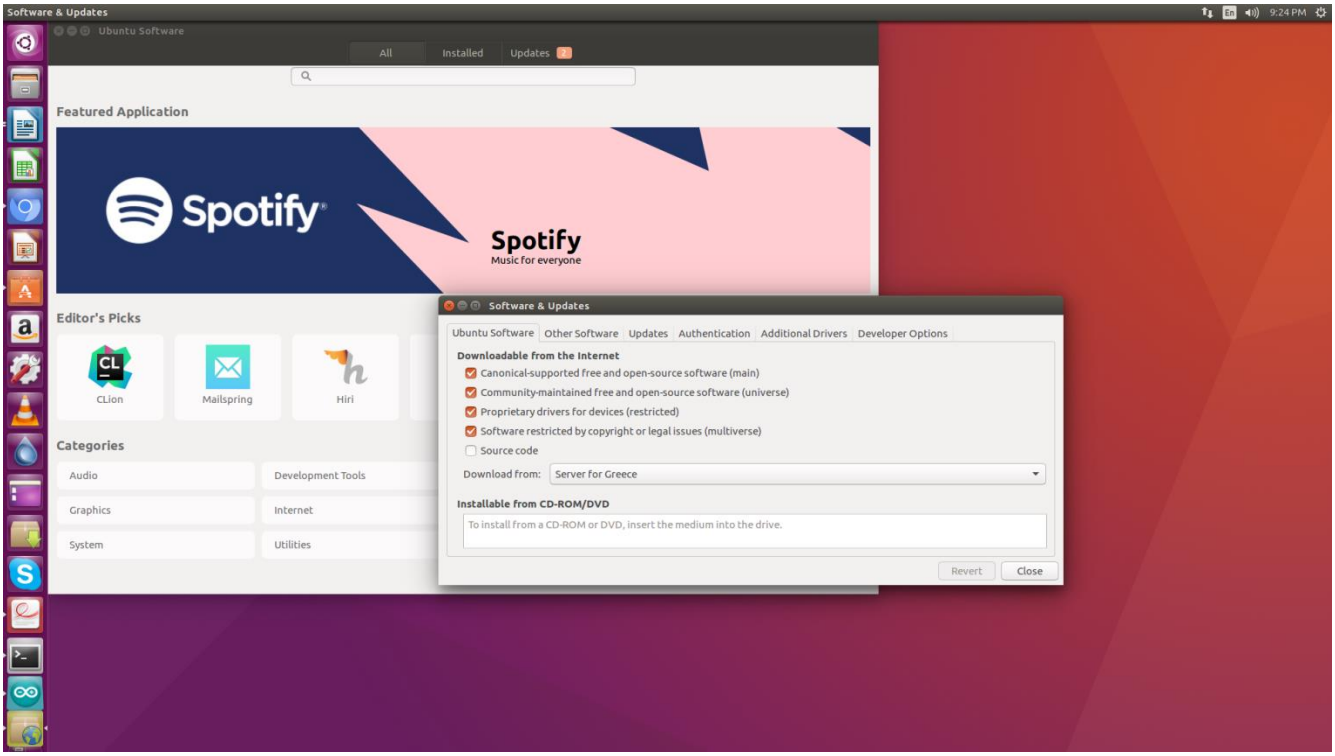
Το ROS λειτουργεί με βέλτιστο τρόπο σε συστήματα αρχιτεκτονικής Unix όπως Linux και MacOS. Η έκδοση ROS που θα χρησιμοποιηθεί είναι η ROS Kinetic Kame η οποία υποστηρίζεται από τα λειτουργικά συστήματα Wily (Ubuntu 15.10), Xenial (Ubuntu 16.04) και Jesie (Debian 8) τα οποία αποτελούν τις πιο γνωστές διανομές Linux. Η εγκατάσταση του Ubuntu 16.04 καθώς και ο οδηγός εγκατάστασης δίνονται παρακάτω:

https://www.ubuntu.com/download/desktop – Εγκατάσταση Ubuntu 16.04 LTS (Long Term Support)

https://tutorials.ubuntu.com/tutorial/tutorial-install-ubuntu-desktop#0 – Οδηγός εγκατάστασης

Εφόσον η εγκατάσταση είναι επιτυχής είναι απαραίτητο να γίνει η ρύθμιση των αποθετηρίων (repositories) του λειτουργικού συστήματος ώστε να επιτρέπεται η χρήση πακέτων πληροφοριών που μπορεί να μην υποστηρίζεται από το σύστημα. Repositories ονομάζονται τα αρχεία λογισμικού, τα οποία διευκολύνουν την εγκατάσταση νέου υλικού παρέχοντας ταυτόχρονα προστασία στον χρήστη. Επιλέγεται εικονίδιο Ubuntu Software → Software and Updates όπου εμφανίζεται ένα μενού επιλογών. Σημειώνονται οι επιλογές που εικονίζονται στην παρακάτω εικόνα (βλ. Εικόνα 16). Το σύστημα είναι έτοιμο για την εγκατάσταση του ROS [71].

[71] URL: https://help.ubuntu.com/community/Repositories/Ubuntu



Εικόνα 14. Repositories

4.4. Εγκατάσταση του ROS Kinetic

Το πρώτο βήμα είναι η ενημέρωση του συστήματος ώστε να δέχεται λογισμικό από τη σελίδα packages.ros.org. Σε ένα τερματικό (Ctrl + Alt + T ή Δεξί Κλίκ → Άνοιγμα Τερματικού) πληκτρολογείται η παρακάτω εντολή:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

Το επόμενο βήμα είναι απαραίτητο για την επιβεβαίωση της προέλευσης του κώδικα και τη διαβεβαίωση πως οι πληροφορίες δεν έχουν τροποποιηθεί χωρίς την συναίνεση του χρήστη. Κανονικά όταν δηλώνει κανείς έναν νέο αποθηκευτικό χώρο πρέπει να προσθέτει ένα κλειδί ώστε να θεωρηθεί έμπιστος από το σύστημα. Για τον συγκεκριμένο σκοπό πληκτρολογείται η εντολή:

sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116

Πριν την εγκατάσταση είναι απαραίτητη η ενημέρωση του λειτουργικού συστήματος με τις πιο πρόσφατες ενημερώσεις. Πληκτρολογούνται οι ακόλουθες εντολές:

sudo apt-get update

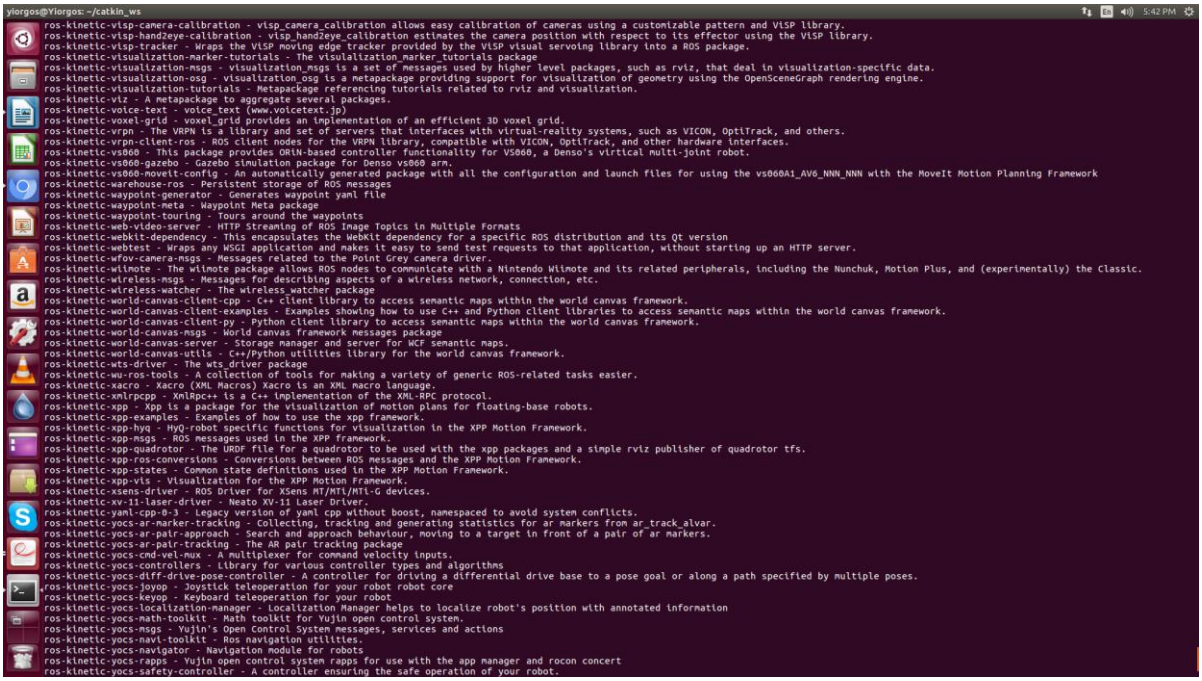
sudo apt-get upgrade

Για την έναρξη της εγκατάστασης πληκτρολογείται η εντολή:

sudo apt-get install ros-kinetic-desktop-full

Με την εγκατάσταση του ROS ο χρήστης έχει πρόσβαση στην αποθήκη πακέτων τα οποία χρεώνονται λειτουργίες για διάφορες περιπτώσεις (βλ. Εικόνα 17). Για την πρόσβαση στην συγκεκριμένη αποθήκη πληκτρολογείται η εντολή:

apt-cache search ros-kinetic



Εικόνα 15. Αποθήκη Πακέτων του ROS

Μερικές φορές τα πακέτα του ROS χρειάζονται βιβλιοθήκες και εργαλεία τα οποία παρέχονται από το λειτουργικό σύστημα. Τα αρχεία αυτά αναφέρονται και ως εξαρτήσεις (dependencies) και συμβαίνει αρκετές φορές να μην έχουν εγκατασταθεί από την αρχή. Το ROS παρέχει το εργαλείο rosdep το οποίο εγκαθιστά εξαρτήσεις στο σύστημα με την χρήση των ακόλουθων εντολών [72]:

```
sudo rosdep init

rosdep update
```

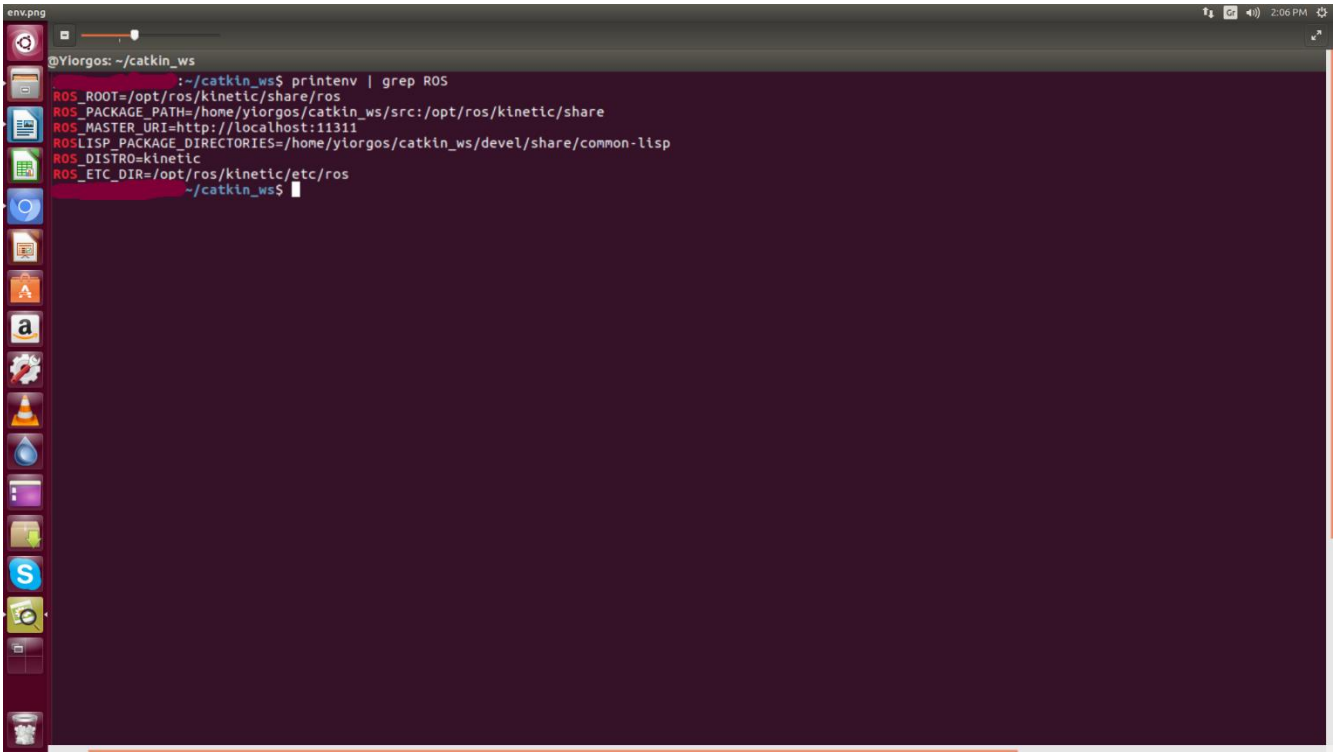
72 URL: http://wiki.ros.org/ROS/Tutorials/rosdep

4.5. Διαχείριση του Περιβάλλοντος Εργασίας

Αρχικά πρέπει να βεβαιωθεί κανείς πως οι μεταβλητές περιβάλλοντος (environment variables) έχουν τεθεί. Οι μεταβλητές περιβάλλοντος επηρεάζουν την συμπεριφορά του ROS και εξυπηρετούν διάφορους ρόλους όπως την εύρεση πακέτων, τροποποιήσεις στην εκτέλεση κόμβων και στην ανάπτυξη του συστήματος κ.α. Πληκτρολογείται η παρακάτω εντολή:

printenv | grep ROS

Εάν εμφανιστούν οι παρακάτω μεταβλητές το περιβάλλον εργασίας έχει εγκατασταθεί με επιτυχία (βλ. [73].



Εικόνα 16. Environmental Variables

73 URL: http://wiki.ros.org/ROS/EnvironmentVariables

Έπειτα είναι απαραίτητη η ενσωμάτωση κάποιων αρχείων τύπου setup (τα οποία παράγονται ως απορροή της ανάπτυξης ή της εγκατάστασης πακέτων catkin) στο κέλυφος του περιβάλλοντος εργασίας ώστε να μπορεί κανείς να έχει πρόσβαση στις εντολές του ROS. Το κέλυφος (shell) είναι μια διαδραστική σύνδεση όπου φέρνει τον χρήστη σε επαφή με τον πυρήνα του λειτουργικού συστήματος. Στο λειτουργικό σύστημα Ubuntu τον ρόλο αυτό εκπληρώνει το τερματικό (γραμμή εντολών). Πληκτρολογείται η παρακάτω εντολή:

```
echo “source /opt/ros/kinetic/setup.bash”>> ~/.bashrc

source ~/.bashrc
```

Η παραπάνω εντολή γράφεται στο αρχείο .bashrc το οποίο είναι υπεύθυνο για την ρύθμιση παραμέτρων σε κάθε τερματικό που ανοίγει ο χρήστης. Με αυτό τον τρόπο κάθε φορά που εκτελείται ένα νέο τερματικό το σύστημα γνωρίζει την τοποθεσία των αρχείων που είναι απαραίτητα για την σωστή λειτουργία του προγράμματος.

Απαραίτητο πακέτο το οποίο δεν περιλαμβάνεται στην αρχική εγκατάσταση είναι το rosinstall το οποίο προσφέρει απαραίτητη βοήθεια στο κατέβασμα πολλαπλών πακέτων.

Πληκτρολογείται η εντολή:

```
sudo apt-get install python-rosinstall
```

4.6. Το Σύστημα Ανάπτυξης του ROS (ROS Development System)

Το Σύστημα ανάπτυξης είναι ένα εργαλείο λογισμικού το οποίο έχει σχεδιαστεί για την αυτοματοποίηση της μεταγλώττισης ενός προγράμματος. Στον πυρήνα του ένα σύστημα

ανάπτυξης είναι υπεύθυνο για την μετατροπή πηγαίου κώδικα σε μορφή εκτελέσιμου αρχείου. Η πρώτη εντύπωση για ένα σύστημα ανάπτυξης είναι πως κάθε πράξη πχ κάθε εντολή με την ίδια είσοδο και τις ίδιες επιλογές παράγει την ίδια έξοδο. Αυτή η θεώρηση επιτρέπει στο σύστημα να απομνημονεύει τις πράξεις που έχει ήδη εκτελέσει και να αναπτύσσει μόνο αρχεία τα οποία έχουν υποστεί αλλαγή. Για την παραγωγή αρχείων αλλά και τη σωστή διαχείριση το σύστημα πρέπει να γνωρίζει ποιοι πόροι χρησιμοποιούνται σε κάθε εντολή. Έτσι πληροφορίες όπως η θέση του πηγαίου κώδικα, οι εξαρτήσεις του κώδικα, η θέση των εξαρτήσεων, ποια αρχεία πρέπει να αναπτυχθούν, που πρέπει να αναπτυχθούν και να εγκατασταθούν είναι απαραίτητες[74].

Κάποια γνωστά συστήματα ανάπτυξης είναι το Cmake, το GNU make, το Make κ.α. Το σύστημα ανάπτυξης του ROS είναι το catkin το οποίο χρησιμοποιεί εντολές Cmake και κώδικα Python. Η ροή εργασιών του είναι παρόμοια με εκείνη του Cmake, με την διαφορά πως το catkin είναι σε θέση να διαχειριστεί πακέτα πληροφοριών τα οποία χρησιμοποιούν διαφορετικές γλώσσες προγραμματισμού, εργαλεία και κανόνες. Με λίγα λόγια παρέχει υποστήριξη για αυτόματη οργάνωση και ανάπτυξη πολλαπλών πακέτων κάνοντας ευκολότερη και πιο απλοποιημένη την διαδικασία ακόμη και για εκείνους που δεν έχουν εμπειρία στην ανάπτυξη λογισμικού [75].

[74] URL: https://www.cs.virginia.edu/~dww4s/articles/build_systems.html

[75] URL : http://wiki.ros.org/catkin/conceptual_overview

4.7. Δημιουργία του Χώρου Εργασίας Catkin (Creation of a Catkin Workspace)

Τα πακέτα catkin μπορούν να δημιουργηθούν ως αυτοδύναμα project με τον ίδιο τρόπο που θα μπορούσε να δημιουργηθεί μια εργασία με την χρήση του Cmake. Το σύστημα ανάπτυξης catkin παρέχει την έννοια του χώρου εργασίας (workspace). Ένα catkin workspace είναι ένας φάκελος όπου μπορούν να δημιουργηθούν πολλαπλά πακέτα αλληλοεξαρτώμενα μεταξύ τους και περιέχει τέσσερις φακέλους όπου ο καθένας εξυπηρετεί έναν συγκεκριμένο ρόλο στην διαδικασία της ανάπτυξης λογισμικού [76].

Πηγαίος Χώρος(Source Space)

Ο Πηγαίος Χώρος περιέχει τον πηγαίο κώδικα των πακέτων catkin. Ένας φάκελος μέσα στο source space περιέχει ένα ή περισσότερα πακέτα. Αυτός ο χώρος δεν επηρεάζεται από την διαμόρφωση ή την ανάπτυξη των πακέτων και περιέχει ένα σύνδεσμο με το αρχείο CmakeLists.txt. Αυτός ο φάκελος καλείται κατά την διαμόρφωση των αρχείων catkin στον χώρο εργασίας.

Χώρος Ολοκλήρωσης Ανάπτυξης (Build Space)

Ο χώρος ολοκλήρωσης ανάπτυξης είναι ο φάκελος όπου καλείται το λογισμικό Cmake για την ανάπτυξη πακέτων στον χώρο πηγής. Ο χώρος ανάπτυξης δεν είναι απαραίτητο να περιέχεται μέσα στον χώρο εργασίας αλλά συνιστάται.

Χώρος Εξέλιξης Ανάπτυξης (Development or Devel Space)

[76] URL: http://wiki.ros.org/catkin/workspaces#Catkin_Workspaces

Ο χώρος εξέλιξης ανάπτυξης είναι το μέρος όπου τοποθετούνται τα αρχεία που έχουν αναπτυχθεί και είναι έτοιμα για εγκατάσταση. Η διάταξη με την οποία οργανώνονται τα αρχεία στο devel space είναι ίδια με εκείνη της εγκατάσταση τους στον χώρο εργασίας. Αυτός ο τρόπος παρέχει ένα χρήσιμο τεστ για την προσομοίωση της εγκατάστασης.

Χώρος Εγκατάστασης (Install Space)

Στον χώρο εγκατάστασης τα αρχεία που έχουν υποστεί ανάπτυξη μπορούν να εγκατασταθούν. Ο χώρος εγκατάστασης δεν είναι απαραίτητο να βρίσκεται μέσα στον χώρο εργασίας.

Για την δημιουργία ενός catkin workspace πληκτρολογούνται οι παρακάτω εντολές:

```
mkdir -p ~/catkin_ws/src

cd ~/catkin_ws/

catkin_make
```

Η εντολή catkin_make είναι ένα βολικό εργαλείο για να δουλεύει κανείς σε ένα catkin workspace. Την πρώτη φορά που θα εκτελεστεί η εντολή θα δημιουργήσει ένα αρχείο CmakeLists.txt στον φάκελο src (source space).

Επιπλέον στον φάκελο του χώρου εργασίας με το όνομα catkin_ws έχουν δημιουργηθεί οι φάκελοι build και devel. Στον φάκελο build επικαλείται το Cmake και ο φάκελος devel περιέχει τα παραγόμενα αρχεία που έχουν δημιουργηθεί καθώς και αρχεία τύπου setup.*sh. Εντοπίζοντας

κάποιο από αυτά τα αρχεία setup έχει ως αποτέλεσμα την επικάλυψη του workspace στο περιβάλλον εργασίας, δηλαδή ο χώρος εργασίας catkin_ws θα είναι ο κύριος χώρος εργασίας[77].

Η συγκεκριμένη διαδικασία πραγματοποιείται πληκτρολογώντας την εντολή:

source devel/setup.bash

Για να σιγουρευτεί κανείς πως ο χώρος εργασίας επικαλύφτηκε πρέπει να σιγουρευτεί πως περιέχεται στην μεταβλητή περιβάλλοντος ROS_PACKAGE_PATH.

echo $ROS_PACKAGE_PATH

Εμφανίζει τα παρακάτω μονοπάτια:

/home/youruser/catkin_ws/src:/opt/ros/kinetic/share

Η άνω κάτω τελεία δηλώνει τον διαχωρισμό των δυο μονοπατιών. Σε περίπτωση που δεν βρει την μεταβλητή στο πρώτο κομμάτι να την αναζητήσει στο δεύτερο.

4.8. Επίπεδο Αρχείων (Filesystem Level)

Τα αρχεία του ROS οργανώνονται παρόμοια με εκείνα ενός λειτουργικού συστήματος. Παρατίθενται παρακάτω οι κύριες έννοιες του επιπέδου αρχείων:

Πακέτα (Packages): Το λογισμικό του ROS οργανώνεται σε πακέτα. Ένα πακέτο μπορεί να περιέχει κόμβους, βιβλιοθήκες, αρχεία ρύθμισης παραμέτρων, λογισμικό από εξωτερικό φορέα ή οτιδήποτε το οποίο απαρτίζει μια μονάδα λογισμικού. Η φιλοσοφία των πακέτων είναι η παροχή

[77] URL: http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

λειτουργικότητας με τέτοιο τρόπο ώστε οι δυνατότητές τους να χρησιμοποιούνται συχνά. Το προφίλ τους είναι παρόμοιο με την αρχή της "Χρυσομαλλούσας" (Goldilocks priciple)[78], όπου ένα πακέτο περιέχει αρκετές ιδιότητες ώστε να είναι χρήσιμο, αλλά όχι πάρα πολλές ώστε να είναι βαρύ και δύσκολο να χρησιμοποιηθεί από το λογισμικό. Τα πακέτα είναι η πιο βασική μονάδα ανάπτυξης και έκδοσης του ROS[79].

Μεταπακέτα (Metapackages): Τα μεταπακέτα είναι ειδικά πακέτα τα οποία σε αντίθεση με τα κανονικά πακέτα δεν περιέχουν εγκατεστημένους φακέλους, κώδικα ή άλλα αντικείμενα. Ένα μεταπακέτο είναι ένα σημείο αναφοράς ενός ή περισσότερων πακέτων τα οποία συμπεριφέρονται ως ένα ενιαίο πακέτο για την εκτέλεση μιας συγκεκριμένης λειτουργίας[80].

Μανιφέστο Πακέτου (Package Manifest): Το μανιφέστο πακέτου είναι ένα αρχείο τύπου XML με το όνομα package.xml και συμπεριλαμβάνεται στον φάκελο ενός πακέτου. Το αρχείο package.xml αναγράφει τις ιδιότητες ενός πακέτου όπως το όνομα, την έκδοση, τον συγγραφέα, τον συντηρητή και τις εξαρτήσεις του. Σε περίπτωση όπου οι εξαρτήσεις ενός πακέτου λείπουν ή δεν έχουν δηλωθεί σωστά στο αρχείο package.xml έχει ως επακόλουθο την μη σωστή λειτουργία του πακέτου κατά την έκδοση του στην κοινότητα του ROS[81].

Τύποι Μηνυμάτων (Messages Types): Ο τύπος μηνυμάτων είναι μια γλώσσα περιγραφής που χρησιμοποιείται για να περιγράψει τα δεδομένα (μηνύματα) που εκδίδουν οι κόμβοι του ROS. Η περιγραφή αυτή διευκολύνει τα εργαλεία του ROS στην προσπάθεια τους να παράγουν πηγαίο

78 URL: https://judithcurry.com/2012/12/22/the-goldilocks-principle/

79 URL: http://wiki.ros.org/Packages

80 URL: http://wiki.ros.org/Metapackages

81 URL: http://wiki.ros.org/Manifest

κώδικα για ορισμένες γλώσσες προγραμματισμού όπως python και C++. Οι περιγραφές των μηνυμάτων αποθηκεύονται σε αρχεία τύπου .msg στο εκάστοτε πακέτο[82].

Τύποι Υπηρεσιών (Service Types): Οι τύποι υπηρεσιών είναι μια γλώσσα περιγραφής που χρησιμοποιεί το ROS για να περιγράψει τους τύπους υπηρεσιών. Συσχετίζεται με την δομή των τύπων μηνυμάτων και επιτρέπει την επικοινωνία τύπου αίτησης/απόκρισης (request/response) μεταξύ των κόμβων. Οι περιγραφές των υπηρεσιών αποθηκεύονται σε αρχεία τύπου srv[83].

4.8.1. Περιήγηση του Συστήματος Αρχείων

Το ROS παρέχει εντολές για την περιήγηση και τον έλεγχο των πακέτων. Οι χρησιμότερες από αυτές είναι οι εξής:

Η εντολή rospack επιτρέπει την συλλογή πληροφοριών για τα πακέτα του ROS, για την ακρίβεια εμφανίζει το μονοπάτι του συγκεκριμένου πακέτου. Η σύνταξη της εντολής είναι η εξής:

rospack find [package name]

Παράδειγμα:

rospack find roscpp

Η εντολή θα επιστρέψει το μονοπάτι το οποίο οδηγεί στο συγκεκριμένο πακέτο.

/opt/ros/kinetic/share/roscpp

[82] URL: http://wiki.ros.org/msg

[83] URL: http://wiki.ros.org/srv

Η εντολή roscd επιτρέπει την μετακίνηση ανάμεσα στους φακέλους των πακέτων και μεταπακέτων του ROS. Η σύνταξη της εντολής είναι η εξής:

```
roscd [locationname[/subdir]]
```

Παράδειγμα:

```
roscd roscpp
```

Με την εντολή pwd μπορεί κανείς να εκτυπώσει στο τερματικό το μονοπάτι του συγκεκριμένου φακέλου. Η σύνταξη της εντολής είναι η εξής:

```
pwd
```

Θα εμφανίσει το παρακάτω μονοπάτι.

```
/opt/ros/kinetic/share/roscpp
```

Το roscd μπορεί να μετακινηθεί και σε έναν υποφάκελο ενός πακέτου ή ενός μεταπακέτου.

```
roscd roscpp/cmake
```

```
pwd
```

Θα εμφανίσει το παρακάτω μονοπάτι.

```
/opt/ros/kinetic/share/roscpp/cmake
```

Για την αναφορά της επόμενης εντολής χρειάζεται η εγκατάσταση του πακέτου ros_tutorials. Στο τερματικό πληκτρολογείται η ακόλουθη εντολή:

```
sudo apt-get install ros-kinetic-ros-tutorials
```

Η εντολή rosls εμφανίζει τα περιεχόμενα ενός πακέτου. Η σύνταξη της εντολής είναι εξής:

rosls [locationname[/subdir]]

Παράδειγμα:

rosls roscpp_tutorials

Θα εμφανίσει τα περιεχόμενα του πακέτου.

cmake launch package.xml srv

Κάποιες φορές είναι κουραστικό και χρονοβόρο να γράφει κανείς ολόκληρο το όνομα ενός πακέτων, πόσο μάλλον όταν επαναλαμβάνεται συνεχώς η διαδικασία. Το ROS δίνει την δυνατότητα της αυτόματης συμπλήρωσης μιας εντολής με την χρήση του πλήκτρου Tab[84].

Παράδειγμα:

roscd roscpp_tu (πάτημα του πλήκτρου Tab)

Μετά το πάτημα του πλήκτρου Tab η γραμμή εντολών θα έχει αυτή την μορφή:

roscd roscpp_tutorials/

4.8.2. Δημιουργία ενός Πακέτου ROS (Creation of a ROS Package)

Τα πακέτα του ROS ακολουθούν μια κοινή δομή. Οι φάκελοι και τα αρχεία που θα συναντήσει κανείς αλλά και δημιουργήσει στην πορεία είναι τα εξής:

[84] URL: http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem

Φάκελος include: Περιέχει επικεφαλίδες και βιβλιοθήκες τα οποία είναι απαραίτητα σε ένα πακέτο.

Φάκελος msg: Περιέχει τα αρχεία των τύπων μηνυμάτων.

Φάκελος srv: Περιέχει τα αρχεία των τύπων υπηρεσιών.

Φάκελος scripts: Περιέχει κείμενα γραμμένα σε γλώσσα Python

Φάκελος src: Περιέχει αρχεία πηγαίου κώδικα γραμμένα σε γλώσσα C++.

Φάκελος launch: Περιέχει εκτελέσιμα αρχεία.

Αρχείο package.xml: Είναι το μανιφέστο ενός πακέτου. Η ετικέτα <build_depend></build_depend> περιέχει τα πακέτα που είναι απαραίτητα για την ανάπτυξη του πακέτου. Η ετικέτα <run_depend></run_depend> περιέχει τα πακέτα τα οποία είναι απαραίτητα για την εκτέλεση ρων κόμβων ενός πακέτου.

Αρχείο CmakeLists.txt: Αποτελεί την είσοδο προς το σύστημα ανάπτυξης Cmake, όπου δίνει πληροφορίες για την ανάπτυξη ενός πακέτου και την τοποθεσία της εγκατάστασής του[85].

Το πρώτο βήμα για την δημιουργία ενός πακέτου είναι η μετάβαση στο χώρο εργασίας catkin που έχει δημιουργηθεί και συγκεκριμένα στο source space. Στο τερματικό πληκτρολογείται η παρακάτω εντολή:

```
cd ~/catkin_ws/src
```

[85] URL: http://wiki.ros.org/Packages

Για την δημιουργία ενός πακέτου χρησιμοποιείται η εντολή catkin_create_pkg, η οποία θα δημιουργήσει ένα πακέτο με το όνομα beginner_tutorials και θα έχει ως εξαρτήσεις το πακέτο std_msgs και τις βιβλιοθήκες rospy και roscpp. Η σύνταξη της εντολής είναι η εξής:

catkin_create_pkg [package name] [dependencies]

Οπότε στο τερματικό πληκτρολογείται η εντολή με την εξής μορφή:

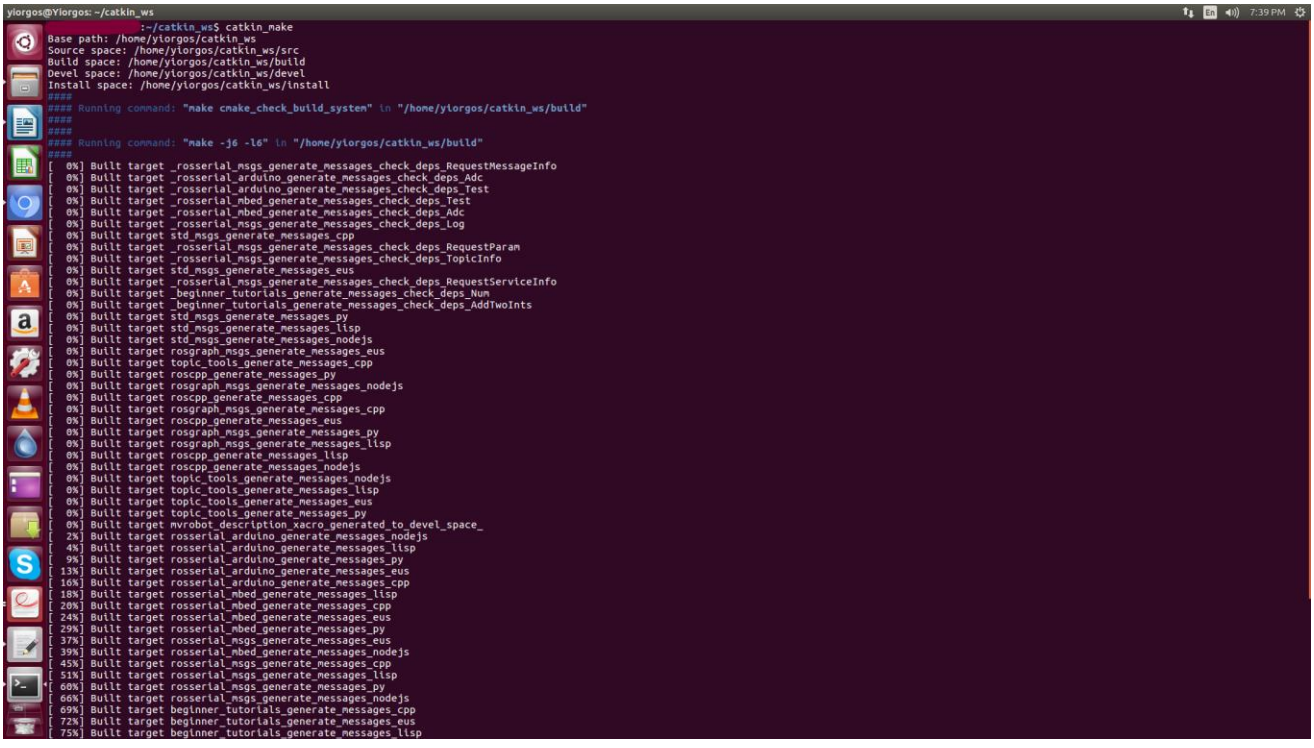
catkin_create_pkg beginner_tutorials std_msgs rospy roscpp

Με την εκτέλεση της εντολής στον χώρο src έχει δημιουργηθεί ο φάκελος του πακέτου beginner_tutorials όπου περιέχει τα αρχεία package.xml και CmakeLists.txt. Το επόμενο βήμα είναι η ανάπτυξη του πακέτου με την χρήση της εντολής catkin_make.

Σε ένα τερματικό πληκτρολογούνται οι εντολές:

cd ~/catkin_ws

catkin_make



Εικόνα 17. Μεταγλώττιση του περιβάλλοντος εργασίας

Όπως έχει αναφερθεί παραπάνω δεν είναι δυνατή η ανάπτυξη ενός πακέτου στο source space οπότε είναι απαραίτητη η μετακίνηση στον φάκελο του χώρου εργασίας και έπειτα η έναρξη της διαδικασίας. Εφόσον η ανάπτυξη ήταν επιτυχημένη μπορεί να διακρίνει κανείς στους χώρους build και devel την δημιουργία φακέλων με το ίδιο όνομα όπως το πακέτο που δημιουργήθηκε. Μετά την δημιουργία του πακέτου ο χώρος εργασίας (workspace) έχει αλλάξει, για την προσθήκη του νέου πια χώρου εργασίας στο περιβάλλον του ROS πληκτρολογείται η εντολή[86]:

```
source ~/catkin_ws/devel/setup.bash
```

Μετά από κάθε αλλαγή που έχει υποστεί ο χώρος εργασίας είναι απαραίτητη η επίκληση της παραπάνω εντολής. Για την εμφάνιση των εξαρτήσεων ενός πακέτου πληκτρολογείται σε τερματικό η παρακάτω εντολή:

```
rospack depends1 beginner_tutorials
```

Θα εμφανιστούν οι εξαρτήσεις που προστέθηκαν κατά την δημιουργία του πακέτου:

roscpp

rospy

std_msgs

4.8.3. Δημιουργία ενός Μηνύματος (Creation of a Message)

[86] URL: http://wiki.ros.org/ROS/Tutorials/CreatingPackage

Ο τύπος μηνυμάτων χρησιμοποιεί αρχεία .txt τα οποία αποτελούνται από τον τύπο δεδομένων και το όνομα του μηνύματος. Χρησιμοποιούνται για την παραγωγή πηγαίου κώδικα μηνυμάτων σε διαφορετικές γλώσσες. Τα αρχεία αποθηκεύονται στον φάκελο μηνυμάτων ενός πακέτου. Παρακάτω δίνεται η τυπική μορφή ενός μηνύματος:

int64 A

float32 B

Οι τύποι δεδομένων που χρησιμοποιεί το ROS είναι οι εξής:

Primitive Type	**Serialization**	**C++**	**Python2**	**Python3**
bool (1)	unsigned 8-bit int	uint8_t (2)	bool	
int8	signed 8-bit int	int8_t	int	
uint8	unsigned 8-bit int	uint8_t	int (3)	
int16	signed 16-bit int	int16_t	int	
uint16	unsigned 16-bit int	uint16_t	int	
int32	signed 32-bit int	int32_t	int	
uint32	unsigned 32-bit int	uint32_t	int	
int64	signed 64-bit int	int64_t	long	int
uint64	unsigned 64-bit int	uint64_t	long	int
float32	32-bit IEEE float	float	float	
float64	64-bit IEEE float	double	float	
string	ascii string (4)	std::string	str	bytes
time	secs/nsecs unsigned 32-bit ints	ros::Time	rospy.Time	
duration	secs/nsecs signed 32-bit ints	ros::Duration	rospy.Duration	

Ένας επιπλέον τύπος δεδομένων που υπάρχει στο ROS είναι ο Header (επικεφαλίδα), ο οποίος περιέχει πληροφορίες για την ώρα αποστολής, όνομα και τον αριθμό του μηνύματος κ.α. Η τυπική μορφή ενός header είναι η εξής:

Header header

string child_frame_id

geometry_msgs/PoseWithCovariance pose

geometry_msgs/TwistWithCovariance twist

Για την δημιουργία ενός μηνύματος πρέπει να μεταφερθεί κανείς στον φάκελο του πακέτου beginner_tutorials που δημιουργήθηκε στην προηγούμενη ενότητα, να δημιουργήσει ένα φάκελο με το όνομα “msg” και έπειτα να δημιουργήσει ένα αρχείο με το όνομα Num. Για την εκτέλεση της διαδικασίας πληκτρολογούνται σε έναν τερματικό οι ακόλουθες εντολές:

roscd beginner_tutorials

mkdir msg

echo “int64 num” > msg/Num.msg

Εφόσον η δημιουργία του φακέλου και του αρχείου μηνυμάτων είναι επιτυχής, το επόμενο βήμα είναι η τροποποίηση των αρχείων CmakeLists.txt και package.xml. Στο package.xml προσθέτονται δύο εξαρτήσεις απαραίτητες για την ανάπτυξη και την εκτέλεση μηνυμάτων:

<build_depend>message_generation</build_depend>

<exec_depend>message_runtime</exec_depend>

Εικόνα 20. Αρχείο package.xml

Στο αρχείο CmakeLists.txt διαμορφώνεται η συνάρτηση find_package ώστε να είναι δυνατή η παραγωγή μηνυμάτων από το συγκεκριμένο πακέτο. Στο ήδη υπάρχον κείμενο προστίθεται η εξάρτηση message_generation, η μορφή της συνάρτησης είναι η ακόλουθη:

find_package(catkin REQUIRED COMPONENTS

roscpp

rospy

std_msgs

message_generation

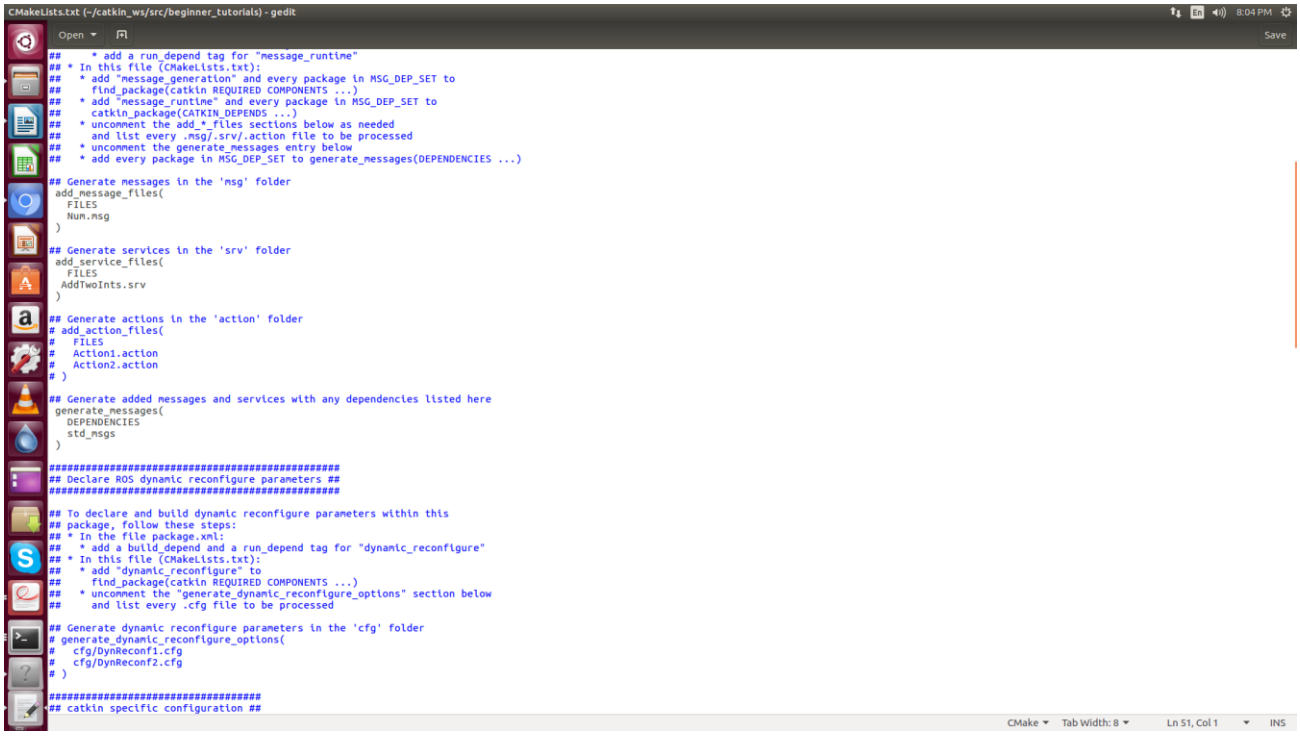
)

Για την εξαγωγή της εξάρτησης η συνάρτηση catkin_package πρέπει να έχει την παρακάτω μορφή:

```
catkin_package(
#  INCLUDE_DIRS include
#  LIBRARIES beginner_tutorials
   CATKIN_DEPENDS roscpp rospy std_msgs message_runtime
#  DEPENDS system_lib
)
```

Το επόμενο βήμα είναι η χειροκίνητη προσθήκη του αρχείου Num.msg στην συνάρτηση add_message_files καθώς και του πακέτου std_msgs στην συνάρτηση generate_messages. Το πακέτο std_msgs περιέχει τους τύπους μηνυμάτων του ROS. Η μορφή των συναρτήσεων πρέπει να είναι η ακόλουθη:

```
 add_message_files(
   FILES
   Num.msg
generate_messages(
   DEPENDENCIES
```



std_msgs

Εικόνα 21. Αρχείο CmakeLists.txt

Μετά την ολοκλήρωση της διαδικασίας είναι απαραίτητη η ανάπτυξη του περιβάλλοντος εργασίας.

```
cd ~/catkin_ws
```

```
catkin_make
```

Για την εμφάνιση του τύπου ενός αρχείου μηνυμάτων χρησιμοποιείται η εντολή rosmsg show. Η μορφή της είναι η εξής:

```
rosmsg show [package_name]/[file_name]
```

Στο τερματικό πληκτρολογείται η εντολή με την ακόλουθη μορφή:

rosmsg show beginner_tutorials/Num

Θα εμφανίσει:

int 64 num

Στην περίπτωση που γνωρίζει κάποιος το όνομα του αρχείου αλλά δεν γνωρίζει το όνομα του πακέτου στο οποίο περιέχεται πληκτρολογείται στο τερματικό η ακόλουθη εντολή:

rosmsg show Num

Θα εμφανίσει:

[beginner_tutorials/Num]

int 64 num

Παρακάτω γίνεται αναφορά σε κάποιες παραλλαγές της εντολής rosmsg:

rosmsg list: Εμφανίζει μια λίστα με όλα τα μηνύματα του ROS.

rosmsg package [package_name]: Εμφανίζει τα μηνύματα σε ένα πακέτο.

rosmsg packages: Εμφανίζει πακέτα τα οποία περιέχουν μηνύματα.

4.8.3. Δημιουργία μιας Υπηρεσίας (Creation of a Service)

Οι υπηρεσίες του ROS είναι μια επικοινωνία τύπου αίτησης/απάντησης μεταξύ κόμβων. Ένας κόμβος στέλνει μια αίτηση περιμένοντας μια απάντηση από τον άλλον κόμβο. Στο φάκελο του πακέτου beginner_tutorials δημιουργείται ένας φάκελος με το όνομα srv.

roscd beginner_tutorials

mkdir srv

Στην προκειμένη περίπτωση ο χρήστης δεν δημιουργεί έναν νέο φάκελο υπηρεσιών αλλά αντιγράφει έναν υπάρχων φάκελο από ένα άλλο πακέτο. Η εντολή roscp αντιγράφει αρχεία από ένα πακέτο σε κάποιο άλλο.

Η μορφή της εντολής είναι η εξής:

roscp [package_name] [file_to_copy] [copy_path]

Στο τερματικό πληκτρολογείται η παρακάτω εντολή:

roscp rospy_tutorials AddTwoInts.srv srv/AddTwoInts.srv

Για την ενεργοποίηση του αρχείου υπηρεσιών που μόλις δημιουργήθηκε στο αρχείο CmakeLists.txt προσθέτουμε τον όνομα του αρχείου υπηρεσιών.

```
add_service_files(
   FILES
   AddTwoInts.srv
)
```

Ομοίως με τον τύπο μηνυμάτων μπορεί κανείς να εμφανίσει τον τύπο δεδομένων ενός αρχείου υπηρεσιών με την εντολή rossrv.

rossrv show beginner_tutorials/AddTwoInts

Θα εμφανίσει:

int64 a

int64 b

- - -

int64 sum

Το πρώτο μέρος του αρχείου δηλώνει τον τύπο και τον αριθμό των μεταβλητών που δέχεται το πακέτο σαν αίτηση και το δεύτερο τον τύπο και τον αριθμό των μεταβλητών που στέλνει ως απάντηση σε κάποιο αίτημα[87]

4.9. Επίπεδο Υπολογισμών (Computation Level)

Το επίπεδο υπολογισμών είναι υπεύθυνο για την υλοποίηση διαδικασιών μέσα από ένα δίκτυο οντοτήτων τα οποία ονομάζονται κόμβοι του ROS. Παρατίθενται παρακάτω οι κύριες έννοιες του επιπέδου υπολογισμών:

Κόμβοι (Nodes): Ένας κόμβος είναι μια διαδικασία η οποία εκτελεί υπολογισμούς. Σε ένα ρομποτικό σύστημα υπάρχουν πολλοί κόμβοι όπου ο καθένας τους, εκτελεί διαφορετικά καθήκοντα και χρησιμοποιώντας τις μεθόδους επικοινωνίας του ROS, έχουν την δυνατότητα να επικοινωνούν και να ανταλλάσουν δεδομένα μεταξύ τους. Ένας από τους στόχους κατά την ανάπτυξη ενός κόμβου είναι η δημιουργία μιας απλής διαδικασίας για βέλτιστη λειτουργία και εύκολη αποσφαλμάτωση

[87] URL: http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv

Κυρίαρχος Κόμβος (Master): Ο κυρίαρχος κόμβος είναι υπεύθυνος για την ονοματοθεσία και επίβλεψη των υπόλοιπων κόμβων. Οι κόμβοι δεν μπορούν να επικοινωνήσουν ο ένας με τον άλλον, να ανταλλάξουν μηνύματα ή να στείλουν αιτήματα υπηρεσιών χωρίς τον κυρίαρχο κόμβο. Σε ένα κατανεμημένο σύστημα ο κυρίαρχος κόμβος πρέπει να τρέχει σε ένα τερματικό και οι υπόλοιποι κόμβοι θα μπορούν να επικοινωνούν μεταξύ τους[88].

Διακομιστής Παραμέτρων (Parameter Server): Ο διακομιστής παραμέτρων είναι ένας χώρος προσβάσιμος μέσω της διεπαφής προγραμματισμού εφαρμογών του ROS. Οι κόμβοι χρησιμοποιούν τον συγκεκριμένο διακομιστή για αποθήκευση και λήψη παραμέτρων κατά την εκτέλεση τους. Αποτελεί κομμάτι του κυρίαρχου κόμβου (master)[89].

Μηνύματα (Messages): Τα μηνύματα είναι οι πληροφορίες τις οποίες οι κόμβοι του ROS μεταδίδουν ο ένας προς τον άλλο. Το ROS υποστηρίζει τους βασικούς τύπους δεδομένων αλλά μπορεί κανείς να δημιουργήσει τον δικό του.[90]

Θέματα (Topics): Τα θέματα είναι διάδρομοι οι οποίοι χρησιμοποιούνται από τους κόμβους για την αποστολή μηνυμάτων. Ένας κόμβος εκδίδει (publish) ένα θέμα όταν στέλνει ένα μήνυμα μέσω ενός θέματος και εγγράφεται (subscribe) σε ένα θέμα όταν λαμβάνει μήνυμα. Ο κόμβος που εκδίδει και ο κόμβος που λαμβάνει δεν γνωρίζουν ο ένας την παρουσία του άλλου οπότε για να ανταλλάξουν πληροφορίες μεταξύ τους πρέπει να εγγραφούν σε ένα θέμα. Κάθε θέμα έχει μοναδικό όνομα.

[88] URL: http://wiki.ros.org/Master

[89] URL: http://wiki.ros.org/Parameter%20Server

[90] URL: http://wiki.ros.org/Messages

Οποιοσδήποτε κόμβος μπορεί να στείλει ή να λάβει μηνύματα μέσω ενός θέματος αρκεί ο τύπος δεδομένων των μηνυμάτων και του θέματος να είναι ίδιοι[91]. Υπηρεσίες (Services): Σε μερικές εφαρμογές ένα μοντέλο επικοινωνίας έκδοσης/συνδρομής (publish/subscribe) μπορεί να μην είναι αρκετά ευέλικτο και να χρειάζεται μια αλληλεπίδραση αίτησης/ανταπόκρισης (request/response). Το μοντέλο αίτησης/απάντησης γίνεται μέσω μιας υπηρεσίας η οποία χωρίζεται σε δύο μέρη, ένα για την αίτηση και ένα για την απάντηση. Ένας κόμβος εξυπηρετητής παρέχει μια υπηρεσία υπό ένα όνομα και ένας κόμβος πελάτης καλεί την υπηρεσία στέλνοντας μια αίτηση και περιμένοντας μια απάντηση. Η αλληλεπίδραση μιας υπηρεσίας μοιάζει με εκείνη της συνάρτησης[92].

Bags: Τα bags είναι ένα είδος αρχείου το οποίο αποθηκεύει και επιστρέφει δεδομένα. Είναι ιδανικός αποθηκευτικός χώρος για την αποθήκευση δεδομένων αισθητήρων τα οποία είναι απαραίτητη για την ανάπτυξη και δοκιμή ρομποτικών εφαρμογών.[93]

4.9.1. Περαιτέρω Κατανόηση των Κόμβων και του Κυρίαρχου Κόμβου (Further Understanding of ROS Nodes and Master)

Ένας κόμβος είναι ένα πρόγραμμα μέσα σε ένα πακέτο το οποίο χρησιμοποιεί τις βιβλιοθήκες του ROS όπως η roscpp και η rospy για να εκτελεί υπολογισμούς. Έχει την δυνατότητα να επικοινωνεί με άλλους κόμβους χρησιμοποιώντας τα θέματα, τις υπηρεσίες και τον διακομιστή παραμέτρων του ROS.

[91] URL: http://wiki.ros.org/Topics

[92] URL: http://wiki.ros.org/Services

[93] URL: http://wiki.ros.org/Bags

Ένα ρομποτικό σύστημα μπορεί να περιέχει πολλούς κόμβους οι οποίοι να εκτελούν διαφορετικές εργασίες, για παράδειγμα, ένας κόμβος να επεξεργάζεται λήψεις μιας κάμερας, ένας άλλος κόμβος να επεξεργάζεται δεδομένα που έχουν συλλεχθεί μέσω αισθητήρων, ένας άλλος κόμβος να υπολογίζει την απόσταση που έχει διανύσει το ρομπότ μέσω οδομετρίας και πολλά άλλα.

Η χρήση κόμβων αυξάνει την ανοχή ενός συστήματος σε τυχόν σφάλματα που μπορεί να προκύψουν, διότι στην περίπτωση που ένας κόμβος δεν αποδίδει το επιθυμητό αποτέλεσμα ή προκύψει κάποιο σφάλμα στην λειτουργία του, οι υπόλοιποι κόμβοι θα συνεχίσουν να δουλεύουν κανονικά. Αυτή η δυνατότητα επιτρέπει την εύκολη αποσφαλμάτωση και την μέγιστη λειτουργία σε ρομποτική εφαρμογή.

Ο κυρίαρχος κόμβος ή master μοιάζει με έναν διακομιστή DNS. Ένας κόμβος καθώς ενεργοποιείται ψάχνει τον κυρίαρχο κόμβο ώστε να καταχωρίσει σε εκείνον το όνομά του. Με αυτό τον τρόπο ο κυρίαρχος κόμβος είναι ενήμερος για όλους τους κόμβους που είναι ενεργοί στο σύστημα του ROS. Σε περίπτωση που τα στοιχεία οποιουδήποτε κόμβου αλλάξουν ο κυρίαρχος κόμβος ενημερώνεται άμεσα.

Όταν ένας κόμβος ξεκινά την έκδοση μηνυμάτων σε ένα θέμα παράλληλα ενημερώνει τον κυρίαρχο κόμβο για το όνομα και τον τύπο του θέματος. Στην περίπτωση όπου δύο κόμβοι είναι εγγεγραμμένοι στο ίδιο θέμα και θέλουν να ανταλλάξουν πληροφορίες, ο κυρίαρχος κόμβος στέλνει πληροφορίες που αφορούν τον εξυπηρετητή προς τον παραλήπτη. Εφόσον ο παραλήπτης λάβει αυτές τις πληροφορίες οι δύο κόμβοι επικοινωνούν μεταξύ τους με την χρήση του πρωτοκόλλου TCPROS, το οποίο είναι βασισμένο στο πρωτόκολλο επικοινωνίας TCP/IP.

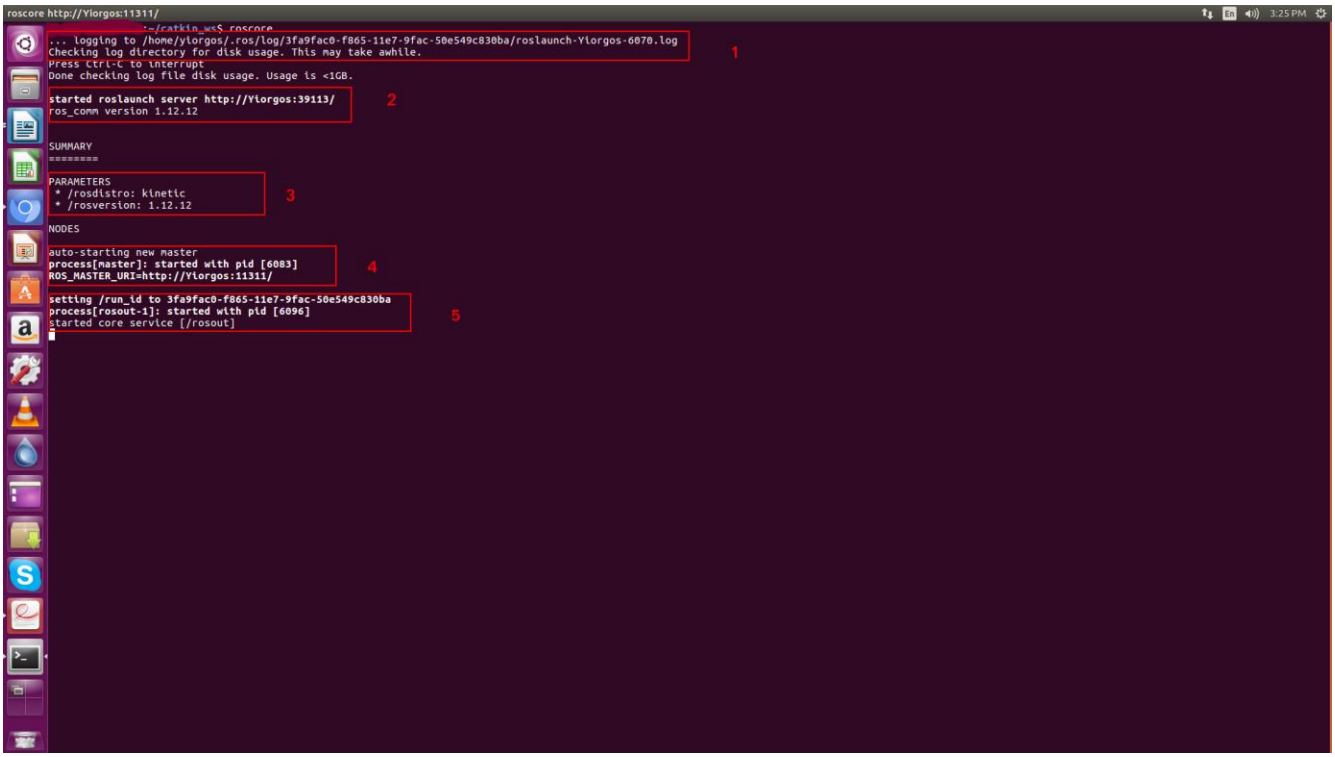
Όταν η σύνδεση τους ολοκληρωθεί ο κυρίαρχος κόμβος θα ειδοποιηθεί, το ίδιο ισχύει και για την επικοινωνία αίτησης/απάντησης.

Πριν την έναρξη οποιουδήποτε κόμβου είναι απαραίτητη η ενεργοποίηση του κυρίαρχου κόμβου και του διακομιστή παραμέτρων με την εντολή roscore. Η εντολή ενεργοποιεί επίσης και τον κόμβο rosout. Ο rosout είναι ένας κόμβος ο οποίος συλλέγει σε αρχεία καταγραφής (log files) καταγεγραμμένα μηνύματα τα οποία εκδίδουν οι υπόλοιποι κόμβοι μέσω του θέματος /rosout και αν χρειαστεί τα επιστρέφει μέσω του θέματος /rosout_agg[94].

Πληκτρολογείται στο τερματικό η εντολή:

roscore

[94] Lentin J. (2015). *Mastering ROS for Robotics Programming*. Packt Publishing Ltd. p.19 – 22.



Εικόνα 18. Roscore. (Πηγή από βιβλίο Mastering ROS for Robotics Programming)

Μετά την εκτέλεση της εντολής στο τερματικό έχει την μορφή της παραπάνω εικόνας.

Ακολουθεί εξήγηση κάθε τμήματος του συγκεκριμένου τερματικού.

1. Στο πρώτο κομμάτι δημιουργείται ένα αρχείο καταχώρησης (log file) μέσα στον φάκελο ~/.ros/log για την συλλογή καταχωρημένων μηνυμάτων από τους κόμβους του ROS.

2. Στο δεύτερο κομμάτι εκτελείται ένα αρχείο τύπου launch με όνομα roscore.xml και δείχνει την διεύθυνση του διακομιστή παραμέτρων μέσα στην θύρα. Κατά την εκτέλεση ενός launch γίνεται αυτόματα η εκκίνηση του κυρίαρχου κόμβου και του διακομιστή παραμέτρων. Η έναρξη λειτουργίας αυτών των δύο είναι εφικτή μέσω της εντολής roslaunch.

3. Στο τρίτο κομμάτι διακρίνονται οι παράμετροι rosdistro και rosversion.

4. Στο τέταρτο κομμάτι ο κόμβος rosmaster εκτελεί την μεταβλητή περιβάλλοντος ROS_MASTER_URI.

5. Στο πέμπτο κομμάτι ενεργοποιείται ο κόμβος rosout και ξεκινά τα θέματα /rosout και /rosout_agg.

6. Η εντολή rosnode προβάλει πληροφορίες για τους ενεργούς κόμβους του ROS, συγκεκριμένα η εντολή rosnode list εμφανίζει τους ενεργούς κόμβους. Ανοίγεται ένα νέο τερματικό και πληκτρολογείται η εντολή:

```
rosnode list
```

Στην προκειμένη περίπτωση θα εμφανίσει:

```
/rosout
```

Με την εντολή rosnode info μπορεί κανείς να λάβει πληροφορίες για οποιονδήποτε ενεργό κόμβο. Στην προκειμένη περίπτωση η σύνταξη της εντολής είναι η εξής:

```
rosnode info /rosout
```

Θα εμφανίσει τα θέματα στα οποία εκδίδει μηνύματα, τα θέματα από τα οποία λαμβάνει μηνύματα και τις υπηρεσίες που χρησιμοποιεί:

Node [/rosout]

Publications:

* /rosout_agg [rosgraph_msgs/Log]

Subscriptions:

* /rosout [unknown type]

Services:

 * /rosout/get_loggers

 * /rosout/set_logger_level

contacting node http://User:39977/ ...

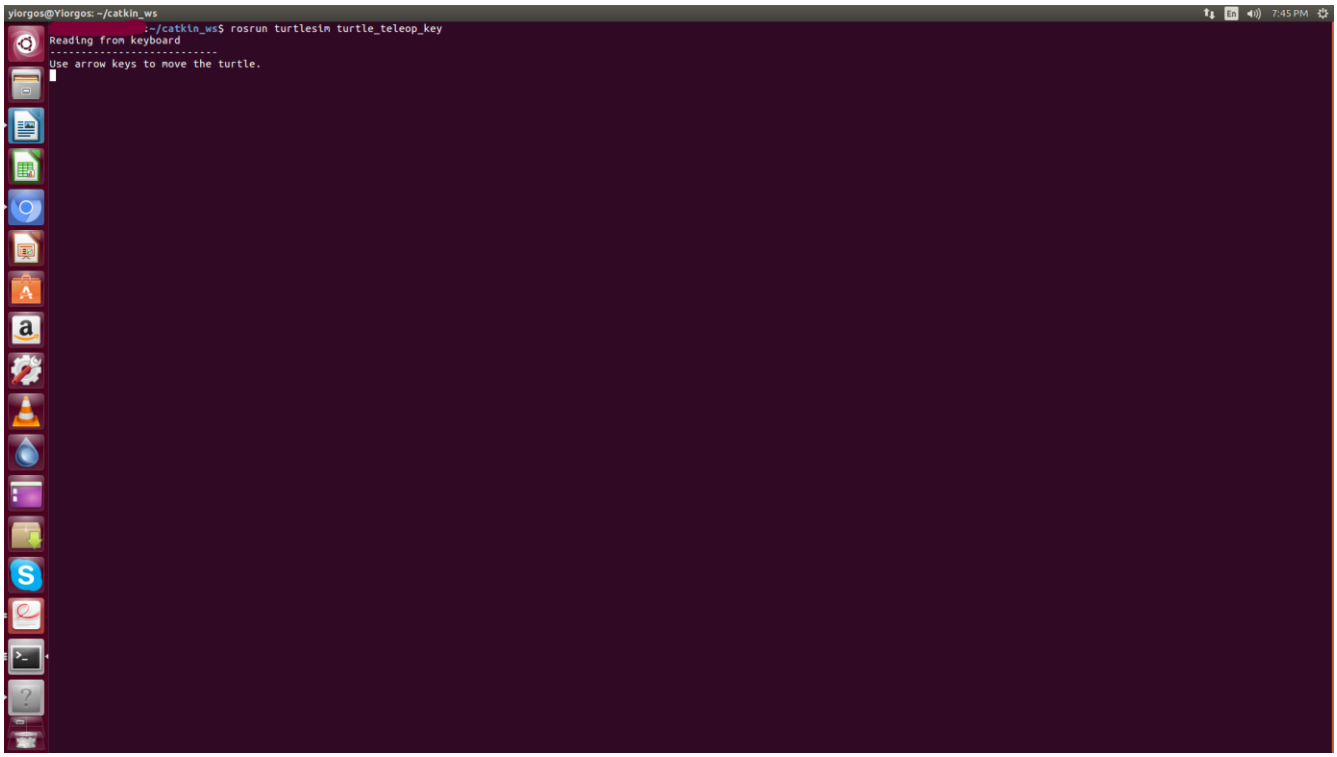
Pid: 24676

Η εντολή rosrun επιτρέπει την απευθείας εκτέλεση ενός κόμβου μέσα από ένα πακέτο, χωρίς να χρειάζεται να γνωρίζει κανείς το μονοπάτι ή την τοποθεσία του πακέτου. Η σύνταξη της εντολής είναι η εξής:

rosrun [package_name] [node_name]

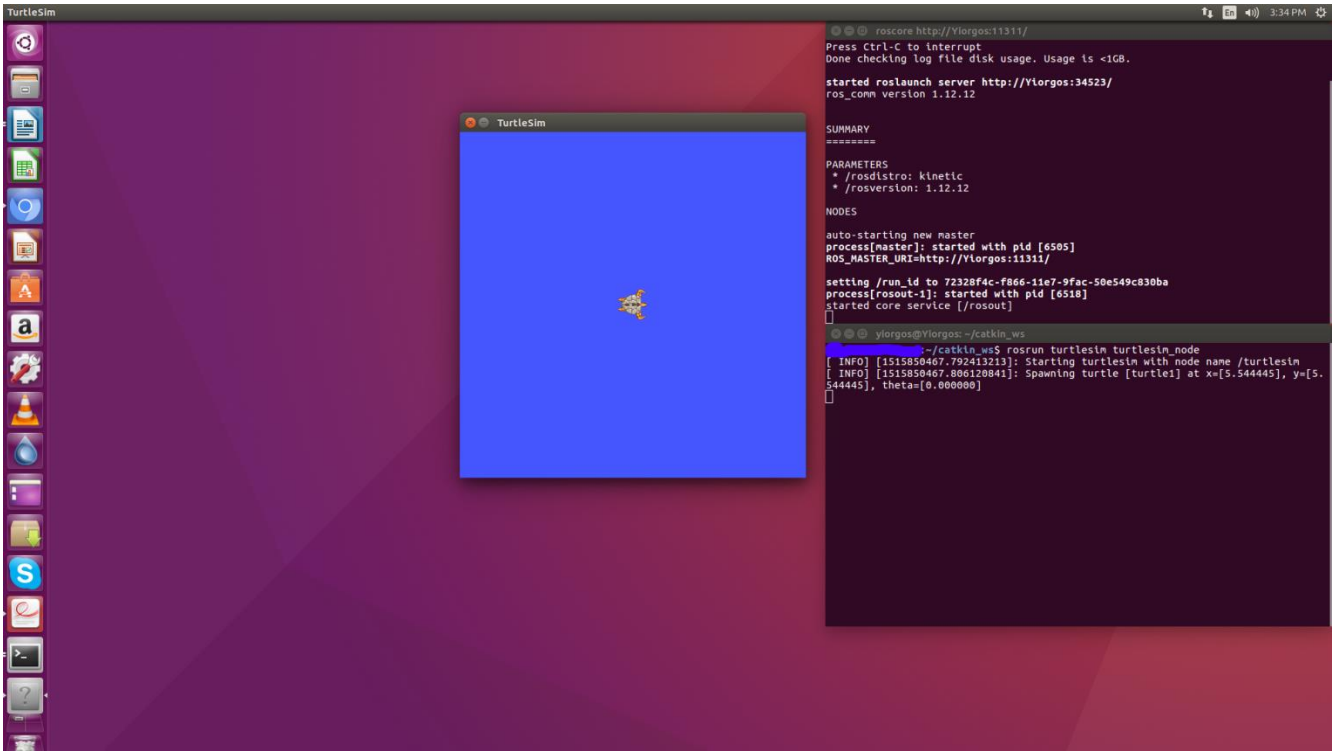
Στην συνέχεια θα επικαλεστεί ο κόμβος turtlesim_node από το πακέτο turtlesim. Πρόκειται για έναν κόμβο ο οποίος προσομοιώνει μια χελώνα που κινείται στον χώρο και έχει καθαρά εκπαιδευτικό στόχο. Στο ήδη ανοιχτό τερματικό πληκτρολογείται η εντολή:

rosrun turtlesim turtlesim_node



Εικόνα 19. Εκτέλεση του Κόμβου Turtlesim_node

Η εκτέλεση της εντολής ανοίγει το παράθυρο προσομοίωσης στο οποίο εικονίζεται μια χελώνα.



Εικόνα 20. Turtlsesim

Για να μπορέσει να δει κανείς τον νέο ενεργό κόμβο σε ένα νέο τερματικό πληκτρολογείται ξανά η εντολή rosnode list όπου θα εμφανίσει:

/rosout

/turtlesim

Με την εντολή rosnode info μπορεί να δει κανείς πληροφορίες σχετικά με τον νέο κόμβο. Στο ίδιο τερματικό πληκτρολογείται η εντολή:

rosnode info /turtlesim

Όπως και στην προηγούμενη περίπτωση εμφανίζει τα θέματα στα οποία στέλνει και από τα οποία λαμβάνει μηνύματα, τον τύπο των μηνυμάτων, τις υπηρεσίες και με ποιον κόμβο είναι συνδεδεμένος[95].

Node [/turtlesim]

Publications:

* /rosout [rosgraph_msgs/Log]

* /turtle1/color_sensor [turtlesim/Color]

* /turtle1/pose [turtlesim/Pose]

Subscriptions:

* /turtle1/cmd_vel [unknown type]

[95] URL: http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes

Services:

 * /clear

 * /kill

 * /reset

 * /spawn

 * /turtle1/set_pen

 * /turtle1/teleport_absolute

 * /turtle1/teleport_relative

 * /turtlesim/get_loggers

* /turtlesim/set_logger_level

contacting node http://User:42625/ ...

Pid: 11202

Connections:

 * topic: /rosout

    * to: /rosout

    * direction: outbound

* transport: TCPROS

4.9.2. Περαιτέρω κατανόηση των θεμάτων του ROS(Further Understanding of ROS Topics)

Τα θέματα είναι διάδρομοι οι οποίοι χρησιμοποιούνται από τους κόμβους το ROS για την ανταλλαγή μηνυμάτων. Στα θέματα, η παραγωγή και η κατανάλωση μηνυμάτων είναι διαχωρισμένη, που σημαίνει πως ένα θέμα μπορεί να στέλνει και να δέχεται μηνύματα ταυτόχρονα. Επειδή η επικοινωνία σε ένα θέμα είναι αμφίδρομη υπάρχει ο εναλλακτικός τρόπος επικοινωνίας των υπηρεσιών. Οι κόμβοι του ROS επικοινωνούν μεταξύ τους χρησιμοποιώντας το πρωτόκολλο μεταφοράς TCPROS το οποίο είναι βασισμένο στο πρωτόκολλο επικοινωνίας TCP/IP[96].

Η εντολή rostopic χρησιμοποιείται για την ανάκτηση πληροφοριών που αφορά τα θέματα του ROS. Η σύνταξη της εντολής είναι η εξής:

rostopic bw /topic: Εμφανίζει το εύρος ζώνης ενός θέματος.

rostopic echo /topic: Τυπώνει το περιεχόμενο ενός θέματος.

rostopic find /message_type: Εμφανίζει τα θέματα που χρησιμοποιούν τον συγκεκριμένο τύπο μηνυμάτων.

rostopic info /topic: Εμφανίζει πληροφορίες για ένα ενεργό θέμα.

rostopic list: Εμφανίζει όλα τα ενεργά θέματα.

[96] Lentin J. (2015). p. 17.

rostopic pub /topic message_type args: Εκδίσει σε ένα θέμα μια μεταβλητή με τον συγκεκριμένο τύπο μηνύματος

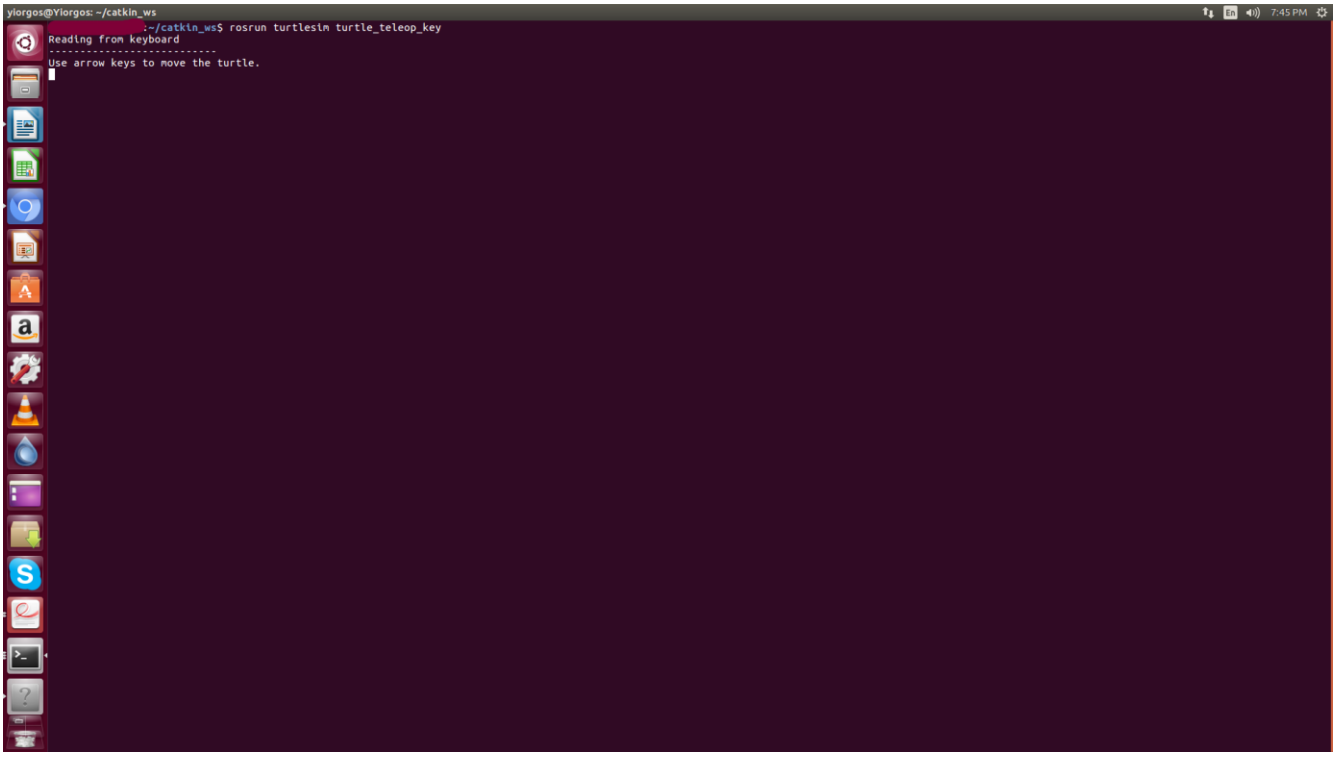
rostopic type /topic: Εμφανίζει τον τύπο μηνυμάτων ενός θέματος

Το πρώτο βήμα είναι η επαναφορά της προσομοίωσης που υπήρχε στην προηγούμενη ενότητα. Σε ένα τερματικό ενεργοποιούμε τον κυρίαρχο κόμβο και σε ένα δεύτερο τερματικό εκτελείται η εντολή για την επαναφορά του παραθύρου προσωμοίωσης. Η ακολουθία των εντολών είναι η εξής:

roscore

rosrun turtlesim turtlesim_node

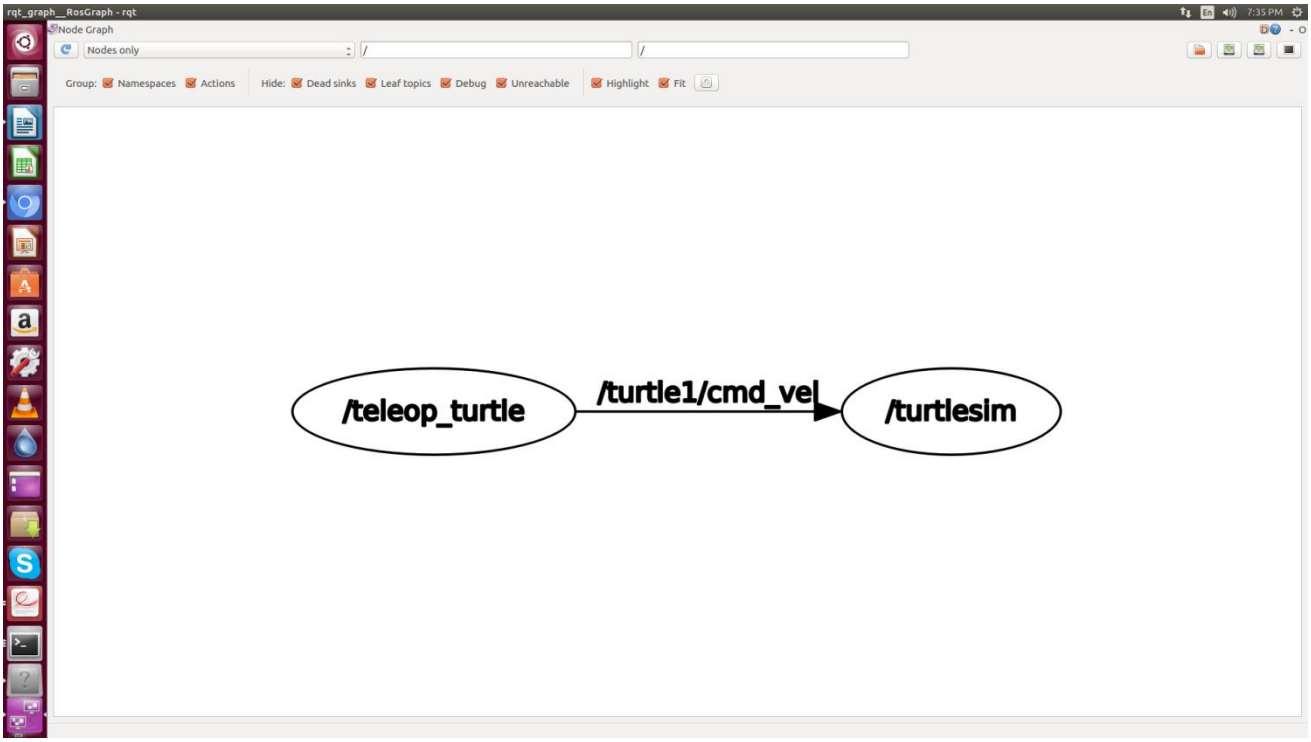
Το επόμενο βήμα είναι ο έλεγχος της χελώνας με την χρήση του κόμβου turtle_teleop_key. Πατώντας τα πλήκτρα περιήγησης με τα σύμβολα πάνω, κάτω, αριστερά και δεξία ο χρήστης έχει την δυνατότητα να ελέγξει την χελώνα αρκεί να είναι επιλεγμένο το τερματικό στο οποίο τρέχει ο κόμβος. Σε ένα νέο τερματικό και πληκτρολογείται η εντολή:



rosrun turtlesim turtle_teleop_key

Εικόνα 21. Εκτέλεση του Κόμβου turtle_teleop_key

Οι κόμβοι turtlesim_node και turtle_teleop_key επικοινωνούν μεταξύ τους με την χρήση ενός θέματος. Ο κόμβος turtle_teleop_key εκδίδει τις εντολές που δίνει ο χρήστης με την μορφή μηνυμάτων μέσω του θέματος /turtle1/cmd_vel (βλ. Εικόνα 25) και ο κόμβος turtlesim_node εγγράφεται στο συγκεκριμένο θέμα ώστε να λάβει τα μηνύματα. Με την εκτέλεση του κόμβου rqt_graph ο οποίος βρίσκεται στο πακέτο rqt_graph μπορεί κανείς μέσα από μια γραφική διεπαφή χρήστη να δει τα στοιχεία του επιπέδου υπολογισμού. Αυτό σημαίνει πως είναι δυνατή η αναπαράσταση των κόμβων και του θέματος που έχουν χρησιμοποιηθεί σε αυτή την προσομοίωση. Πληκτρολογείται σε ένα νέο τερματικό η παρακάτω εντολή:

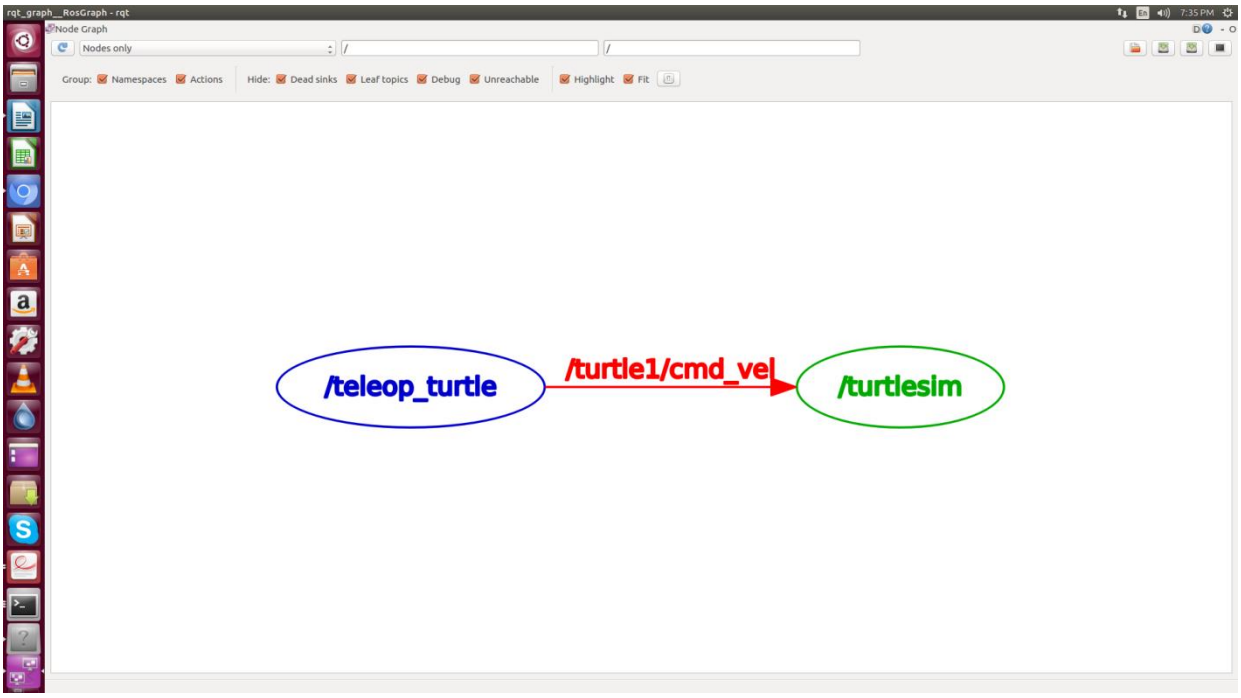


rosrun rqt_graph rqt_graph

Εικόνα 22. rqt_graph

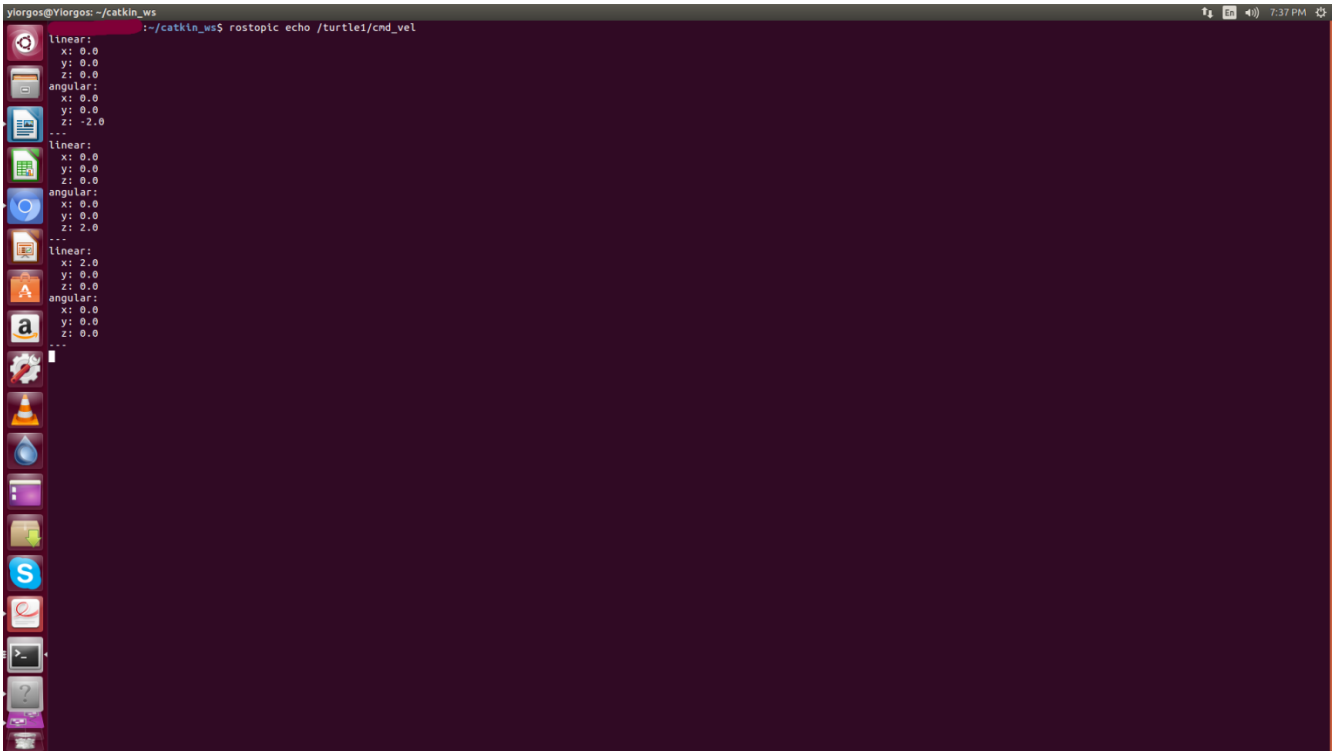
Εάν τοποθετήσει ο χρήστης τον κέρσορα στο θέμα /turtle1/command_velocity οι δύο κόμβοι θα πάρουν το χρώμα μπλε και πράσινο ενώ το θέμα θα γίνει κόκκινο. Εφόσον είναι γνωστό το όνομα του θέματος το οποίο χρησιμοποιείται για την επικοινωνία των δύο κόμβων, είναι εφικτή η εμφάνιση των δεδομένων που δημοσιεύονται στο συγκεκριμένο θέμα με την χρήση της εντολής rostopic echo. Σε ένα νέο τερματικό πληκτρολογείται η εντολή:

rostopic echo /turtle1/cmd_vel



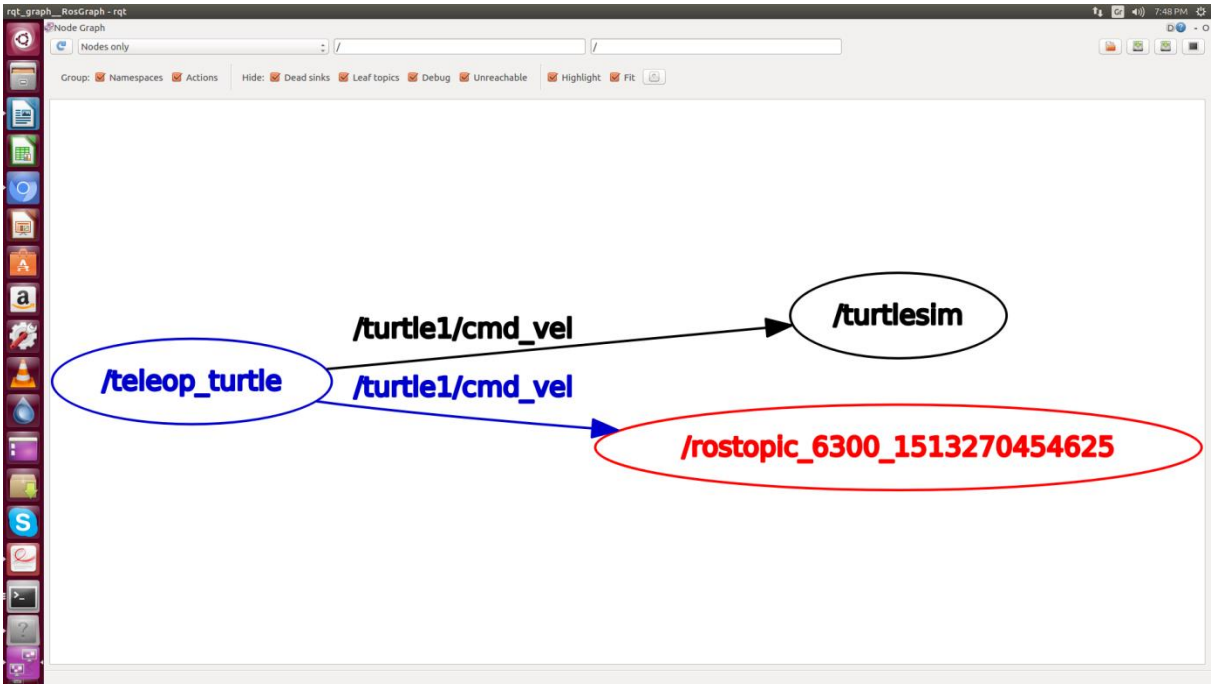
Εικόνα 23. rqt_graph

Μετά την εκτέλεση της εντολής δεν παρατηρείται κάποια αλλαγή επειδή δεν εκδίδονται δεδομένα στο θέμα. Ο χρήστης δεν έχει παρά να επιλέξει το τερματικό στο οποίο τρέχει ο κόμβος turtle_teleop_key και να πατήσει κάποιο πλήκτρο ώστε να ελέγξει την χελώνα. Μετά το πάτημα ενός πλήκτρου στο τερματικό όπου τρέχει το θέμα διακρίνονται τα δεδομένα που μεταφέρονται από τον κόμβο turtle_teleop_key στον κόμβο turtlesim_node.



Εικόνα 24. Δεδομένα του Θέματος /turtle1/cmd_vel

Στην προκειμένη περίπτωση το τερματικό στο οποίο εκτελέστηκε η εντολή έδρασε ως ένας κόμβος, ο οποίος έκανε εγγραφή στο θέμα /turtle1/cmd_vel για να έχει πρόσβαση στα δεδομένα του. Πατώντας το εικονίδιο της ανανέωσης στο παράθυρο του κόμβου rqt-graph μπορεί κανείς να διακρίνει την επικοινωνία μεταξύ του κόμβου teleop_turtle και του προσωρινού κόμβου που δημιούργησε η εντολή rostopic echo (βλ. Εικόνα 24).



Εικόνα 25. rqt_graph

Με την εντολή rostopic list εμφανίζονται όλα τα θέματα στα οποία στέλνονται και λαμβάνονται πληροφορίες εντολή rostopic type επιστρέφει τον τύπο μηνύματος που χρησιμοποιεί ένα θέμα.

Σε ένα τερματικό πληκτρολογείται η εντολή:

rostopic type /turtle1/cmd_vel

Θα εμφανίσει:

geometry_msgs/Twist

Η εντολή rosmsg show εμφανίζει περισσότερες λεπτομέρειες για τον τύπο δεδομένων του μηνύματος:

rosmsg show geometry_msgs/Twist

Θα εμφανίσει:

geometry_msgs/Vector3 linear

float64 x

float64 y

float64 z

geometry_msgs/Vector3 angular

float64 x

float64 y

float64 z

Με την εντολή rostopic pub μπορεί κανείς να εκδώσει δεδομένα σε ένα θέμα. Στην προκειμένη περίπτωση η σύνταξη της εντολής είναι η εξής:

rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'

Λόγω της πολύπλοκης φύσης της εντολής δίνεται επεξήγηση των στοιχείων της

rostopic pub → Η εντολή

-1 → Ο αριθμός δηλώνει τον αριθμό των μηνυμάτων που θα αποσταλούν

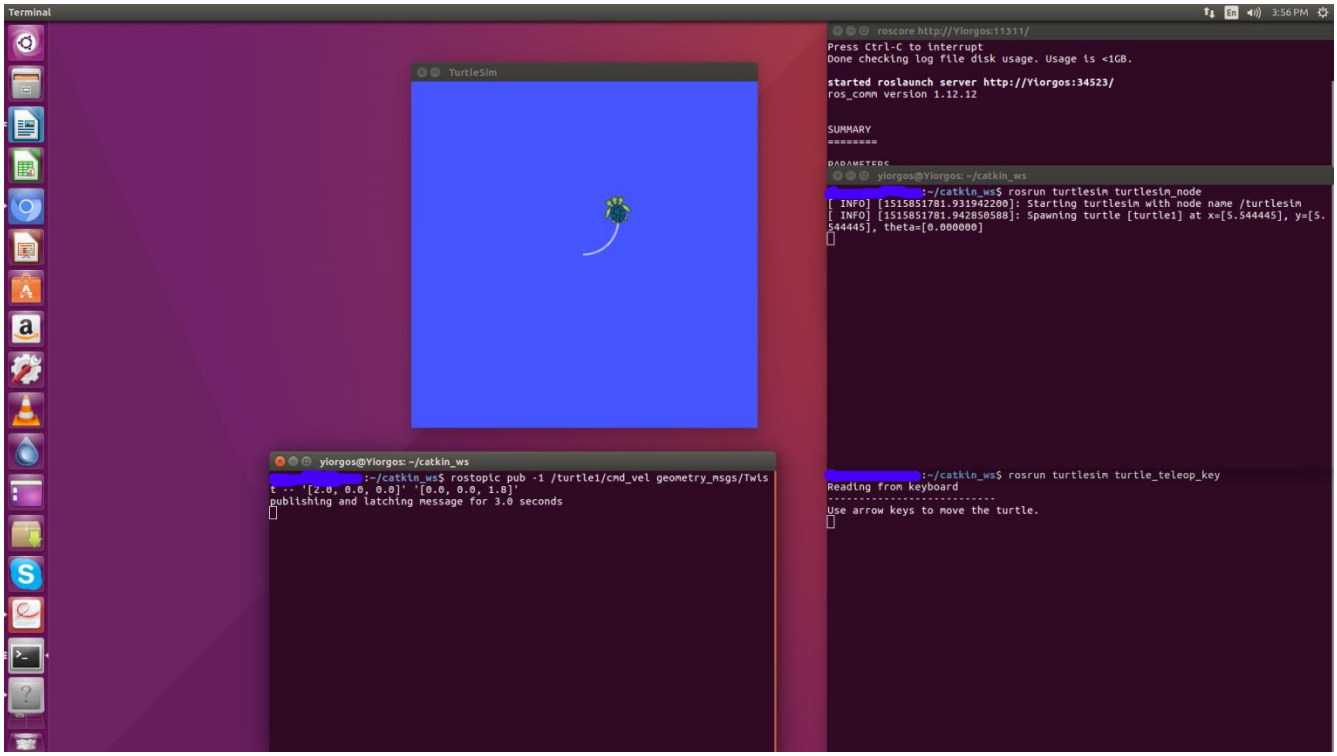
/turtle1/cmd_vel → Το όνομα του θέματος

geometry_msgs/Twist → Ο τύπος του μηνύματος

- - → Το συγκεκριμένο σύμβολο έχει συντακτικό συμβολισμό, δηλώνει πως κάποιος αριθμός δεν είναι επιλογή της εντολής (για παράδειγμα ο αριθμός των μηνυμάτων που θα αποσταλούν)

(δεν υπάρχει κενό ανάμεσα στις δύο γραμμές γράφτηκε έτσι λόγω μιας ιδιαιτερότητας του Microsoft Word)

‘[0.0, 0.0, 0.0]’ ‘[0.0, 0.0, 0.0]’ → ‘Οπως φαίνεται παραπάνω ο τύπος μηνυμάτων geometry_msgs/Twist έχει δύο διαφορετικές φορές διανύσματων, γραμμικό και γωνιακό όπου οι τιμές εκφράζουν τα επίπεδα x,y και z αντίστοιχα. Ο πρώτος πίνακας εκφράζει το γραμμικό διάνυσμα και ο δεύτερος το γωνιακό διάνυσμα.

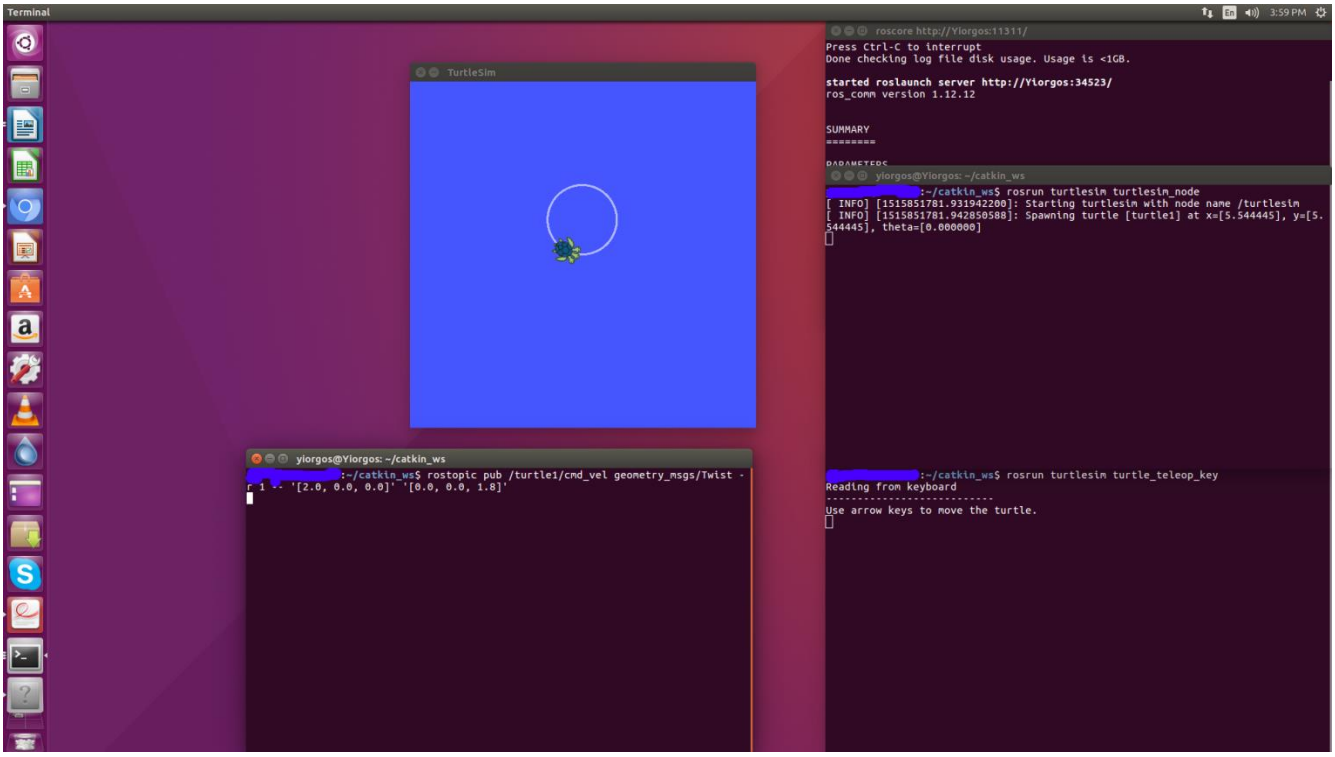


Εικόνα 26. Κίνηση χελώνας με την χρήση της εντολής rostopic pub

Μετά την εκτέλεση της εντολής η χελώνα θα διαγράψει μια μικρή κυκλική κίνηση. Για την συνεχή κίνηση της χελώνας πρέπει να οριστεί μια συχνότητα. Στο παρακάτω παράδειγμα η συχνότητα με την οποία θα μεταδίδεται το μήνυμα στο θέμα /turtle1/cmd_vel θα είναι 1 Hz[97].

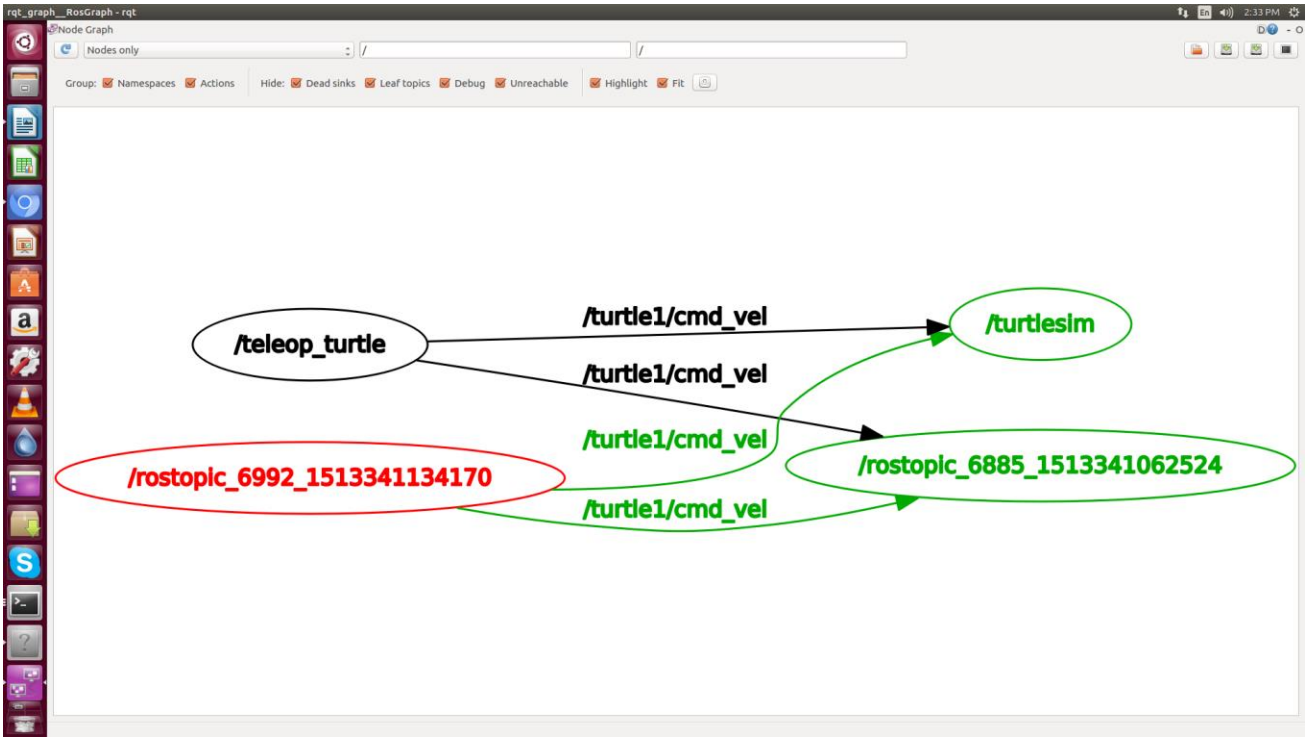
97 URL: http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics

rostopic pub /turtle1/cmd_vel geometry_msgs/Twist -r 1 -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'



Εικόνα 27. Κίνηση χελώνας με την χρήση της εντολής rostopic pub

Στο διάγραμμα του κόμβου rqt_graph ο προσωρινός κόμβος τον οποίο δημιουργεί η εντολή rostopic pub εκδίδει μηνύματα στον κόμβο turtlesim_node αλλά και στον προσωρινό κόμβο που έχει δημιουργήσει η εντολή rostopic echo.



Εικόνα 28. rqt_graph

4.9.3. Περαιτέρω Κατανόηση των Υπηρεσιών του ROS (Further Understanding of ROS Services)

Με την χρήση των υπηρεσιών του ROS γίνεται εφικτή η επικοινωνία αίτησης/απάντησης (request/response) μεταξύ των κόμβων. Τα θέματα του ROS δεν υποστηρίζουν αυτού του είδους την επικοινωνία επειδή δεν υπάρχει συγκεκριμένη κατεύθυνση κατά την μεταφορά δεδομένων. Οι υπηρεσίες του χαρακτηρίζονται από ένα ζευγάρι μηνυμάτων, ένα για την αίτηση που γίνεται προς τον κόμβο εξυπηρετητή και ένα για την απάντηση που δίνεται προς τον αιτούμενο κόμβο (πελάτη).

Παρακάτω δίνονται παραλλαγές της εντολής rosservice:

rosservice call /service args: Καλεί μια υπηρεσία

rosservice find service_type: Βρίσκει εντολές με τον συγκεκριμένο τύπο υπηρεσιών

rosservice info /services: Εμφανίζει πληροφορίες για την συγκεκριμένη υπηρεσία

rosservice list: Εμφανίζει τις ενεργές υπηρεσίες

rosservice type /service: Εμφανίζει τον τύπο υπηρεσιών μιας υπηρεσίας[98]

Σε αυτή την ενότητα είναι απαραίτητη η προσομοίωση της προηγούμενης ενότητας με την εξής ακολουθία:

roscore

rosrun turtlesim turtlesim_node

rosrun turtlesim turtle_teleop_key

Σε ένα νέο τερματικό πληκτρολογείται η εντολή:

rosservice list

Εμφανίζονται οι παρακάτω υπηρεσίες:

/clear

/kill

[98] Lentin J. (2015). p. 18.

/reset

/rosout/get_loggers

/rosout/set_logger_level

/spawn

/teleop_turtle/get_loggers

/teleop_turtle/set_logger_level

/turtle1/set_pen

/turtle1/teleport_absolute

/turtle1/teleport_relative

/turtlesim/get_loggers

/turtlesim/set_logger_level

Η υπηρεσία /clear είναι ‘άδεια’, αυτό σημαίνει πως δεν χρειάζεται δεδομένα όταν γίνεται μια αίτηση και δεν λαμβάνει δεδομένα όταν λαμβάνει μια απάντηση. Πληκτρολογείται στο τερματικό η εντολή:

rosservice type /clear

Θα εμφανίσει:

std_srvs/Empty

Στο παράθυρο προσομοίωσης κατά την κίνηση της χελώνας δημιουργείται μια λευκή γραμμή. Για να εξαφανιστεί πληκτρολογείται η παρακάτω εντολή:

```
rosservice call /clear
```

Το επόμενο βήμα είναι η ανάλυση της υπηρεσίας /spawn. Χρησιμοποιώντας την εντολή rosservice type σε συνδυασμό με την εντολή rossrv show είναι εφικτή η ανάκτηση περισσότερων πληροφοριών για την υπηρεσία. Πληκτρολογείται η εντολή:

```
rosservice type /spawn | rossrv show
```

Θα εμφανίσει:

```
float32 x

float32 y

float32 theta

string name

---

string name
```

Η συγκεκριμένη υπηρεσία δημιουργεί μια νέα χελώνα και δέχεται ως δεδομένα το σημείο εκκίνησης με συντεταγμένες στους άξονες x και y, την γωνία και το όνομα. Σε ένα τερματικό πληκτρολογείται η εντολή:

```
rosservice call /spawn 2 2 0.2 ""
```

Θα εμφανίσει στο σημείο με τις συντεταγμένες που δόθηκαν μια νέα χελώνα με το όνομα turtle2.

4.9.4. Περαιτέρω Κατανόηση του Διακομιστή Παραμέτρων (Further Understanding of the ROS Parameter)

Κατά τον προγραμματισμό μιας ρομποτικής εφαρμογής είναι πιθανή η χρήση παραμέτρων για την σωστή λειτουργία της, για παράδειγμα το κέρδος ενός ελεγκτή PID. Όταν ο αριθμός των παραμέτρων αυξάνεται το σύνηθες είναι η αποθήκευσή τους σε αρχεία όπου αποθηκεύονται στον διακομιστή παραμέτρων. Ένας κόμβος έχει πρόσβαση στα συγκεκριμένα αρχεία και μπορεί να αποθηκεύσει, διαβάσει, γράψει αλλά και να διαγράψει τιμές. Με την εντολή rosparam μπορεί κανείς να έχι πρόσβαση στις παραμέτρους του ROS:

rosparam set [parameter_name] [value]: Θέτει μια τιμή στην συγκεκριμένη παράμετρο

rosparam get [parameter_name]: Επιστρέφει την τιμή μιας παραμέτρου

rosparam load [YAML file]: Φορτώνει αποθηκευμένες παραμέτρους από αρχείου τύπου .YAML

rosparam dump [YAML file]: Αποθηκεύει τις υπάρχουσες παραμέτρους σε ένα αρχείο. YAML

rosparam delete [parameter_name]: Διαγράφει μια παράμετρο

rosparam list: Εμφανίζει τις υπάρχουσες παραμέτρους[99]

Σε αυτή την ενότητα είναι απαραίτητη η προσομοίωση της προηγούμενης ενότητας με την εξής ακολουθία:

[99] Lentin J. (2015). p. 21.

roscore

rosrun turtlesim turtlesim_node

rosrun turtlesim turtle_teleop_key

Πληκτρολογείται η εντολή:

rosparam list

Θα εμφανίσει τις παραμέτρους που χρησιμοποιούνται για την προσομοίωση:

background_b

/background_g

/background_r

/rosdistro

/roslaunch/uris/host_yiorgos__40338

/rosversion

/run_id

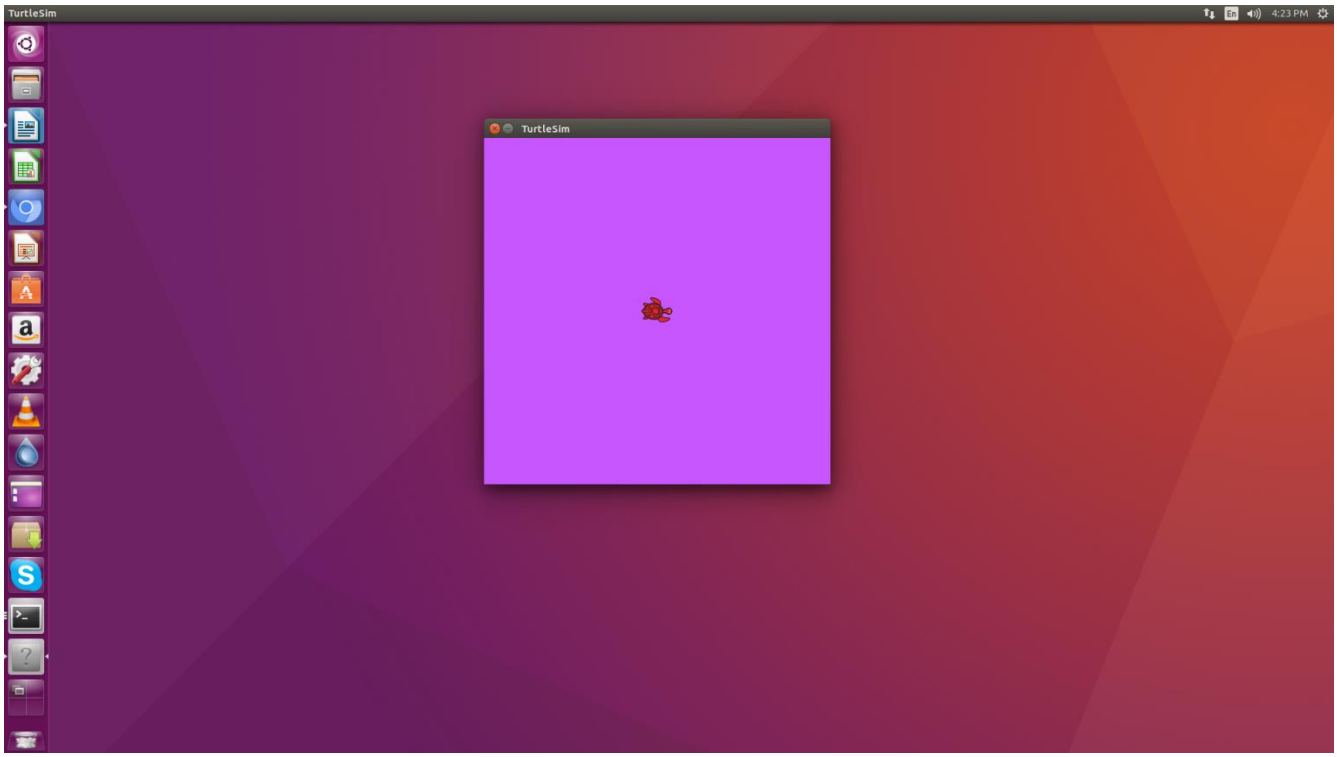
Παρατηρεί κανείς πως ο κόμβος turtlesim_node χρησιμοποιεί τρεις παραμέτρους για το υπόβαθρο του παραθύρου της προσομοίωσης. Με την τροποποίηση της παραμέτρου /background_r είναι εφικτή η αλλαγή του χρώματος υποβάθρου του παραθύρου προσομοίωσης.

Στο τερματικό πληκτρολογείται η εντολή:

```
rosparam set /background_r 200
```

Παρατηρεί κανείς πως το χρώμα του υπόβαθρου δεν άλλαξε. Για την οριστικοιποίηση των αλλαγών είναι απαραίτητη η κλήση της υπηρεσίας /clear (βλ. Εικόνα 32). Στο τερματικό πληκτρολογείται η εντολή:

```
rosservice call /clear
```



Εικόνα 29. Background change

4.10. Δημιουργία Κόμβου Εκδότη και Συνδρομητή (Creation of Publisher and Subscriber Node)

Όπως έχει αναφερθεί και σε προηγούμενη ενότητα ο κόμβος είναι ένα εκτελέσιμο αρχείο το οποίο είναι στην συνδεδεμένο στο δίκτυο του ROS. Σε αυτή την ενότητα θα δημιουργηθούν δύο

αρχεία τύπου .cpp, ένας κόμβος εκδότης που θα εκπέμπει συνεχώς ένα μήνυμα και ένας κόμβος συνδρομητής που θα λαμβάνει το μήνυμα. Το πρώτο βήμα είναι η μετάβαση στον φάκελου του πακέτου beginner_tutorials και η δημιουργία ενός φακέλου με το όνομα src. Σε αυτόν τον φάκελο θα αποθηκεύονται αρχεία πηγαίου κώδικα που αφορούν το συγκεκριμένο πακέτο. Σε ένα τερματικό πληκτρολογείται η εντολή:

```
mkdir -p src
```

Δημιουργείται ένα κενό αρχείο με το όνομα publisher.cpp. Ο κώδικας του κόμβου εκδότη είναι ο εξής:

```
#include "ros/ros.h"

#include "std_msgs/String.h"

#include <sstream>

int main(int argc, char **argv)

{

ros::init(argc, argv, "publisher");

ros::NodeHandle n;

ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);

ros::Rate loop_rate(10);

int count = 0;

while (ros::ok())
```

```
{

std_msgs::String msg;

std::stringstream ss;

ss << "hello world " << count;

msg.data = ss.str();

ROS_INFO("%s", msg.data.c_str());

chatter_pub.publish(msg);

ros::spinOnce();

loop_rate.sleep();

++count;

}

return 0;

}
```

Περαιτέρω ανάλυση του κώδικα:

```
#include "ros/ros.h"
```

Η βιβιοθήκη ros/ros.h είναι ένα βολικό εργαλείο το οποίο περιλαμβάνει όλες τις επικεφαλίδες οι οποίες είναι απαραίτητες για την χρήση των βασικών χαρακτηριστικών του συστήματος ROS.

#include "std_msgs/String.h"

Καλείται το μήνυμα std_msgs/String.h το οποίο βρίσκεται στο πακέτο std_msgs. Πρόκειται για μια επικεφαλίδα η οποία καλείται από το αρχείο μηνύματος String.msg.

ros::init(argc, argv, "publisher");

Η συγκεκριμένη εντολή αρχικοποιεί το ROS και ορίζει το όνομα του κόμβου το οποίο είναι μοναδικό.

ros::NodeHandle n;

Η λαβή (node handle) είναι το κεντρικό σημείο επικοινωνίας με το ROS. Η πρώτη που τυπώνεται λαβή αρχικοποιεί τον κόμβο ενώ η δεύτερη λαβή σταματά τον κόμβο.

ros::Publisher chatter_pub = n.advertise<std_msgs::String>("chatter", 1000);

Δημιουργείται ένα αντικείμενο της κλάσης Publisher το οποίο ενημερώνει τον κυρίαρχο κόμβο πως πρόκειται να εκδώσει μηνύματα τύπου String μέσω του θέματος chatter. Το δεύτερο κομμάτι της εντολής αποθηκεύει τα χίλια πρώτα μηνύματα που δέχεται και έπειτα διαγράφει τα παλαιότερα.

ros::Rate loop_rate(10);

Το αντικείμενο ros::Rate ορίζει μια συχνότητα με την οποία εκτελείται ο βρόγχος και στέλνει μηνύματα.

int count = 0;

while (ros::ok())

Εάν το ros::ok() γίνει false η διαδικάσια τερματίζεται. Ο μετρητής count μετρά τον αριθμό των μηνυμάτων που έχουν σταλεί.

std_msgs::String msg;

std::stringstream ss;

ss << "hello world " << count;

msg.data = ss.str();

Δημιουργείται ένα αλφαριθμητικό αντικείμενο τύπου String στο οποίο καταχωρείται το μήνυμα που πρόκειται να σταλεί.

ROS_INFO("%s", msg.data.c_str());

Τυπώνεται το μήνυμα

chatter_pub.publish(msg);

Εκδίδεται το μήνυμα σε οποιονδήποτε είναι συνδεδεμένος στο θέμα

ros::spinOnce();

Εξυπηρετεί συναρτήσεις τύπου callback.

loop_rate.sleep();

Το πρόγραμμα σταματά να λειτουργεί ώστε να ο χρόνος έκδοσης να είναι 10 Hz.

Δημιουργείται ένα κενό αρχείο με το όνομα subscriber.cpp. Ο κώδικας του κόμβου συνδρομητή είναι ο εξής:

```
# include "ros/ros.h"

# include "std_msgs/String.h"

void chatterCallback(const std_msgs::String::ConstPtr& msg)

{

ROS_INFO("I heard: [%s]", msg->data.c_str());

}

int main(int argc, char **argv)

{

ros::init(argc, argv, "subscriber");

ros::NodeHandle n;

ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);

ros::spin();

return 0;

}
```

Περαιτέρω ανάλυση του κώδικα:

```
void chatterCallback(const std_msgs::String::ConstPtr& msg)
```

{

ROS_INFO("I heard: [%s]", msg->data.c_str());

}

Διακόπτει την ροή του προγράμματος κάθε φορά που λαμβάνεται ένα νέο μήνυμα.

ros::Subscriber sub = n.subscribe("chatter", 1000, chatterCallback);

Δηλώνεται ένα αντικείμενο της κλάσης Subscriber το οποίο ενημερώνει τον κυρίαρχο κόμβο πως επιθυμεί να εγγραφεί στο θέμα chatter και αφού καταχωρήσει χίλια μηνύματα να διαγράψει τα παλαιότερα.

Στο τέλος του αρχείου CmakeLists.txt προστίθενται οι παρακάτω γραμμές κώδικα:

include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(publisher src/publisher.cpp)

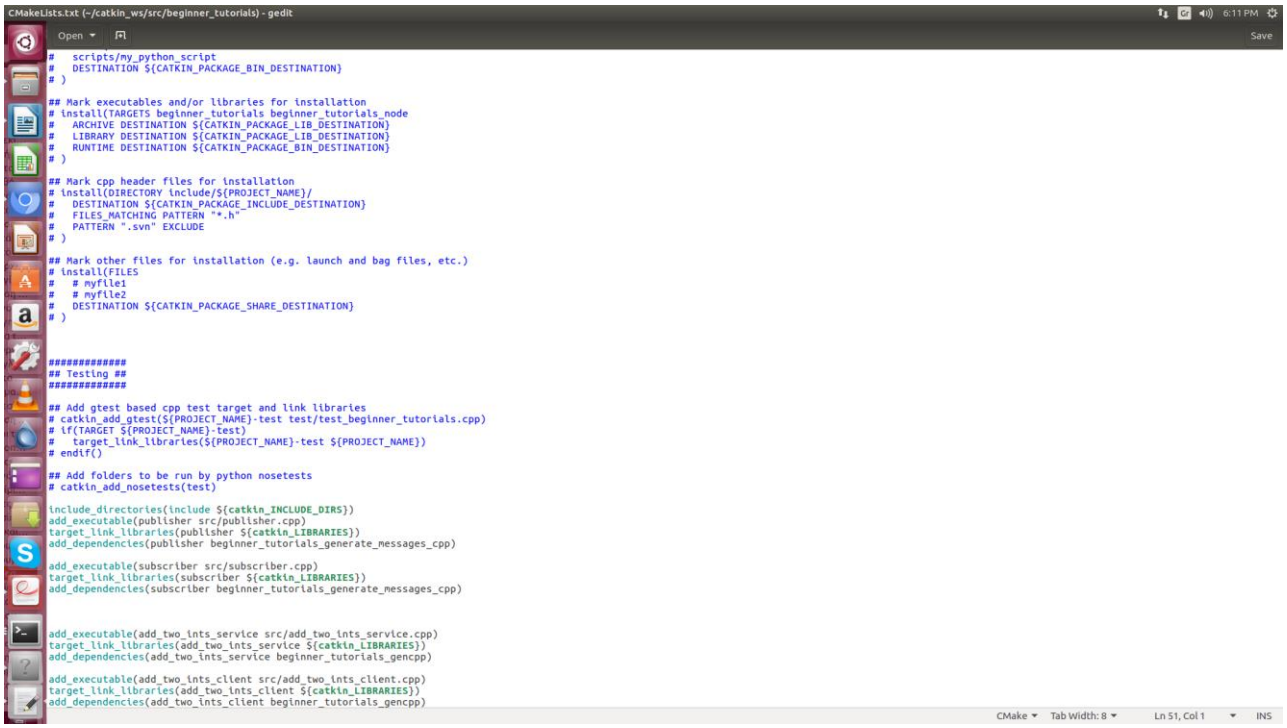
target_link_libraries(publisher ${catkin_LIBRARIES})

add_dependencies(publisher beginner_tutorials_generate_messages_cpp)

add_executable(subscriber src/subscriber.cpp)

target_link_libraries(subscriber ${catkin_LIBRARIES})

add_dependencies(subscriber beginner_tutorials_generate_messages_cpp)



Εικόνα 30. Αρχείο CmakeLists.txt

Στην συνέχεια το περιβάλλον εργασίας πρέπει να μεταγλωττιστεί. Πληκτρολογείται σε ένα τερματικό η εντολή:

catkin_make

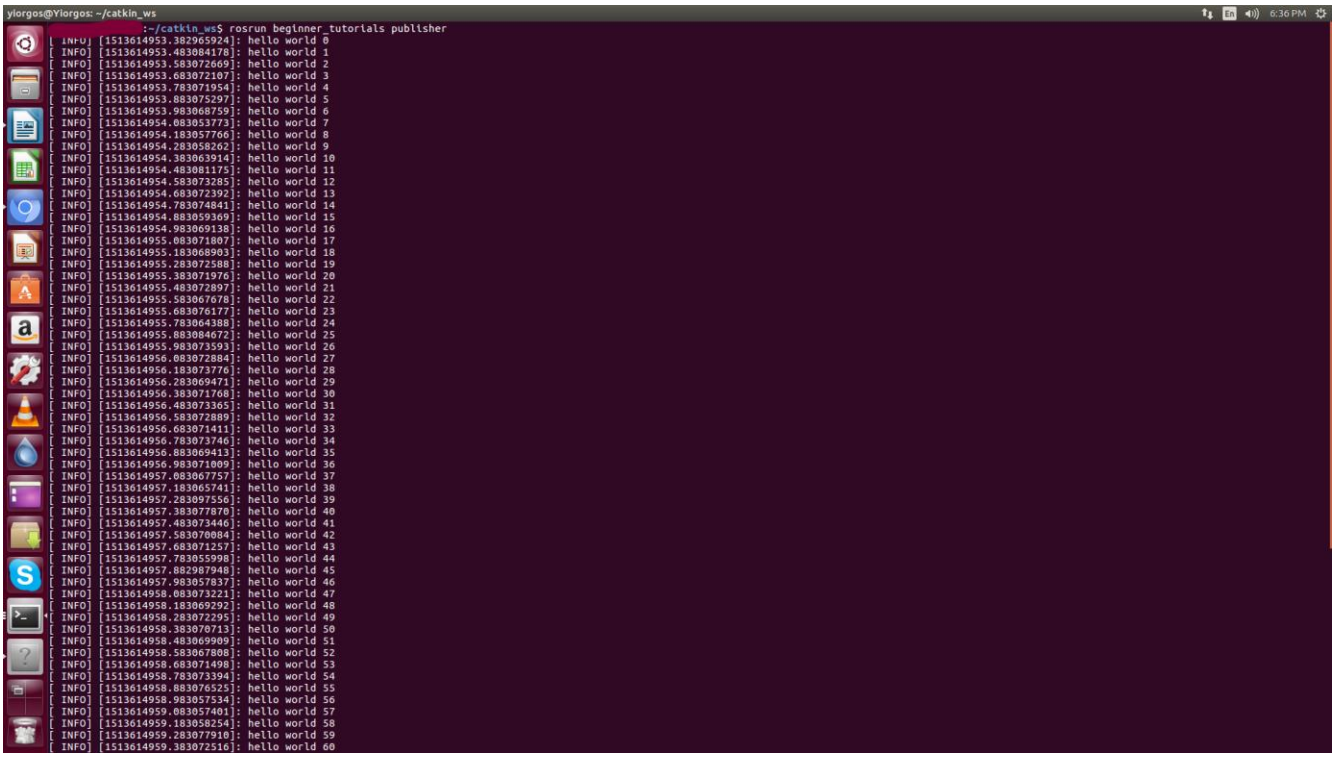
source ./devel/setup.bash

Εφόσον δεν εμφανιστεί κάποιο λάθος η μετάφραση ήταν επιτυχημένη[100].

Το επόμενο βήμα είναι η εκτέλεση των δύο κόμβων. Για την ενεργοποίηση του κόμβου εκδότη πληκτρολογείται η εντολή:

rosrun beginner_tutorials publisher

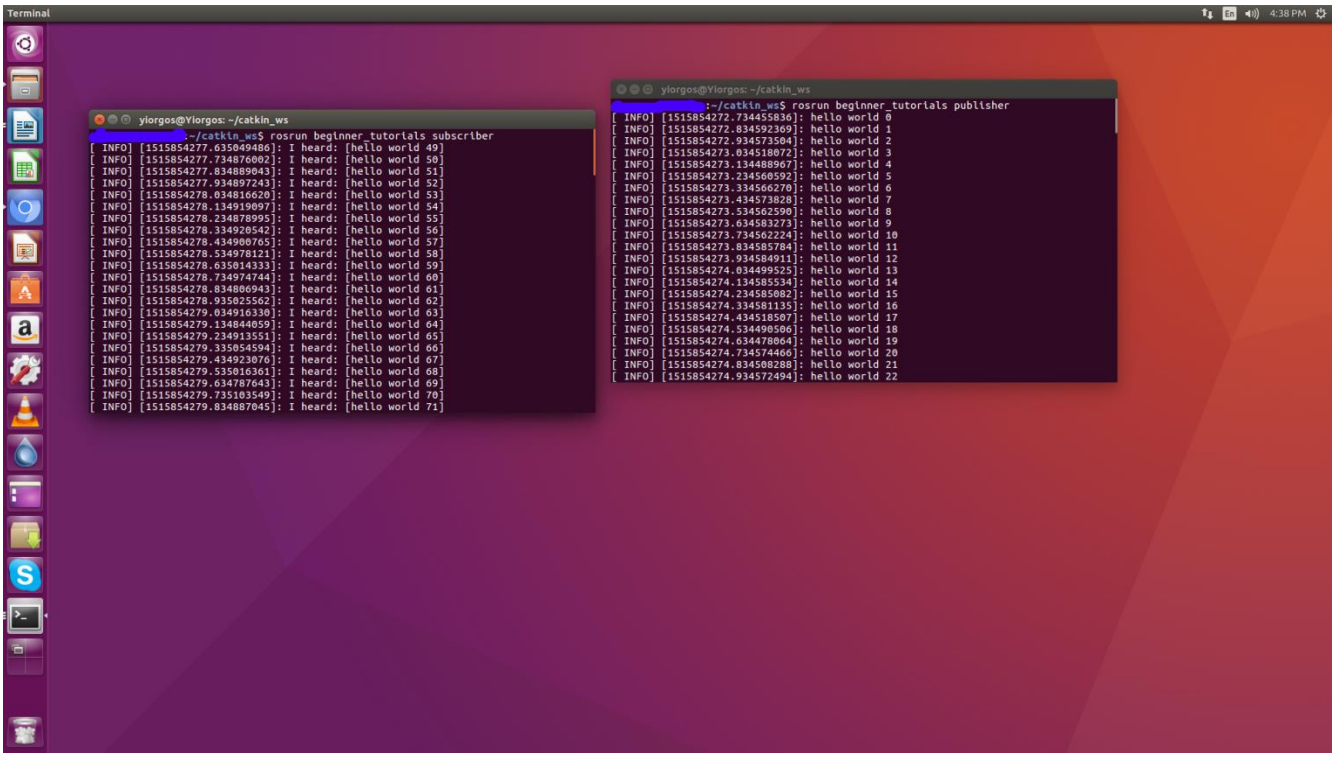
100 URL:http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29



Εικόνα 31. Κόμβος Εκδότης

Για την ενεργοποίηση του κόμβου συνδρομητή σε ένα άλλο τερματικό πληκτρολογείται η εντολή:

rosrun beginner_tutorials subscriber



Εικόνα 32. Κόμβος Συνδρομητής

4.11. Δημιουργία Κόμβου Υπηρεσίας (Creation of Service Node)

Σε αυτή την ενότητα στόχος είναι η δημιουργία ενός κόμβου υπηρεσίας με το όνομα “add_two_ints_service” η οποία θα λαμβάνει δυο ακέραιους αριθμούς και θα επιστρέφει το άθροισμά τους και ενός κόμβου πελάτη με το όνομα “add_two_ints_client” ο οποίος θα λαμβάνει το άθροισμά του κόμβου υπηρεσίας.Το πρώτο βήμα είναι η μετάβαση στον φάκελο beginner tutorials. Δημιουργείται ένα κενό αρχείο με το όνομα add_two_ints_service.cpp και προστίθεται ο παρακάτω κώδικας.

```
#include "ros/ros.h"

#include "beginner_tutorials/AddTwoInts.h"

bool add(beginner_tutorials::AddTwoInts::Request &req,
```

```
beginner_tutorials::AddTwoInts::Response &res)

{

res.sum = req.a + req.b;

ROS_INFO("request: x=%ld, y=%ld", (long int)req.a, (long int)req.b);

ROS_INFO("sending back response: [%ld]", (long int)res.sum);

return true;

}

int main(int argc, char **argv)

{

ros::init(argc, argv, "add_two_ints_server");

ros::NodeHandle n;

ros::ServiceServer service = n.advertiseService("add_two_ints", add);

ROS_INFO("Ready to add two ints.");

ros::spin();

return 0;

}
```

Περαιτέρω ανάλυση του κώδικα:

#include "beginner_tutorials/AddTwoInts.h"

Η beginner_tutorials/AddTwoInts.h είναι η επικεφαλίδα η οποία παράγεται από το αρχείο υπηρεσίας AddTwoInts.srv που δημιουργήθηκε σε προηγούμενη ενότητα.

bool add(beginner_tutorials::AddTwoInts::Request &req,

beginner_tutorials::AddTwoInts::Response &res)

Αυτή η συνάρτηση παρέχει την υπηρεσία και επιστρέφει μια μεταβλητή boolean true ή false όσο αναφορά την διεκπεραίωση της υπηρεσίας.

ros::ServiceServer service = n.advertiseService("add_two_ints", add);

Δημιουργείται ένα αντικείμενο της κλάσης ServerService όπου δημιουργεί την υπηρεσία add_two_ints και την αποστέλει στον κόμβο πελάτη.

Δημιουργείται ένα κενό αρχείο με το όνομα add_two_ints_client.cpp και προστίθεται ο παρακάτω κώδικας.

#include "ros/ros.h"

#include "beginner_tutorials/AddTwoInts.h"

#include <cstdlib>

int main(int argc, char **argv)

{

```
ros::init(argc, argv, "add_two_ints_client");

if (argc != 3)

{

ROS_INFO("usage: add_two_ints_client X Y");

return 1;

}

ros::NodeHandle n;

ros::ServiceClient client = n.serviceClient<beginner_tutorials::AddTwoInts>("add_two_ints");

beginner_tutorials::AddTwoInts srv;

srv.request.a = atoll(argv[1]);

srv.request.b = atoll(argv[2]);

if (client.call(srv))

{

ROS_INFO("Sum: %ld", (long int)srv.response.sum);

41}

else

{
```

```
ROS_ERROR("Failed to call service add_two_ints");

return 1;

}

return 0;

}
```

Περαιτέρω ανάλυση του κώδικα:

```
ros::ServiceClient client =n.serviceClient<beginner_tutorials::AddTwoInts>("add_two_ints");
```

Δημιουργείται ένα αντικείμενο της κλάσης ServiceClient το οποίο στέλνει αίτηση προς την υπηρεσία add_two_ints.

```
beginner_tutorials::AddTwoInts srv;

srv.request.a = atoll(argv[1]);

srv.request.b = atoll(argv[2]);
```

Ορίζεται το αρχείο υπηρεσίας AddTwoInts.srv καθώς και οι δυο τιμές του οι οποίες είναι υπεύθυνες για την αίτηση (request) και απάντηση (response).

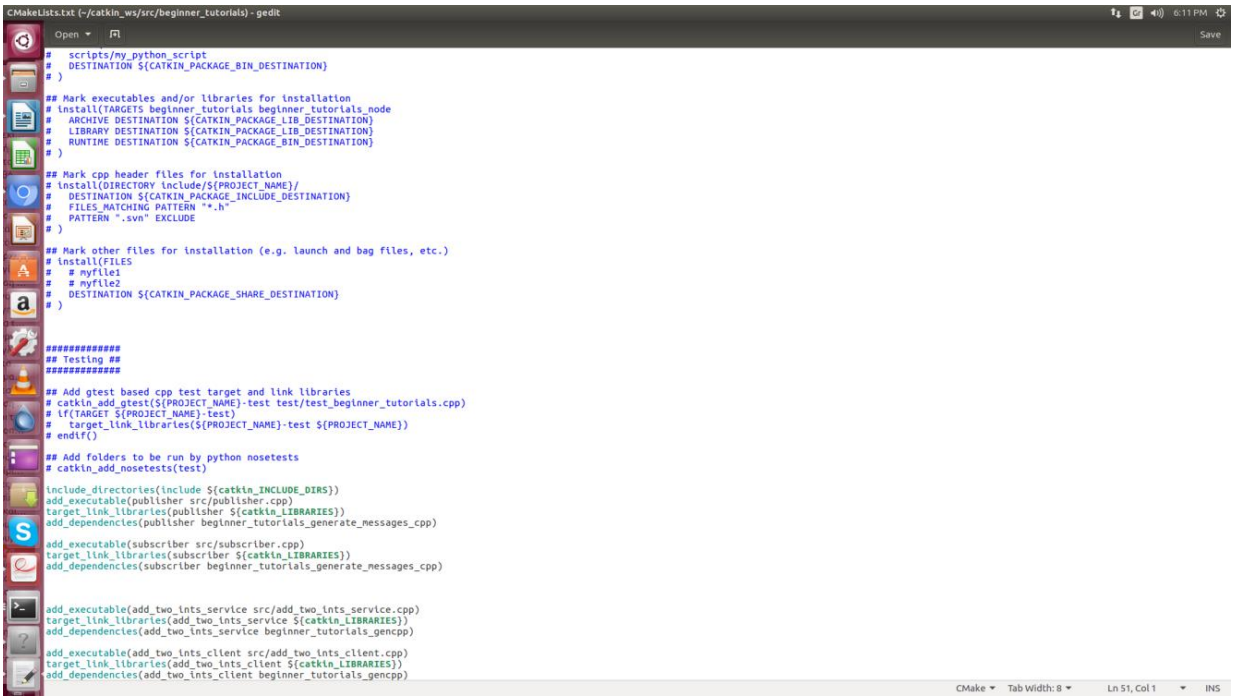
```
if (client.call(srv))
```

Εφόσον το κάλεσμα της συνάρτησης ήταν επιτυχημένο θα επιστρέψει την τιμή true αλλιώς την τιμή false[101].

Οι παρακάτω γραμμές κώδικα προστίθενται στο αρχείο CmakeLists.txt:

```
add_executable(add_two_ints_service src/add_two_ints_service.cpp)

target_link_libraries(add_two_ints_service ${catkin_LIBRARIES})

add_dependencies(add_two_ints_service beginner_tutorials_gencpp)

add_executable(add_two_ints_client src/add_two_ints_client.cpp)

target_link_libraries(add_two_ints_client ${catkin_LIBRARIES})

add_dependencies(add_two_ints_client beginner_tutorials_gencpp)
```

[101] URL: http://wiki.ros.org/ROS/Tutorials/WritingServiceClient%28c%2B%2B%29



Εικόνα 33. Αρχείο CmakeLists.txt

Έπειτα γίνεται μετάφραση του χώρου εργασίας:

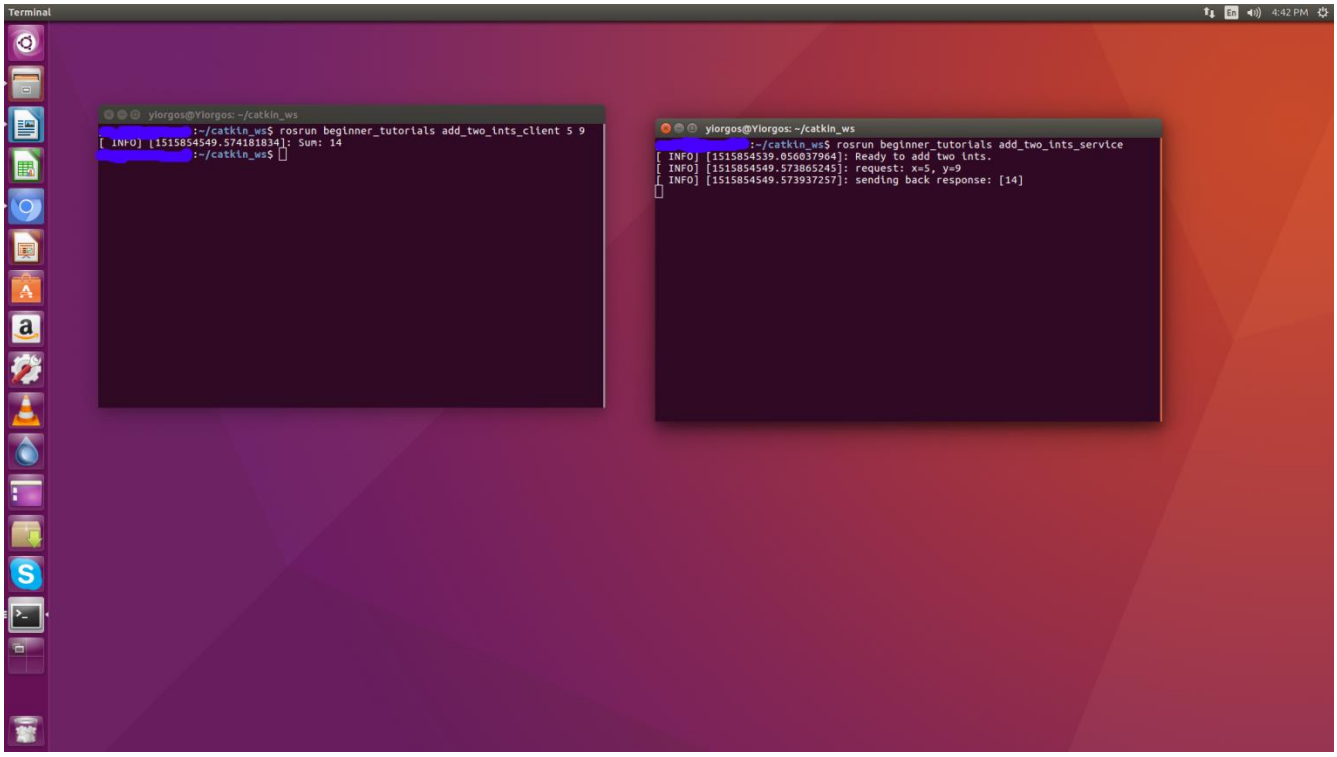
catkin_make

Το επόμενο βήμα είναι η έναρξη του κυρίαρχου κόμβου και σε διαφορετικά τερματικά η εκτέλεση του κόμβου υπηρεσίας και του κόμβου πελάτη, η σειρά εκτέλεσης των εντολών είναι η εξής:

roscore

rosrun beginner_tutorials add_two_ints_service

rosrun beginner_tutorials add_two_ints_client 5 9



Εικόνα 34. Service – Client

5. Επικοινωνία Arduino - ROS

Σε αυτή την ενότητα ο χρήστης έρχεται σε επαφή με την επικοινωνία μεταξύ Arduino – ROS και μαθαίνει τις εντολές με τις οποίες γίνεται εφικτή. Επιπλέον παρατίθενται παραδείγματα και περαιτέρω ανάλυσή τους.

5.1. Σύνδεση ROS με Arduino (Connention ROS with Arduino)

Οι περισσότεροι τρόποι επικοινωνίας μεταξύ ενός υπολογιστή και ενός μικροελεγκτή σε μια ρομποτική εφαρμογή διεκπεραιώνονται μέσω του πρωτοκόλλου επικοινωνίας UART (UART protocol). Κατά την επικοινωνία χρησιμοποιούνται προγράμματα για την μετάφραση των σειριακών εντολών από την μια συσκευή στην άλλη. Η διεπαφή ROS – Arduino είναι ένας σίγουρος τρόπος επικοινωνίας μεταξύ μικροελεγκτών Arduino και υπολογιστή και είναι εφικτή μέσω του rosserial.

Το rosserial είναι ένα πακέτο λογισμικού το οποίο περιέχει πρωτόκολλα επικοινωνίας για την σύνδεση του ROS με περιφερειακές συσκευές μέσω σειριακής θύρας. Έχει την ικανότητα να μετατρέπει τους τύπους μηνυμάτων και υπηρεσιών που χρησιμοποιεί το ROS σε τύπους δεδομένων που χρησιμοποιούνται από την αντίστοιχη συσκευή. Με την βιβλιοθήκη rosserial_client η οποία είναι ενσωματωμένη στο πακέτο rosserial, είναι εφικτή η δημιουργία κόμβων σε περιβάλλον Arduino και Linux. Για την δημιουργία ενός διαύλου επικοινωνίας μεταξύ του υπολογιστή και ενός ελεγκτή Arduino χρησιμοποιείται το πακέτο rosserial_arduino. Για την εγκατάσταση του πακέτου rosserial πληκτρολογείται η εντολή:

sudo apt-get install ros-kinetic-rosserial-arduino

sudo apt-get install ros-kinetic-rosserial

Το επόμενο βήμα είναι η εγκατάσταση της βιβλιοθήκης rosserial_client στο περιβάλλον του Arduino για την επικοινωνία των προγραμμάτων με το ROS.

Sketch -> Include Library -> Manage Library και αναζητείται η βιβλιοθήκη rosserial (βλ. Κεφάλαιο 3.5)[102].

5.2. Δημιουργία ενός Κόμβου Εκδότη (Creating a Publisher Node)

Η πρώτη επαφή με το rosserial είναι η δημιουργία ενός προγράμματος το οποίο θα δημοσιεύει το μήνυμα "Hello World". Ο χρήστης έχει πρόσβαση στο πρόγραμμα μέσω της βιβλιοθήκης Rosserial Arduino Library επιλέγοντας File -> Examples -> Rosserial Arduino Library -> Hello World. Παρακάτω παρατίθεται το πρόγραμμα και περαιτέρω επεξήγηση του κώδικα.

```
#include <ros.h>

#include <std_msgs/String.h>


ros::NodeHandle  nh;


std_msgs::String str_msg;

ros::Publisher chatter("chatter", &str_msg);
```

[102] URL: http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup

```
char hello[13] = "hello world!";


void setup()
{
  nh.initNode();
  nh.advertise(chatter);
}


void loop()
{
  str_msg.data = hello;
  chatter.publish( &str_msg );
  nh.spinOnce();
  delay(1000);
}
```

Ανάλυση του κώδικα:

#include <ros.h>

#include <std_msgs/String.h>

Η εντολή #include είναι μια εντολή η οποία απευθύνεται στον προεπεξεργαστή ώστε να συμπεριλάβει το πηγαίο πρόγραμμα ros.h στο υπάρχον πηγαίο πρόγραμμα. Ο προεπεξεργαστής (preprocessor) είναι ένα πρόγραμμα ενσωματωμένο με τον μεταγλωττιστή το οποίο είναι

υπεύθυνο για την συμπεριλήψη κώδικα από άλλα προγράμματα, όπως αρχεία κεφαλίδες. Τα αρχεία κεφαλίδες (header files) περιέχουν δηλώσεις μεταβλητών ή συναρτήσεων οι οποίες μπορούν να χρησιμοποιηθούν από πολλά αρχεία. Αυτή η μέθοδος εξοικονομεί χρόνο διότι στην περίπτωση που είναι απαραίτητη η χρήση μιας συγκεκριμένης συνάρτησης, δεν χρειάζεται ο χρήστης να την δηλώσει στο υπάρχον πρόγραμμά του απλά ορίζει τις παραμέτρους της.

ros::NodeHandle nh;

Με την χρήση της κλάσης ros::NodeHandle δημιουργείται ένα αντικείμενο με το όνομα nh, το οποίο είναι υπεύθυνο για την εκκίνηση του κόμβου και την εγκατάσταση σειριακής επικοινωνίας.

std_msgs::String str_msg;

ros::Publisher chatter("chatter", &str_msg);

Καλείται ο χώρος ονομάτων std_msgs για να δηλωθεί το αντικείμενο str_msg της κλάσης String. Στην επόμενη εντολή καλείται η κλάση ros::Publisher για την δημιουργία ενός αντικειμένου με το όνομα chatter. Μέσα στην παρένθεση αναφέρεται το όνομα του θέματος (“chatter”) και η μεταβλητή η οποία θα χρησιμοποιηθεί για την έκδοση μηνυμάτων.

```
void setup()
{
  nh.initNode();
  nh.advertise(chatter);
}
```

Καλείται η συνάρτηση μέλος initNode() της κλάσης ros::NodeHandle ώστε να ξεκινήσει η λειτουργία του κόμβου. Η συνάρτηση advertise() ενεργοποιεί τον εκδότη chatter.

```
void loop()
{
  str_msg.data = hello;
  chatter.publish( &str_msg );
  nh.spinOnce();
  delay(1000);
}
```

Μέσα στην συνάρτηση loop() αρχικοποιείται στο αντικείμενο str_msg η λέξη hello. Ο εκδότης chatter το μήνυμα. Η συνάρτηση spinOnce() είναι υπεύθυνη για την επανάληψη των

συναρτήσεων callback. Οι συναρτήσεις callback ορίζονται στο πρόγραμμα που χρησιμοποιεί ο χρήστης, αλλά έχουν δηλωθεί σε ένα διαφορετικό πρόγραμμα για παράδειγμα μια βιβλιοθήκη.

Το επόμενο βήμα είναι η εκτέλεση της εφαρμογής. Εφόσον το πρόγραμμα στο περιβάλλον Arduino έχει μεταγλωττιστεί, ανοίγει κανείς τρία τερματικά. Στο πρώτο πληκτρολογείται η εντολή roscore ώστε να γίνει η εκκίνηση των κόμβων του ROS. Στο δεύτερο τερματικό εκτελείται η εντολή:

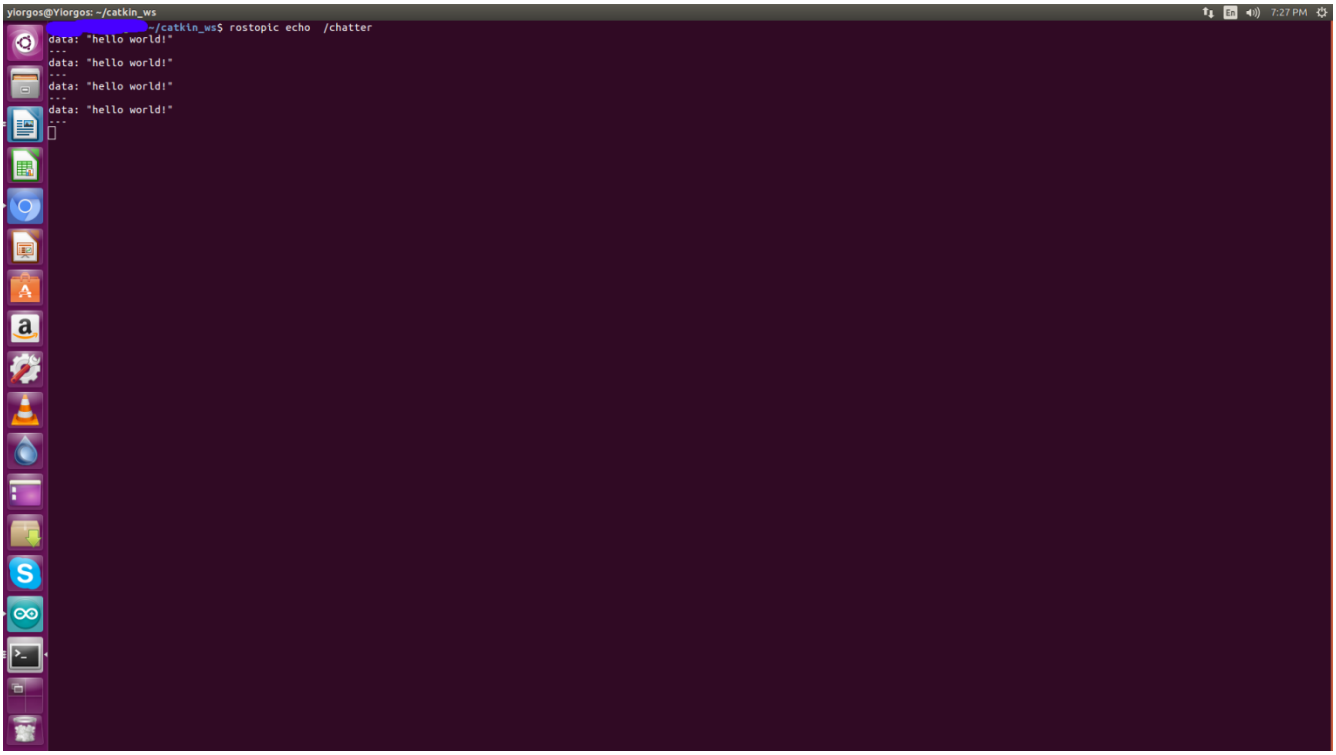
rosrun rosserial_python serial_node.py /dev/ttyUSB0

Με αυτό τον τρόπο τα μηνύματα που στέλνει το Arduino προωθούνται στο ROS μέσω της εκάστοτε σειριακής θύρας[103].

Τέλος για την εμφάνιση του μηνύματος μέσω του θέματος chatter πληκτρολογείται η εντολή:

rostopic echo chatter (βλ. Εικόνα 37).

[103] URL: http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World



Εικόνα 37. Θέμα /chatter

5.3. Δημιουργία Κόμβου Συνδρομητή (Creation of Subscriber Node)

Το επόμενο βήμα είναι η δημιουργία ενός προγράμματος συνδρομητή. Στο περιβάλλον του Arduino, στην βιβλιοθήκη Rosserial Arduino Library ανοίγεται το πρόγραμμα Blink. Παρακάτω παρατίθεται το πρόγραμμα και περαιτέρω επεξήγηση του κώδικα.

```
#include <ros.h>

#include <std_msgs/Empty.h>

ros::NodeHandle  nh;

void messageCb( const std_msgs::Empty& toggle_msg){
```

```
  digitalWrite(13, HIGH-digitalRead(13));   // blink the led
}
ros::Subscriber<std_msgs::Empty> sub("toggle_led", &messageCb );

void setup()
{
  pinMode(13, OUTPUT);
  nh.initNode();
  nh.subscribe(sub);
}

void loop()
{
  nh.spinOnce();
  delay(1);
}
```

Ανάλυση του κώδικα:

```
void messageCb( const std_msgs::Empty& toggle_msg){
  digitalWrite(13, HIGH-digitalRead(13));   // blink the led
}
```

Δημιουργείται μια συνάρτηση callback για τον συνδρομητή, η οποία δέχεται ως όρισμα μια σταθερά μηνύματος τύπου Empty. Η επόμενη εντολή χρησιμοποιεί τον ακροδέκτη 13 του Arduino, ώστε κάθε φορά που διαβάζει μήνυμα να ανάβει το ενσωματομένο led.

```
ros::Subscriber<std_msgs::Empty> sub("toggle_led", &messageCb );
```

Δημιουργείται ένα αντικείμενο της κλάσης Subscriber με όνομα sub και τύπο δεδομένων Empty. Οι λέξεις ros και std_msgs αναφέρονται στους χώρους ονομάτων που χρησιμοποιούν η κλάση και ο τύπος δεδομένων. Μέσα στην παρένθεση αναφέρεται το όνομα του θέματος και η συνάρτηση callback που χρησιμοποιεί για την λήψη μηνυμάτων.

```
void setup()
{
  pinMode(13, OUTPUT);
  nh.initNode();
  nh.subscribe(sub);
}
```

Δηλώνεται ο ακροδέκτης 13 ως έξοδος και καλείται η συνάρτηση μέλος initNode() της κλάσης ros::NodeHandle ώστε να ξεκινήσει η λειτουργία του κόμβου. Η συνάρτηση advertise() ενεργοποιεί τον συνδρομητή sub.

Το επόμενο βήμα είναι η εκτέλεση της εφαρμογής. Εφόσον το πρόγραμμα στο περιβάλλον Arduino έχει μεταγλωττιστεί, ανοίγει κανείς τρία τερματικά. Στο πρώτο πληκτρολογείται η εντολή roscore ώστε να γίνει η εκκίνηση των κόμβων του ROS. Στο δεύτερο τερματικό εκτελείται η εντολή:

rosrun rosserial_python serial_node.py /dev/ttyUSB0

Με αυτό τον τρόπο τα μηνύματα που στέλνει το Arduino προωθούνται στο ROS μέσω της εκάστοτε σειριακής θύρας[104].

Τέλος για να αναβοσβήσει το led ο χρήστης εκδίδει στο θέμα toggle_led:

rostopic pub toggle_led std_msgs/Empty –once

5.4. Τηλεχειρισμός Οχήματος (Teleoperation of an Mobile Vehicle)

Πριν την εισαγωγή παρούσα ενότητα είναι απαραίτητη η εγκατάσταση του πακέτου teleop_twist_keyboard. Πληκτρολογείται η εντολή:

sudo apt-get install ros-kinetic-teleop-twist-keyboard

[104] URL: http://wiki.ros.org/rosserial_arduino/Tutorials/Blink

Σε αυτή την ενότητα στόχος είναι ο τηλεχειρισμός ενός MV (Mobile Vehicle). Παρακάτω παρατίθεται ο κώδικας και περαιτέρω ανάλυσή του.

```
#include <ros.h>

#include <geometry_msgs/Twist.h>


ros::NodeHandle nh;

geometry_msgs::Twist msg;


float move1;

float move2;


void callback(const geometry_msgs::Twist& cmd_vel)

{

  move1 = cmd_vel.linear.x;

  move2 = cmd_vel.angular.z;

  if (move1 > 0 && move2 == 0)

  {
```

```
  front();
}
else if (move1 > 0 && move2 > 0 )
{
  left();
}
else if (move1 > 0 && move2 < 0 )
{
  right();
}
else if (move1 < 0)
{
  back();
}
else
{
  die();
```

```
  }
}
ros::Subscriber <geometry_msgs::Twist> sub("/cmd_vel", callback);


const int rightforw = 2;
const int leftback = 3;
const int leftforw = 4;
const int rightback = 5;

void setup() {
pinMode(leftforw, OUTPUT);
pinMode(leftback, OUTPUT);
pinMode(rightforw, OUTPUT);
pinMode(rightback, OUTPUT);

nh.initNode();
nh.subscribe(sub);
```

```
}


void loop() {
nh.spinOnce();
delay(1);
}


void front()
{
   digitalWrite(leftforw, HIGH);
   digitalWrite(rightforw, HIGH);
   digitalWrite(leftback, LOW);
   digitalWrite(rightback, LOW);
   delay(100);
   die();

}
```

```
void back()
{
   digitalWrite(leftforw, LOW);
   digitalWrite(rightforw, LOW);
   digitalWrite(leftback, HIGH);
   digitalWrite(rightback, HIGH);
   delay(100);
   die();
}
void left()
{
   digitalWrite(leftforw, LOW);
   digitalWrite(rightforw, HIGH);
   digitalWrite(leftback, LOW);
   digitalWrite(rightback, LOW);
   delay(100);
   die();
```

```
}
void right()
{
   digitalWrite(leftforw, HIGH);
   digitalWrite(rightforw, LOW);
   digitalWrite(leftback, LOW);
   digitalWrite(rightback, LOW);
   delay(100);
   die();
}
void die()
{
   digitalWrite(leftforw, LOW);
   digitalWrite(rightforw, LOW);
   digitalWrite(leftback, LOW);
   digitalWrite(rightback, LOW);
}
```

Ανάλυση του κώδικα:

```
#include <ros.h>

#include <geometry_msgs/Twist.h>
```

Περιλαμβάνονται στο πηγαίο πρόγραμμα η βιβλιοθήκη ros.h και η βιβλιοθήκη geometry_msgs/Twist.h.

```
ros::NodeHandle nh;

geometry_msgs::Twist msg;
```

Δημιουργείται ένα αντικείμενο της κλάσης NodeHandle με όνομα nh το οποίο ανήκει στο χώρο ονομάτος ros και ένα αντικείμενο της κλάσης Twist με όνομα msg το οποίο ανήκει στο χώρο στον χώρο ονόματος geometry_msgs.

```
void callback(const geometry_msgs::Twist& cmd_vel)

{

  move1 = cmd_vel.linear.x;

  move2 = cmd_vel.angular.z;

  if (move1 > 0 && move2 == 0)

  {
```

```
  front();
}
else if (move1 == 0 && move2 > 0 )
{
  left();
}
else if (move1 == 0 && move2 < 0 )
{
  right();
}
else if (move1 < 0)
{
  back();
}
else
{
  die();
```

```
  }
}
```

Στην συνάρτηση callback ορίζεται ως παράμετρος η μεταβλητή cmd_vel με τύπο δεδομένων Twist. Η μεταβλητή move1 ισοδυναμεί με την μεταβλητή cmd_vel.linear.x και η μεταβλητή move2 ισοδυναμεί με την μεταβλητή cmd_vel.angular.z. Ένα μήνυμα τύπου geometry_msgs::Twist εκφράζει την ταχύτητα στον χώρο υπό μορφή γραμμικών και γωνιακών διανυσμάτων στους άξονες x,y και z. Στην συγκεκριμένη περίπτωση το μέλος δεδομένων linear.x της μεταβλητής cmd_vel εκφράζει την τιμή του γραμμικού διανύσματος στον άξονα x και το μέλος δεδομένων angular.z την τιμή του γωνιακού διανύσματος στον άξονα z. Η κατεύθυνση που ακολουθεί το MV δίνεται υπό μορφή τιμών, για παράδειγμα όταν το όχημα ακολουθεί ευθεία πορεία το γραμμικό διάνυσμα x έχει τιμή μεγαλύτερη από μηδέν, ενώ το γωνιακό διάνυσμα z έχει τιμή ίση με μηδέν. Αντίστοιχα η ίδια λογική ακολουθείται και στις υπόλοιπες περιπτώσεις. Η μέθοδος γίνεται περισσότερο κατανοητή στην ενότητα 4.9.2.

```
ros::Subscriber <geometry_msgs::Twist> sub("/cmd_vel", callback);
```

Δημιουργείται ένα αντικείμενο της κλάσης Subscriber με όνομα sub στο οποίο εκδίδονται μηνύματα τύπου Twist. Μέσα στην παρένθεση αναγράφονται το όνομα του θέματος (cmd_vel) και η συνάρτηση callback η οποία εκπληρώνει τον ρόλο της παραμέτρου, λαμβάνοντας τα μηνύματα που εκδίδονται στο θέμα από το ROS.

```
pinMode(leftforw, OUTPUT);
pinMode(leftback, OUTPUT);
pinMode(rightforw, OUTPUT);
```

pinMode(rightback, OUTPUT);

Οι σταθερές δηλώνονται ως εξόδοι.

nh.initNode();

nh.subscribe(sub);

Η συνάρτηση αρχικοποιεί τον κόμβο nh ενώ η συνάρτηση subscribe() εντάσει τον κόμβο nh ως συνδρομητή στο θέμα του αντικειμένου sub.

void loop() {

nh.spinOnce();

delay(1);

}

Η συνάρτηση spinOnce() ενεργοποιεί τις συναρτήσεις του προγράμματος ώστε να τρέχουν διαρκώς.

Οι συναρτήσεις front(), back(), left(), right() και die() χρησιμοποιούνται για την αντίστοιχη κίνηση που θέλει να κάνει το όχημα όπως για παράδειγμα ευθεία ή δεξιά.

Το επόμενο βήμα είναι η εκτέλεση της εφαρμογής. Εφόσον το πρόγραμμα στο περιβάλλον Arduino έχει μεταγλωττιστεί, ανοίγει κανείς τρία τερματικά. Στο πρώτο πληκτρολογείται η

εντολή roscore ώστε να γίνει η εκκίνηση των κόμβων του ROS. Στο δεύτερο τερματικό εκτελείται η εντολή:

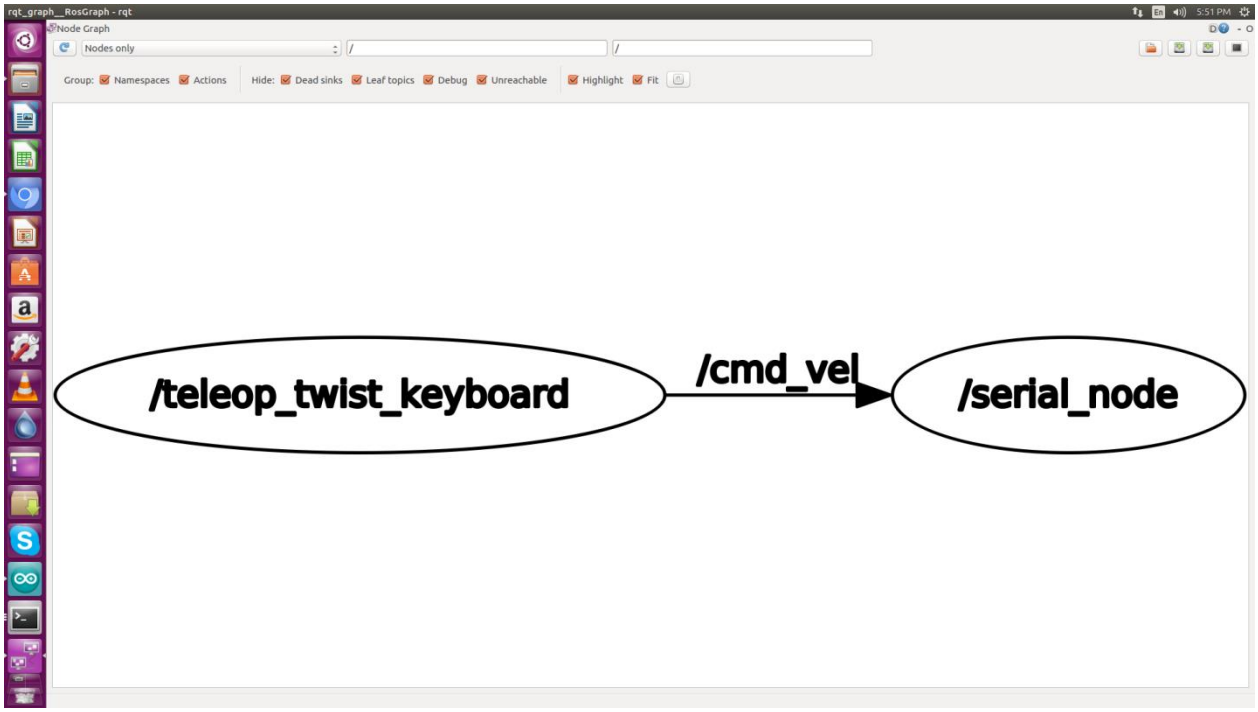
rosrun rosserial_python serial_node.py /dev/ttyUSB0

Με αυτό τον τρόπο τα μηνύματα που στέλνει το Arduino προωθούνται στο ROS μέσω της εκάστοτε σειριακής θύρας.

Τέλος για την εμφάνιση του πληκτρολογίου μέσα από το οποίο ο χρήστης ελέγχει το όχημα πληκτρολογείται η εντολή:

rosrun teleop_twist_keyboard teleop_twist_keyboard.py

Όπως φαίνεται και στην παρακάτω εικόνα το περιβάλλον του Arduino γίνεται κόμβος που



επικοινωνεί με τον κόμβο teleop_twist_keyboard.

Εικόνα 36. Rqt-graph με τους ενεργούς κόμβους του ROS

5.5. Αισθητήρας Υπερήχων (Ultrasonic Sensor)

Σε αυτή την ενότητα δημιουργείται ένα πρόγραμμα για τον υπολογισμό της απόστασης ανάμεσα στο ρομπότ και σε πιθανά εμπόδια με την χρήση ενός αισθητήρα υπερήχων.Οι μετρήσεις του αισθητήρα εκδίδονται στον μέσω ενός θέματος ROS. Παρακάτω παρατίθεται το πρόγραμμα και περαιτέρω ανάλυσή του.

```
#include <ros.h>

#include <ros/time.h>

#include <sensor_msgs/Range.h>

#define USE_USBCON

ros::NodeHandle  nh;


sensor_msgs::Range range_msg;

ros::Publisher pub_range( "/ultrasound", &range_msg);


int pingPin = 12;

int inPin = 13;

int ledPin = 11;
```

```
long range_time;
char frameid[] = "/ultrasound";


void setup() {
  nh.initNode();
  nh.advertise(pub_range);
  range_msg.radiation_type = sensor_msgs::Range::ULTRASOUND;
  range_msg.header.frame_id =  frameid;
  range_msg.field_of_view = 0.1;  // fake
  range_msg.min_range = 0.0;
  range_msg.max_range = 4.47;
}
void loop()
{
  if ( millis() >= range_time ){
      int r =0;
      range_msg.range = getRange() / 100;
```

```
      range_msg.header.stamp = nh.now();
      pub_range.publish(&range_msg);
      range_time =  millis() + 50;
    }
    nh.spinOnce();
    delay(1);
}

long microsecondsToCentimeters(long microseconds)
{
return microseconds / 29.1 / 2;
}

float getRange()
{
  long duration, cm;
```

```
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(3);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(pingPin, LOW);
  pinMode(inPin, INPUT);
  duration = pulseIn(inPin, HIGH);
  return microsecondsToCentimeters(duration);
}
```

Ανάλυση του κώδικα:

```
#include <ros.h>
#include <ros/time.h>
#include <sensor_msgs/Range.h>
```

Περιλαμβάνονται στο πρόγραμμα οι βιβλιοθήκες ros.h, sensor_msgs/Range.h και ros/time.h.

```
ros::NodeHandle  nh;
```

Με την χρήση της κλάσης ros::NodeHandle δημιουργείται ένα αντικείμενο με το όνομα nh, το οποίο είναι υπεύθυνο για την εκκίνηση του κόμβου και την εγκατάσταση σειριακής επικοινωνίας.

```
sensor_msgs::Range range_msg;
```

Με την χρήση της κλάσης Range του χώρου ονομάτων ros δημιουργείται το αντικείμενο range_msgs. Η κλάση Range περιέχει μέλη δεδομένων και συναρτήσεις-μέλη τα οποία συλλέγουν πληροφορίες για τον υπερηχητικό αισθητήρα και τα δεδομένα που επιστρέφει.

```
ros::Publisher pub_range( "/ultrasound", &range_msg);
```

Με την χρήση της κλάσης Publisher δημιουργείται το αντικείμενο pub_range.Μέσα στην παρένθεση αναφέρεται το όνομα του θέματος (“/ultrasound”) και το όρισμα range_msg το οποία θα χρησιμοποιηθεί για την έκδοση μηνυμάτων τύπου Range.

```
  nh.initNode();
  nh.advertise(pub_range);
  range_msg.radiation_type = sensor_msgs::Range::ULTRASOUND;
  range_msg.header.frame_id =  frameid;
  range_msg.field_of_view = 0.1;  // fake
```

```
  range_msg.min_range = 0.0;

  range_msg.max_range = 4.47;
```

Η συνάρτηση initNode() χρησιμοποιείται για την εκκίνηση του κόμβου.

Η συνάρτηση advertise(pub_range) ενεργοποιεί το θέμα pub_range.

Το μέλος δεδομένων radiation_type ισοδυναμεί με το αντικείμενο ULTRASOUND. Με αυτό τον τρόπο ενημερώνεται το πρόγραμμα πως πρόκειται πως ο αισθητήρας χρησιμοποιεί υπερηχητικά κύματα για την μέτρηση της απόστασης.

Το μέλος δεδομένων field_of_view καταχωρεί την ακτίνα του αισθητήρα μέσα στην οποία οι μετρήσεις θεωρούνται έγκυρες.

Τα min_range και max_range καταχωρούν την μικρότερη και την μέγιστη αντίστοιχα εμβέλεια που καλύπτει ο αισθητήρας.

```
float getRange()

{

  long duration, cm;

  pinMode(pingPin, OUTPUT);

  digitalWrite(pingPin, LOW);

  delayMicroseconds(3);

  digitalWrite(pingPin, HIGH);
```

```
  delayMicroseconds(10);
  digitalWrite(pingPin, LOW);
  pinMode(inPin, INPUT);
  duration = pulseIn(inPin, HIGH);


  return microsecondsToCentimeters(duration);
}
```

Η ακροδέκτης pingPin δηλώνεται ως έξοδος και η κατάστασή του μετατρέπεται σε LOW. Αυτό γίνεται ώστε η κατάσταση του ακροδέκτη να είναι σίγουρα LOW πριν μεταβεί σε HIGH για να παράγει παλμοσειρά. Μετά από καθυστέρηση τριων microseconds η κατάσταση του pingPin μετατρέπεται σε HIGH για περίπου 10 microseconds πριν μεταβεί ξανά σε LOW. Μέσα σε αυτό το διάστημα ο αισθητήρας έχει εκπέμψει παλμό ο οποίος μόλις συναντήσει εμπόδιο θα επιστρέψει ξανά προς τον αισθητήρα. Δηλώνοντας τον ακροδέκτη inPin ως είσοδο η λήψη του παλμού γίνεται με την συνάρτηση pulseIn. Όσο η τιμή του ακροδέκτη Pingpin είναι HIGH ο ακροδέκτης inPin θα κάνει λήψη της παλμοσειράς που θα επιστρέψει πίσω στον αισθητήρα η οποία καταχωρείται στην μεταβλητή duration. Η συνάρτηση microsecondsToCentimeters() μετατρέπει την διάρκεια του παλμού που μετρήθηκε σε εκατοστά.

```
 if ( millis() >= range_time ){
     int r =0;
     range_msg.range = getRange() / 100;
```

```
    range_msg.header.stamp = nh.now();
    pub_range.publish(&range_msg);
    range_time =  millis() + 50;
  }
```

Η συνθήκη if ενεργοποιείται όταν ο χρόνος που έχει μετρήσει η συνάρτηση millis() είναι μεγαλύτερος από την μεταβλητή range_time.

Το μέλος δεδομένων range ισοδυναμεί με την συνάρτηση getRange() η οποία ενεργοποιεί τους ακροδέκτες του αισθητήρα για την παραγωγή παλμοσειράς. Με αυτό τον τρόπο γίνεται λήψη της απόστασης η οποία μετρήθηκε και διαιρείται με τον αριθμό εκατό για την μετατροπή των εκατοστών σε μέτρα.

Το μέλος header.stamp εμφανίζει το χρονόσημο (timestamp) ενός μηνύματος sensor_msgs/Range, δηλαδή την ώρα κατά την οποία δημιουργήθηκε η τελευταία μέτρηση του αισθητήρα και προστίθεται στην επικεφαλίδα. Η συνάρτηση-μέλος now() ανήκει στην κλάση Time και επιστρέφει την τρέχουσα ώρα. Προκύπτει πως στο μέλος header.stamp καταχωρείται η ώρα που έγινε η λήψη του μηνύματος.

Το αντικείμενο pub_range εκδίδει το μήνυμα μέσω του θέματος /ultrasound.

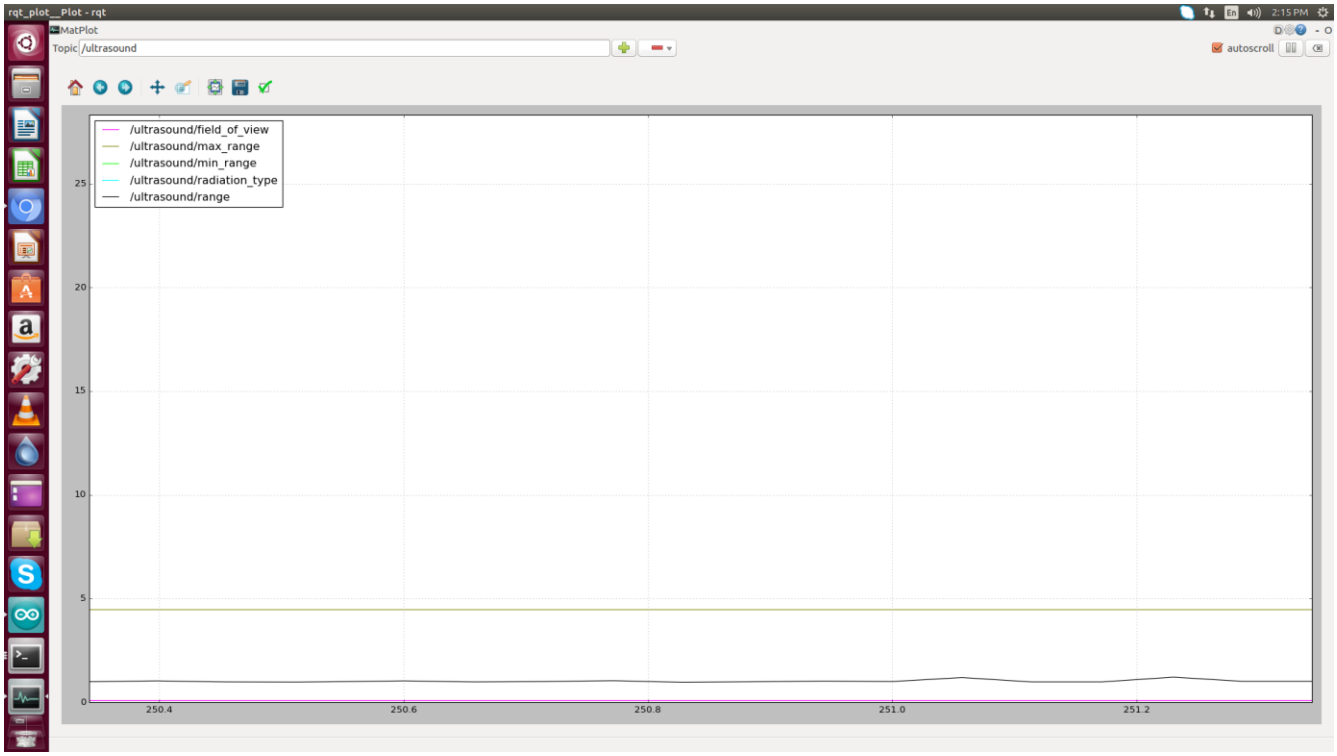
Το επόμενο βήμα είναι η εκτέλεση της εφαρμογής. Εφόσον το πρόγραμμα στο περιβάλλον Arduino έχει μεταγλωττιστεί, ανοίγει κανείς τρία τερματικά. Στο πρώτο πληκτρολογείται η εντολή roscore ώστε να γίνει η εκκίνηση των κόμβων του ROS. Στο δεύτερο τερματικό εκτελείται η εντολή:

rosrun rosserial_python serial_node.py /dev/ttyUSB0

Με αυτό τον τρόπο τα μηνύματα που στέλνει το Arduino προωθούνται στο ROS μέσω της εκάστοτε σειριακής θύρας.

Τέλος για την εμφάνιση του διαγράμματος το οποίο αναπαριστά τα στοιχεία του θέματος /ultrasonic πληκτρολογείται η εντολή:

rqt_plot /ultrasound



Εικόνα 39. Θέμα /ultrasonic

Παρατηρεί κανείς την μέγιστη και ελάχιστη απόσταση που καλύπτει ο αισθητήρας, την ακτίνα εμβέλειάς του καθώς και την απόσταση που χωρίζει το ρομπότ από ένα εμπόδιο.

5.6. Τηλεχειρισμός με την Χρήση Υπερηχητικού Αισθητήρα (Teleopration with Use of Ultrasonic Sensor)

Το παρακάτω πρόγραμμα αποτελεί συνδυασμός των δυο προγραμμάτων για τα οποία έγινε αναφορά παραπάνω. Χρησιμοποιείται για τον τηλεχειρισμό του οχήματος ενώ παράλληλα εκδίδει στο ROS τις ενδείξεις του υπερηχητικού αισθητήρα. Παρακάτω παρατίθεται το πρόγραμμα:

```
#include <ros.h>

#include <geometry_msgs/Twist.h>

#include <ros/time.h>

#include <sensor_msgs/Range.h>

#define USE_USBCON


ros::NodeHandle  nh;


sensor_msgs::Range range_msg;

geometry_msgs::Twist msg;


float move1;
```

```
float move2;


ros::Publisher pub_range( "/ultrasound", &range_msg);


int pingPin = 12;
int inPin = 13;
int ledPin = 11;
long range_time;
char frameid[] = "/ultrasound";


void callback(const geometry_msgs::Twist& cmd_vel)
{
  move1 = cmd_vel.linear.x;
  move2 = cmd_vel.angular.z;
if (move1 > 0 && move2 == 0)
  {
    front();
```

```
}
else if (move1 == 0 && move2 > 0 )
{
  left();
}
else if (move1 == 0 && move2 < 0 )
{
  right();
}
else if (move1 < 0)
{
  back();
}
else
{
  die();
}
```

```
}


ros::Subscriber <geometry_msgs::Twist> sub("/cmd_vel", callback);


const int rightforw = 2;
const int leftback = 3;
const int leftforw = 4;
const int rightback = 5;
void setup() {
  nh.initNode();
  nh.advertise(pub_range);
  nh.subscribe(sub);
  range_msg.radiation_type = sensor_msgs::Range::ULTRASOUND;
  range_msg.header.frame_id =  frameid;
  range_msg.field_of_view = 0.1;  // fake
  range_msg.min_range = 0.0;
  range_msg.max_range = 4.47;
```

```
  pinMode(leftforw, OUTPUT);
  pinMode(leftback, OUTPUT);
  pinMode(rightforw, OUTPUT);
  pinMode(rightback, OUTPUT);
}
void loop()
{
  if ( millis() >= range_time ){
      int r =0;
      range_msg.range = getRange() / 100;
      range_msg.header.stamp = nh.now();
      pub_range.publish(&range_msg);
      range_time =  millis() + 50;
    }
  nh.spinOnce();
  delay(1);
}
```

```
long microsecondsToCentimeters(long microseconds)
{
return microseconds / 29.1 / 2;
}

float getRange()
{
  pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(3);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(pingPin, LOW);

  pinMode(inPin, INPUT);
  duration = pulseIn(inPin, HIGH);
```

```
  return microsecondsToCentimeters(duration);
}

void front()
{
  digitalWrite(rightenable, 180);
  digitalWrite(leftenable, 180);
  digitalWrite(rightforw, HIGH);
  digitalWrite(rightback, LOW);
  digitalWrite(leftforw, HIGH);
  digitalWrite(leftback, LOW);
  delay(500);
  die();
}
void back()
{
```

```
  digitalWrite(rightenable, 180);
  digitalWrite(leftenable, 180);
  digitalWrite(rightforw, LOW);
  digitalWrite(rightback, HIGH);
  digitalWrite(leftforw, LOW);
  digitalWrite(leftback, HIGH);
  delay(500);
  die();
}
void left()
{
digitalWrite(rightenable, 180);
  digitalWrite(leftenable, 180);
 digitalWrite(rightforw, LOW);
  digitalWrite(rightback, HIGH);
  digitalWrite(leftforw, HIGH);
  digitalWrite(leftback, LOW);
```

```
  delay(300);
  die();
}
void right()
{
digitalWrite(rightenable, 180);
  digitalWrite(leftenable, 180);
 digitalWrite(rightforw, HIGH);
  digitalWrite(rightback, LOW);
  digitalWrite(leftforw, LOW);
  digitalWrite(leftback, HIGH);
    delay(300);
  die();
}
void die()
{
  digitalWrite(leftforw, LOW);
```

```
  digitalWrite(rightforw, LOW);

  digitalWrite(leftback, LOW);

  digitalWrite(rightback, LOW);

}
```

6. Όχημα που ακολουθεί προκαθορισμένη διαδρομή

Σε αυτή την ενότητα παρατίθεται τα στοιχεία του οχήματος καθώς και μια σύντομη περιγραφή για το καθένα. Στην συνέχεια περιγράφεται η διαδικασία της υλοποίησης καθώς ανάλυση του προγράμματος που χρησιμοποιεί το όχημα.

6.1. Εισαγωγή

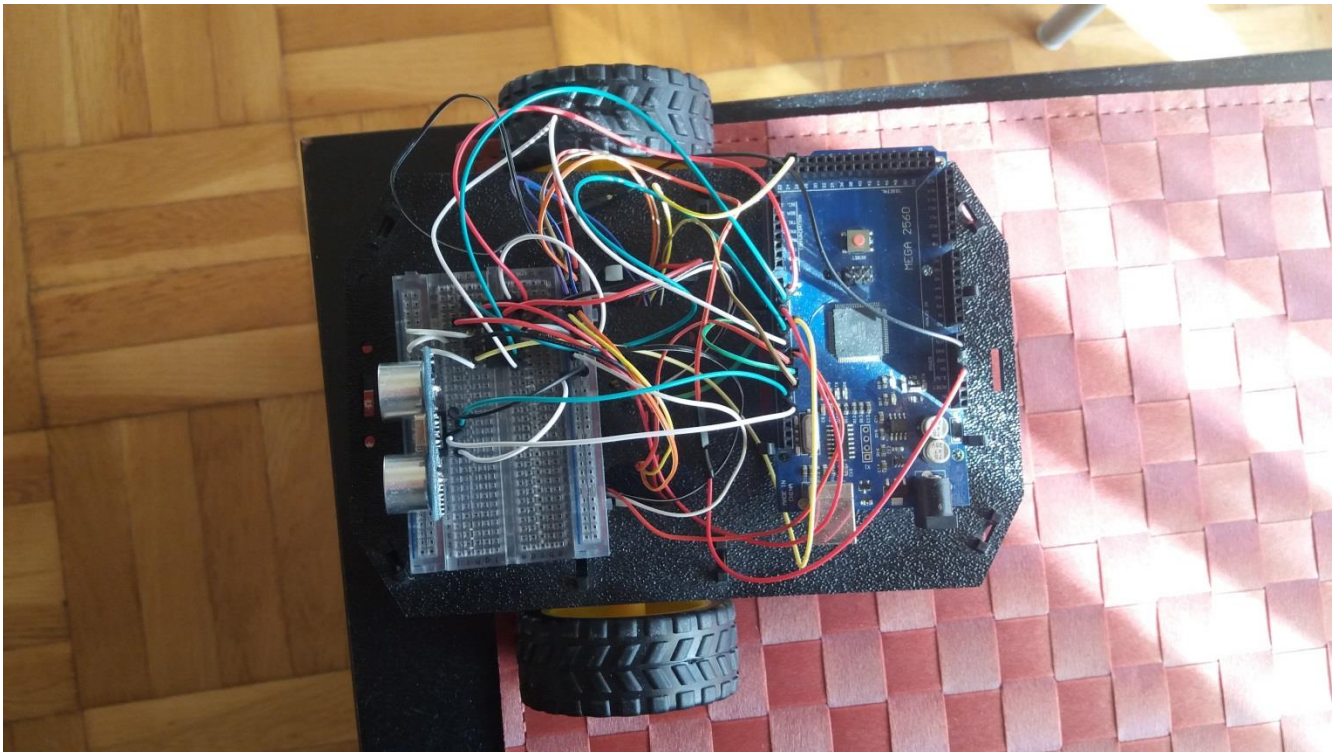
Line follower robot ονομάζεται ένα κινητό ρομπότ το οποίο έχει την δυνατότητα να ανιχνεύει και να ακολουθεί μια γραμμή σχεδιασμένη στο δάπεδο. Το μονοπάτι που θα ακολουθήσει είναι προσχεδιασμένο και μπορεί είτε να είναι μια μαύρη γραμμή πάνω σε μια λευκή επιφάνεια είτε ένα αόρατο μονοπάτι όπως ένα μαγνητικό πεδίο. Η ανίχνευση της γραμμής είναι εφικτή μέσω υπέρυθρων αισθητήρων οι οποίοι έχουν εγκατασταθεί στο ρομπότ. Αφού οι πληροφορίες συλλεχθούν μεταφέρονται στον μικροελεγκτή ο οποίος επεξεργάζεται τα δεδομένα και χαράζει την πορεία του ρομπότ, μεταφέροντας στον οδηγό των κινητήρων (motor driver) μέσω ενός ηλεκτρικού μοτίβου εντολών οδηγίες ώστε το όχημα να ακολουθεί την προγεγραμμένη πορεία του.

Η οδήγηση του οχήματος γίνεται διαφορικά, δηλαδή σε ένα σταθερό άξονα οι τροχοί κινούνται ανεξάρτητα. Έτσι ρυθμίζοντας τις στροφές του κάθε τροχού το ρομπότ κινείται στην επιθυμητή κατεύθυνση, για παράδειγμα σε μια στροφή με κατεύθυνση προς τα αριστερά ο αριστερός τροχός θα μειώσει την ταχύτητά του ενώ ο δεξιός τροχός θα ανεβάσει την ταχύτητά του, η ίδια διαδικασία ακολουθείται και για την αντίθετη περίπτωση. Στην προκειμένη περίπτωση ο ένας ελεγκτής PID είναι υπεύθυνος για τον υπολογισμό της κίνησης.

Ο ελεγκτής PID είναι ένας γενικός μηχανισμός ελέγχου με ανατροφοδότηση ο οποίος χρησιμοποιείται σε βιομηχανικές και ρομποτικές εφαρμογές και προσπαθεί να διορθώσει το σφάλμα μεταξύ μιας μετρημένης μεταβλητής σε σχέση με ένα επιθυμητό σημείο λειτουργίας. Ο ελεγκτής περιέχει τρεις όρους τον όρο αναλογίας(P), τον όρο ολοκλήρωσης (I) και τον όρο διαφόρισης. Το κέρδος του αναλογικού ελεγκτή καθορίζει την αντίδραση στο τρέχον σφάλμα, το κέρδος του όρου ολοκλήρωσης ελεγκτή καθορίζει την αντίδραση στο συνεχές άθροισμα των σφαλμάτων και το κέρδος του όρου διαφόρισης ελεγκτή καθορίζει την αντίδραση στο ποσοστό του σφάλματος που έχει αλλάξει[105]. Στην προκειμένη περίπτωση το επιθυμητό σημείο λειτουργίας είναι η παραμονή του ρομπότ στο κέντρο της μαύρης γραμμής κατά την διαδικασία της οδήγησης και το σφάλμα είναι η απόκλιση του ρομπότ από το κέντρο της γραμμής[106].

[105] URL: http://hlektrologia.gr/%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CE%BF-%CE%B5%CE%BB%CE%B5%CE%B3%CE%BA%CF%84%CE%AE%CF%82-pid/

[106] URL: http://ieeexplore.ieee.org/abstract/document/5380235/



Εικόνα 40. Line Follower Robot

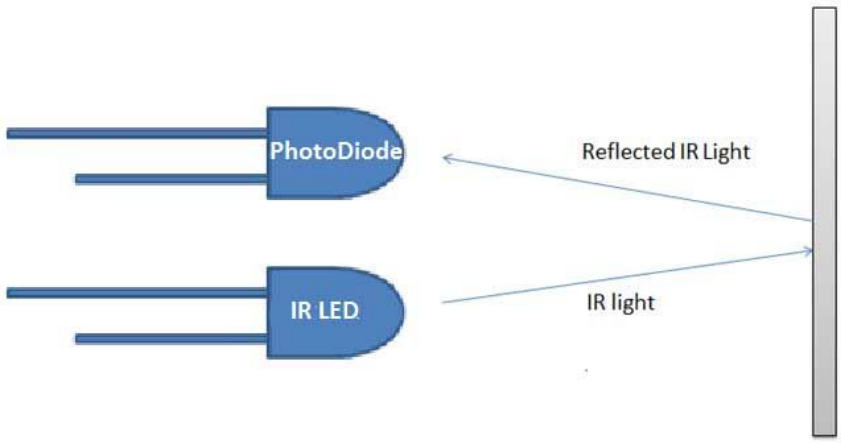
6.2. Ανάλυση Στοιχείων ενός Line Follower Robot

Παρακάτω αναλύονται ηλεκτρονικά στοιχεία που χρησιμοποιούνται σε ένα Line Follower Robot.

6.2.1. Υπέρυθρος Αισθητήρας (Infrared Sensor)

Ο υπέρυθρος αισθητήρας είναι ένα ηλεκτρονικό στοιχείο το οποίο χρησιμοποιείται σε πολλές εφαρμογές απομακρυσμένου έλεγχου, ανίχνευσης κίνησης, απαρίθμησης προϊόντων, ρομπότ γραμμής κ.α. αποτελείται από μια υπέρυθρη λυχνία και μια φωτοδίοδο. Η λειτουργία του υπέρυθρου αισθητήρα βασίζεται στην εκπομπή ακτινοβολίας από την υπέρυθρη λυχνία και στην συλλογή της από την φωτοδίοδο. Η αντίσταση της φωτοδιόδου αλλάζει ανάλογα με την

ποσότητα της ακτινοβολίας που δέχεται και κατά συνέπεια η έξοδος είναι ανάλογη, αλλά υπάρχει και η δυνατότητα λήψης ψηφιακής εξόδου[107].



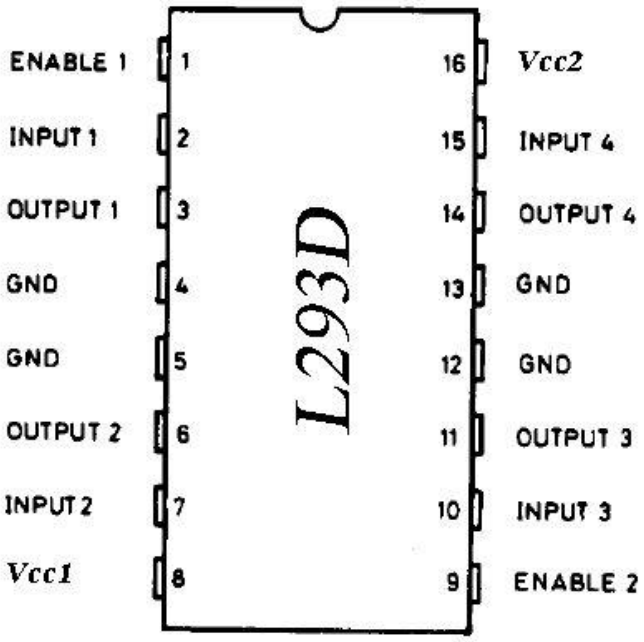
Εικόνα 41. Infrared Sensor (Πηγή circuitdigest.com)

6.2.2. Motor Driver L293D

Ο l293D είναι ένας οδηγός κινητήρων που αποτελείται από 4 κανάλια με 16 ακροδέκτες και είναι σε θέση να ελέγχει 2 DC κινητήρες ταυτόχρονα. Έχει σχεδιαστεί ώστε να παρέχει αμφίδρομα ρεύματα της τάξης των 600 mA ανά κανάλι και τάσης από 4.5 V έως 36 V στον ακροδέκτη 8, με τους υπόλοιπους ακροδέκτες του να υποστηρίζουν τάση έως 7 V. Κάθε ζευγάρι καναλιών σχηματίζει μια πλήρη H bridge. H bridge είναι ένα ηλεκτρικό κύκλωμα το οποίο

[107] URL: https://circuitdigest.com/electronic-circuits/ir-sensor-circuit-diagram

επιτρέπει στην τάση να εφαρμόζεται σε φορτίο σε οποιαδήποτε κατεύθυνση σε μια έξοδο, για παράδειγμα έναν κινητήρα[108].



Εικόνα 42. L293D Motor Driver (Πηγή google.gr)

6.2.3. Κινητήρας DC (DC Motor)

Ένας κινητήρας dc είναι μια ηλεκτρική μηχανή η οποία μετατρέπει την ηλεκτρική ενέργεια σε μηχανική. Οι περισσότεροι τύποι κινητήρων dc βασίζουν την λειτουργία τους στις δυνάμεις που

[108] URL; http://www.ti.com/lit/ds/symlink/l293.pdf

παράγονται από ένα μαγνητικό πεδίο. Σχεδόν όλοι οι τύποι κινητήρων dc έχουν ένα εσωτερικό μηχανισμό είτε μηχανικό είτε ηλεκτρικό ο οποίος είναι υπεύθυνος για την αλλαγή φοράς του ρεύματος στον κινητήρα[109].

Εικόνα 43. DC Motors (Πηγή grobotronics.com)

6.2.4. Αισθητήρας Υπερήχων(Ultrasonic Sensor)

Ένας αισθητήρας υπερήχων είναι μια συσκευή η οποία μετρά την απόσταση από ένα αντικείμενο στέλνοντας ένα ηχητικό κύμα με συγκεκριμένη συχνότητα και περιμένοντας να επιστρέψει υπολογίζει την απόσταση από το συγκεκριμένο αντικείμενο.

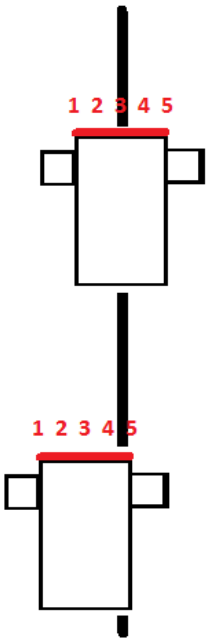
Εικόνα 44. Ultrasonic Sensor HC-SRF05

[109] URL: http://wv stics-of-dc-motors.html

6.3. PID Ελεγκτής (PID Controller)

Ο ελεγκτής PID (Proportional – Integral – Derivative) είναι ο ένας από τους πιο κοινούς ελεγκτές που χρησιμοποιούνται στην βιομηχανία και έχουν παγκόσμια αποδοχή στον βιομηχανικό έλεγχο. Η δημοτικότητα των ελεγκτών PID χαρακτηρίζεται εν μέρει από την εύρωστη απόδοσή τους σε ένα ευρύ φάσμα συνθηκών λειτουργίας και έπειτα από την λειτουργική τους απλότητα, η οποία επιτρέπει την χρήση τους με απλό και ξεκάθαρο τρόπο.

Ο ελεγκτής PID αποτελείται από τρεις όρους : τον αναλογικό (proportional), τον όρο ολοκλήρωσης (integral) και τον όρο διαφόρισης (derivative) οι οποίοι ποικίλουν ανάλογα με την επιθυμητή απόκριση. Η βασική ιδέα πίσω από την υλοποίηση ενός ελεγκτή PID είναι η ανάγνωση της τιμής ενός αισθητήρα, έπειτα ο υπολογισμός της επιθυμητής εξόδου υπολογίζοντας τον αναλογικό, τον ολοκληρωτικό και το διαφορικό όρο για την απόκριση του συστήματος και τέλος η άθροιση των τριών όρων για τον υπολογισμό της τελικής εξόδου.



Εικόνα 45a. Η κίνηση του οχήματος πάνω στην γραμμή

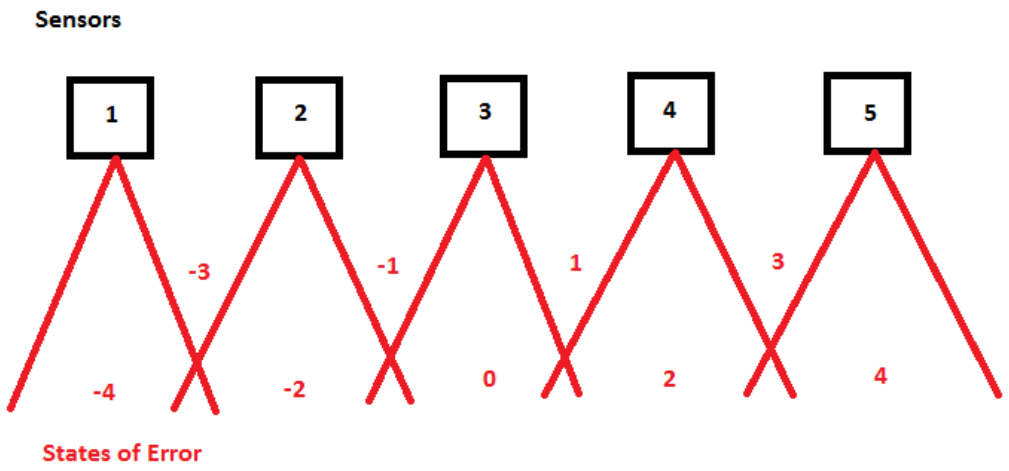
Υπάρχουν όμως και άλλοι παράμετροι που μπορούν να επηρεάσουν την απόδοση ενός συστήματος όπως η τριβή, περιβαλλοντικοί παράγοντες ακόμη και η ίδια η φύση του συστήματος.

Ο αναλογικός όρος εξαρτάται από την διαφορά του σημείου αναφοράς με την μεταβλητή διαδικασίας. Η μεταξύ τους διαφορά αναφέρεται ως σφάλμα. Το αναλογικό κέρδος (Kp) καθορίζει την αναλογία απόκρισης της εξόδου σε σχέση με το σφάλμα. Για παράδειγμα, εάν το σφάλμα έχει τιμή δέκα και το αναλογικό κέρδος έχει οχτώ, τότε η αναλογική απόκριση προς το σφάλμα θα ήταν ογδόντα. Γενικά αυξάνοντας το αναλογικό κέρδος αυξάνεται και η αντίδραση του συστήματος σε ενδεχόμενο σφάλμα. Παρόλα αυτά μια μεγάλη αύξηση του αναλογικού κέρδους θα έχει ως αποτέλεσμα την παρουσία ταλαντώσεων και ενδεχομένως η κατάσταση του συστήματος να γίνει ασταθής.

Ο όρος ολοκλήρωσης αθροίζει συνεχώς το σφάλμα που προκύπτει, με αποτέλεσμα ακόμη και μια μικρή αλλαγή στην τιμή του σφάλματος προκαλεί την σταδιακή αύξηση του. Η απόκριση ολοκλήρωσης συνεχίζει να αυξάνεται όσο η τιμή του σφάλματος είναι διάφορη του μηδενός, έτσι σκοπός του συντελεστή είναι η εξάλειψη του σφάλματος σταθερής κατάστασης (steady – state error). Στην περίπτωση όπου η τιμή του όρου ολοκλήρωσης είναι αρκετά υψηλή, μια απότομη αλλαγή στην κατάσταση του σφάλματος μπορεί να έχει ως αποτέλεσμα την υπερύψωση του συστήματος, καθώς αυξάνεται συνεχώς η τιμή του συντελεστή για να μηδενίσει το σφάλμα (integral windup).

Ο όρος διαφόρισης προκαλεί την μείωση της εξόδου στην περίπτωση που η μεταβλητή διαδικασίας αυξάνεται ραγδαία. Η παραγωγική απόκριση είναι αναλογική σε σχέση με τον ρυθμό της αλλαγής της μεταβλητής διαδικασίας. Η αύξηση του κέρδους διαφόρισης (Kd) έχει ως αντίκτυπο το σύστημα να αντιδρά με ισχυρότερη συμπεριφορά σε ενδεχόμενες αλλαγές του σφάλματος, αλλά και την αύξηση της ολικής αντίδρασης του συστήματος. Στα περισσότερα συστήματα το διαφορικό κέρδος είναι σχετικά μικρό, καθώς η αντίδραση του συντελεστή είναι εξαιρετικά ευαίσθητη. Εάν το σύστημα δέχεται πολλές διαταραχές ή ο ρυθμός του κλειστού βρόγχου δεν είναι ιδιαίτερα γρήγορος, το σύστημα κινδυνεύει να γίνει ασταθές[1].

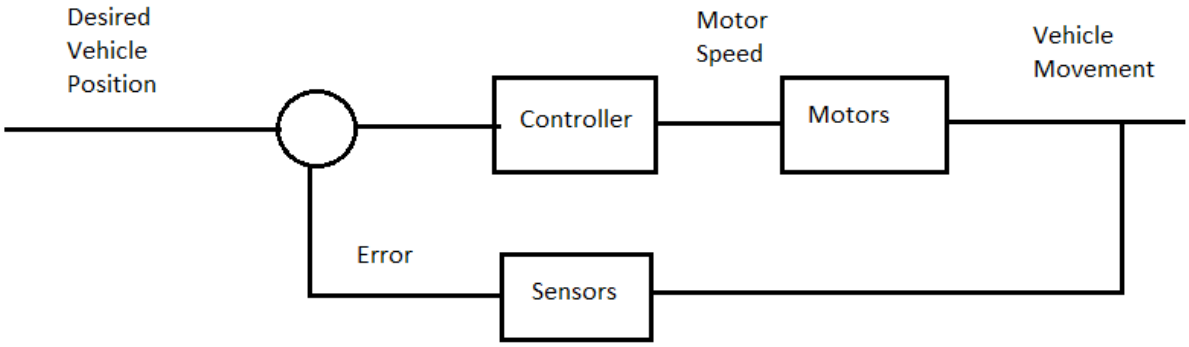
Στην παρούσα εργασία οι πειραματικές τιμές που έχουν τα κέρδη του αναλογικού, ολοκληρωτικού και διαφορικού όρου είναι 23, 17 και 7 αντίστοιχα. Η επιθυμητή θέση η οποία έχει οριστεί για το όχημα είναι όταν ο μεσαίος υπέρυθρος αισθητήρας βρίσκεται πάνω από την μαύρη γραμμή. Όταν το όχημα παρεκκλίνει από την επιθυμητή θέση δημιουργείται σφάλμα, το οποίο παίρνει αντίστοιχη τιμή ανάλογα με την απόσταση που απέχει το όχημα από το σημείο αναφοράς.



Εικόνα 45b. Καταστάσεις του σφάλματος

Ας υποθέσουμε πως η μεταβλητή του σφάλματος έχει λάβει την τιμή -4. Στη σχέση P = error η οποία παρατίθεται στην συνέχεια της εργασίας, στην μεταβλητή του αναλογικού όρου P ορίζεται η τιμή του υπάρχοντος σφάλματος, δηλαδή -4. Στη σχέση I = I + previous_error στη μεταβλητή του όρου ολοκλήρωσης I ορίζεται το συνεχόμενο άθροισμα του σφάλματος, επειδή όμως δεν υπάρχει προηγούμενο σφάλμα η τιμή του θα είναι -4. Στη σχέση D = error – previous_error στην μεταβλητή του διαφορικού όρου D ορίζεται η τιμή του υπάρχοντος σφάλματος μείον την τιμή του προηγούμενου σφάλματος, όμως σε αυτή την περίπτωση η τιμή του όρου είναι πάλι -4 μιας που δεν υπάρχει προηγούμενο σφάλμα. Εκτελώντας την μαθηματική πράξη του ελεγκτή προκύπτει η τιμή -168 , η οποία εφαρμόζεται στους κινητήρες με την σχέση left_motor_speed = initial_motor_speed – PID_value για τον αριστερό κινητήρα και right_motor_speed = initial_motor_speed + PID_value για τον δεξιό κινητήρα, με αποτέλεσμα το όχημα να

επιστρέφει στο ορισμένο σημείο αναφοράς το οποίο είναι η τιμή , ή με απλά λόγια η μετάβαση του κεντρικού υπέρυθρου αισθητήρα πάνω από την μαύρη γραμμή.



Εικόνα 46. Διάγραμμα ελέγχου του οχήματος

Στην περίπτωση η επόμενη τιμή του σφάλματος αλλάξει, για παράδειγμα γίνει δυο, στην τιμή του ολοκληρωτικού όρου θα προστεθεί το προηγούμενο σφάλμα, μείον τέσσερα, με το υπάρχον σφάλμα το οποίο έχει την τιμή δύο. Στην περίπτωση του διαφορικού όρου θα αφαιρεθεί η τιμή του υπάρχοντος σφάλματος με εκείνη του προηγούμενου, η τιμή που θα προκύψει θα είναι μείον δυο.

Μια απλή μέθοδος για ρύθμιση ενός pid ελεγκτή είναι κατά την διάρκεια της λειτουργίας του συστήματος. Αρχικά μηδενίζονται τα κέρδη Ki και Kd και στην συνέχεια ο χρήστης αυξάνει το κέρδος Kp μέχρι το σύστημα να φτάσει στην ταλάντωση. Έπειτα γίνεται η ρύθμιση του κέρδους

Ki ώστε το σύστημα να φτάσει στην ευστάθεια και στην συνέχεια ρυθμίζεται το κέρδος Kd για πιο γρήγορη απόκριση[2].

1URL: http://www.ni.com/white-paper/3782/en/ - 17.2.2018

2URL: https://www.elprocus.com/the-working-of-a-pid-controller/ - 17.2.2018

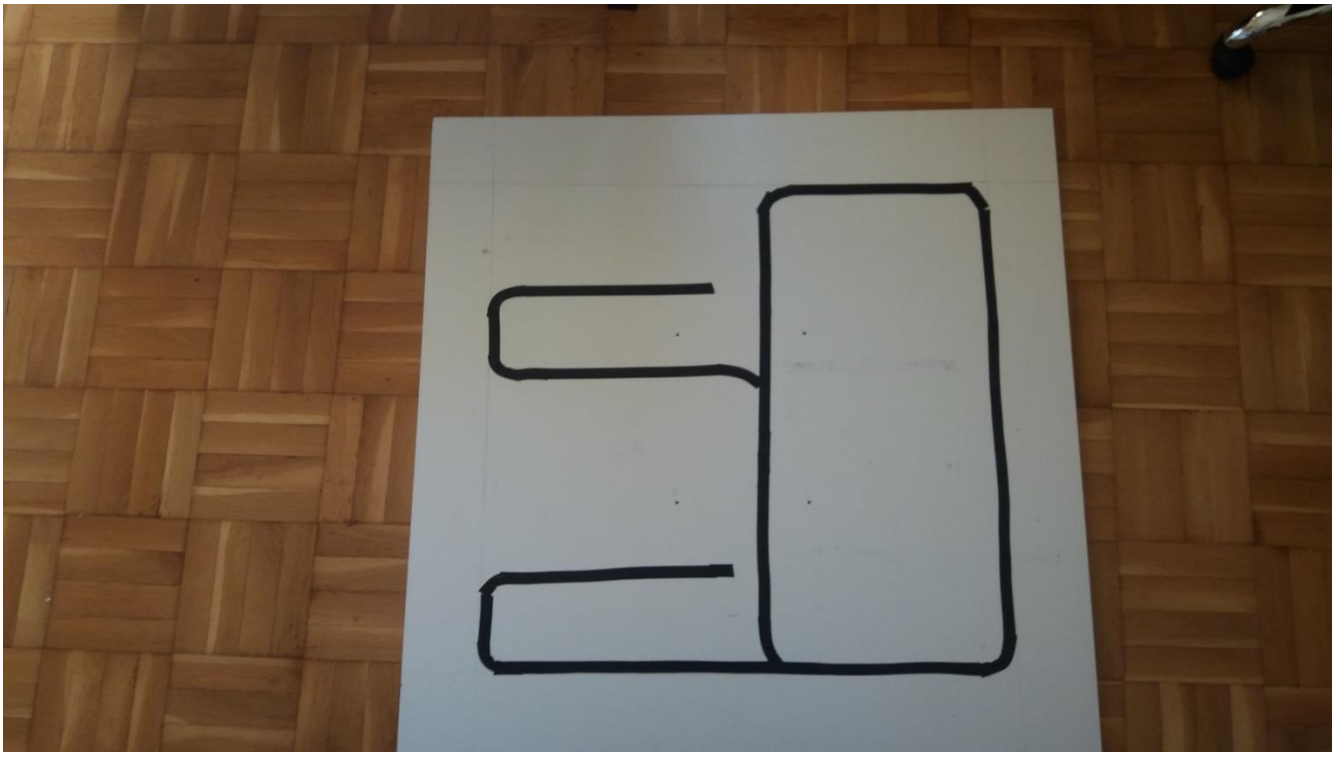
6.4. Υλοποίηση (Implementation)

Η πρώτη κίνηση ήταν η συγκέντρωση των υλικών για την κατασκευή του οχήματος. Μετά την ολοκλήρωση της κατασκευής έγινε η τοποθέτηση των αισθητήρων. Στην αρχή έγιναν πειράματα με αισθητήρες φωτός (light sensors) καθώς η αρχική σκέψη ήταν η οδήγηση του οχήματος με τον προγραμματισμό των αισθητήρων, ώστε να αντιλαμβάνονται το μαύρο της γραμμής και δίνουν ανάλογο σήμα στους κινητήρες για την υλοποίηση της κίνησης.

Η μέθοδος αυτή σύντομα απορρίφθηκε καθώς δεν ήταν αξιόπιστη η κίνηση του οχήματος και στην πορεία αντικαταστάθηκαν από υπέρυθρους αισθητήρες. Όταν η ακτινοβολία που εκπέμπει ο αισθητήρας περάσει πάνω από μαύρη επιφάνεια η επιστρεφόμενη ακτινοβολία είναι μικρότερη σε σχέση με την περίπτωση όπου ο αισθητήρας βρίσκεται πάνω από μια λευκή επιφάνεια. Με αυτή την προσθήκη η λήψη δεδομένων για τον προσανατολισμό του οχήματος καθώς και η πλοήγησή του είδαν τεράστια βελτίωση.

Στη συνέχεια δημιουργήθηκε το πρόγραμμα του ελεγκτή PID για την καλύτερη κίνηση του οχήματος. Η λειτουργία απαρτίζεται από τρία στάδια: την καταγραφή της θέσης του οχήματος πάνω στην γραμμή μέσω των ενδείξεων που δίνουν οι αισθητήρες, τον υπολογισμό της τιμής του ελεγκτή και την απόδοση της τιμής στους κινητήρες.

Έπειτα σχεδιάστηκε πάνω η πίστα πάνω στην οποία κινείται το όχημα, η οποία αποτελείται από μια κεντρική κυκλική διαδρομή και δυο παρακάμψεις. Το AGV κινείται στην κεντρική διαδρομή και όταν σε δυο συγκεκριμένα σημεία συναντήσει εμπόδιο σταματά και ακολουθεί την παράκαμψη για να μην σταματήσει η πορεία του. Η αρχική σκέψη είναι πως σε ένα χώρο εργασίας η παραγωγική διαδικασία πρέπει να έχει μια συγκεκριμένη ροή, οπότε ένα αυτοοδηγούμενο όχημα πρέπει να είναι σε θέση να αλλάζει πορεία όταν το απαιτεί η παραγωγική διαδικασία ή για την αποφυγή ενός εργατικού ατυχήματος. Σε ένα υποθετικό σενάριο το όχημα θα μπορούσε να αλλάζει πορεία λόγω μιας εργασίας σε οποιοδήποτε σημείο της γραμμής παραγωγής. Στο πρώιμο στάδιο της εφαρμογής η αρχική ιδέα ήταν το AGV να κάνει στάσεις σε καθορισμένα σημεία στα οποία θα υπήρχαν κομμάτια χαρτιού με χρώματα όπως κόκκινο, πράσινο, μπλε κ.α με την χρήση ενός αισθητήρα χρώματος (color sensor). Μετά από πολλές δοκιμές καταλήξαμε στο συμπέρασμα πως ο αισθητήρας δεν μπορούσε να αναγνωρίσει τα χρώματα λόγω της απόστασης που χώριζε εκείνο και την επιφάνεια της πίστας, όπως και κατα διαστήματα η έλλειψη φωτισμού στον χώρο. Μπορεί να φανταστεί κανείς πως σε πραγματικές συνθήκες μια παρόμοια εφαρμογή θα δημιουργούσε προβλήματα κατά την διάρκεια της νύχτας ή σε μέρη με λιγοστό φωτισμό. Ο υπερηχητικός αισθητήρας έδωσε αξιοπιστία στην εφαρμογή καθώς σε καθορισμένη απόσταση σταματά ακαριαία, δημιουργώντας με αυτόν τον τρόπο ασφαλείς εργασιακές συνθήκες και ακρίβεια στις βιομηχανικές εργασίες.



Εικόνα 47. Η επιφάνεια εργασίας και η πορεία που ακολουθεί το Robot

6.5. Κίνηση Ενός Οχήματος Που Ακολουθεί Γραμμή (Movement of a Line Follower AGV)

Παρακάτω παρατίθεται το πρόγραμμα με το οποίο γίνεται εφικτή η λειτουργία του οχήματος καθώς και ανάλυση των στοιχείων του.

```
#include <NewPing.h>

const float Kp = 23, Ki = 17 , Kd = 7 ;

float error = 0, P = 0, I = 0, D = 0, PID_value = 0;

float previous_error = 0, previous_I = 0;

int initial_motor_speed = 100;
```

```
unsigned long  le = 0;

int pingPin = 12, inPin = 13;


NewPing sonar (pingPin, inPin, 5);


void setup()

{

  pinMode(6, OUTPUT); //PWM Pin 1

  pinMode(7, OUTPUT); //PWM Pin 2

  pinMode(2, OUTPUT); //Left Motor Pin 1

  pinMode(3, OUTPUT); //Left Motor Pin 2

  pinMode(4, OUTPUT); //Right Motor Pin 1

  pinMode(5, OUTPUT); //Right Motor Pin 2

  pinMode(8, INPUT); //Far Left IR Sensor

  pinMode(9, INPUT); //Left IR Sensor

  pinMode(10, INPUT); //Central IR Sensor

  pinMode(11, INPUT); //Right IR Sensor

  pinMode(22, INPUT); //Far Right IR Sensor

  pinMode(pingPin, OUTPUT); //Trigger Pin

  pinMode(inPin, INPUT); //Echo Pin
```

```
  Serial.begin(9600); //Enable Serial Communications
}


void loop()
{
  read_sensor_values();
  calculate_pid();
  motor_control();
  le = micros();
  while (micros() - le < 15)
  {
    Serial.print("Ping: ");
    Serial.print(sonar.ping_cm());
    Serial.println("cm");
  }
  if (sonar.ping_cm() == 5)
  {
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
```

```
    digitalWrite(5, LOW);
    delay(1000);
    analogWrite(6, 170);
    analogWrite(7, 170);
    digitalWrite(2, HIGH);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
    delay(400);
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    delay(500);
  }
}
void read_sensor_values()
{
  if ((digitalRead(8)) && (!digitalRead(9)) && (!digitalRead(10)) && (!digitalRead(11)) &&
(!digitalRead(22)))
```

```
  error = 4;
 else  if ((digitalRead(8)) && (digitalRead(9)) && (!digitalRead(10)) && (!digitalRead(11)) && (!digitalRead(22)))
  error = 3;
 else if ((!digitalRead(8)) && (digitalRead(9)) && (!digitalRead(10)) && (!digitalRead(11)) && (!digitalRead(22)))
  error = 2;
 else if ((!digitalRead(8)) && (digitalRead(9)) && (digitalRead(10)) && (!digitalRead(11)) && (!digitalRead(22)))
  error = 1;
 else if ((!digitalRead(8)) && (!digitalRead(9)) && (digitalRead(10)) && (!digitalRead(11)) && (!digitalRead(22)))
  error = 0;
 else if ((!digitalRead(8)) && (!digitalRead(9)) && (digitalRead(10)) && (digitalRead(11)) && (!digitalRead(22)))
  error = -1;
 else if ((!digitalRead(8)) && (!digitalRead(9)) && (!digitalRead(10)) && (digitalRead(11)) && (!digitalRead(22)))
  error = -2;
 else if ((!digitalRead(8)) && (!digitalRead(9)) && (!digitalRead(10)) && (digitalRead(11)) && (digitalRead(22)))
```

```
    error = -3;
  else if ((!digitalRead(8)) && (!digitalRead(9)) && (!digitalRead(10)) && (!digitalRead(11))
&& (digitalRead(22)))
    error = -4;
}
void calculate_pid()
{
  P = error;
  I = I + previous_I;
  D = error – previous_error;

  PID_value = (Kp * P) + (Ki * I) + (Kd * D);

  previous_I = I;
  previous_error = error;
}
void motor_control()
{
  // Calculating the effective motor speed:
  int left_motor_speed = initial_motor_speed - PID_value;
```

```
  int right_motor_speed = initial_motor_speed + PID_value;


  // The motor speed should not exceed the max PWM value
  constrain(left_motor_speed, 0, 255);
  constrain(right_motor_speed, 0, 255);


  analogWrite(6, initial_motor_speed - PID_value); //Left Motor Speed
  analogWrite(7, initial_motor_speed + PID_value); //Right Motor Speed


  digitalWrite(2, LOW);
  digitalWrite(3, HIGH);
  digitalWrite(4, LOW);
  digitalWrite(5, HIGH);
}
```

Ανάλυση του κώδικα:

```
#include <NewPing.h>
```

Η βιβλιοθήκη NewPing.h περιέχει συναρτήσεις οι οποίες χρησιμοποιούνται για την καταγραφή της παλμοσειράς που εκπέμπει ο υπερηχητικός αισθητήρας και την μετατροπή της σε microseconds, cm κ.α.

const float Kp = 23, Ki = 17 , Kd = 7 ;

float error = 0, P = 0, I = 0, D = 0, PID_value = 0;

float previous_error = 0, previous_I = 0;

Πρόκειται για τις μεταβλητές Kp, Ki και Kd του ελεγκτή PID καθώς και οι μεταβλητές με τις οποίες γίνεται ο υπολογισμός της τιμής του ελεγκτή.

NewPing sonar (pingPin, inPin, 5);

Δηλώνεται το αντικείμενο sonar της κλάσης NewPing το οποίο έχει ως παραμέτρους την μεταβλητή η οποία αντιστοιχεί στον ακροδέκτη Trig, μια μεταβλητή η οποία αντιστοιχεί στον ακροδέκτη Echo και την απόσταση μέχρι την οποία εκπέμπει παλμοσειρά για την αναγνώριση ενός εμποδίου σε εκατοστά, στην προκειμένη περίπτωση πέντε.

void read_sensor_values()

Η συνάρτηση read_sensor_values() λαμβάνει τα δεδομένα από τους υπέρυθρους αισθητήρες και ανάλογα με την τιμή του εκάστοτε αισθητήρα καθορίζει το σφάλμα, δηλαδή κατά πόσο η θέση του οχήματος απέχει από την επιθυμητή θέση, η οποία γίνεται αντιληπτή από τον μικροελεγκτή όταν η τιμή της μεταβλητής error είναι μηδέν.

void calculate_pid()

Η συνάρτηση calculate_pid() υπολογίζει την τιμή του ελεγκτή PID.

void motor_control()

Η συνάρτηση motor_control() επιβάλλει την τιμή του ελεγκτή PID στους κινητήρες για την εκτέλεση της επιθυμητής κίνησης. Οι μεταβλητές left_motor_speed και right_motor_speed δηλώνουν την ταχύτητα στον αριστερό και τον δεξιό τροχό αντίστοιχα. Η συνάρτηση constrain οριοθετεί την τιμή των μεταβλητών μεταξύ 0 και 255, τα όρια τάσης τα οποία μπορεί να επιβάλλει ο μικροελεγκτής στους ακροδέκτες 6 και 7 για τον έλεγχο των κινητήρων. Οι ακροδέκτες 2,3,4, και 5 αντιπροσωπεύουν τους ακροδέκτες των κινητήρων.

```
void loop()
{
 read_sensor_values();
 calculate_pid();
 motor_control();
 le = micros();
 while (micros() - le < 15)
 {
  Serial.print("Ping: ");
  Serial.print(sonar.ping_cm());
  Serial.println("cm");
 }
 if (sonar.ping_cm() == 5)
 {
```

```
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    delay(1000);
    analogWrite(6, 170);
    analogWrite(7, 170);
    digitalWrite(2, HIGH);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, HIGH);
    delay(400);
    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
    delay(500);
  }
}
```

Στην συνάρτηση loop() λαμβάνει χώρα η λειτουργία του προγράμματος. Επικαλούνται οι συναρτήσεις read_sensor_value, calculate_pid και motor_control για την οδήγηση του οχήματος πάνω στην γραμμή. Έπειτα στην μεταβλητή le καταχωρείται η συναρτηση micros(), όπου μετρά microseconds κατά την διάρκεια του προγράμματος. Στην συνθήκη while() εκτελείται η συνάρτηση ping_cm() του αντικειμένου sonar, όπου επιστρέφει το μήκος της παλμοσειράς σε εκατοστά. Η συνθήκη επαναλαμβάνεται μέχρι η διαφορά ανάμεσα στην συνάρτηση micros() και στην μεταβλητή le να ξεπεράσει τα 15 microseconds. Στην συνέχεια εάν ο αισθητήρας λάβει ένδειξη πως σε απόσταση πέντε εκατοστών από το όχημα υπάρχει εμπόδιο, το AGV σταματά, κάνει στροφή προς τα δεξιά για 400 milliseconds και έπειτα σταματά ξανά. Μόλις τελειώσει η συνθήκη if ξεκινά πάλι η διαδικασία από την αρχή με την λειτουργία των τριών βασικών συναρτήσεων που οδηγούν το όχημα πάνω στην γραμμή.

7. Βιβλιογραφία

Anderson M & S. L. Anderson (ed.) (2011). *Machine Ethics*. Cambridge University Press, p. 268.

Asimov, I. (1942). *Runaround. Robot series,* Astounding Science Fiction, Street & Smith.

Asimov I. (1950). *I, Robot.* Gnome Press

Breazeal C. L.. (2002.) *Designing Social Robots*. MIT Press.

Cheney M. (1981). *Tesla: Man Out of Time,* Touchstone.

Cork S. J. (2011). From Prometheus to Pistorious: a genealogy of physical ability. A thesis submitted to the Department of Sociology In conformity with the requirements for the degree of Masters of Arts, Queen's University Kingston, Ontario, Canada

Čapek, K. (1920). *Rossum's Universal Robot (R.U.R).* Translated by Paul Selver and Nigel Playfair Mineola, New York: Dover Publications.

Gilbert W. S.. (1886). *Pygmalion and Galatea: An original mythological comedy in three acts*. Chicago Publishing Company.

Grafton, A. (1999). *Natural Particulars: Nature and the Disciplines in Renaissance Europe*. MIT Press.

Etzioni O & Weld D. (1994). *The First Law of Robotics: (a call to arms*), AAAI p. 92Technical Report SS-94-03, Spring, p.17.

Halperin , D. Kavraki , Latombe, J. (1997). *Robotics CRC Handbook of Discrete and Computational Geometry*, J.E. Goodman and J. O'Rourke (eds.), CRC Press, Boca Raton, FL, Chapter 41, p. 755-778.

Heath F. G. Origins of the binary code, Scientific American JSTOR, Vol. 227, p. 76-83.

Hyman A. (1985). Charles Babbage: Pioneer of the Computer, Princeton University Press.

Idel M. (1990). Golem: Jewish Magical and Mystical Traditions on the Artificial Anthropoid. Albany, New York: State University of New York Press.

Kurfess Th. R. (2004). Robotics and Automation Handbook, CRC Press.

Landels J. G. (1979). Water-Clocks and Time Measurement in Classical Antiquity, Endeavour, Vol. 3, p. 32-37.

Lang Fritz. (2003). Interviews ed. by B. K. Grant, University Press of Mississippi.

Nocks L. (2007). The Robot: The Life Story of a Technology, Greenwood Press.

North E. H (1951). The Day the Earth stood still . The movie screenplay.

Schwartzman R. (1999) Engenderneered Machines in Science Fiction. Studies in Popular

Culture. Vol. 22, No. 1 p. 75-87.

Shelley, M. (1818) . Frankenstein or, The Modern Prometheus . Lackington, Hughes, Harding, Mavor & Jones.

Wadhwa A.S. (2012). Role of it in manufacturing sector. *Proceedings of the National Conference on Trends and Advances in Mechanical Engineering*, YMCA University of

Science & Technology, Faridabad, Haryana, Oct 19-20, p. 550-554.

URL:http://americansocietyofarmscollectors.org/wpcontent/uploads/2013/05/B039_Lewis.pdf 06.08.2017

URL: http://www.etymonline.com/index.php?term=robot - 09.08.2017

URL:www.npr.org/2011/04/22/135634400/science-diction-the-origin-of-the-word-robot – 09.08.2017

URL: https://www.biography.com/people/isaac-asimov-9190737 – 12.8.2017

Noble. J (2009). Programming Interactivity: A Designer's Guide to Processing, Arduino and Openframeworks. O'Reily Media, Inc,p 92

URL: http://smithsonianchips.si.edu/augarten/p38.htm- 14.8.2017

URL: http://www.arduino-tutorials.com/arduino-pwm/- 16.8.2017

URL: https://www.tutorialspoint.com/arduino/arduino_board_description.htm- 18.8.2017

URL: https://www.arduino.cc/en/Main/Software- 21.8.2017

URL: https://www.arduino.cc/en/hacking/libraries- 21.8.2017

URL: https://www.computerhope.com/unix/uchmod.htm - 22.8.2017

URL: http://www.tldp.org/LDP/Linux-Filesystem-Hierarchy/html/dev.html - 24.8.2017

URL: https://processing.org/overview/ - 16.11.2017

Kernighan. W. B, Dennis M. R (1978). The C Programming Language. Prentice Hall Software Series.

Noble. J (2009).Programming Interactivity: A Designer's Guide to Processing, Arduino and Openframeworks. O'Reily Media, Inc, p. 105.

Noble. J (2009).Programming Interactivity: A Designer's Guide to Processing, Arduino and Openframeworks. O'Reily Media, Inc,p 106

URL: http://searchmicroservices.techtarget.com/definition/data-type - 17.9.2017

URL: https://www.arduino.cc/reference/en/language/structure/control-structure/for/-17.9.2017

URL: https://www.arduino.cc/reference/en/language/structure/control-structure/while/ - 17.9.2017

URL: https://www.arduino.cc/reference/en/language/structure/control-structure/dowhile/ - 20.9.2017

URL: https://www.arduino.cc/reference/en/language/structure/control-structure/continue/- 20.9.2017

URL: https://www.arduino.cc/en/Reference.Break - 20.9.2017

URL: https://www.arduino.cc/reference/en/language/variables/data-types/array/

- 23.9.2017

URL: https://www.arduino.cc/reference/en/language/variables/data-types/string/ -24.9.2017

URL: https://www.arduino.cc/en/Reference/FunctionDeclaration -28.9.2017

URL: http://www.ros.org/about-ros/ -28.9.2017

Lentin J. (2015). Mastering ROS for Robotics Programming. Packt Publishing Ltd. p.3.

URL: https://help.ubuntu.com/community/Repositories/Ubuntu -2.10.2017

URL: http://wiki.ros.org/ROS/Tutorials/rosdep -2.10.2017

URL: http://wiki.ros.org/ROS/EnvironmentVariables -5.10.2017

URL : http://wiki.ros.org/catkin/conceptual_overview -7.10.2017

URL: https://www.cs.virginia.edu/~dww4s/articles/build_systems.html 10.10.2017

URL: http://wiki.ros.org/catkin/workspaces#Catkin_Workspaces -10.10.2017

URL: http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment -15.10.2017

URL: https://judithcurry.com/2012/12/22/the-goldilocks-principle/ - 17.10.2017

URL: http://wiki.ros.org/Packages -25.10.2017

URL: http://wiki.ros.org/Metapackages- 27.10.2017

URL: http://wiki.ros.org/Manifest -28.10.2017

URL: http://wiki.ros.org/msg -30.10.2017

URL: http://wiki.ros.org/srv -5.11.2017

URL: http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem -7.11.2017

URL: http://wiki.ros.org/ROS/Tutorials/CreatingPackage -8.11.2017

URL: http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv -10.11.2017

URL: http://wiki.ros.org/Master -13.11.2017

URL: http://wiki.ros.org/Parameter%20Server -15.11.2017

URL: http://wiki.ros.org/Messages -17.11.2017

URL: http://wiki.ros.org/Topics -19.11.2017

URL: http://wiki.ros.org/Services -20.11.2017

URL: http://wiki.ros.org/Bags -25.11.2017

Lentin J. (2015). Mastering ROS for Robotics Programming. Packt Publishing Ltd. p.19 – 22.

URL: http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes -27.11.2017

Lentin J. (2015).Mastering ROS for Robotics Programming. Packt Publishing Ltd. p. 17.

URL: http://wiki.ros.org/ROS/Tutorials/UnderstandingTopics 28.11.2017

Lentin J. (2015).Mastering ROS for Robotics Programming. Packt Publishing Ltd. p. 18.

Lentin J. (2015).Mastering ROS for Robotics Programming. Packt Publishing Ltd. p. 21.

URL:http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29

URL: http://wiki.ros.org/ROS/Tutorials/WritingServiceClient%28c%2B%2B%29 -30.11.2017

URL: http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup -2.12.2017

URL: http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World -5.12.2017

URL: http://wiki.ros.org/rosserial_arduino/Tutorials/Blink - 7.12.2017

URL: http://hlektrologia.gr/%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CE%BF-%CE%B5%CE%BB%CE%B5%CE%B3%CE%BA%CF%84%CE%AE%CF%82-pid/- 10.12.2017

URL: http://ieeexplore.ieee.org/abstract/document/5380235/ -15.12.2017

URL: https://circuitdigest.com/electronic-circuits/ir-sensor-circuit-diagram -20.12.2017

URL; http://www.ti.com/lit/ds/symlink/l293.pdf - 24.12.2017

URL: http://www.electricaleasy.com/2014/07/characteristics-of-dc-motors.html -4.1.2018

URL: http://www.ni.com/white-paper/3782/en/ - 17.2.2018

URL: https://www.elprocus.com/the-working-of-a-pid-controller/ - 17.2.2018