



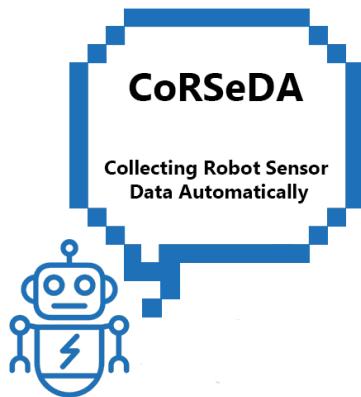
ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εργαστήριο Επεξεργασίας Πληροφορία και Υπολογισμών

Αυτόματη παραγωγή διεπαφής υψηλού επιπέδου για
συλλογή δεδομένων αισθητήρων ρομπότ αξιοποιώντας την
πλατφόρμα R4A



Διπλωματική Εργασία της
Ντζιώνη Δήμητρας
Α.Ε.Μ: 8209

Επιβλέποντες:
Επίκουρος καθηγητής: Ανδρέας Λ. Συμεωνίδης
Υποψήφιος διδάκτωρ: Ζολώτας Χριστόφορος

ΑΥΤΟΜΑΤΗ ΠΑΡΑΓΩΓΗ ΔΙΕΠΑΦΗΣ ΥΨΗΛΟΥ ΕΠΙΠΕΔΟΥ ΓΙΑ ΣΥΛΛΟΓΗ ΔΕΔΟΜΕΝΩΝ ΑΙΣΘΗΤΗΡΩΝ ΡΟΜΠΟΤ
ΑΞΙΟΠΟΙΟΝΤΑΣ ΤΗΝ ΠΛΑΤΦΟΡΜΑ R4A

Ευχαριστίες

Θα ήθελα να ευχαριστήσω αρχικά τον κ. Ανδρέα Συμεωνίδη, για την εμπιστοσύνη που έδειξε στο πρόσωπο μου, αναθέτοντας μου την παρούσα διπλωματική εργασία, καθώς και για την ευκαιρία που μου έδωσε να ανακαλύψω νέες ενδιαφέρουσες πτυχές της επιστήμης λογισμικού. Έπειτα, θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα Χριστόφορο Ζολώτα, για την άριστη συνεργασία, την υπομονή και την καθοδήγησή του, καθώς ακόμη και για τις παρατηρήσεις και τη βοήθεια που μου προσέφερε. Τέλος, θα ήθελα να ευχαριστήσω όλα τα μέλη του Εργαστηρίου Επεξεργασίας Πληροφορίας και Υπολογισμών για την οποιαδήποτε βοήθεια μου προσέφεραν κατά τη διάρκεια εκπόνησης της διπλωματικής αυτής.

Πέραν των επιστημονικών συνεργατών, θα ήθελα να εκφράσω την ευγνωμοσύνη μου σε όλους εκείνους τους ανθρώπους, την οικογένεια και τους φίλους μου, που στάθηκαν στο πλευρό μου και με στήριζαν με οποιονδήποτε τρόπο καθ' όλη τη διάρκεια των φοιτητικών μου χρόνων και ιδιαίτερα κατά την περίοδο εκπόνησης της παρούσας διπλωματικής.

Αφιερωμένη σε όσους ήταν πάντα εκεί

ΑΥΤΟΜΑΤΗ ΠΑΡΑΓΩΓΗ ΔΙΕΠΑΦΗΣ ΥΨΗΛΟΥ ΕΠΙΠΕΔΟΥ ΓΙΑ ΣΥΛΛΟΓΗ ΔΕΔΟΜΕΝΩΝ ΑΙΣΘΗΤΗΡΩΝ ΡΟΜΠΟΤ
ΑΞΙΟΠΟΙΟΝΤΑΣ ΤΗΝ ΠΛΑΤΦΟΡΜΑ R4A

Περίληψη

Στην επιστήμη του λογισμικού, ο όρος αυτόματος προγραμματισμός προσδιορίζει ένα μηχανισμό ο οποίος δημιουργεί ένα πρόγραμμα που επιτρέπει στους επιστήμονες να συντάξουν κώδικα σε υψηλότερο επίπεδο αφαίρεσης. Σήμερα, οι ρομποτικές εφαρμογές σε εταιρικό και οικιακό περιβάλλον κερδίζουν όλο και μεγαλύτερο έδαφος, αυξάνοντας συνεχώς την ανάγκη για αυτόματη παραγωγή λογισμικού χωρίς σφάλματα. Όπως είναι γνωστό, τα ρομπότ φέρουν ένα πλήθος αισθητήρων, οι οποίοι παίζουν καθοριστικό ρόλο στη λειτουργία τους και στην επίτευξη ορισμένων εργασιών. Για το λόγο αυτό, πρέπει συχνά να ελέγχονται τα δεδομένα που παράγονται, ώστε να σχεδιαστούν περαιτέρω συστήματα λογισμικού. Η συγκεκριμένη διπλωματική εργασία φιλοδοξεί να θέσει τα πρώτα βήματα προς την κατεύθυνση της αυτοματοποίησης της διαδικασίας ανάπτυξης διεπαφών για συλλογή δεδομένων από αισθητήρες ρομπότ.

Στα πλαίσια αυτής της προσπάθειας, στην παρούσα διπλωματική εργασία, αξιοποιείται η μηχανική οδηγούμενη από μοντέλα, MDE (*Model Driven Engineering*). Συγκριμένα, αφού οριστεί ένα αφαιρετικό μοντέλο, πραγματοποιείται μια σειρά από μετασχηματισμούς, με τελικό αποτέλεσμα μια πλήρως λειτουργική εφαρμογή. Με αυτό τον τρόπο, επιταχύνεται διαδικασία ανάπτυξης λογισμικού και παράγεται λογισμικό με μεγαλύτερη αξιοπιστία.

Η παρούσα διπλωματικής εργασία, υλοποιεί το σύστημα CoRSeDA (*Collecting Robot Sensor Data Automatically*) στο οποίο ο χρήστης, μέσω ενός φιλικού γραφικού περιβάλλοντος αλληλοεπιδρά και ορίζει παραμέτρους για τους αισθητήρες ενός ρομπότ. Με βάση τους αισθητήρες και τις παραμέτρους που θέτει, δημιουργείται εκτελέσιμος κώδικας, βασισμένος στην πλατφόρμα R4A, για τη συλλογή δεδομένων από το ρομπότ και παράλληλα παράγεται πλήρως λειτουργική διεπαφή που παρέχει πληροφορίες ολόκληρου του συστήματος. Τις πληροφορίες αυτές μπορεί να τις δει, μαζί με οποιαδήποτε πληροφορία παράγεται, σε μία web εφαρμογή που έχει δημιουργηθεί στα πλαίσια της συγκεκριμένης εργασίας ακριβώς για το σκοπό αυτό.

Για την δοκιμή και αξιολόγηση αυτού του συστήματος, εκτελέστηκαν πειράματα στο πραγματικό ρομπότ NAO, ένα αυτόνομο, προγραμματιζόμενο ανθρωποειδές που αναπτύχθηκε από την Aldebaran Robotics.

ΑΥΤΟΜΑΤΗ ΠΑΡΑΓΩΓΗ ΔΙΕΠΑΦΗΣ ΥΨΗΛΟΥ ΕΠΙΠΕΔΟΥ ΓΙΑ ΣΥΛΛΟΓΗ ΔΕΔΟΜΕΝΩΝ ΑΙΣΘΗΤΗΡΩΝ ΡΟΜΠΟΤ
ΑΞΙΟΠΟΙΟΝΤΑΣ ΤΗΝ ΠΛΑΤΦΟΡΜΑ R4A

Abstract

“Automatic generation of high-level interfaces to collect robot sensor data using the R4A platform”

In software engineering, the term automatic programming describes a mechanism that creates a program which in turn allows scientists to code at a higher level of abstraction. Nowadays, robot applications, both in business and home environments are gaining traction, increasing the need for automatic software generation without errors. It is well known that robots are equipped with multitude of sensors, which play a key role in their operation and in accomplishing certain tasks. For this reason, it is often necessary to control the produced data, in order to build software systems. This diploma thesis aspires to take the first steps towards the process of automating the development of ready-to-run interfaces to collect robot sensor data.

Towards this direction, *MDE (Model Driven Engineering)* is employed. More specifically, once a subtractive model has been defined, a series of transformations takes place, resulting in a fully functional system. This way, the software development process is accelerated and software is produced with greater reliability.

Within the context of this diploma thesis, we have designed and implemented CoRSeDA (*Collecting Robot Sensor Data Automatically*), a system where the user interacts through a friendly graphical user interface and defines the features for the desired sensors of a robot. Based on the sensors and their parameters, the system generates automatically executable code, based on the R4A platform, to collect data from the specific robot. At the same time, a fully functional interface is generated providing information for the whole system. The data of the system and its sensors are displayed, along with any other information generated, in a web application created for that purpose.

To test and evaluate this system, experiments were performed on the robot NAO, an autonomous, programmable humanoid robot developed by Aldebaran Robotics.

ΑΥΤΟΜΑΤΗ ΠΑΡΑΓΩΓΗ ΔΙΕΠΑΦΗΣ ΥΨΗΛΟΥ ΕΠΙΠΕΔΟΥ ΓΙΑ ΣΥΛΛΟΓΗ ΔΕΔΟΜΕΝΩΝ ΑΙΣΘΗΤΗΡΩΝ ΡΟΜΠΟΤ
ΑΞΙΟΠΟΙΟΝΤΑΣ ΤΗΝ ΠΛΑΤΦΟΡΜΑ R4A

Περιεχόμενα

Ευχαριστίες	2
Περίληψη	4
Abstract.....	6
Λίστα Σχημάτων	12
Λίστα Αλγορίθμων.....	13
Λίστα Πινάκων.....	13
Λεξικό Όρων	18
Αρκτικόλεξα.....	18
1. Εισαγωγή.....	19
1.1 Κίνητρο	19
1.2 Περιγραφή Προβλήματος	20
1.3 Στόχοι της διπλωματικής εργασίας.....	22
1.4 Διάρθρωση του εγγράφου	23
2. Υπόβαθρο	24
2.1 Θεωρητικό Υπόβαθρο	24
2.1.1 Τεχνολογία Λογισμικού	24
2.1.2 ROS (Robot Operating System)	25
2.1.3 MDE (Model Driven Engineering).....	25
2.1.6 REST (Represantational State Tranfer)	32
2.2 Τεχνολογικό Υπόβαθρο	34
2.2.2 Acceleo Template Language	34
2.2.3 EMF (Ecore Modeling Framework).....	35
2.2.4 Ecore Meta-Model	36
2.2.5 Redis Database	37
2.2.6 Flask Framework	39

2.2.7 R4A Platform	39
3 Σχετική Βιβλιογραφία (<i>State of the art</i>)	41
3.1 SmartSoft	41
3.2 Proteus	42
3.3 MathWorks Simulink®	43
3.4 DEPOSIT (Data collecTion POlicies for Sensing InfrasTructures)	44
3.5 TargetLink	45
4 Μεθοδολογία	46
4.1 Συναρτήσεις της R4A Platform	47
4.1.1 Συναρτήσεις κατηγορίας απόστασης	47
4.1.2 Συναρτήσεις κατηγορίας ήχου	48
4.1.3 Συναρτήσεις κατηγορίας ταχύτητας	48
4.1.4 Συναρτήσεις κατηγορίας θέσης	49
4.1.5 Συναρτήσεις κατηγορίας πίεσης	50
4.1.6 Συναρτήσεις κατηγορίας όρασης	51
4.1.7 Συναρτήσεις κατηγορίας ηλεκτρισμού	52
4.1.8 Συναρτήσεις γενικού περιεχομένου	53
4.2 Ορισμός του Μετα-μοντέλου	53
4.2.1 SensorSystem	54
4.2.2 Subsystem	55
4.2.3 DistanceSystem	57
4.2.4 ElectricSystem	58
4.2.5 PositionSystem	59
4.2.6 PressureSystem	59
4.2.7 SpeedSystem	60
4.2.6 VisionSystem	61

4.2.7 AcousticSystem	62
4.2.8 Sensor	65
4.2.9 AcousticSensor	66
4.2.10 ElectricSensor.....	67
4.2.11 VisionSensor.....	69
4.2.12 DistanceSensor.....	70
4.2.12 SpeedSensor.....	73
4.2.14 PressureSensor.....	74
4.2.15 PressureSensorHead.....	75
4.2.16 PressureSensorHand	76
4.2.17 PressureSensorFoot.....	77
4.2.18 PositionSensor.....	79
4.2.19 PostureSensor	79
4.2.20 TframeSensor	81
4.2.21 PositionSensorHead	83
4.2.22 PositionSensorHand	84
4.2.23 PositionSensorFoot	85
4.2.24 PositionSensorRobot	86
4.3 Eclipse – Sirius User Interface.....	87
4.3.1 Διάγραμμα DevelopSystem	88
4.3.2 Κανόνες επικύρωσης διαγράμματος	95
4.4 Acceleo Project	97
4.5 Redis.....	102
4.5.1 Συναρτήσεις Αποθήκευσης	102
4.5.2 Συναρτήσεις Ανάκτησης	104
4.6 Flask REST API	106

4.6.1 Endpoints γενικού περιεχομένου του συστήματος	107
4.6.2 Endpoints μεμονωμένων αισθητήρων.....	111
4.7 Web Application	138
5 Παράδειγμα Χρήσης Συστήματος.....	142
5.1 Παρουσίαση επιλεγμένων στοιχείων	142
5.2 Αποτελέσματα	143
6 Συμπεράσματα & Μελλοντικές Επεκτάσεις.....	146
6.1 Συμπεράσματα	146
6.2 Μελλοντικές Επεκτάσεις	146
7 Βιβλιογραφία	148

Λίστα Σχημάτων

Σχήμα 1: Ραγδαία αύξηση του πλήθους ρομπότ βιομηχανικών εφαρμογών	20
Σχήμα 2: Σύγκριση της ζήτησης μεταξύ των services των ρομπότ από το 2016 έως σήμερα και πρόβλεψη για τα επόμενα έτη, αναδεικνύοντας την ανάγκη για services αναφορικά με την αποθήκευση και τη διαχείριση αποθεμάτων.....	21
Σχήμα 3: Σύγκριση της ζήτησης μεταξύ των services των ρομπότ από το 2016 έως σήμερα και πρόβλεψη για τα επόμενα έτη, αναδεικνύοντας την ανάγκη για services αναφορικά με τους εξωτερικούς σκελετούς	21
Σχήμα 4: Σύγκριση της ζήτησης μεταξύ των services των ρομπότ από το 2016 έως σήμερα και πρόβλεψη για τα επόμενα έτη, αναδεικνύοντας την ανάγκη για services αναφορικά με οικιακές εφαρμογές	22
Σχήμα 5: Βασικές αρχές αντικειμενοστραφούς και μοντελοκεντρικής λογικής.....	26
Σχήμα 6: Η αρχιτεκτονική MOF 4 επιπέδων	28
Σχήμα 7: Μετασχηματισμός M2M	29
Σχήμα 8: Μετασχηματισμός M2T.....	29
Σχήμα 9: Θετικές πτυχές της MDE, σε έρευνα αναφορικά με την Ιταλική Βιομηχανία.....	31
Σχήμα 10: Δυσκολίες και προβλήματα που περιορίζουν την υιοθέτηση της μεθόδου MDE	32
Σχήμα 11: Παρουσίαση .mtl αρχείου	34
Σχήμα 12: EMF και ενοποιήση των Java, XML, UML	36
Σχήμα 13: Απλοποιημένο παράδειγμα Ecore meta-model	37
Σχήμα 14: R4A Model of Robot Resource Component.....	40
Σχήμα 15: Στιγμότυπο οθόνης του SmartMDSD Toolchain κατά τη μοντελοποίηση ενός εξαρτήματος	41
Σχήμα 16: Επισκόπηση αρχιτεκτονικής DEPOSIT	44
Σχήμα 17: Στιγμότυπο δημιουργίας μοντέλου χρησιμοποιώντας το TargetLink	45
Σχήμα 18: High-level διάγραμμα του συστήματος CoRSeDA	46
Σχήμα 19: Παρουσίαση ολόκληρου του μετα-μοντέλου για το σύστημα CoRSeDA	54
Σχήμα 20: AcousticTimeID Enumeration	55
Σχήμα 21: Enumeration Category.....	56
Σχήμα 22: Enumeration AcousticID	66
Σχήμα 23: Enumeration ElectricID.....	68
Σχήμα 24: Enumeration VisionID και Enumeration ResolutionID	70
Σχήμα 25: Enumeration DistanceID	72
Σχήμα 26: Enumeration SpeedID.....	74
Σχήμα 27: Enumeration PressureID_Head.....	76

Σχήμα 28: Enumeration PressureID_Hand.....	77
Σχήμα 29: Enumeration PressureID_Foot.....	78
Σχήμα 30: Enumeration PostureID	80
Σχήμα 31: EnumerationTframeOrigings και EnumerationTframeTargets	82
Σχήμα 32: Enumeration PositionID_Head.....	83
Σχήμα 33: Enumeration PositionID_Hand	84
Σχήμα 34: Enumeration PositionID_Foot.....	86
Σχήμα 35: Enumeration PositionID_Robot	87
Σχήμα 36: Στιγμότυπο της αρχικής σελίδας της web εφαρμογής	138
Σχήμα 37: Στιγμότυπο της σελίδας Data Measuring της εφαρμογής	139
Σχήμα 38: Στιγμότυπο της εφαρμογής μία στιγμή, έπειτα από το πάτημα του κουμπιού load	139
Σχήμα 39: Στιγμότυπο της εφαρμογής που παρουσιάζονται μετρήσεις του αισθητήρα τάδε	140
Σχήμα 40: Αναπαράσταση συστήματος στο γραφικό περιβάλλον Eclipse Sirius	142
Σχήμα 41: Στιγμότυπο της web εφαρμογής κατά τη συλλογή δεδομένων από αισθητήρα πίεσης του ρομπότ ..	144
Σχήμα 42: Στιγμότυπο της web εφαρμογής κατά τη συλλογή δεδομένων από αισθητήρα ήχου του ρομπότ ..	145
Σχήμα 43: Στιγμότυπο της web εφαρμογής κατά τη συλλογή δεδομένων από αισθητήρα ήχου του ρομπότ ..	145

Λίστα Αλγορίθμων

Αλγόριθμος 1: Υλοποίηση συστήματος αισθητήρων βάσει των παραμέτρων από το μοντέλο του χρήστη.....	99
Αλγόριθμος 2: Δημιουργία ενός instance της κλάσης DistanceSensor βάσει των παραμέτρων που θέτει ο χρήστης στο μοντέλο και καταχώρηση αυτού στο σύστημα	100
Αλγόριθμος 3: Δημιουργία ενός instance της κλάσης AcousticSystem βάσει των παραμέτρων που θέτει ο χρήστης στο μοντέλο και καταχώρηση αυτού στο σύστημα	101

Λίστα Πινάκων

Πίνακας 1: Παράθεση σύγκρισης διαφόρων βάσεων δεδομένων	38
Πίνακας 2: Ιδιότητες του SensorSystem του μετα-μοντέλου	54
Πίνακας 3: Συσχετίσεις του SensorSystem του μετα-μοντέλου	55
Πίνακας 4: Ιδιότητες του Subsystem του μετα-μοντέλου	56
Πίνακας 5: Συσχετίσεις του Subsystem του μετα-μοντέλου	56

Πίνακας 6: Συσχετίσεις του DistanceSystem του μετα-μοντέλου	57
Πίνακας 7: Συσχετίσεις του ElectricSystem του μετα-μοντέλου	58
Πίνακας 8: Συσχετίσεις του PositionSystem του μετα-μοντέλου	59
Πίνακας 9: Συσχετίσεις του PressureSystem του μετα-μοντέλου	60
Πίνακας 10: Συσχετίσεις του SpeedSystem του μετα-μοντέλου	60
Πίνακας 11: Συσχετίσεις του VisionSystem του μετα-μοντέλου	61
Πίνακας 12: Ιδιότητες του AcousticSystem του μετα-μοντέλου.....	62
Πίνακας 13: Συσχετίσεις του AcousticSystem του μετα-μοντέλου	62
Πίνακας 14: Ιδιότητες του Sensor του μετα-μοντέλου	65
Πίνακας 15: Ιδιότητες του AcousticSensor του μετα-μοντέλου	66
Πίνακας 16: Συσχετίσεις του AcousticSensor του μετα-μοντέλου.....	66
Πίνακας 17: Ιδιότητες του ElectricSensor του μετα-μοντέλου	67
Πίνακας 18: Συσχετίσεις του ElectricSensor του μετα-μοντέλου	68
Πίνακας 19: Ιδιότητες του VisionSensor του μετα-μοντέλου	69
Πίνακας 20: Συσχετίσεις τουVisionSensor του μετα-μοντέλου	70
Πίνακας 21: Ιδιότητες του DistanceSensor του μετα-μοντέλου	71
Πίνακας 22: Συσχετίσεις του DistanceSensor του μετα-μοντέλου	72
Πίνακας 23: Ιδιότητες του SpeedSensor του μετα-μοντέλου	73
Πίνακας 24: Συσχετίσεις του SpeedSensor του μετα-μοντέλου	74
Πίνακας 25: Συσχετίσεις του PressureSensor του μετα-μοντέλου	75
Πίνακας 26: Ιδιότητες του PressureSensorHead του μετα-μοντέλου	75
Πίνακας 27: Συσχετίσεις του PressureSensorHead του μετα-μοντέλου	76
Πίνακας 28: Ιδιότητες του PressureSensorHand του μετα-μοντέλου.....	76
Πίνακας 29: Συσχετίσεις του PressureSensorHand του μετα-μοντέλου	77
Πίνακας 30: Ιδιότητες του PressureSensorFoot του μετα-μοντέλου	78
Πίνακας 31: Συσχετίσεις του PressureSensorFoot του μετα-μοντέλου	78
Πίνακας 32: Συσχετίσεις του PositionSensor του μετα-μοντέλου	79
Πίνακας 33: Ιδιότητες του PostureSensor του μετα-μοντέλου	80
Πίνακας 34: Συσχετίσεις του PostureSensor του μετα-μοντέλου.....	80
Πίνακας 35: Ιδιότητες του TframeSensor του μετα-μοντέλου	81
Πίνακας 36: Συσχετίσεις του TframeSensor του μετα-μοντέλου	81
Πίνακας 37: Ιδιότητες του PositionSensorHead του μετα-μοντέλου	83

Πίνακας 38: Συσχετίσεις του PositionSensorHead του μετα-μοντέλου	83
Πίνακας 39: Ιδιότητες του PositionSensorHand του μετα-μοντέλου.....	84
Πίνακας 40: Συσχετίσεις του PositionSensorHand του μετα-μοντέλου	84
Πίνακας 41: Ιδιότητες του PositionSensorFoot του μετα-μοντέλου.....	85
Πίνακας 42: Συσχετίσεις του PositionSensorFoot του μετα-μοντέλου	85
Πίνακας 43: Ιδιότητες του PositionSensorRobot του μετα-μοντέλου	86
Πίνακας 44: Συσχετίσεις του PositionSensorRobot του μετα-μοντέλου.....	87
Πίνακας 45: Απεικόνιση AcousticSystem στο Eclipse Sirius UI	88
Πίνακας 46: Απεικόνιση DistanceSystem στο Eclipse Sirius UI.....	88
Πίνακας 47: Απεικόνιση ElectricSystem στο Eclipse Sirius UI.....	89
Πίνακας 48: Απεικόνιση VisionSystem στο Eclipse Sirius UI.....	89
Πίνακας 49: Απεικόνιση SpeedSystem στο Eclipse Sirius UI.....	89
Πίνακας 50: Απεικόνιση PressureSystem στο Eclipse Sirius UI.....	90
Πίνακας 51: Απεικόνιση PositionSystem στο Eclipse Sirius UI.....	90
Πίνακας 52: Απεικόνιση Acoustic Sensor στο Eclipse Sirius UI	90
Πίνακας 53: Απεικόνιση Distance Sensor στο Eclipse Sirius UI.....	91
Πίνακας 54: Απεικόνιση Electric Sensor στο Eclipse Sirius UI.....	91
Πίνακας 55: Απεικόνιση Position Sensor Robot στο Eclipse Sirius UI	91
Πίνακας 56: Απεικόνιση Position Sensor Posture στο Eclipse Sirius UI.....	92
Πίνακας 57: Απεικόνιση Position Sensor Head στο Eclipse Sirius UI.....	92
Πίνακας 58: Απεικόνιση Position Sensor tFrame στο Eclipse Sirius UI.....	92
Πίνακας 59: Απεικόνιση Position Sensor Hand στο Eclipse Sirius UI.....	93
Πίνακας 60: Απεικόνιση Position Sensor Foot στο Eclipse Sirius UI.....	93
Πίνακας 61: Απεικόνιση Pressure Sensor Head στο Eclipse Sirius UI.....	93
Πίνακας 62: Απεικόνιση Pressure Sensor Hand στο Eclipse Sirius UI.....	93
Πίνακας 63: Απεικόνιση Pressure Sensor Foot στο Eclipse Sirius UI.....	94
Πίνακας 64: Απεικόνιση Speed Sensor στο Eclipse Sirius UI.....	94
Πίνακας 65: Απεικόνιση Vision Sensor στο Eclipse Sirius UI.....	94
Πίνακας 66: Απεικόνιση SensorSystem στο Eclipse Sirius UI.....	94
Πίνακας 67: Επισκόπηση των modules του μετασχηματισμού	97
Πίνακας 68: Αίτημα GET για την ανάκτηση της λίστας categories των αισθητήρων.....	107
Πίνακας 69: Αίτημα GET για την ανάκτηση της λίστας χαρακτηριστικών όλων των αισθητήρων	107

Πίνακας 70: Αίτημα GET για την ανάκτηση της λίστας ονομάτων όλων των αισθητήρων.....	108
Πίνακας 71: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Acoustic. .	108
Πίνακας 72: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Distance..	108
Πίνακας 73: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Electric. ...	109
Πίνακας 74: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Pressure..	109
Πίνακας 75: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Position...	110
Πίνακας 76: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Speed.....	110
Πίνακας 77: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Vision.....	110
Πίνακας 78: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Distance.	111
Πίνακας 79: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Acoustic.	112
Πίνακας 80: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Electric.	112
Πίνακας 81: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Position.	113
Πίνακας 82: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Position - Posture. ..	113
Πίνακας 83: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Position - Tframe.	114
Πίνακας 84: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Position - Robot.....	115
Πίνακας 85: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Pressure.	115
Πίνακας 86: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Speed.	115
Πίνακας 87: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Vision.	116
Πίνακας 88: Αίτημα GET για την ανάκτηση των ορισμένων χαρακτηριστικών του ρομπότ.	117
Πίνακας 89: Αίτημα GET για την ανάκτηση των τεχνικών χαρακτηριστικών ενός αισθητήρα τύπου Distance...	117
Πίνακας 90: Αίτημα GET για την ανάκτηση των τεχνικών χαρακτηριστικών ενός αισθητήρα τύπου Acoustic... .	118
Πίνακας 91: Αίτημα GET για την ανάκτηση των τεχνικών χαρακτηριστικών ενός αισθητήρα τύπου Electric.....	119
Πίνακας 92: Αίτημα GET για την ανάκτηση των τεχνικών χαρακτηριστικών ενός αισθητήρα τύπου Position....	119
Πίνακας 93: Αίτημα GET για την ανάκτηση των τεχνικών χαρακτηριστικών ενός αισθητήρα τύπου Pressure... .	120
Πίνακας 94: Αίτημα GET για την ανάκτηση των τεχνικών χαρακτηριστικών ενός αισθητήρα τύπου Vision.....	120
Πίνακας 95: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Distance.....	121
Πίνακας 96: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Acoustic.....	122
Πίνακας 97: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Electric.....	122
Πίνακας 98: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Position.	123
Πίνακας 99: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Position - Posture....	124
Πίνακας 100: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Position - Tframe. .	124
Πίνακας 101: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Position - Robot. ...	125

Πίνακας 102: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Pressure.....	125
Πίνακας 103: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Speed.	126
Πίνακας 104: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Vision.....	127
Πίνακας 105: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Distance.....	127
Πίνακας 106: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Acoustic.	128
Πίνακας 107: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Electric.....	128
Πίνακας 108: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Position.....	129
Πίνακας 109: Αίτημα GET για την ανάκτηση τελευταίας μέτρησης ενός αισθητήρα τύπου PositionPosture.....	129
Πίνακας 110: Αίτημα GET για την ανάκτηση τελευταίας μέτρησης ενός αισθητήρα τύπου PositionTframe.	130
Πίνακας 111: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου PositionRobot.	130
Πίνακας 112: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Pressure.....	131
Πίνακας 113: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Speed.....	131
Πίνακας 114: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Vision.....	131
Πίνακας 115: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Distance.....	132
Πίνακας 116: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Acoustic.	132
Πίνακας 117: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Electric.....	133
Πίνακας 118: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Position.	133
Πίνακας 119: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου PositionPosture. .	133
Πίνακας 120: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου PositionTframe. ..	134
Πίνακας 121: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου PositionRobot.	134
Πίνακας 122: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Pressure.....	135
Πίνακας 123: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Speed.	135
Πίνακας 124: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Vision.....	135
Πίνακας 125: Αίτημα GET για την ανάκτηση αρχείου εικόνας ενός αισθητήρα τύπου Vision.	136
Πίνακας 126: Αίτημα GET για την ανάκτηση αρχείου ήχου ενός αισθητήρα τύπου Acoustic.	136
Πίνακας 127: Κλήσεις του API που παράχθηκαν μέσω του συστήματος.....	143

Λεξικό Όρων

Αρκτικόλεξα

API: Application Programming Interface

ATL: Atlas Transformation Language (a declarative M2M transformation language)

CIM: Computational Independent Model

CSS: Cascading Styling Sheets

EMF: Eclipse Modelling Framework

GUI: Graphical User Interface

HTTP: Hypertext Transfer Protocol

JSON: JavaScript Object Notation (Τύπος media format)

M2M: Model to Model transformation

M2T: Model to Text transformation

MDA: Model Driven Architecture

MDE: Model Driven Engineering

MOF: Meta-Object Facility

OMG: Object Management Group

PIM: Platform Independent Model

PSM: Platform Specific Model

REST: Representational State Transfer

SDLC: Software Development Life Cycle

UML: Unified Modeling Language

URL: Uniform Resource Locator

VSM: Viewpoint Specification Model

XML: Extensible Markup Language

1. Εισαγωγή

1.1 Κίνητρο

Αδιαμφισβήτητα, η κοινωνία και οι δομές της επηρεάζονται και αναδιαμορφώνονται βάσει των τεχνολογικών μέσων αλλά και των μέσων επικοινωνίας που υπάρχουν. Όταν εμφανίζεται λοιπόν κάποιο νέο μέσο, υπάρχει άμεση επίδραση, κυρίως λόγω των νέων δυνατοτήτων που παρέχει αυτό. Λίγοι θα διαφωνήσουν με τη θέση πως το διαδίκτυο, το οποίο μεταφέρει πληθώρα πληροφοριών από κάθε γωνία του πλανήτη, αποτελεί ένα από τα σημαντικότερα μέσα επικοινωνίας της σύγχρονης εποχής και έχει ενταχθεί σε κάθε κοινωνική δομή.

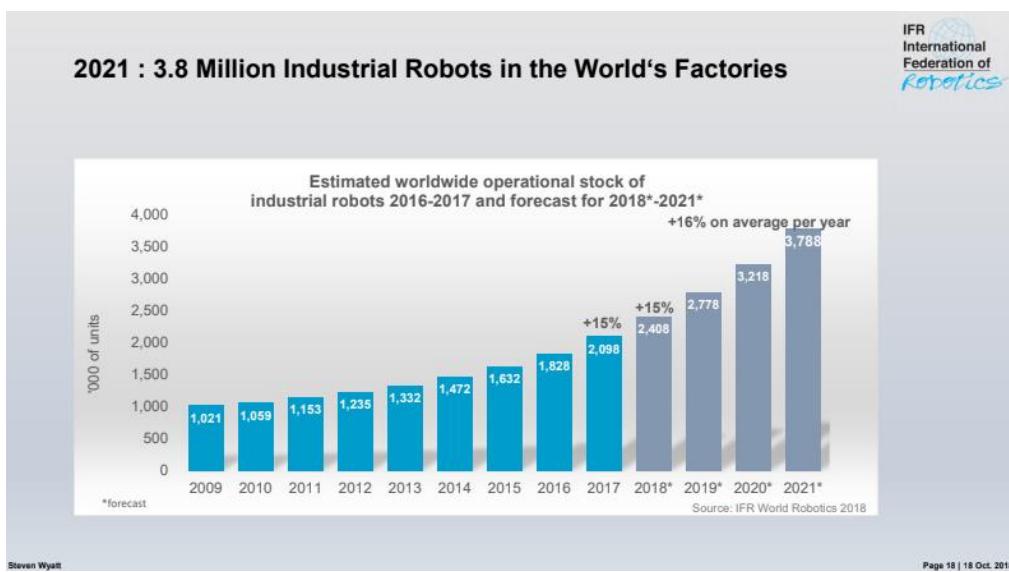
Σε αυτά έρχεται να προστεθεί πλέον και η τεχνολογία λογισμικού, μία επιστήμη που συνδυάζει αρχές και γνώσεις διαφορετικών γνωστικών αντικειμένων, με στόχο την ανάπτυξη βέλτιστων συστημάτων λογισμικού. Το λογισμικό συνιστά ένα εργαλείο που κατασκεύασε ο άνθρωπος τον προηγούμενο αιώνα, αλλάζοντας άρδην την προοπτική που είχε ο άνθρωπος σχετικά με την επεξεργασία της πληροφορίας και των επικοινωνιών. Πράγματι, ακόμη και σήμερα βιώνουμε αυτήν τη συνεχόμενη εξέλιξη, καθώς βρισκόμαστε συχνά, εκούσια ή ακούσια, στη θέση του χρήστη μίας ή πολλών εφαρμογών λογισμικού.

Αυτή η ολοένα αυξανόμενη ζήτηση λογισμικού δημιουργεί την ανάγκη για γρηγορότερη ανάπτυξη ορθού λογισμικού. Ο αυτοματισμός είναι μία πτυχή της ανάπτυξης λογισμικού που μειώνει σημαντικά το χρόνο παραγωγής του και η Οδηγούμενη από Μοντέλα Μηχανική (*Model Driven Engineering*) είναι μία σχετικά νέα μέθοδος ανάπτυξης λογισμικού, η οποία βασίζεται σε μετασχηματισμούς μοντέλων μεταξύ των διαφορετικών φάσεων ανάπτυξης και πετυχαίνει την ταχεία παραγωγή λογισμικού για μία πληθώρα εφαρμογών.

Σε όλα τα παραπάνω ήρθε τα τελευταία χρόνια να προστεθεί μία νέα ανάγκη για λογισμικό, αυτή των ρομποτικών εφαρμογών. Η ολοένα αυξανόμενη διεισδυτικότητα των ρομπότ σε εταιρικό και οικιακό περιβάλλον δημιουργούν τη ζήτηση για ταχεία παραγωγή λογισμικού για ρομπότ, ο οποίος θα είναι κατά το δυνατόν χωρίς σφάλματα. Ωστόσο η περίπλοκη φύση της εργασίας αυτής σε συνδυασμό με την ποικιλομορφία που παρουσιάζουν οι ρομποτικές εφαρμογές, ακόμη και στην περίπτωση που επιτελούν παρόμοιες εργασίες, καθιστούν συχνά ιδιαιτέρως δύσκολη την ανάπτυξη ορθού κώδικα για ρομπότ.

1.2 Περιγραφή Προβλήματος

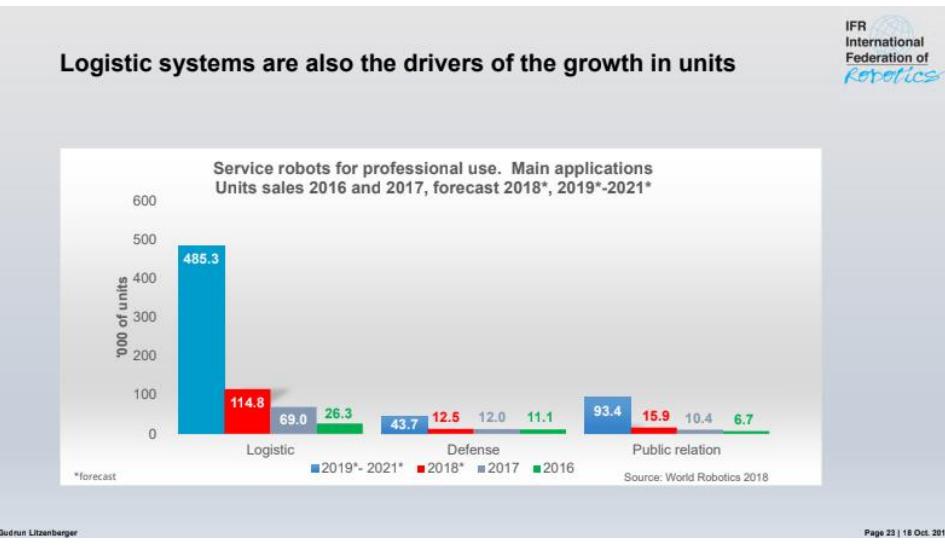
Η δημιουργία και η χρήση ρομπότ δε συνιστούν απαραίτητα νέα τεχνολογικά επιτεύγματα, καθώς χρησιμοποιούνται ευρέως εδώ και αρκετά χρόνια. Ωστόσο, την τελευταία δεκαετία έχει παρατηρηθεί μία ραγδαία ανάπτυξη στον τομέα αυτό, τόσο σε πλήθος, όσο και σε επίπεδο πολυπλοκότητας αλλά κυρίως ποικιλομορφίας. Πιο αναλυτικά, παρακάτω παρουσιάζονται ορισμένα δεδομένα δεδομένα για τα τελευταία δέκα χρόνια από το IFR⁽¹⁾, αναφορικά με την αύξηση ζήτησης του πλήθους των ρομπότ για βιομηχανικές και οικιακές, καθώς και ποιες είναι οι ανάγκες αυτές [1].



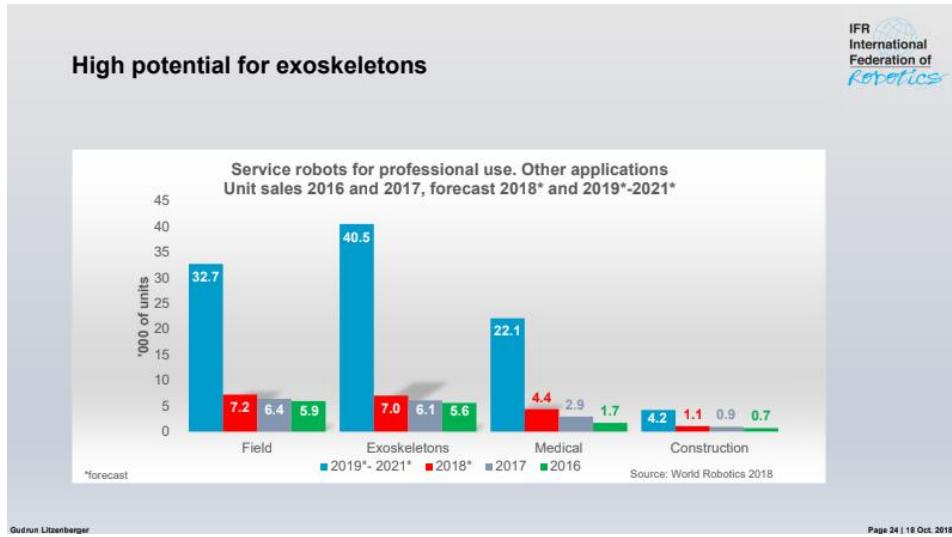
Σχήμα 1: Ραγδαία αύξηση του πλήθους ρομπότ βιομηχανικών εφαρμογών

Όπως είναι εμφανές στο Σχήμα 1, σε διάρκεια μόλις οκτώ ετών υπήρξε διπλασιασμός των ρομπότ στο χώρο της βιομηχανίας, ξεπερνώντας σε αριθμό τα 2εκ. ρομπότ που βρίσκονται σε λειτουργία, προκειμένου να καλυφθούν οι ανάγκες της αγοράς. Στα σχήματα που ακολουθούν, εύκολα μπορεί να παρακολουθήσει κανείς την αυξανόμενη ανάγκη για παραγωγή λογισμικού για ρομποτικές εφαρμογές. Αυτή η γκάμα αντικειμένων, δημιουργεί πολύπλευρες ανάγκες για την άμεση παραγωγή λογισμικών, ικανών να καλύπτουν κάθε ανάγκη που υπάρχει στο χώρο της ιατρικής, της εκπαίδευσης, της βιομηχανίας, ακόμη και της διασκέδασης.

⁽¹⁾ IFR: Η Διεθνής Ομοσπονδία Ρομποτικής (International Federation of Robotics) ιδρύθηκε ως μη κερδοσκοπικός οργανισμός το 1987 και συνδέει τον κόσμο της ρομποτικής σε όλο τον κόσμο. Τα μέλη προέρχονται από τη βιομηχανία ρομποτικής, εθνικές ή διεθνείς βιομηχανικές ενώσεις και ινστιτούτα έρευνας και ανάπτυξης. Το τμήμα στατιστικών της ομοσπονδίας είναι η κύρια πηγή παγκοσμίως για δεδομένα σχετικά με τη ρομποτική.



Σχήμα 2: Σύγκριση της ζήτησης μεταξύ των services των ρομπότ από το 2016 έως σήμερα και πρόβλεψη για τα επόμενα έτη, αναδεικνύοντας την ανάγκη για services αναφορικά με την αποθήκευση και τη διαχείριση αποθεμάτων

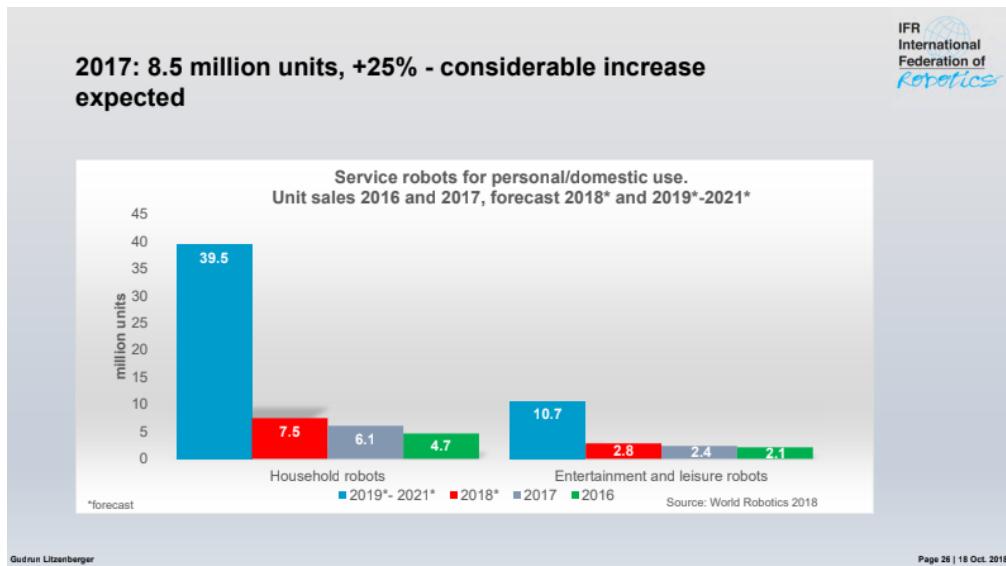


Σχήμα 3: Σύγκριση της ζήτησης μεταξύ των services των ρομπότ από το 2016 έως σήμερα και πρόβλεψη για τα επόμενα έτη, αναδεικνύοντας την ανάγκη για services αναφορικά με τους εξωτερικούς σκελετούς

Το πρόβλημα λοιπόν που παρουσιάζεται έγκειται στο γεγονός πως δεν γνωρίζουν όλοι οι άνθρωποι να χειρίζονται και να προγραμματίζουν τα ρομπότ, ώστε να καλύπτουν τις επιμέρους ανάγκες που προκύπτουν.

Αυτό οφείλεται πιθανώς στην έλλειψη γνώσεων προγραμματισμού, καθώς ακόμη και στην άγνοια των δυνατοτήτων των ρομπότ. Αυτό είναι ένα πρόβλημα που οι μηχανικοί λογισμικού καλούνται να επιλύσουν,

αναπτύσσοντας λογισμικό χωρίς σφάλματα που θα βοηθά τους χρήστες, οι οποίοι δεν κατέχουν την απαραίτητη γνώση, να χρησιμοποιούν τα ρομπότ. Επιπλέον, οι ανάγκες αλλάζουν και προσαρμόζονται παράλληλα με τη δημιουργία νέων τεχνολογικών επιτευγμάτων, διευρύνοντας έτσι το έργο τους. Σε μεγάλο βαθμό το παραγόμενο λογισμικό θα αξιοποιηθεί επιμέρους από εξειδικευμένο προσωπικό (προγραμματιστές, μηχανικούς κ.λ.π.), ώστε να μειωθεί ο χρόνος και το κόστος ανάπτυξης άλλων συστημάτων λογισμικού. Αυτό επιτυγχάνεται παρέχοντας υψηλού επιπέδου διεπαφές για συγκεκριμένες εφαρμογές.



Σχήμα 4: Σύγκριση της ζήτησης μεταξύ των services των ρομπότ από το 2016 έως σήμερα και πρόβλεψη για τα επόμενα έτη, αναδεικνύοντας την ανάγκη για services αναφορικά με οικιακές εφαρμογές

1.3 Στόχοι της διπλωματικής εργασίας

Η παρούσα διπλωματική έχει ως στόχο τη σχεδίαση και την ανάπτυξη συστήματος λογισμικού μοντελοστραφούς λογικής, το οποίο θα επιτρέπει στους χρήστες του να μοντελοποιούν ένα σύστημα συλλογής δεδομένων αισθητήρων ρομπότ. Πιο αναλυτικά, οι χρήστες θα μπορούν αναζητώντας υπάρχοντα μοντέλα αισθητήτων διαφόρων τύπων που υπάρχουν σε ένα ρομπότ να επιλέγουν όσα χρειάζονται και να τα παραμετροποιούν με βάση την εκάστοτε επιθυμητή συμπεριφορά του αισθητήρα (π.χ. refresh rates). Στη συνέχεια, το σύστημα λογισμικού θα παράγει αυτόματα τον απαιτούμενο κώδικα, σε Python 2.7, ο οποίος κατά τη λειτουργία του ρομπότ θα παρέχει την απαιτούμενη λειτουργικότητα ως προς το συγκεκριμένο αισθητήρα λαμβάνοντας περιοδικά τα δεδομένα από αυτόν, αποθηκεύοντάς τα στην τοπική μνήμη του ρομπότ ώστε να είναι αξιοποιήσιμα από άλλα υποσυστήματα του ρομπότ, και τέλος, κάνοντας διαθέσιμη την πληροφορία του

αισθητήρα στο διαδίκτυο μέσω παραγωγής διεπαφής REST ώστε να μπορεί ο αισθητήρας να αποτελεί ένα αντικείμενο IoT (Internet of Things).

Το σύστημα αυτό καταλήγει σε ένα γραφικό περιβάλλον στο οποίο κάθε χρήστης, ανεξαρτήτως του γνωστικού του επιπέδου, πρέπει να μπορεί να δημιουργεί ένα σύστημα αισθητήρων από τους οποίους επιθυμεί να συλλέγει δεδομένα, βάσει των παραμέτρων που επιθυμεί. Το αξιοσημείωτο είναι πως το γραφικό αυτό περιβάλλον πρέπει να είναι εύχρηστο, ώστε να μπορεί ο χρήστης εύκολα να το προσαρμόζει στις ανάγκες του. Μέσα από αυτή τη διαδικασία αποκτήθηκαν γνώσεις αναφορικά με τη μοντελοστραφή σχεδίαση λογισμικού και ορισμένων εξειδικευμένων εργαλείων και προγραμμάτων, όπως είναι τα εξής: Eclipse/Ecore και Acceleo Template Language.

1.4 Διάρθρωση του εγγράφου

Η παρούσα διπλωματική εργασία αποτελείται από έξι κεφάλαια. Σε αυτό το κεφάλαιο γίνεται μια σύντομη εισαγωγή στο αντικείμενο που καλείται να αναλύσει η συγκεκριμένη διπλωματική εργασία. Στο κεφάλαιο 2 γίνεται περιγραφή του θεωρητικού και τεχνολογικού υπόβαθρου που χρειάζεται ο αναγνώστης προκειμένου να γίνει ευκολότερη η κατανόηση της παρούσας εργασίας. Πιο αναλυτικά, περιγράφονται λακωνικά οι ορισμοί και θεωρητικά στοιχεία, ενώ στη συνέχεια επισημαίνονται οι τεχνολογίες, τα εργαλεία και τα προγράμματα που χρησιμοποιήθηκαν. Στο κεφάλαιο 3 γίνεται σύντομη αναφορά σε ορισμένες από τις πιο σύγχρονες και επιτυχημένες υλοποίησεις στον τομέα της σχεδίασης ενός συστήματος και στην αυτόματη παραγωγή κώδικα. Στο κεφάλαιο 4 παρουσιάζεται η μεθοδολογία που ακολουθήθηκε για την επίτευξη των στόχων αυτής της διπλωματικής. Γίνεται λεπτομερής περιγραφή των εργαλείων που υλοποιήθηκαν, καθώς επεξηγούνται οι λόγοι για τους οποίους επιλέχθηκαν οι συγκεκριμένες μέθοδοι, ενώ παρουσιάζονται όλα τα σημεία του τελικού αποτελέσματος, προκειμένου να γίνει κατανοητός ο τρόπος χρήσης και λειτουργίας του. Στο κεφάλαιο 5 παρουσιάζονται ορισμένα παραδείγματα και τα αποτελέσματα που προέκυψαν από τη χρήση του συστήματος. Για να γίνει περισσότερο κατανοητό παρουσιάζονται ορισμένα στιγμιότυπα από τη χρήση του συστήματος σε κάθε στάδιο της εφαρμογής με επεξηγήσεις. Ολοκληρώνονται την εργασία αυτή, στο κεφάλαιο 6 παρατίθενται τα συμπεράσματα από τη δημιουργία και τη χρήση του συστήματος και περιγράφονται ορισμένες βελτιώσεις, ενώ καταγράφονται ορισμένα ανοιχτά θέματα και πιθανές μελλοντικές επεκτάσεις με νέες λειτουργίες για την εφαρμογή. Τέλος, στο κεφάλαιο 7 γίνεται αναφορά στις πηγές οι οποίες τεκμηριώνουν όσα αναφέρονται στη συγκεκριμένη διπλωματική εργασία.

2. Υπόβαθρο

2.1 Θεωρητικό Υπόβαθρο

2.1.1 Τεχνολογία Λογισμικού

Όπως επισημάνθηκε στο προηγούμενο κεφάλαιο, οι κλάδοι της τεχνολογίας, της επικοινωνίας και της πληροφορίας εξελίσσονται με ιλιγγιώδης ρυθμούς. Η εξέλιξη αυτή έρχεται να καλύψει τις ανάγκες των ανθρώπων μέσα από μία πληθώρα αυτοματοποιημένων τεχνολογικών προϊόντων, τα οποία απαιτούν με τη σειρά τους ορθό και κατάλληλα σχεδιασμένο λογισμικό. Είναι εύκολα κατανοητό λοιπόν, πως η εξέλιξη της τεχνολογίας ακολουθείται από τη συνεχώς αυξανόμενη ανάγκη για λογισμικό. Στο σημείο αυτό είναι σκόπιμο να επισημάνουμε τον ορισμό της τεχνολογίας λογισμικού τη σημασία της, καθώς επίσης και τον ορισμό του λογισμικού.

Η *Τεχνολογία Λογισμικού (Software Engineering)* ορίζεται ως μια συστηματική, πειθαρχημένη και ποσοτικοποιήσιμη προσέγγιση της ανάπτυξης, της εκτέλεσης και της συντήρησης του λογισμικού. Στη βιβλιογραφία ορίζεται και λίγο διαφορετικά ως μία συστηματική εφαρμογή των επιστημονικών και τεχνολογικών γνώσεων, των μεθόδων και των εμπειριών στο σχεδιασμό, την υλοποίηση, τη δοκιμή και την τεκμηρίωση του λογισμικού [2]. Πρόκειται για έναν τομέα που ασχολείται με την ανάπτυξη μεθόδων και διαδικασιών με σκοπό την ανάπτυξη γρήγορου, φθηνού, αξιόπιστου και ποιοτικού λογισμικού που επιλύει όλο και πιο περίπλοκα προβλήματα.

Όσον αφορά το λογισμικό, συχνά γίνεται μία παρερμηνεία των όρων λογισμικό και πρόγραμμα. Ένα πρόγραμμα αποτελείται από ένα σύνολο εντολών που εκτελούν μια συγκεκριμένη εργασία όταν αυτό θα εκτελείται από έναν υπολογιστή. Ως λογισμικό ορίζεται ένα σύνολο από προγράμματα υπολογιστών, διαδικασίες, κανόνες, ενώ ενδεχομένως και σχετική τεκμηρίωση και δεδομένα αναφορικά με τη λειτουργία ενός υπολογιστικού συστήματος. Συνεπώς, η δυσκολία της επιστήμης της τεχνολογίας λογισμικού έγκειται σε όλα τα παραπάνω. Τα πλεονεκτήματα που προσφέρει η επιστήμη της τεχνολογίας λογισμικού ποικίλουν και παρατηρούνται σε όλες τις εφαρμογές της. Τα οφέλη της χρήσης του *SDLC (Software Development Life Cycle)*, μία από τις πιο βασικές διαδικασίες ανάπτυξης λογισμικού, που εισήγαγε η επιστήμη της τεχνολογίας λογισμικού, παρατηρούνται σε όλους τους τύπους οργανισμών που παράγουν λογισμικό. Αν το πρότυπο ακολουθηθεί σωστά, τότε μειώνεται η πολυπλοκότητα παραγωγής του λογισμικού, αποφεύγονται σφάλματα και βελτιώνεται η ποιότητα του τελικού έργου το οποίο είναι σύμφωνο με τις αρχικές προαπαιτήσεις.

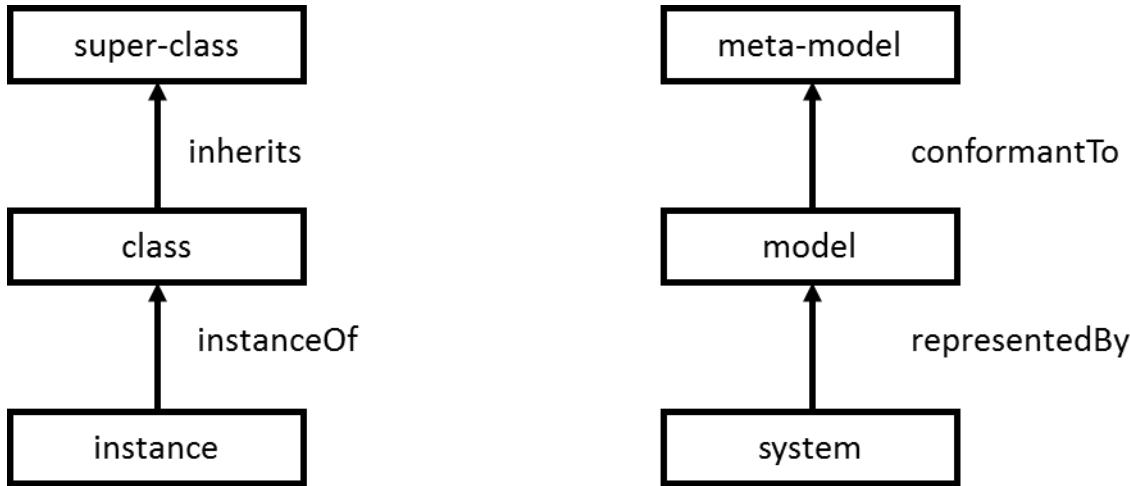
2.1.2 ROS (Robot Operating System)

Το λειτουργικό σύστημα ρομπότ (*Robot Operating System* ή *ROS*) είναι ένα ευέλικτο πλαίσιο για τη σύνταξη λογισμικού ρομπότ, ή αλλιώς ένα μετα-λειτουργικό σύστημα ανοιχτού κώδικα για ένα ρομπότ. Παρέχει τις υπηρεσίες που θα περίμενε κανείς από ένα λειτουργικό σύστημα, συμπεριλαμβανομένης της αφαίρεσης υλικού, του ελέγχου των συσκευών χαμηλού επιπέδου, της εφαρμογής κοινώς χρησιμοποιούμενων λειτουργιών, της μετάδοσης μηνυμάτων μεταξύ των διαδικασιών και της διαχείρισης πακέτων. Πρόκειται για μια συλλογή εργαλείων, βιβλιοθηκών και συμβάσεων που στοχεύουν στην απλοποίηση του έργου δημιουργίας περίπλοκων και ισχυρών συμπεριφορών ρομπότ σε μια μεγάλη ποικιλία ρομποτικής πλατφόρμας. Γενικότερα, η δημιουργία ενός πραγματικά εύρωστου λογισμικού ρομπότ γενικής χρήσης είναι δύσκολη. Από τη σκοπιά του ρομπότ, τα προβλήματα που φαίνονται τετριμένα στον άνθρωπο συχνά ποικίλουν μεταξύ περιπτώσεων εργασιών και περιβαλλόντων. Η ενασχόληση με αυτές τις παραλλαγές είναι αρκετά δύσκολη, καθώς απαιτείται ο συνδυασμός πολλών γνωστικών αντικειμένων. Για το σκοπό αυτό, δημιουργήθηκε το *ROS*, προκειμένου δηλαδή να ενθαρρύνει την ανάπτυξη συνεργατικής ρομποτικής λογισμικού [3].

2.1.3 MDE (Model Driven Engineering)

Είναι πολύ σημαντικό για τη συγκεκριμένη διπλωματική εργασία να έχουν κατανοηθεί οι βασικές αρχές της μοντελο-κεντρικής μηχανικής. Εάν ο αναγνώστης κατέχει αυτές τις απαιτούμενες γνώσεις μπορεί να προσπεράσει τη συγκεκριμένη και την επόμενη παράγραφο [11].

Η Μοντελο-κεντρική Μηχανική (*Model Driven Engineering* ή *MDE*) ήταν το ακόλουθο βήμα από τον αντικειμενοστραφή προγραμματισμό, κατά τον οποίο η βασική αρχή αναφέρει πως όλα μπορούν να χαρακτηριστούν ως αντικείμενα. Η *MDE* υιοθετεί μία παρόμοια λογική με τη σημαντική διαφορά πως όλα πλέον αναγνωρίζονται ως μοντέλα και όχι ως αντικείμενα.



Σχήμα 5: Βασικές αρχές αντικειμενοστραφούς και μοντελοκεντρικής λογικής

Η *MDE* βασίζεται στη μετάβαση από την κωδικο-κεντρική λογική στην μοντελο-κεντρική με στόχο την βελτιστοποίηση την παραγωγικότητας μέσω αυτοματοποιημένων διαδικασιών. Στην *MDE* η έννοια του μοντέλου δεσπόζει ως το κυρίαρχο εργαλείο καθ' όλη τη ζωή ενός έργου λογισμικού. Πιο αναλυτικά, κατά τη δημιουργία λογισμικού με *MDE*, λαμβάνεται υπόψιν πως κάθε σύστημα θα πρέπει να αναπαρασταθεί από μοντέλα και το κάθε ένα μοντέλο θα συμμορφώνεται συντακτικά σε ένα μετα-μοντέλο. Στη συνέχεια περιγράφονται αναλυτικά οι έννοιες των μοντέλων, των μετα-μοντέλων, καθώς ακόμη και οι μετασχηματισμοί μοντέλων.

2.1.3.1 Μοντέλα, Μέτα-Μοντέλα και Μετα-Μέτα-Μοντέλα

Αναφερόμενοι πάντα στο πλαίσιο της *MDE*, η έννοια του μοντέλου ορίζεται ως μία απλουστευμένη αναπαράσταση ενός αντικειμένου ή συστήματος, η οποία όμως διατηρεί τις βασικές ιδιότητες τους, ώστε να επιτρέπει τη συστηματική ανάλυση τους για ένα συγκεκριμένο σκοπό. Συχνά, απαιτείται να δημιουργηθούν πολλά μοντέλα για ένα σύστημα, προκειμένου να αναπαρασταθούν καλύτερα όλες οι όψεις του συστήματος. Αξιοσημείωτο είναι εδώ πως το μοντέλο είναι κι αυτό ένα σύστημα με δικά του χαρακτηριστικά στοιχεία και δική του πολυπλοκότητα.

Συνεχίζοντας, υπάρχουν πολλοί ορισμοί για την έννοια του μετα-μοντέλου, οι οποίοι όμως δεν το περιγράφουν ικανοποιητικά. Για την παρούσα εργασία, όταν θα γίνεται αναφορά σε ένα μετα-μοντέλο, τότε αυτό θα

εικλαμβάνεται ως το σύνολο των συμβόλων, των όρων και των περιορισμών που επιτρέπεται να χρησιμοποιηθούν, προκειμένου να σχεδιαστεί ένα έγκυρο μοντέλο, το οποίο θα συμμορφώνεται με τα στοιχεία του μετα-μοντέλου. Η συγκεκριμένη περιγραφή δεν αντιστοιχεί σε κανέναν αυστηρό ορισμό, απλώς επεξηγεί την έννοια του όρου. Πρόκειται δηλαδή για την αφηρημένη σύνταξη μιας γλώσσας περιγραφής μοντέλων (*modeling language*).

Τι ισχύει όμως όταν φτάνει κανείς στη σύνταξη και τον καθορισμό του μετα-μοντέλου; Ο προβληματισμός αυτός αντιμετωπίζεται με τη χρήση μιας γλώσσας η οποία σε κάποιο σημείο της, θα περιγράφει τα δικά της χαρακτηριστικά στη δική της γλώσσα. Ο μηχανισμός που υλοποιεί τα παραπάνω αποτελεί ένα τρίτο επίπεδο στην ιεραρχία της περιγραφής και ονομάζεται μετα-μετα-μοντέλο. Σε αυτό το επίπεδο θα αναφέρονται οι απαραίτητες σημασιολογικές έννοιες για την περιγραφή ενός μετα-μοντέλου. Γίνεται έτσι κατανοητό πως με τον ίδιο τρόπο που ένα μοντέλο συμμορφώνεται με ένα μετα-μοντέλο, έτσι κι ένα μετα-μοντέλο συμμορφώνεται με ένα μετα-μετα-μοντέλο.

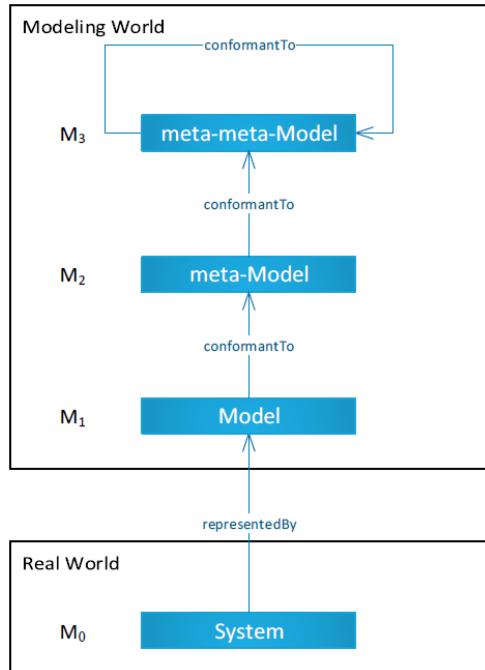
Για να μπορέσουν να εφαρμοστούν όλα τα παραπάνω αναπτύχθηκε το *MOF*(*Meta Object Facility*), το οποίο παρουσιάζεται αμέσως μετά.

2.1.4.2 *Meta Object Facility (MOF)*

To *OMG (Object Management Group)* ανέπτυξε το 2000 το *MOF* το οποίο υποστηρίζει μία αρχιτεκτονική τεσσάρων στρωμάτων. Ουσιαστικά, πρόκειται για έναν τρόπο μοντελοποίησης ενός συστήματος λογισμικού. Αυτό που απαιτείται από το *MOF*, και από κάθε άλλο παρόμοιο μηχανισμό, είναι να μπορεί να υποστηρίζει όλων των ειδών μετα-δεδομένων και ακόμη να παρέχει τη δυνατότητα να προστεθεί κι άλλο είδος. Για να επιτευχθεί η συγκεκριμένη ανάγκη, το *MOF* έχει δημιουργηθεί με μία αρχιτεκτονική τεσσάρων επιπέδων.

Στο *Σχήμα 6* που ακολουθεί απεικονίζεται η αρχιτεκτονική αυτή των τεσσάρων επιπέδων. Πιο αναλυτικά, στο τελευταίο επίπεδο M_0 βρίσκεται το πραγματικό σύστημα, το οποίο περιέχει και τις πληροφορίες. Ας επισημανθεί εδώ, πως το συγκεκριμένο επίπεδο διαχωρίζεται με τα υπόλοιπα, καθώς τα άλλα τρία επίπεδα αναφέρονται στον κόσμο του μοντέλου, ενώ το M_0 στον πραγματικό κόσμο. Στη συνέχεια, ξεκινώντας από το επίπεδο M_1 , αυτό αντιπροσωπεύει το μοντέλο ή αλλιώς τα μετα-δεδομένα που περιγράφουν τις πληροφορίες του επιπέδου M_0 . Το M_1 συμμορφώνεται με το μετα-μοντέλο που βρίσκεται στο επίπεδο M_2 και το οποίο περιέχει τους κανονισμούς και τους περιορισμούς για τα μετα-δεδομένα του επιπέδου M_1 . Τέλος, στην κορυφή της ιεραρχίας βρίσκεται το επίπεδο M_3 , που αποτελεί το μετα-μετα-μοντέλο.

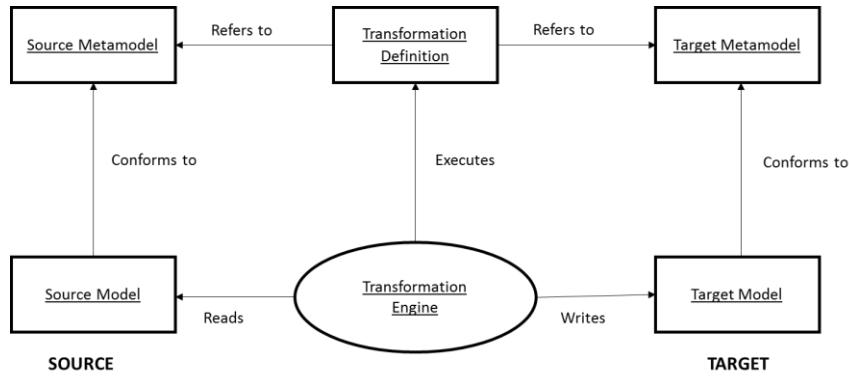
Όπως αναφέρθηκε και στην περιγραφή ενός μετα-μετα-μοντέλου, έτσι κι εδώ το M_3 είναι ουσιαστικά μια περιγραφή του μετα-μοντέλου του επιπέδου M_2 . Τέλος, δύο είναι οι βασικές σχέσεις του MOF, η *representedBy* που δηλώνει «αναπαρίσταται ως» και η *conformantTo* «συμμορφώνεται σε».



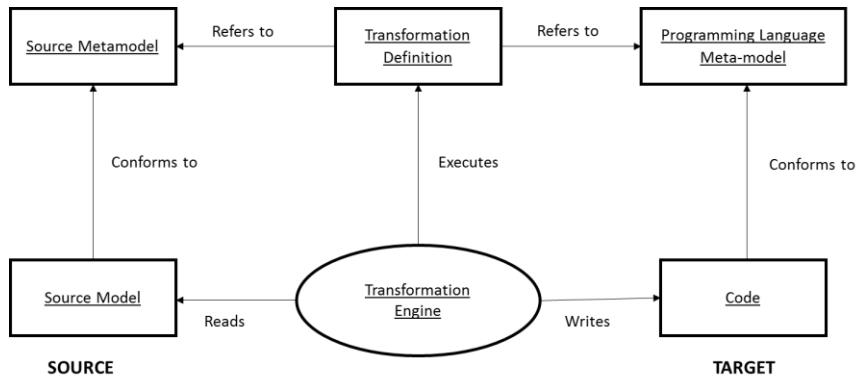
Σχήμα 6: Η αρχιτεκτονική MOF 4 επιπέδων

2.1.4.3 Μετασχηματισμοί

Ο μετασχηματισμός αποτελείται από ένα πλήθος κανόνων με τους οποίους θα μετατραπεί αυτόματα ένα μοντέλο εισόδου(*source*) σε ένα μοντέλο εξόδου(*target*). Υπάρχουν δύο ειδών μετασχηματισμοί, ο μετασχηματισμός από μοντέλο σε μοντέλο(*Model to model transformation* ή *M2M*) και ο μετασχηματισμός από μοντέλο σε κώδικα (*Model to text transformation* ή *M2T*). Ένας μετασχηματισμός από μοντέλο σε μοντέλο αποτελείται από όσους κανόνες μετασχηματισμού χρειάζονται ώστε κάθε όρος του μοντέλου εισόδου να μετασχηματίζεται κατάλληλα σε ένα σύνολο όρων του μοντέλου εξόδου. Ένας μετασχηματισμός από μοντέλο σε κώδικα αποτελείται από ένα σύνολο προτύπων κώδικα, τα οποία ο *M2T* μετασχηματισμός «συμπληρώνει», χρησιμοποιώντας τις πληροφορίες από το μοντέλο εισόδου. Ο παραγόμενος κώδικας μπορεί να είναι σε οποιαδήποτε γλώσσα προγραμματισμού υψηλού επιπέδου, όπως Python, Java, C++ [6].



Σχήμα 7: Μετασχηματισμός M2M



Σχήμα 8: Μετασχηματισμός M2T

2.1.4.4 Πλεονεκτήματα και μειονεκτήματα της μεθόδου MDE

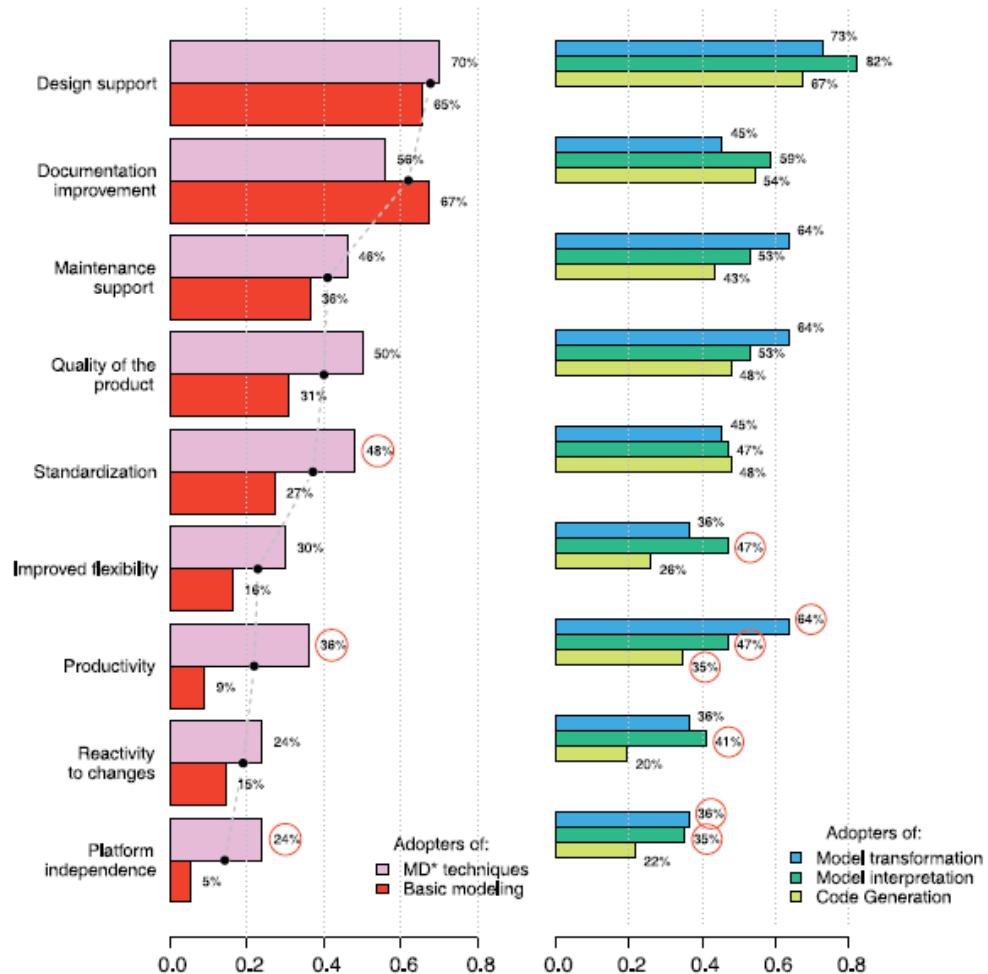
Η χρήση της μεθόδου αγγίζει πλέον νέες πτυχές στη βιομηχανία αλλά και στην ακαδημαϊκή έρευνα, στις οποίες η ανάγκη για αυτοματοποίηση και βελτίωση της απόδοσης αυξάνονται ιλιγγιωδώς. Ωστόσο, δεν έχει ξεκαθαριστεί ακόμη εάν η χρήση της μπορεί να χαρακτηριστεί ως βέλτιστη, αφού παρουσιάζει πολλά θετικά στοιχεία επιλύοντας μια πληθώρα προβλημάτων, όμως ταυτόχρονα εισάγει νέες αρνητικές πλευρές. Παρακάτω θα γίνει αναφορά τόσο στη θετική όσο και στην αρνητική σημασία της MDE.

Ξεκινώντας από τα πλεονεκτήματα της μεθόδου, δε θα μπορούσε να παραληφθεί η συμβολή της στην αύξηση της παραγωγικότητας. Πιο αναλυτικά, οι προγραμματιστές των συστημάτων έχουν πλέον τη δυνατότητα να χρησιμοποιούν τη MDE για να παράγουν αυτόματα και γρήγορα λογισμικό απαραίτητο για άλλες εφαρμογές που υλοποιούν, κερδίζοντας σημαντικό χρόνο και επικεντρώνοντας περισσότερο στην ουσία του έργου τους.

Παράλληλα, βελτιώνονται οι πτυχές της συντήρησης και της φορητότητας, καθώς πλέον παράγονται μοντέλα υψηλού επιπέδου, τα οποία δεν περιέχουν τέτοιου είδους περιορισμούς. Με αυτόν τον τρόπο διευκολύνεται η κατανόηση του έργου από άλλους και ταυτόχρονα καθίσταται απλή η αλλαγή της πλατφόρμας λειτουργίας του λογισμικού. Τα παραπάνω έχουν άμεσο αντίκτυπο στους οικονομικούς πόρους που χρειάζονται για την παραγωγή του έργου. Συγκεκριμένα, παρατηρείται μείωση του κόστους τόσο της ανάπτυξης, όσο και της συντήρησης του έργου. Παράλληλα, το προσωπικό που απασχολείται βελτιώνει το γνωστικό του επίπεδο αλλά και την προσωπική του απόδοση.

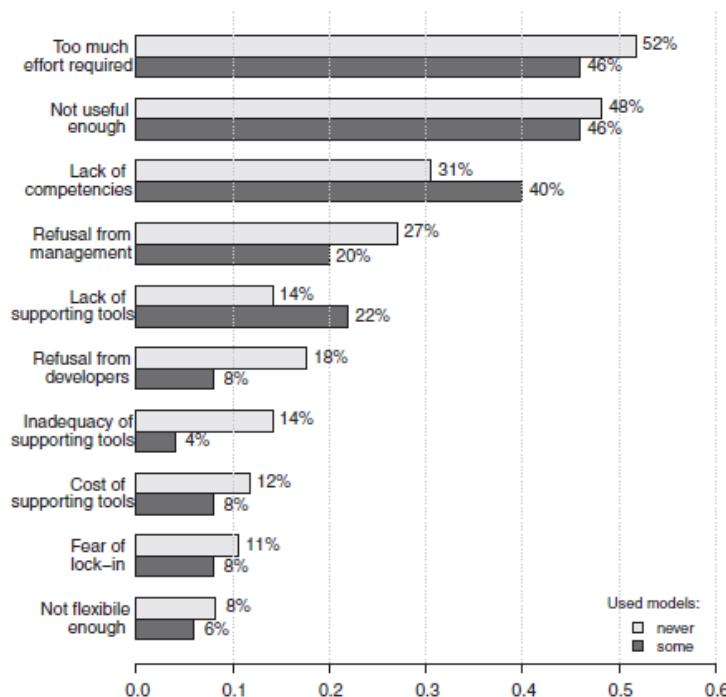
Ωστόσο, η χρήση της μεθόδου παρουσιάζει και αρνητικά σημεία. Λόγω της αυτοματοποίησης και άλλων προδιαγραφών, απαιτείται η παραγωγή μοντέλων, τα οποία θα έχουν γκάμα εφαρμογών και αντίστοιχα η δημιουργία των απαραίτητων μετασχηματισμών. Συνεπώς, το προσωπικό που θα υλοποιήσει αυτές τις ανάγκες πρέπει να έχει καλή γνώση του αντικειμένου, ώστε να μπορεί γρήγορα να προσαρμόσει σωστά τους μετασχηματισμούς για το εκάστοτε ζήτημα, επιβαρύνοντας έτσι το έργο με επιπλέον κόστος και χρόνο εκπαίδευσης. Παράλληλα, απαιτείται σημαντικός χρόνος για τη δοκιμή των μετασχηματισμών αυτών, καθώς υπάρχουν ορισμένοι φυσικοί περιορισμοί που δε σχετίζονται με το υλικό αλλά την πρακτική χρήση της εφαρμογής.

Τέλος, υπάρχουν επιπλέον πλεονεκτήματα και μειονεκτήματα της μεθόδου, τα οποία δεν παρουσιάζονται σε αυτήν την παράγραφο. Για να μπορέσει ο αναγνώστης να έχει μία πιο πλήρη εικόνα των πλεονεκτημάτων, αλλά και των μειονεκτημάτων της χρήσης της μεθόδου, παρουσιάζονται στη συνέχεια συγκεντρωμένα ορισμένα αποτελέσματα που σε έρευνα του πανεπιστημίου Università di Genova της Ιταλίας. Η έρευνα αφορά τη χρήση της μεθόδου στην ιταλική βιομηχανία και συγκρίνει τη νέα μέθοδο με άλλες παλιότερες οι οποίες χρησιμοποιούνται ακόμη στον τομέα της παραγωγής. Όπως είναι φανερό η μέθοδος παρουσιάζει καλύτερα αποτελέσματα στην πλειονότητα των συγκρινόμενων τιμών [7].



Σχήμα 9: Θετικές πτυχές της MDE, σε έρευνα αναφορικά με την Ιταλική Βιομηχανία

Σημαντικό είναι σε αυτό το σημείο να τονιστεί πως η MDE εφαρμόζεται τόσο από εξειδικευμένο προσωπικό, όσο κι από νέους χρήστες της μεθόδου και σε κάθε περίπτωση, η χρήση της μεθόδου επηρεάζει το προσωπικό. Στην ίδια έρευνα πραγματοποιήθηκε συλλογή και σύγκριση δεδομένων αναφορικά με τα προβλήματα και τις δυσκολίες που παρουσιάστηκαν κατά τη χρήση της μεθόδου. Πιο αναλυτικά, τα αποτελέσματα παρουσιάζονται στο παρακάτω Σχήμα 10 [13].



Σχήμα 10: Δυσκολίες και προβλήματα που περιορίζουν την υιοθέτηση της μεθόδου MDE

Συμπερασματικά, δε θα ήταν σωστό να θεωρήσουμε πως η MDE συνιστά τη λύση σε όλα τα προβλήματα και τα αντικείμενα των εφαρμογών. Βέβαια, παρόλο που φαίνεται αποθαρρυντικό, σύμφωνα με τους χρήστες, να εφαρμόζει κανείς τη μέθοδο, η MDE αποκτά ολοένα και μεγαλύτερο έδαφος στους τομείς της βιομηχανίας. Αυτό οφείλεται στο γεγονός πως τα πλεονεκτήματα της είναι περισσότερα και ιδιαίτερα σημαντικά στον τομέα της επιστήμης και της τεχνολογίας λογισμικού, έναντι των δυσκολιών που παρουσιάζει. Τέλος, πολλοί υποστηρίζουν πως οι παράγοντες που θα καθορίσουν τη χρήση της μεθόδου που θα ακολουθηθεί σε κάθε περίπτωση είναι αρκετοί και πρέπει να ελέγχονται διεξοδικά.

2.1.6 REST (Representational State Transfer)

Παρόλο που η REST είναι ιδιαίτερος κλάδος με πολλές πτυχές και προσεγγίσεις, για τους σκοπούς της παρούσας διπλωματικής, θα επικεντρωθούμε μόνο στα πολύ βασικά σημεία, τα οποία θα βοηθήσουν τον αναγνώστη της στα επόμενα κεφάλαια [14][15].

Αρχικά, ας ορίσουμε τον όρο *REST*. Η *Μεταφορά Αναπαραστάσεων Κατάστασης* (*Representational State Transfer* ή *REST*) συνιστά έναν τύπο αρχιτεκτονικής διαδικτυακών υπηρεσιών. Περιγράφεται ουσιαστικά ένας τρόπος διαλειτουργικότητας μεταξύ υπολογιστικών συστημάτων στο διαδίκτυο. Ο όρος *REST* πρωτοεμφανίστηκε το 2000 από τον Roy Fielding, ο οποίος περιέγραψε τον τρόπο με τον οποίο λειτουργούν τα διανεμημένα συστήματα πληροφορίας, όπως το διαδίκτυο. Ένα από τα σημαντικότερα πλεονεκτήματα αυτής της τεχνολογίας είναι η εκμετάλλευση μικρού *bandwidth*, καθιστώντας την ως την πιο κατάλληλη για χρήση στο Διαδίκτυο. Αυτός είναι και ο λόγος που η τεχνολογία *REST* γενικά προτιμάται από την τεχνολογία *SOAP* (*Simple Object Access Protocol*).

Ο κώδικας που επιτρέπει σε δύο προγράμματα λογισμικού να επικοινωνούν μεταξύ τους συνιστά μια *Διεπαφή Προγράμματος Εφαρμογής* (*Application Programming Interface* ή *API*) για τον ιστότοπο. Το *API* εξηγεί τον κατάλληλο τρόπο για έναν προγραμματιστή να γράψει ένα πρόγραμμα που ζητά υπηρεσίες από ένα λειτουργικό σύστημα ή άλλη εφαρμογή. Τα *API* λειτουργούν χρησιμοποιώντας αιτήσεις(*requests*) και απαντήσεις(*responses*). Όταν ένα *API* ζητά πληροφορίες από μια εφαρμογή ιστού ή έναν διακομιστή ιστού, θα λάβει μια απάντηση. Ένα *RESTful API* - που αναφέρεται επίσης ως μια υπηρεσία *RESTful web* - είναι λοιπόν ένα *API* που χρησιμοποιεί *HTTP requests*, ώστε να ανακτήσει ορισμένα δεδομένα (*GET*), να ανανεώσει τα αποθηκευμένα δεδομένα (*PUT*), να αποθηκεύσει ορισμένα δεδομένα (*POST*) ή να τα διαγράψει (*DELETE*).

Τέλος, το τελευταίο σημείο που θα επικεντρωθούμε εδώ είναι *API endpoints*. Τα ονόματα των *API endpoints* είναι η πιο ορατή πλευρά ενός *API* που συναντούν οι χρήστες και επιλέγονται με τρόπο που να εξυπηρετεί την καλύτερη επικοινωνία με το χρήστη, παρά με οποιοδήποτε άλλο χαρακτηριστικό του *Restful API*. Τι ακριβώς είναι όμως ένα *endpoint*? Είναι το ένα άκρο ενός καναλιού επικοινωνίας. Όταν ένα *API* αλληλοεπιδρά με ένα άλλο σύστημα, τα σημεία επαφής αυτής της επικοινωνίας θεωρούνται *endpoints*. Για τα *API*, ένα *endpoint* μπορεί να περιλαμβάνει μια διεύθυνση URL ενός διακομιστή ή μιας υπηρεσίας και από εκεί μπορούν να έχουν πρόσβαση στους πόρους(*resources*) που χρειάζονται για να εκτελέσουν τη λειτουργία τους.

2.2 Τεχνολογικό Υπόβαθρο

2.2.2 Acceleo Template Language

Η γλώσσα *Acceleo* είναι μία ανοιχτού κώδικα γλώσσα, ανεπτυγμένη στην πλατφόρμα *Eclipse* από την *Obeo - Eclipse Strategic Member* - η οποία επιτρέπει στους προγραμματιστές να χρησιμοποιούν με ευκολία μια προσέγγιση βασισμένη στο μοντέλο για την κατασκευή εφαρμογών. Πρόκειται για μια γλώσσα που βασίζεται στη χρήση προτύπων κειμένου και αποτελεί μια εφαρμογή του προτύπου *MOFM2T*, για την πραγματοποίηση μετασχηματισμού μοντέλου προς κείμενο [27].

Πιο συγκεκριμένα, η τελευταία έκδοση *Acceleo 3*, είναι ένα πρότυπο παραγωγής κώδικα, το οποίο επιτρέπει στον προγραμματιστή την παραγωγή γενικών προτύπων κώδικα. Ως είσοδος στα πρότυπα αυτά εισάγονται ένα ή περισσότερα μοντέλα, τα οποία συμμορφώνονται σε ένα ή περισσότερα μετά-μοντέλα και έχουν όλες τις απαραίτητες πληροφορίες. Η *Acceleo* αποτελείται από δύο βασικούς τύπους δομών, τα πρότυπα (*template*) και τα ερωτήματα (*queries*). Μία μονάδα *Acceleo* είναι ένα αρχείο *.mtl* το οποίο περιέχει *templates*, τα οποία θα παράγουν τον ζητούμενο κώδικα, και *queries*, τα οποία χρησιμοποιούνται για την εξαγωγή πληροφοριών από τα τελικά μοντέλα. Στο Σχήμα 11, που ακολουθεί παρουσιάζεται ένα στιγμιότυπο ενός *.mtl* αρχείου.

```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/3.0.0/UML')]

[template public generate(aClass : Class)
[file (aClass.classFileName(), false)]
[package [aClass.containingPackages().name->sep('.')]]

// [protected ('imports')]
// [/protected]

public class [aClass.name.toUpperFirst()]
[for (p: Property | aClass.attribute) separator('\n')
    private [p.type.name/] [p.name/];
[/for]

[for (p: Property | aClass.attribute) separator('\n')
    public [p.type.name/] get[p.name.toUpperFirst()]/() {
        return this.[p.name];
    }
[/for]

[for (o: Operation | aClass.ownedOperation) separator('\n')
    public [o.type.name/] [o.name/] () {
        // [protected (o.name)]
        // TODO should be implemented
        // [/protected]
    }
[/for]
}
[/file]
[/template]

[query public classFileName(aClass : Class) : String =
    aClass.qualifiedName().replaceAll('\\.', '/').concat('.java')
/]

[query public qualifiedName(aClass : Class) : String =
    aClass.containingPackages().name->sep('.')->including('..')->including(aClass.name)->toString()
/]

[query public containingPackages(aClass : Class) : Sequence(Package) =
    aClass.ancestors(Package)->reject(oclIsKindOf(Model))->reverse()
/]
```

Σχήμα 11: Παρουσίαση *.mtl* αρχείου

2.2.3 EMF (Ecore Modeling Framework)

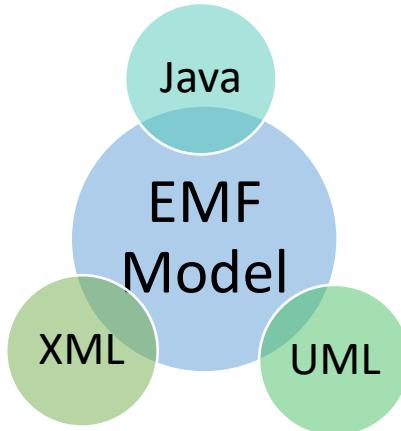
Το *EMF*(*Ecore Modeling Framework*) είναι ένα πλαίσιο μοντελοποίησης και δημιουργίας κώδικα για την κατασκευή εργαλείων και άλλων εφαρμογών που βασίζονται σε ένα δομημένο μοντέλο δεδομένων. Από μια προδιαγραφή μοντέλου που περιγράφεται σε μορφή *XMI*, το *EMF* παρέχει εργαλεία και *run time* στιγμιότυπα για την παραγωγή ενός συνόλου *Java* κλάσεων για το μοντέλο, μαζί με ένα σύνολο *Adapter* κλάσεων που επιτρέπουν την προβολή και επεξεργασία βασισμένων σε εντολές του μοντέλου και έναν βασικό επεξεργαστή.

Το *EMF* είναι ένα κοινό πλαίσιο για μοντέλα δεδομένων και αποτελείται από τρία βασικά τμήματα:

- *EMF*: το βασικό πλαίσιο *EMF* περιλαμβάνει ένα μετα-μοντέλο (*Ecore*) για την περιγραφή μοντέλων και υποστήριξης χρόνου εκτέλεσης για τα μοντέλα.
- *EMF.Edit*: το πλαίσιο *EMF.Edit* περιλαμβάνει γενικές επαναχρησιμοποιούμενες τάξεις για *building editors* για μοντέλα *EMF*. Το *EMF.Edit* προσφέρει ένα πλαίσιο εντολών, που περιλαμβάνει ένα σύνολο γενικών τάξεων υλοποίησης εντολών για *building editors* που υποστηρίζουν πλήρως αυτόματη αναίρεση και επανάληψη.
- *EMF.Codegen*: Η ευκολία παραγωγής κώδικα *EMF* είναι σε θέση να παράγει όλα όσα απαιτούνται για την κατασκευή ενός πλήρους *editor* για ένα μοντέλο *EMF*. Περιλαμβάνει ένα *GUI* από το οποίο μπορούν να οριστούν κάποιες επιλογές και να επικαλεστούν οι γεννήτριες κώδικα. Η εγκατάσταση παραγωγής αξιοποιεί το στοιχείο *JDT (Java Development Tooling)* του *Eclipse*.

Υποστηρίζονται τρία επίπεδα δημιουργίας κώδικα:

- *Model*: παρέχει *Java* διασυνδέσεις και τάξεις υλοποίησης για όλες τις κλάσεις του μοντέλου, καθώς και κλάση υλοποίησης πακέτου και κλάση εργοστασίου (*meta data*).
- *Adapter*: παράγει τάξεις υλοποίησης (που ονομάζονται προμηθευτές αντικειμένων) που προσαρμόζουν τις κατηγορίες μοντέλων για επεξεργασία και εμφάνιση.
- *Editor*: παράγει ένα σωστά δομημένο πρόγραμμα επεξεργασίας που συμμορφώνεται με το προτεινόμενο στυλ για επεξεργαστές μοντέλων *Eclipse EMF* και χρησιμεύει ως σημείο εκκίνησης από το οποίο μπορεί να ξεκινήσει η προσαρμογή.

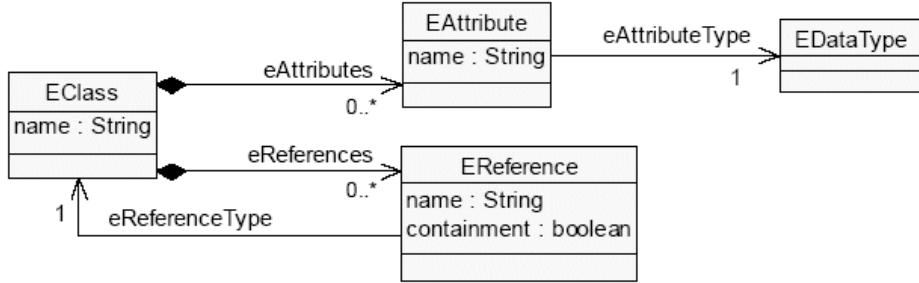


Σχήμα 12: EMF και ενοποιήση των Java, XML, UML

Τέλος, το αξιοσημείωτο για το *EMF* είναι πως ενοποιεί τις τεχνολογίες *Java*, *XML* και *UML* δίνοντας με αυτόν τον τρόπο τη δυνατότητα στο χρήστη-προγραμματιστή να εναλλάσσεται μεταξύ τους. Αυτό συμβαίνει καθώς πρόκειται για τη ίδια πληροφορία με διαφορετικό τρόπο παρουσίασης [16].

2.2.4 Ecore Meta-Model

Το μετα-μοντέλο *Ecore* παρουσιάζεται ως βασικό στοιχείο πολλών μοντελοστραφών εφαρμογών και επεκτάσεων στο *Eclipse*. Στο Σχήμα 13 που ακολουθεί, παρουσιάζεται ένα απλοποιημένο διάγραμμα κλάσεων του *Ecore* μεταμοντέλου. Το στοιχείο *EClass* μοντελοποιεί μια κλάση η οποία θα έχει ένα όνομα. Επιπρόσθετα, το στοιχείο *EAttribute* αναπαριστά το χαρακτηριστικό που μπορεί να έχει μία κλάση. Είναι δυνατό, μια κλάση να έχει από κανένα έως πολλά χαρακτηριστικά. Το στοιχείο *EReferences* αντιπροσωπεύει τις συσχετίσεις μεταξύ των κλάσεων *EClass* τόσο στην περίπτωση που η συσχέτιση είναι περιοριστική, όσο και όχι. Τέλος, το στοιχείο *EDataType* αντιστοιχεί στην πληροφορία για τον τύπο του κάθε χαρακτηριστικού μίας κλάσης [16].



Σχήμα 13: Απλοποιημένο παράδειγμα Ecore meta-model

2.2.5 Redis Database

To *Redis* είναι η βάση δεδομένων που χρησιμοποιείται στη συγκεκριμένη διπλωματική εργασία. Πιο αναλυτικά, πρόκειται για μια *in-memory*, πολύ γρήγορη, μη σχεσιακή βάση δεδομένων που μπορεί να αποθηκεύει μια χαρτογράφηση(*Mapping*) κλειδιών σε πέντε διαφορετικούς τύπους τιμών. To *Redis* υποστηρίζει μόνιμη αποθήκευση σε μνήμη στο δίσκο, αναπαραγωγή σε κλίμακα επιδόσεων ανάγνωσης και απόκρυψη από την πλευρά του πελάτη σε κλίμακα απόδοσης γραφής. Οι λόγοι για τους οποίους επιλέχθηκε το *Redis* είναι αρκετοί.

Αρχικά, για την αποθήκευση δεδομένων χρησιμοποιούνται είτε ένα *LIST*, είτε ένα *SET*, οπότε η προσθήκη και η αφαίρεση στοιχείων μπορεί να γίνει απευθείας. Ο κώδικας που απαιτείται είναι εν τέλει μικρότερος, πιο κατανοητός και πιο εύκολος στη συντήρηση, αλλά παράλληλα και πιο γρήγορος, καθώς δεν χρειάζεται να διαβάζεται μια βάση δεδομένων για να ενημερωθούν τα δεδομένα. Υπάρχουν πολλές άλλες περιπτώσεις όπου το *Redis* είναι πιο αποδοτικό ή πιο εύκολο στη χρήση από τις σχεσιακές βάσεις δεδομένων. Για παράδειγμα, με το *Redis*, μπορεί να αποφευχθεί η χρήση γραμμών(*rows*) για την αποθήκευση μακροσκελών τμημάτων δεδομένων τα οποία συχνά αναφέρονται ως συναθροίσεις(*aggregates*). Πιο απλά, με το *Redis* χρησιμοποιούνται απλές *build-in* εντολές, κάνοντας την όλη διαδικασία πολύ πιο εύκολη και γρήγορη. Παράλληλα, ένα ακόμη παράδειγμα είναι το γεγονός ότι με τη χρήση του *Redis*, μπορεί να αποφευχθεί η αποθήκευση προσωρινών δεδομένων, η προσπέλαση αυτών και η διαγραφή τους, βελτιώνοντας έτσι τη συνολική απόδοση της βάσης. Τέλος, η λειτουργία του *Redis* υποστηρίζεται σε πολλές γλώσσες προγραμματισμού, καθιστώντας το ιδιαίτερα εύκολη στη χρήση.

Παρακάτω παρατίθεται μία σύγκριση ορισμένων πολύ διαδεδομένων, σχεσιακών και μη, βάσεων δεδομένων, για να γίνει πιο κατανοητή η διαφορά που υπάρχουν μεταξύ τους. Η σύγκριση γίνεται χρησιμοποιώντας την αγγλική ορολογία για καλύτερη κατανόηση των όρων [17].

Πίνακας 1: Παράθεση σύγκρισης διαφόρων βάσεων δεδομένων

Name	Type	Data storage options	Query types	Additional features
Redis	In-memory non-relational database	Strings, lists, sets, hashes, sorted sets	Commands for each data type for common access patterns, with bulk operations, and partial transaction support	Commands for each data type for common access patterns, with bulk operations, and partial transaction support
memcached	In-memory key-value cache	Mapping of keys to values	Commands for create, read, update, delete, and a few others	Multithreaded server for additional performance
MySQL	Relational database	Databases of tables of rows, views over tables, spatial and third-party extensions	SELECT, INSERT, UPDATE, DELETE, functions, stored procedures	ACID compliant (with InnoDB), master/slave and master/master replication
PostgreSQL	Relational database	Databases of tables of rows, views over tables, spatial and third-party extensions, customizable types	SELECT, INSERT, UPDATE, DELETE, built-in functions, custom stored procedures	ACID compliant, master/slave replication, multi-master replication (third party)
MongoDB	On-disk non-relational document store	Databases of tables of schema-less BSON documents	Commands for create, read, update, delete, conditional queries, and more	Supports map-reduce operations, master/slave replication, sharding, spatial indexes

2.2.6 Flask Framework

Το *Flask* είναι ένα micro web framework υλοποιημένο στην Python και είναι σχεδιασμένο βάσει των *Werkzeug* και *Jinja 2*. Ο χαρακτηρισμός micro framework δηλώνει πως το *Flask* δεν απαιτεί συγκεκριμένα εργαλεία ή άλλες βιβλιοθήκες, εκτός από ορισμένες πολύ βασικές όπως το *bottom.py*. Ακόμη, δε διαθέτει επίπεδο αφαίρεσης βάσης δεδομένων (*database abstraction layer*), φόρμα επικύρωσης (*form validation*) ή άλλα στοιχεία, όπου προϋπάρχουσες βιβλιοθήκες παρέχουν κοινές λειτουργίες. Ωστόσο, το *Flask* υποστηρίζει επεκτάσεις που μπορούν να προσθέσουν λειτουργίες σαν να είχαν σχεδιαστεί για το ίδιο. Ορισμένα από τα βασικά σημεία του *Flask* είναι ότι περιέχει διακομιστή ανάπτυξης(*development server*) και πρόγραμμα εντοπισμού σφαλμάτων(*debugger*), ενώ υποστηρίζει τη δυνατότητα δοκιμών(*testing*). Παράλληλα, παρέχει δυνατότητες ανάπτυξης RESTful requests, αλλά και δυνατότητες επέκτασης προς κάθε επιθυμητή κατεύθυνση. Τέλος, έχει εκτεταμένο documentation, στο οποίο μπορεί να ανατρέξει κανείς για να κατανοήσει τις διάφορες παραμέτρους και λειτουργίες του. Για τους παραπάνω λόγους, χρησιμοποιήθηκε στην παρούσα διπλωματική για την ανάπτυξη του RESTful API, ώστε να μπορεί ο χρήστης να συλλέγει όποια πληροφορία επιθυμεί για τους αισθητήρες που περιέχει το σύστημα που έφτιαξε [18].

2.2.7 R4A Platform

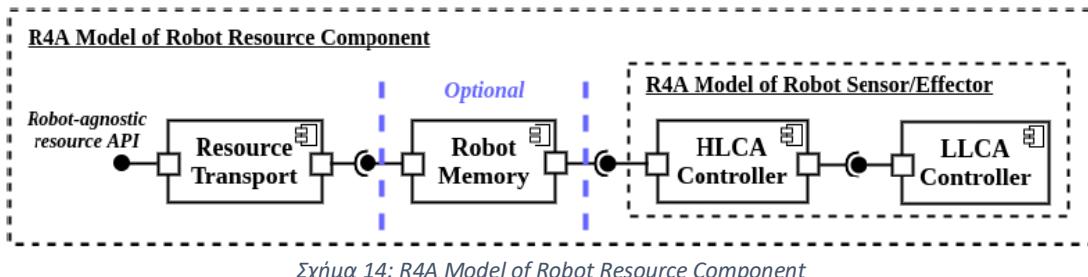
Η παρούσα διπλωματική εργασία είναι βασισμένη στην πλατφόρμα που έχει υλοποιήσει η ομάδα *R4A* του Αριστοτελείου Πανεπιστημίου Θεσσαλονίκης, χάρη στην οποία πήρε και το όνομα της. Η συγκεκριμένη πλατφόρμα έρχεται να φέρει τη λύση στην ανάπτυξη ρομποτικών εφαρμογών μέσω των robot agnostic APIs [19].

Πιο αναλυτικά, η πλατφόρμα αυτή παρέχει τυποποίηση της διαδικασίας ανάπτυξης ρομποτικών εφαρμογών, παρέχοντας ένα *component-based* λογισμικό, που θα επιτρέπει τον έλεγχο ενός ρομπότ και θα συνιστά ένα εργαλείο ικανό να μειώσει το χρόνο παραγωγής λογισμικού για ρομπότ. Η πλατφόρμα έχει υλοποιηθεί σε *Python 2.7* και για το λόγο αυτό και σε αυτή τη διπλωματική υιοθετήθηκε η ίδια γλώσσα προγραμματισμού.

Το API που παρέχεται από τη συγκεκριμένη πλατφόρμα δίνει τη δυνατότητα να ανακτηθούν όλες εκείνες οι απαραίτητες πληροφορίες που απαιτούνται τόσο για το ρομπότ, όσο για όλη τη λειτουργικότητα του αλλά και να επέμβουν σε αυτή. Στην παρούσα εργασία δόθηκε έμφαση συγκεκριμένα στις κλήσεις σχετικά με τη συλλογή δεδομένων των αισθητήρων που υπάρχουν στο ρομπότ *NAO*, στο οποίο δοκιμάστηκε και το παραγόμενο λογισμικό. Με τον τρόπο αυτό έγινε εφικτή η επικοινωνία με το ρομπότ και η επέμβαση σε αυτό, ώστε να

ενεργοποιούνται οι αισθητήρες που επέλεξε ο χρήστης και να συλλέγουν δεδομένα, σύμφωνα με τις προϋποθέσεις που εκείνος έθεσε και πάντοτε συμβατές με τις προκαθορισμένες δυνατότητες των αισθητήρων.

Στη συνέχεια, παρουσιάζεται σύντομα μία αναπαράσταση της πλατφόρμας R4A (σχημ.14). Κάθε τμήμα του παρακάτω μοντέλου αντιστοιχεί σε ένα μοναδικό πόρο ενός ρομπότ. Συνεπώς, όλα αυτά τα τμήματα συνδυάζονται και συνιστούν το τελικό API της πλατφόρμας. Το εμφαλευμένο τμήμα που φαίνεται στο Σχήμα 14, αναπαριστά μία αφαίρεση των αισθητήρων και των κινητήρων ενός ρομπότ. Αυτή η αφαίρεση υλοποιείται μέσω των HLCA Controllers που έχουν παραχθεί για το σκοπό αυτό. Αξιοσημείωτο είναι πως δεν απαιτείται μνήμη για κάθε πόρο του ρομπότ και για το λόγο αυτό δεν είναι υποχρεωτική η χρήση της μνήμη του ρομπότ [28].

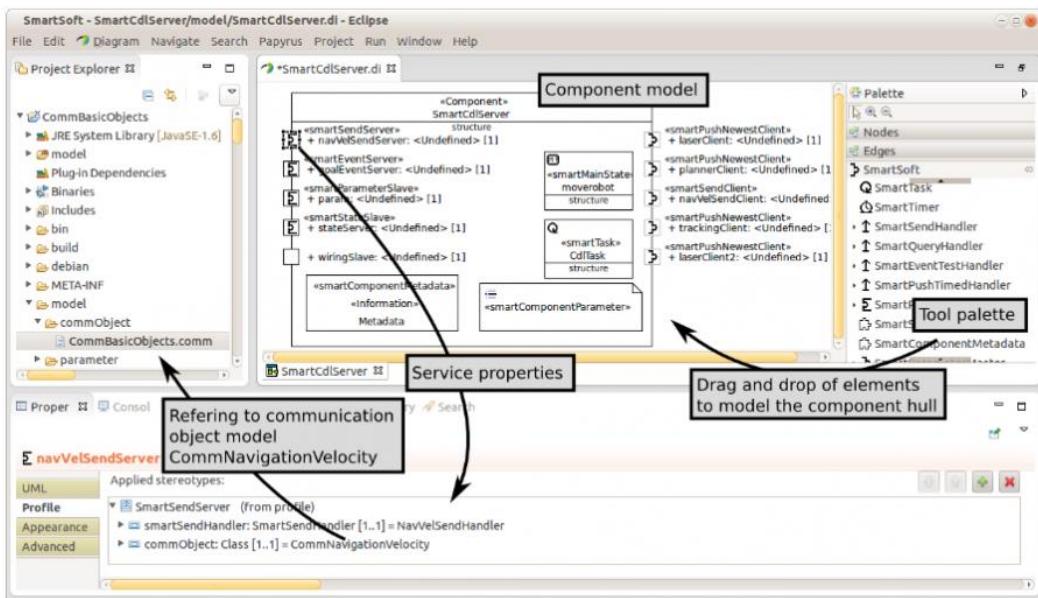


3 Σχετική Βιβλιογραφία (*State of the art*)

3.1 SmartSoft

To *SmartSoft* είναι μια προσέγγιση βασισμένη σε εξαρτήματα(*components*) και επικεντρωμένη σε υπηρεσίες για ρομποτικό λογισμικό. Θεωρεί ως πυρήνα ενός μοντέλου ρομποτικών στοιχείων την παραπάνω προσέγγιση που βασίζεται σε πρότυπα επικοινωνίας. Το *SmartSoft* βοηθά τον προγραμματιστή των εξαρτημάτων, τον κατασκευαστή εφαρμογών και τον τελικό χρήστη να δημιουργεί και να χρησιμοποιεί κατανευμημένα εξαρτήματα με τέτοιο τρόπο ώστε η σημασιολογία της διεπαφής των εξαρτημάτων να είναι προκαθορισμένη από τα πρότυπα επικοινωνίας, ανεξάρτητα από το πού εφαρμόζονται. Η μεγάλη διαφορά με άλλες προσεγγίσεις είναι πως το Software επιτρέπει τη διασύνδεση των εξαρτημάτων δυναμικά, δηλαδή, σε χρόνο εκτέλεσης. Αυτό επιτρέπει τη δημιουργία συστημάτων που βασίζονται σε τυποποιημένα στοιχεία των οποίων η αλληλεπίδραση μπορεί να προσαρμοστεί σύμφωνα με το τρέχον πλαίσιο και τις απαιτήσεις.

To *SmartSoft MDSD Toolchain* είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (*Integrated Development Environment* ή *IDE*) για την ανάπτυξη λογισμικού ρομποτικής που υποστηρίζει τον διαχωρισμό των ρόλων και καλύπτει τη διαδικασία ανάπτυξης μοντέλων αντικειμένων, εξαρτημάτων και συστημάτων επικοινωνίας.



Σχήμα 15: Στιγμιότυπο οθόνης του SmartMDSD Toolchain κατά τη μοντελοποίηση ενός εξαρτήματος

Ο προγραμματιστής εξαρτημάτων υλοποιεί και χρησιμοποιεί τις υπηρεσίες των εξαρτημάτων. Με βάση το διαμορφωμένο πλέον εξάρτημα, το *Smart MDSD⁽³⁾ Toolchain* παράγει το πρότυπο συνιστώσων και παρέχει κενές λειτουργίες για να συμπληρώσει ο χρήστης τον κώδικα για την παρεχόμενη λειτουργικότητα. Έτσι δημιουργείται, ο πηγαίος κώδικας.

3.2 Proteus

Το πρόγραμμα *Proteus⁽⁵⁾* υποστηρίζεται από τη γαλλική ρομποτική κοινότητα (*Groupement De Recherche Robotique*) και αφορά την ανάπτυξη ενός εργαλείου *MDE* για ρομποτικό σχεδιασμό. Αυτό που το διαφοροποιεί είναι η χρήση των οντολογιών για την αντιπροσώπευση γνώσεων σχετικά με τα ρομποτικά συστήματα. Οι οντολογίες *Proteus* είναι μια προσπάθεια να επισημοποιήσουμε το λεξιλόγιο των ειδικών στα κινούμενα ρομποτικά συστήματα, προκειμένου να μπορούμε να μοντελοποιήσουμε τις αρχιτεκτονικές ελέγχου ρομπότ βάσει μίας ιδέας (π.χ. αισθητήρας, προγραμματιστής κίνησης, κλπ.) και όχι μόνο βάσει λογισμικού (αλγόριθμος). Στο σημείο αυτό αξίζει να επισημανθεί πως γίνονται παρόμοιες προσπάθειες από το *IEEE Standards Association⁽⁴⁾*, το οποίο περιέχει το *CORA (Core Ontologies for Robotics and Automation)* [20].

Το έργο *PROTEUS* θα συνδυάσει πολλά εργαλεία για να προσφέρει μια πλήρη ομοιόμορφη εργαλειοθήκη για την ανάπτυξη ρομποτικών εφαρμογών από τη μοντελοποίηση μέχρι την προσομοίωση και την ανάπτυξη λογισμικού σε πραγματικά ρομποτικά συστήματα. Τα περισσότερα από αυτά είναι λογισμικό ανοιχτού κώδικα. Βασίζεται σε ένα πρότυπο *UML* για τον ορισμό του *PIM⁽⁶⁾ (Platform Independent Model)* της λειτουργικής αρχιτεκτονικής ρομπότ. Για την παραγωγή κώδικα, τα στοιχεία του μοντέλου *PIM* πρέπει να κατανεμηθούν σε μια πλατφόρμα εκτέλεσης, όπως ένα middleware και ένα προσομοιωτή. Για παράδειγμα, το στοιχείο που αντιπροσωπεύει τη λειτουργικότητα του ρομπότ και τις δραστηριότητες ελέγχου κατανέμεται σε ενδιάμεσο λογισμικό που βασίζεται σε λογισμικό, ενώ το στοιχείο που αντιπροσωπεύει το ρομποτικό εξοπλισμό κατανέμεται σε έναν προσομοιωτή.

⁽³⁾ <https://wiki.servicerobotik-ulm.de/about-smartsoft:approach>

⁽⁴⁾ Η IEEE Standards Association είναι μια κορυφαία οργάνωση που αναπτύσσει και προωθεί τις παγκόσμιες τεχνολογίες, μέσω της παγκόσμιας κοινότητας ηλεκτρολόγων και ηλεκτρονικών μηχανικών (*Institute of Electrical and Electronics Engineers* ή *IEEE*).

⁽⁵⁾ <http://www.anr-proteus.fr/?q=node/109>

⁽⁶⁾ Στον τομέα της τεχνολογίας λογισμικού, *PIM* ονομάζεται το μοντέλο ενός συστήματος λογισμικού, το οποίο είναι ανεξάρτητο της τεχνολογικής πλατφόρμας που χρησιμοποιήθηκε για να υλοποιηθεί.

3.3 MathWorks Simulink®

To *Simulink*® είναι ένα περιβάλλον μπλοκ διαγράμματος για προσομοίωση πολλαπλών τομέων και σχεδιασμό βάσει μοντέλου. Υποστηρίζει σχεδιασμό σε επίπεδο συστήματος, προσομοίωση, αυτόματη δημιουργία κώδικα και συνεχή δοκιμή και επαλήθευση των ενσωματωμένων συστημάτων. To *Simulink* παρέχει ένα γραφικό επεξεργαστή, προσαρμόσιμες βιβλιοθήκες μπλοκ και λύτες για μοντελοποίηση και προσομοίωση δυναμικών συστημάτων. Είναι ενσωματωμένο με το *MATLAB*®, επιτρέποντάς να ενσωματωθούν οι αλγόριθμοι από αυτό σε μοντέλα και να εξαχθούν αποτελέσματα προσομοίωσης σε αυτό για περαιτέρω ανάλυση.

Αξίζει να τονιστεί πως το *Simulink*®⁽⁷⁾ θεωρείται ένα καλά δοκιμασμένο και πολύ αξιόπιστο εργαλείο λογισμικού το οποίο χρησιμοποιείται ευρέως σε πολλές εφαρμογές στο χώρο της βιομηχανίας. Παρακάτω θα αναφερθούν ενδεικτικά ορισμένα εργαλεία που μπορούν να χρησιμοποιηθούν συνδυαστικά με το *Simulink*® για να παράγουν κώδικα από ένα μοντέλο ή από άλλο κώδικα.

To MATLAB Coder™⁽⁸⁾ παράγει κώδικα C και C++ από κώδικα *MATLAB*® για διάφορες πλατφόρμες υλικού, από ηλεκτρονικούς υπολογιστές μέχρι ενσωματωμένο υλικό. Υποστηρίζει το μεγαλύτερο μέρος της γλώσσας *MATLAB* και ένα ευρύ φάσμα εργαλειοθηκών.

To *Simulink Coder*™⁽⁹⁾ (πρώην *Real-Time Workshop*®) δημιουργεί και εκτελεί κώδικα C και C++ από μοντέλα *Simulink*®, χάρτες *Stateflow*® και λειτουργίες *MATLAB*®. Ο παραγόμενος πηγαίος κώδικας μπορεί να χρησιμοποιηθεί σε εφαρμογές σε πραγματικό χρόνο και σε μη πραγματικό χρόνο, συμπεριλαμβανομένης της επιτάχυνσης προσομοίωσης, της σύντομης σύνθεσης πρωτοτύπων και των δοκιμών hardware-in-the-loop.

To *Embedded Coder*®⁽¹⁰⁾ παράγει κατανοητό και γρήγορο κώδικα C και C++ για ενσωματωμένους επεξεργαστές που χρησιμοποιούνται στη μαζική παραγωγή. Επεκτείνει το *MATLAB Coder*™ και το *Simulink Coder*™ με προηγμένες βελτιστοποιήσεις για τον ακριβή έλεγχο των λειτουργιών, των αρχείων και των δεδομένων που δημιουργούνται. Αυτές οι βελτιστοποιήσεις αυξάνουν την αποτελεσματικότητα του κώδικα και διευκολύνουν την ενσωμάτωση με κωδικό κληρονομιάς, τύπους δεδομένων και παραμέτρους βαθμονόμησης.

⁽⁷⁾ <https://uk.mathworks.com/help/simulink/index.html>

⁽⁸⁾ <https://uk.mathworks.com/products/matlab-coder.html>

⁽⁹⁾ <https://www.mathworks.com/products/simulink-coder.html>

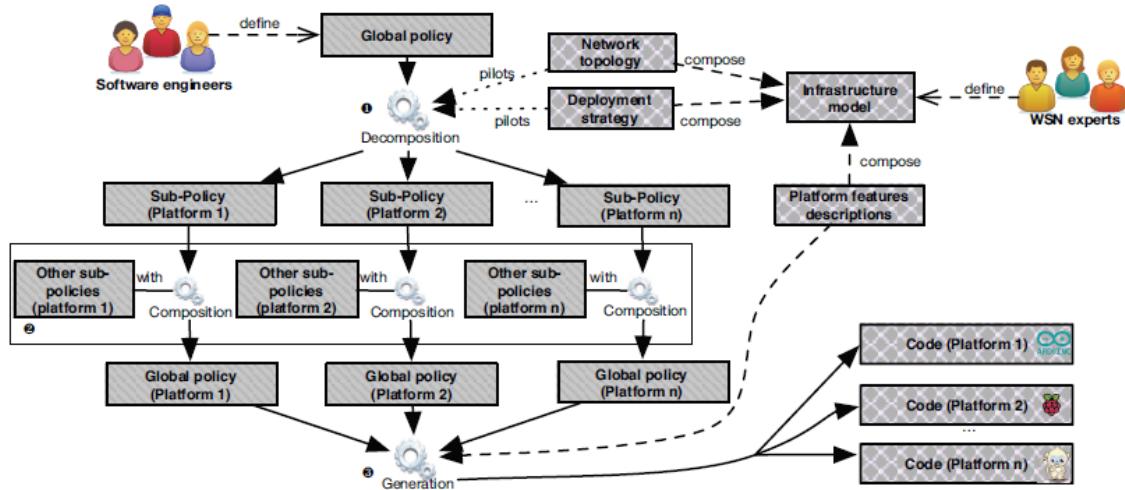
⁽¹⁰⁾ <https://uk.mathworks.com/products/embedded-coder.html>

3.4 DEPOSIT (Data collEction POlicies for Sensing InfrasTructures)

Πρόκειται για μια αυτοματοποιημένη προσέγγιση που υποστηρίζει τον ορισμό των πολιτικών συλλογής δεδομένων σε υψηλότερο επίπεδο αφαίρεσης, την αντιπροσώπευση των διαφόρων πλατφορμών και της τοπολογίας του δικτύου, και την αυτόματη σύνθεση και ανάπτυξη πολιτικών ελέγχοντας ετερογενείς υποδομές αισθητήρων που ακολουθούν διαφορετικές στρατηγικές. Αξιοσημείωτο είναι στο σημείο αυτό πως στο συγκεκριμένο εργαλείο, δε λαμβάνεται υπόψιν για την παραγωγή κώδικα ένας συγκεκριμένος αισθητήρας, αλλά μία πιο αφηρημένη προσέγγιση του. Ακόμη, αξίζει να τονιστεί πως δε μοντελοποιείται μόνο το σύνολο των αισθητήρων, αλλά παράλληλα και το δίκτυο τους.

Σε κάθε περίπτωση, οι μηχανικοί ορίζουν μία γενική πολιτική η οποία μπορεί να προσαρμοστεί σε επιμέρους ανάγκες. Αρχικά επιλέγουν κατηγορίες αισθητήρων για μία πλατφόρμα (μία τοπολογία αισθητηρίων) που επιθυμούν κι έπειτα ορίζουν τις προδιαγραφές του συστήματος. Κάθε φορά, λοιπόν, το λογισμικό DEPOSIT⁽¹¹⁾ δέχεται δύο εισόδους και ως έξοδο δημιουργεί αυτόμata τον απαραίτητο κώδικα. Είναι πολύ σημαντικό να τονιστεί εδώ η σημασία του εργαλείου, πως μπορεί αυτή η παραγωγή κώδικα να αφορά πολλές διαφορετικές πλατφόρμες.

Στο Σχήμα 16 παρακάτω φαίνεται η επισκόπηση της αρχιτεκτονικής του προγράμματος DEPOSIT [21].

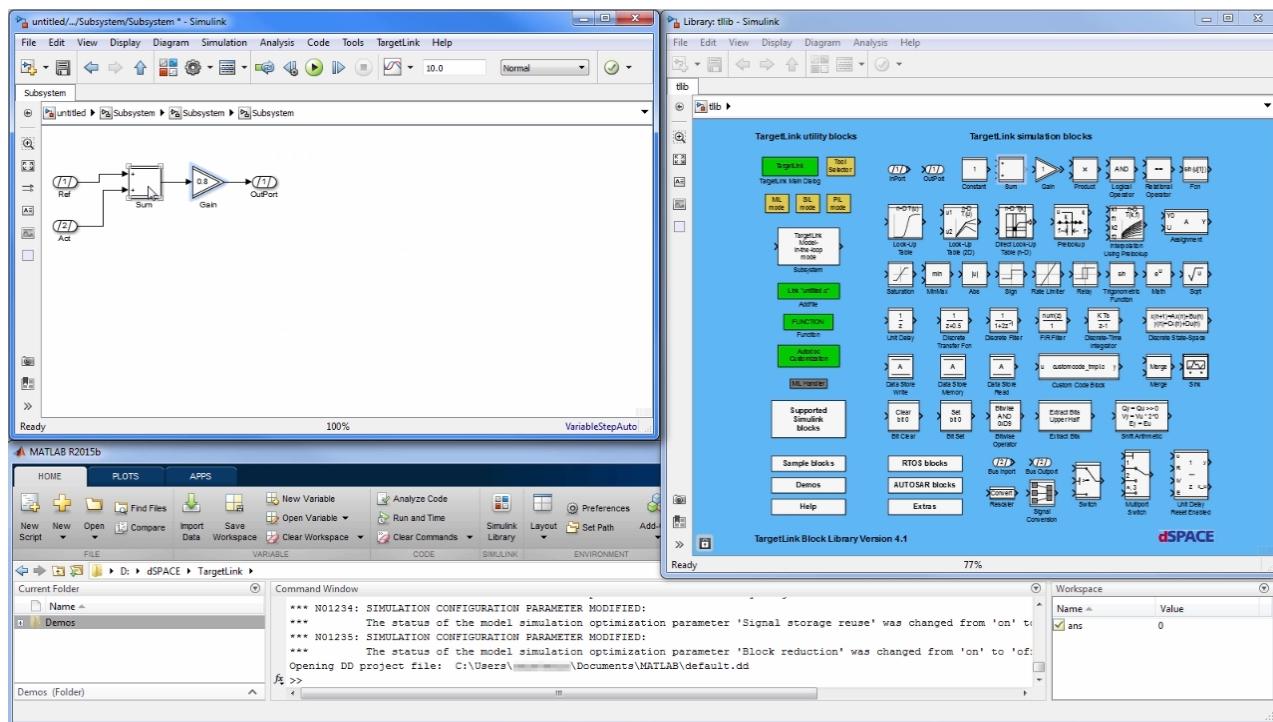


Σχήμα 16: Επισκόπηση αρχιτεκτονικής DEPOSIT

⁽¹¹⁾ <https://github.com/ace-design/DEPOSIT/blob/master/README.md>

3.5 TargetLink

Το TargetLink⁽¹²⁾ είναι ένα πρόγραμμα σχεδιασμένο από την εταιρεία dSpace, το οποίο δημιουργεί κώδικα παραγωγής (κώδικα C) κατευθείαν από το γραφικό περιβάλλον ανάπτυξης MATLAB® / Simulink / Stateflow, δηλαδή μετατρέπει γραφικά μοντέλα σε κώδικα παραγωγής υψηλότερης ποιότητας. Οι επιλογές δημιουργίας κώδικα C κυμαίνονται από τον απλό κώδικα ANSI C έως τον βελτιστοποιημένο κώδικα σταθερής ή κινητής υποδιαστολής για τις πλατφόρμες AUTOSAR. Οι ευέλικτες επιλογές διαμόρφωσης κώδικα διασφαλίζουν ότι ο κώδικας παραγωγής μπορεί να χειριστεί τους περιορισμούς του επεξεργαστή.

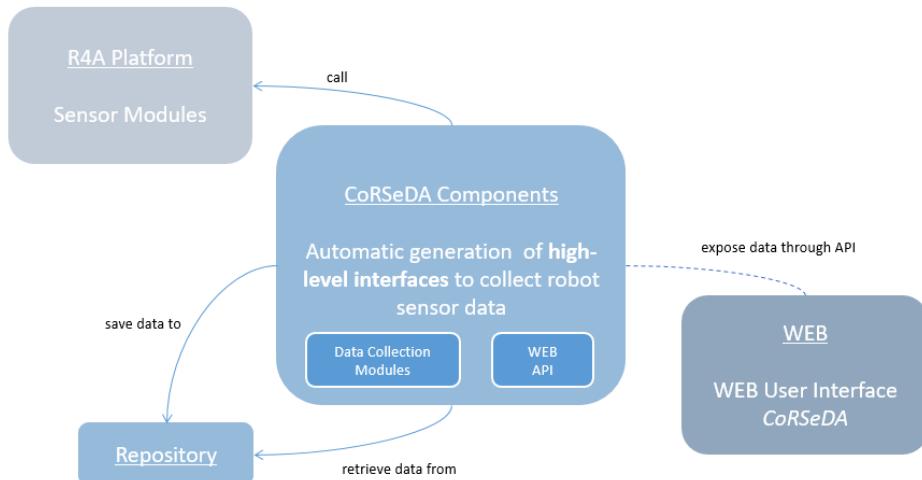


Σχήμα 17: Στιγμιότυπο δημιουργίας μοντέλου χρησιμοποιώντας το TargetLink

⁽¹²⁾ <https://www.dspace.com/en/pub/home/products/sw/pcgs/targetli.cfm>

4 Μεθοδολογία

Στο κεφάλαιο αυτό θα παρουσιαστεί αναλυτικά όλη η διαδικασία για την υλοποίηση του τελικού συστήματος CoRSeDa. Στο *Σχήμα 18* φαίνεται μία high-level αναπαράσταση του συστήματος. Ορισμένα *components* του συστήματος χρησιμοποιούν τις συναρτήσεις αισθητήρων της πλατφόρμας R4A προκειμένου να πραγματοποιηθεί η επικοινωνία με το ρομπότ και να συλλεχθούν τα δεδομένα. Στη συνέχεια τα δεδομένα αυτά αποθηκεύονται στο *repository* του ρομπότ και εκτίθενται στο διαδίκτυο μέσω του παραγόμενου *API*. Με τον τρόπο αυτό είναι δυνατή η παρουσίαση της ζητούμενης πληροφορίας μέσω της *web* εφαρμογής που σχεδιάστηκε στην παρούσα διπλωματική.



Σχήμα 18: High-level διάγραμμα του συστήματος CoRSeDA

Συνεχίζοντας, όπως αναφέρθηκε προηγουμένως, το πρώτο βήμα όταν σχεδιάζει κανείς σύμφωνα με την μεθοδολογία MDE είναι να δημιουργηθεί το κατάλληλο γενικευμένο μετα-μοντέλο, στο οποίο θα βασιστεί όλη η διαδικασία. Για να οδηγηθεί κανείς στη δημιουργία σωστού μετα-μοντέλου πρέπει να μελετήσει αρχικά τους περιορισμούς και τις ανάγκες που θέτει το πρόβλημα που επιχειρεί να λύσει. Στα πλαίσια αυτής της διπλωματικής, μελετήθηκε αρχικά η λειτουργία της πλατφόρμας R4A και πιο συγκεκριμένα η υλοποίηση που αφορά το ρομπότ NAO, καθώς επίσης και το ίδιο το NAO, ώστε να καταγραφούν οι αισθητήρες που φέρει. Στόχος αυτής της μελέτης ήταν να βρεθούν ποιοι αισθητήρες υπάρχουν στο ρομπότ, ποια στοιχεία τους μπορούν να παραμετροποιηθούν, τι επιλογές παρέχει η πλατφόρμα, και ποιες συναρτήσεις θα επιλεγούν για το συγκεκριμένο σκοπό, καθώς και τι χρειάζεται να ορισθεί προκειμένου να γίνουν σωστά οι απαραίτητες κλήσεις προς το ρομπότ. Μέσω της παραπάνω διαδικασίας αναπτύχθηκε, μέσα από διαδοχικές βελτιώσεις, το μετα-μοντέλο της παρούσας διπλωματικής εργασίας.

4.1 Συναρτήσεις της R4A Platform

Σε αυτό το σημείο είναι σημαντικό να αναφερθούν όλες οι συναρτήσεις της R4A που χρησιμοποιήθηκαν στη συγκεκριμένη εργασία. Οι συναρτήσεις όπως έχει αναφερθεί και νωρίτερα είναι γραμμένες στη γλώσσα προγραμματισμού Python 2.7. Έχουν χωριστεί σε υποκατηγορίες ανάλογα με την κατηγορία των αισθητήρων για τους οποίους εκτελούν μια συγκεκριμένη συμπεριφορά. Αυτές οι κατηγορίες είναι η απόσταση(*Distance*), ο ήχος(*Acoustic*), η ταχύτητα(*Speed*), η θέση(*Position*), η πίεση(*Pressure*), η όραση(*Vision*) και ο ηλεκτρισμός(*Electric*). Πιο αναλυτικά αξιοποιήθηκαν οι παρακάτω συναρτήσεις:

4.1.1 Συναρτήσεις κατηγορίας απόστασης

Ο controller που επιλέγεται για τις συγκεκριμένες κλήσεις αυτής της κατηγορίας είναι ο *SensorDistanceApi()*. Στην παρούσα διπλωματική χρησιμοποιούνται δύο συναρτήσεις αυτής της κατηγορίας, οι οποίες επεξηγούνται αμέσως μετά:

getDistanceSensor(id)

Περιγραφή

Επιστρέφει σε μορφή *String* πληροφορίες αναφορικά με τα τεχνικά χαρακτηριστικά του αισθητήρα του ρομπότ.

Παράμετροι εισόδου

Id: *String* που περιέχει το όνομα του αισθητήρα, όπως αυτό ορίζεται στην πλατφόρμα R4A

getDistanceSensorMeasurement(id)

Περιγραφή

Εκτελεί μία μέτρηση μέσω του αισθητήρα με το id που χρησιμοποιείται κατά την κλήση και επιστρέφει σε μορφή *ArrayNumber* πληροφορίες αναφορικά με τα τεχνικά χαρακτηριστικά του αισθητήρα του ρομπότ.

Σημείωση: το *ArrayNumber* είναι τύπος δεδομένων που ορίζεται στην πλατφόρμα R4A.

Παράμετροι εισόδου

Id: *String* που περιέχει το όνομα του αισθητήρα, όπως αυτό ορίζεται στην πλατφόρμα R4A

4.1.2 Συναρτήσεις κατηγορίας ήχου

Ο controller που επιλέγεται για τις συγκεκριμένες κλήσεις αυτής της κατηγορίας είναι ο *SensorsAcousticApi()*. Στην παρούσα διπλωματική χρησιμοποιούνται δύο συναρτήσεις αυτής της κατηγορίας, οι οποίες επεξηγούνται αμέσως μετά:

getAcousticSensor(id)

Περιγραφή

Επιστρέφει σε μορφή *String* πληροφορίες αναφορικά με τα τεχνικά χαρακτηριστικά του αισθητήρα του ρομπότ.

Παράμετροι εισόδου

Id: *String* που περιέχει το όνομα του αισθητήρα, όπως αυτό ορίζεται στην πλατφόρμα *R4A*

recordAudioFile(ids, time)

Περιγραφή

Εκτελεί ηχογράφηση για ένα χρονικό διάστημα *time*, χρησιμοποιώντας τους αισθητήρες, των οποίων τα *ids* θέτει ως παράμετρο στη συνάρτηση. Σε περίπτωση που αφεθεί κενή η συγκεκριμένη παράμετρος, τότε επιλέγει να ενεργοποιήσει όλους τους αισθητήρες του συστήματος.

Παράμετροι εισόδου

Ids: Λίστα από *Strings* που περιέχει τα όνομα των αισθητήρα με τους οποίους θα γίνει η ηχογράφηση, όπως αυτά ορίζεται στην πλατφόρμα *R4A*

Time: Νούμερο, οποιουδήποτε έγκυρου τύπου, για το χρονικό διάστημα της ηχογράφησης.

4.1.3 Συναρτήσεις κατηγορίας ταχύτητας

Ο controller που επιλέγεται για τις συγκεκριμένες κλήσεις αυτής της κατηγορίας είναι ο *SensorsSpeedApi()*. Στην παρούσα διπλωματική χρησιμοποιούνται μία συνάρτηση αυτής της κατηγορίας, η οποία επεξηγείται παρακάτω:

getSpeedSensorMeasurement()

Περιγραφή

Εκτελεί μία μέτρηση της ταχύτητας ολόκληρου του σώματος του ρομπότ, ορισμένο ως σημείο. Επιστρέφει τη γραμμική και γωνιακή ταχύτητα στους τρεις άξονες, σε μορφή *Twist*.

Σημείωση: το *Twist* είναι τύπος δεδομένων που ορίζεται στην πλατφόρμα R4A.

Παράμετροι εισόδου

Δεν δέχεται ορίσματα

4.1.4 Συναρτήσεις κατηγορίας θέσης

Ο controller που επιλέγεται για τις συγκεκριμένες κλήσεις αυτής της κατηγορίας είναι ο *SensorsPositionApi()*. Στην παρούσα διπλωματική χρησιμοποιούνται τέσσερις συναρτήσεις αυτής της κατηγορίας, οι οποίες επεξηγούνται αμέσως μετά:

getPositionSensor(id)

Περιγραφή

Επιστρέφει σε μορφή *String* πληροφορίες αναφορικά με τα τεχνικά χαρακτηριστικά του αισθητήρα του ρομπότ.

Παράμετροι εισόδου

Id: *String* που περιέχει το όνομα του αισθητήρα, όπως αυτό ορίζεται στην πλατφόρμα R4A

getPositionSensorMeasurement(id)

Περιγραφή

Εκτελεί μέτρηση χρησιμοποιώντας έναν αισθητήρα τύπου *pose*. Σε περίπτωση που ο αισθητήρας είναι ενός άξονα, τροποποιεί την τιμή *x*, αφήνοντας τις υπόλοιπες μηδενικές. Διαφορετικά αν είναι δύο αξόνων, τροποποιούνται οι τιμές *x*, *y*, *yaw*, ενώ αν είναι τριών αξόνων οι τιμές *x*, *y*, *z*, *yaw*, *pitch*, *roll*. Σε κάθε περίπτωση επιστρέφεται ένα *Pose3D* που περιέχει όλες τις πληροφορίες.

Σημείωση: το *Pose3D* είναι τύπος δεδομένων που ορίζεται στην πλατφόρμα R4A.

Παράμετροι εισόδου

Id: String που περιέχει το όνομα του αισθητήρα, όπως αυτό ορίζεται στην πλατφόρμα R4A

getTransformation(origin, target)

Περιγραφή

Εκτελεί μία μέτρηση από έναν αισθητήρα τύπου *pose* και επιστρέφει τον μετασχηματισμό των δεδομένων από ένα *tframe* σε ένα άλλο. Τα δεδομένα επιστρέφονται σε μορφή *Pose3D* που περιέχει όλες τις πληροφορίες.

Σημείωση: το *Pose3D* είναι τύπος δεδομένων που ορίζεται στην πλατφόρμα R4A.

Παράμετροι εισόδου

origin: το σημείο του οποίου ζητείται ο μετασχηματισμός

target: το επιθυμητό *tframe*

getRobotPosture()

Περιγραφή

Επιστρέφει σε μορφή *String* σε φυσική γλώσσα τη στάση σώματος του ρομπότ τη δεδομένη χρονική στιγμή. Υπάρχουν προκαθορισμένες στάσεις του ρομπότ οι οποίες έχουν οριστεί από τους κατασκευαστές. Σε περίπτωση που δε βρίσκεται σε μία από αυτές, θα επιστρέψει το *String* “Unknown”.

Παράμετροι εισόδου

Δε δέχεται ορίσματα

4.1.5 Συναρτήσεις κατηγορίας πίεσης

Ο controller που επιλέγεται για τις συγκεκριμένες κλήσεις αυτής της κατηγορίας είναι ο *SensorsPressureApi()*. Στην παρούσα διπλωματική χρησιμοποιούνται δύο συναρτήσεις αυτής της κατηγορίας, οι οποίες επεξηγούνται αμέσως μετά:

getPressureSensor(id)

Περιγραφή

Επιστρέφει σε μορφή *String* πληροφορίες αναφορικά με τα τεχνικά χαρακτηριστικά του αισθητήρα του ρομπότ.

Παράμετροι εισόδου

Id: *String* που περιέχει το όνομα του αισθητήρα, όπως αυτό ορίζεται στην πλατφόρμα *R4A*

getPressureMeasurement(id)

Περιγραφή

Επιστρέφει τη μέτρηση που έγινε από τον αισθητήρα πίεσης. Στην περίπτωση που οι αισθητήρες είναι τύπου κουμπιού ή αισθητήρα αφής, τότε σε περίπτωση πατήματος ή αφής επιστρέφεται η τιμή 1, διαφορετικά επιστρέφεται νούμερο.

Παράμετροι εισόδου

Id: *String* που περιέχει το όνομα του αισθητήρα, όπως αυτό ορίζεται στην πλατφόρμα *R4A*

4.1.6 Συναρτήσεις κατηγορίας όρασης

Ο controller που επιλέγεται για τις συγκεκριμένες κλήσεις αυτής της κατηγορίας είναι ο *SensorsVisionApi()*. Στην παρούσα διπλωματική χρησιμοποιούνται δύο συναρτήσεις αυτής της κατηγορίας, οι οποίες επεξηγούνται αμέσως μετά:

getVisionSensor(id)

Περιγραφή

Επιστρέφει σε μορφή *String* πληροφορίες αναφορικά με τα τεχνικά χαρακτηριστικά του αισθητήρα του ρομπότ.

Παράμετροι εισόδου

Id: *String* που περιέχει το όνομα του αισθητήρα, όπως αυτό ορίζεται στην πλατφόρμα *R4A*

`getRgbImageFile(id, resolutionId)`

Περιγραφή

Χρησιμοποιώντας μία συγκεκριμένη κάμερα, βγάζει μία έγχρωμη φωτογραφία και επιστρέφει το αρχείο.

Παράμετροι εισόδου

`Id: String` που περιέχει το όνομα του αισθητήρα, όπως αυτό ορίζεται στην πλατφόρμα R4A

`resolutionId: String` το οποίο δηλώνει μία έγκυρη και επιθυμητή ανάλυση για τη φωτογραφία

4.1.7 Συναρτήσεις κατηγορίας ηλεκτρισμού

Ο controller που επιλέγεται για τις συγκεκριμένες κλήσεις αυτής της κατηγορίας είναι ο `SensorsElectricApi()`. Στην παρούσα διπλωματική χρησιμοποιούνται δύο συναρτήσεις αυτής της κατηγορίας, οι οποίες επεξηγούνται αμέσως μετά:

`getElectricSensor(id)`

Περιγραφή

Επιστρέφει σε μορφή `Information` πληροφορίες αναφορικά με τα τεχνικά χαρακτηριστικά του αισθητήρα του ρομπότ.

`Σημείωση:` το `Information` είναι τύπος δεδομένων που ορίζεται στην πλατφόρμα R4A.

Παράμετροι εισόδου

`Id: String` που περιέχει το όνομα του αισθητήρα, όπως αυτό ορίζεται στην πλατφόρμα R4A

`getElectricMeasurement(id)`

Περιγραφή

Εκτελεί μέτρηση για ένα συγκεκριμένο αισθητήρα αυτής της κατηγορίας. Οι μονάδες μέτρησης είναι σε Volt για την περίπτωση που των μπαταρίων ή των ανιχνευτών τάσης και Tesla όταν είναι μαγνητόμετρα. Σε κάθε περίπτωση όταν υπάρχει μέτρηση, επιστρέφεται σε μορφή αριθμού.

Παράμετροι εισόδου

`Id: String` που περιέχει το όνομα του αισθητήρα, όπως αυτό ορίζεται στην πλατφόρμα R4A

4.1.8 Συναρτήσεις γενικού περιεχομένου

Ο controller που επιλέγεται για τις συγκεκριμένες κλήσεις αυτής της κατηγορίας είναι ο *RobotApi()*. Στην παρούσα διπλωματική χρησιμοποιείται μία συναρτηση αυτής της κατηγορίας, οι οποία επεξηγείται αμέσως μετά:

getRobotInformation()

Περιγραφή

Επιστρέφει σε μορφή *RobotInfo* πληροφορίες αναφορικά με ορισμένα τεχνικά χαρακτηριστικά του ρομπότ.

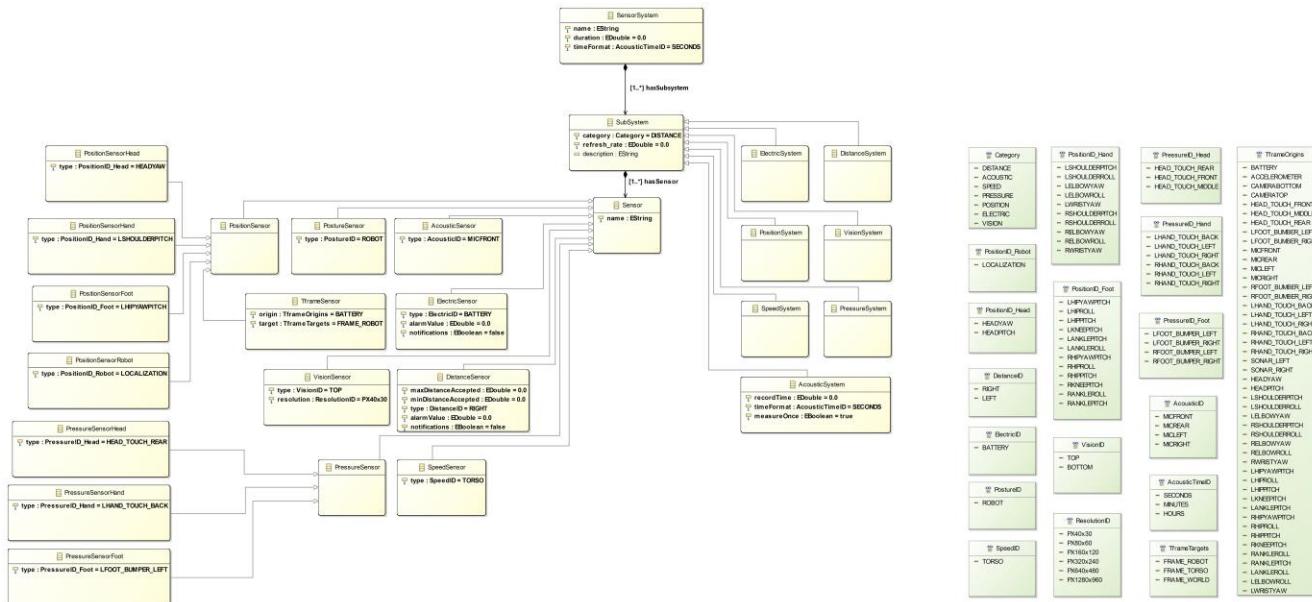
Σημείωση: το *RobotInfo* είναι τύπος δεδομένων που ορίζεται στην πλατφόρμα R4A.

Παράμετροι εισόδου

Δε δέχεται ορίσματα

4.2 Ορισμός του Μετα-μοντέλου

Μέσα από μία διαδικασία επαναλαμβανόμενης σχεδίασης, επιλύθηκαν ποικίλα προβλήματα, και τελικά παράχθηκε το τελικό μετα-μοντέλο το οποίο καθορίζει τις βασικές έννοιες και τους απαραίτητους κανόνες και περιορισμούς του συστήματος. Στο Σχήμα 19 που ακολουθεί φαίνεται το μετα-μοντέλο που σχεδιάστηκε. Στη συνέχεια, για το κάθε στοιχείο του μετα-μοντέλου παρουσιάζονται επίσης και όλες οι λίστες με τις επιλογές παραμετροποίησης ορισμένων εννοιών του συστήματος, γνωστά ως *Eumerations*. Επισημαίνονται ακόμη και ορισμένοι συμπεριφορικοί περιορισμοί που έχουν τεθεί.



Σχήμα 19: Παρουσίαση ολόκληρου του μετα-μοντέλου για το σύστημα CoRSeDA

4.2.1 SensorSystem

Σύνοψη

Αρχίζοντας από τη ρίζα του μετα-μοντέλου, το SensorSystem, είναι το σύστημα που θα κατασκευάσει ο χρήστης και θα περιλαμβάνει όλους τους αισθητήρες της επιλογής του.

Ιδιότητες

Πίνακας 2: Ιδιότητες του SensorSystem του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
name	EString	1..1	Το όνομα του συστήματος
duration	EDouble	1..1	Η διάρκεια συλλογής δεδομένων του συστήματος
timeFormat	AcousticTimeID (enum)	1..1	Η επιλογή του χρήστη αναφορικά με τη μονάδα μέτρησης της διάρκειας

Ο χρήστης μπορεί να επιλέγει τη μονάδα μέτρησης της διάρκειας συλλογής δεδομένων από τους αισθητήρες μέσω ενός *enumeration* που δημιουργήθηκε για το σκοπό αυτό, το λεγόμενο *AcousticTimeID*, το οποίο φαίνεται παρακάτω.

Συσχετίσεις

Πίνακας 3: Συσχετίσεις του SensorSystem του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
hasSubsystem	Composition – Σύνθεση	1..*	Κάθε σύστημα πρέπει να έχει τουλάχιστον ένα υποσύστημα αισθητήρων



Σχήμα 20: AcousticTimeID Enumeration

Αρχικοποίηση τιμών (Default Values)

Δεν περιλαμβάνει αρχικές τιμές.

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.duration > 0.0
Στοιχείο εφαρμογής	SensorSystem
Περιγραφή	Η διάρκεια λειτουργίας του συστήματος, δεν μπορεί να έχει αρνητική ή μηδενική τιμή, καθώς αφορά χρονικό διάστημα

AQL Code	aql: self.hasSubsystem->size() > 0
Στοιχείο εφαρμογής	SensorSystem
Περιγραφή	Το σύστημα πρέπει να έχει τουλάχιστον ένα υποσύστημα για να οριστεί σωστά

4.2.2 Subsystem

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του υποσυστήματος σε ένα ευρύτερο σύστημα αισθητήρων. Η ύπαρξη αυτού του στοιχείου, βοηθά το χρήστη να ομαδοποιεί τους αισθητήρες με βάση την κατηγορία τους και να εφαρμόζει κοινό refresh rate σε αυτούς. Οι κατηγορίες συμφωνούν τις κατηγορίες όπως ορίστηκαν στην πλατφόρμα R4A.

Ιδιότητες

Πίνακας 4: Ιδιότητες του Subsystem του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
description	EString	1..1	Το όνομα του συστήματος
refresh_rate	EDouble	1..1	Η διάρκεια συλλογής δεδομένων του συστήματος
category	Category (enum)	1..1	Η κατηγορία του υποσυστήματος

Προκειμένου να μοντελοποιηθούν οι κατηγορίες αισθητήρων που υποστηρίζει το ρομπότ NAO και η πλατφόρμα R4A, δημιουργήθηκε ένα *enumeration*, με το όνομα *Category*, το οποίο περιλαμβάνει τις κατηγορίες αυτές. Το συγκεκριμένο *enumeration* παρουσιάζεται παρακάτω.

Συσχετίσεις

Πίνακας 5: Συσχετίσεις του Subsystem του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
hasSensor	Composition – Σύνθεση	1..*	Κάθε σύστημα πρέπει να έχει τουλάχιστον έναν αισθητήρα



Σχήμα 21: Enumeration Category

Στις επόμενες επτά ενότητες παρουσιάζονται τα εξειδικευμένα Subsystems του μετα-μοντέλου. Ο σκοπός της προσθήκης αυτών των Subsystems είναι διπλός. Κάθε SubSystem, μπορεί να περιλαμβάνει αποκλειστικά αισθητήρες της ίδιας κατηγορίας με αυτό. Αφενός λοιπόν, εξυπηρετούν στον εύκολο και γρήγορο διαχωρισμό των κατηγοριών που μπορεί να υλοποιεί το σύστημα, συντελώντας στην ορθή ομαδοποίηση των αισθητήρων, βάσει της κατηγορίας τους. Αφετέρου, προσφέρουν δυνατότητες επέκτασης, καθώς σε επόμενο στάδιο θα υπάρχει ο τρόπος να προστεθούν επιπλέον ιδιότητες και χαρακτηριστικά για το κάθε υποσύστημα, χωρίς να χρειάζεται να τροποποιηθεί όλο το μετα-μοντέλο του συστήματος.

Αρχικοποίηση τιμών (Default Values)

Δεν περιλαμβάνει αρχικές τιμές.

Συμπεριφορικό Περιορισμό (Behavioral Constraints)

AQL Code	aql: self.refresh_rate > 0.0
Στοιχείο εφαρμογής	SubSystem
Περιγραφή	Η τιμή του Refresh Rate δεν επιτρέπεται να έχει αρνητική ή μηδενική τιμή

AQL Code	aql: self.eContents() -> size() <> 0
Στοιχείο εφαρμογής	SubSystem
Περιγραφή	Κάθε υποσύστημα που υπάρχει στο σύστημα, πρέπει να περιέχει τουλάχιστον έναν αισθητήρα.

4.2.3 DistanceSystem

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του υποσυστήματος με συγκεκριμένη κατηγορία, αυτήν της απόστασης.

Ιδιότητες

Δεν περιλαμβάνει περαιτέρω ιδιότητες.

Συσχετίσεις

Πίνακας 6: Συσχετίσεις του DistanceSystem του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Subsystem	SuperType – Επέκταση	-	Το στοιχείο DistanceSystem επεκτείνει το στοιχείο Subsystem

Αρχικοποίηση τιμών (Default Values)

description	Distance Sensor Sub System
refresh_rate	0.2 (sec)
category	DISTANCE

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.category.toString() = 'DISTANCE'
Στοιχείο εφαρμογής	DistanceSystem
Περιγραφή	Η κατηγορία αυτού του υποσυστήματος, πρέπει να είναι αποκλειστικά DISTANCE

4.2.4 ElectricSystem

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του υποσυστήματος με συγκεκριμένη κατηγορία, αυτήν του ηλεκτρισμού.

Ιδιότητες

Δεν περιλαμβάνει περαιτέρω ιδιότητες.

Συσχετίσεις

Πίνακας 7: Συσχετίσεις του ElectricSystem του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Subsystem	SuperType – Επέκταση	-	Το στοιχείο ElectricSystem επεκτείνει το στοιχείο Subsystem

Αρχικοποίηση τιμών (Default Values)

description	Electric Sensor Sub System
refresh_rate	0.2 (sec)
category	ELECTRIC

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.category.toString() = 'ELECTRIC'
Στοιχείο εφαρμογής	ElectricSystem
Περιγραφή	Η κατηγορία αυτού του υποσυστήματος, πρέπει να είναι αποκλειστικά ELECTRIC

4.2.5 PositionSystem

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του υποσυστήματος με συγκεκριμένη κατηγορία, αυτήν της θέσης.

Ιδιότητες

Δεν περιλαμβάνει περαιτέρω ιδιότητες.

Συσχετίσεις

Πίνακας 8: Συσχετίσεις του PositionSystem του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Subsystem	SuperType – Επέκταση	-	Το στοιχείο PositionSystem επεκτείνει το στοιχείο Subsystem

Αρχικοποίηση τιμών (Default Values)

description	Position Sensor Sub System
refresh_rate	0.2 (sec)
category	POSITION

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.category.toString() = 'POSITION'
Στοιχείο εφαρμογής	PositionSystem
Περιγραφή	Η κατηγορία αυτού του υποσυστήματος, πρέπει να είναι αποκλειστικά POSITION

4.2.6 PressureSystem

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του υποσυστήματος με συγκεκριμένη κατηγορία, αυτήν της πίεσης.

Ιδιότητες

Δεν περιλαμβάνει περαιτέρω ιδιότητες.

Συσχετίσεις

Πίνακας 9: Συσχετίσεις του PressureSystem του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Subsystem	SuperType – Επέκταση	-	Το στοιχείο PressureSystem επεκτείνει το στοιχείο Subsystem

Αρχικοποίηση τιμών (Default Values)

description	Pressure Sensor Sub System
refresh_rate	0.2 (sec)
category	PRESSURE

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.category.toString() = 'PRESSURE'
Στοιχείο εφαρμογής	PressureSystem
Περιγραφή	Η κατηγορία αυτού του υποσυστήματος, πρέπει να είναι αποκλειστικά PRESSURE

4.2.7 SpeedSystem

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του υποσυστήματος με συγκεκριμένη κατηγορία, αυτήν της ταχύτητας.

Ιδιότητες

Δεν περιλαμβάνει περαιτέρω ιδιότητες.

Συσχετίσεις

Πίνακας 10: Συσχετίσεις του SpeedSystem του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Subsystem	SuperType – Επέκταση	-	Το στοιχείο SpeedSystem επεκτείνει το στοιχείο Subsystem

Αρχικοποίηση τιμών (Default Values)

description	Speed Sensor Sub System
refresh_rate	0.2 (sec)
category	SPEED

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.category.toString() = 'SPEED'
Στοιχείο εφαρμογής	SpeedSystem
Περιγραφή	Η κατηγορία αυτού του υποσυστήματος, πρέπει να είναι αποκλειστικά SPEED

4.2.6 VisionSystem

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του υποσυστήματος με συγκεκριμένη κατηγορία, αυτήν της όρασης.

Ιδιότητες

Δεν περιλαμβάνει περαιτέρω ιδιότητες.

Συσχετίσεις

Πίνακας 11: Συσχετίσεις του VisionSystem του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Subsystem	SuperType – Επέκταση	-	Το στοιχείο VisionSystem επεκτείνει το στοιχείο Subsystem

Αρχικοποίηση τιμών (Default Values)

description	Vision Sensor Sub System
refresh_rate	1.0 (sec)
category	VISION

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.category.toString() = 'VISION'
Στοιχείο εφαρμογής	VisionSystem
Περιγραφή	Η κατηγορία αυτού του υποσυστήματος, πρέπει να είναι αποκλειστικά VISION

4.2.7 AcousticSystem

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του υποσυστήματος με συγκεκριμένη κατηγορία, αυτήν του ήχου.

Ιδιότητες

Πίνακας 12: Ιδιότητες του AcousticSystem του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
recordTime	EDouble	1..1	Ο χρόνος ηχογράφησης
timeFormat	AcousticTimeID (enum)	1..1	Η επιλογή του χρήστη αναφορικά με τη μονάδα μέτρησης της διάρκειας
measureOnce	EBoolean	1..1	Δυνατότητα να πραγματοποιηθεί μόνο μία ηχογράφηση καθ'όλη τη διάρκεια συλλογής δεδομένων του συστήματος

Σημείωση: Το AcousticTimeID παρουσιάστηκε προηγουμένως.

Συσχετίσεις

Πίνακας 13: Συσχετίσεις του AcousticSystem του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Subsystem	SuperType – Επέκταση	-	Το στοιχείο AcousticSystem επεκτείνει το στοιχείο Subsystem

Αρχικοποίηση τιμών (Default Values)

description	Acoustic Sensor Sub System
refresh_rate	0.0 (sec)
category	ACOUSTIC
recordTime	30
timeFormat	SECONDS
measureOnce	TRUE

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.category.toString() = 'ACOUSTIC'
Στοιχείο εφαρμογής	AcousticSystem
Περιγραφή	Η κατηγορία αυτού του υποσυστήματος, πρέπει να είναι αποκλειστικά ACOUSTIC

AQL Code	aql: not(self.description.toString().trim().contains(' '))
Στοιχείο εφαρμογής	AcousticSystem
Περιγραφή	Η περιγραφή αυτού του υποσυστήματος δεν μπορεί να περιέχει κανένα κενό

AQL Code	aql: self.description.size() <> 0
Στοιχείο εφαρμογής	AcousticSystem
Περιγραφή	Η περιγραφή αυτού του υποσυστήματος πρέπει οπωσδήποτε να καθοριστεί

AQL Code	aql: self.recordTime > 0.0
Στοιχείο εφαρμογής	AcousticSystem
Περιγραφή	Η τιμή recordTime αυτού του υποσυστήματος δεν μπορεί να περιέχει μηδενική ή αρνητική τιμή, καθώς αφορά χρονικό διάστημα

AQL Code	aql: self.timeFormat = self.eContainer().timeFormat implies self.recordTime <= self.eContainer().duration
Στοιχείο εφαρμογής	AcousticSystem
Περιγραφή	Η τιμή recordTime αυτού του υποσυστήματος δεν μπορεί να περιέχει τιμή μεγαλύτερη από το συνολικό χρόνο λειτουργία του συστήματος

AQL Code	aql: ((self.timeFormat.toString() = 'MINUTES') and (self.eContainer().timeFormat.toString() = 'SECONDS')) implies ((self.recordTime * 60.0) <= self.eContainer().duration)
Στοιχείο εφαρμογής	AcousticSystem
Περιγραφή	Η τιμή recordTime αυτού του υποσυστήματος δεν μπορεί να περιέχει τιμή μεγαλύτερη από το συνολικό χρόνο λειτουργία του συστήματος

AQL Code	aql: ((self.timeFormat.toString() = 'HOURS') and (self.eContainer().timeFormat.toString() = 'MINUTES')) implies ((self.recordTime * 60.0) <= self.eContainer().duration)
Στοιχείο εφαρμογής	AcousticSystem
Περιγραφή	Η τιμή recordTime αυτού του υποσυστήματος δεν μπορεί να περιέχει τιμή μεγαλύτερη από το συνολικό χρόνο λειτουργία του συστήματος

AQL Code	aql: ((self.timeFormat.toString() = 'HOURS') and (self.eContainer().timeFormat.toString() = 'SECONDS')) implies ((self.recordTime *3600.0) <= self.eContainer().duration)
Στοιχείο εφαρμογής	AcousticSystem
Περιγραφή	Η τιμή recordTime αυτού του υποσυστήματος δεν μπορεί να περιέχει τιμή μεγαλύτερη από το συνολικό χρόνο λειτουργία του συστήματος

AQL Code	aql: self.measureOnce.toString() = 'false' implies (self.timeFormat.toString() = 'SECONDS' implies self.recordTime <= self.refresh_rate)
Στοιχείο εφαρμογής	AcousticSystem
Περιγραφή	Η τιμή recordTime αυτού του υποσυστήματος δεν μπορεί να περιέχει τιμή μεγαλύτερη από αυτή του Refresh Rate

AQL Code	aql: self.measureOnce.toString() = 'false' implies (self.timeFormat.toString() = 'MINUTES' implies ((self.recordTime * 60.0) <= self.refresh_rate))
Στοιχείο εφαρμογής	AcousticSystem
Περιγραφή	Η τιμή recordTime αυτού του υποσυστήματος δεν μπορεί να περιέχει τιμή μεγαλύτερη από αυτή του Refresh Rate

AQL Code	aql: self.timeFormat.toString() = 'HOURS' implies ((self.recordTime *3600.0) <= self.refresh_rate)
Στοιχείο εφαρμογής	AcousticSystem
Περιγραφή	Η τιμή recordTime αυτού του υποσυστήματος δεν μπορεί να περιέχει τιμή μεγαλύτερη από αυτή του Refresh Rate

AQL Code	aql: self.eContainer().eAllContents() -> filter(sensorProject::AcousticSystem) -> select(rs rs<>self) -> forAll(rs rs.description <> self.description)
Στοιχείο εφαρμογής	AcousticSystem
Περιγραφή	Κάθε υποσύστημα ήχου πρέπει να έχει μοναδική περιγραφή

AQL Code	aql: self.eContainer().eAllContents() -> filter(sensorProject::AcousticSystem) -> select(rs rs<>self) -> forAll(rs rs.measureOnce = self.measureOnce)
Στοιχείο εφαρμογής	AcousticSystem
Περιγραφή	Όλα τα υποσυστήματα ήχου πρέπει να έχουν ίδια τιμή για την ιδιότητα MeasureOnce

4.2.8 Sensor

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την πιο βασική έννοια του συγκεκριμένου συστήματος, αυτή του αισθητήρα.

Ιδιότητες

Πίνακας 14: Ιδιότητες του Sensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
name	EString	1..1	Το όνομα που επιλέγει ο χρήστης. Βασική προϋπόθεση να είναι μοναδικό το όνομα αυτό, καθώς λειτουργεί ως κλειδί(key) με το οποίο θα το αναζητεί στη βάση δεδομένων και στο παραγόμενο API.

Συσχετίσεις

Δεν περιέχει καμία συσχέτιση.

Αρχικοποίηση τιμών (Default Values)

Δεν περιλαμβάνει αρχικές τιμές.

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.eContainer().eContainer().eAllContents() -> filter(sensorProject::Sensor) -> select(rs rs <> self) -> forAll(rs rs.name <> self.name)
Στοιχείο εφαρμογής	Sensor
Περιγραφή	Κάθε αισθητήρας του συστήματος πρέπει να έχει ένα μοναδικό όνομα

4.2.9 AcousticSensor

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία του ήχου και συγκεκριμένα αναφέρεται στα τέσσερα μικρόφωνα του ρομπότ NAO.

Ιδιότητες

Πίνακας 15: Ιδιότητες του AcousticSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
type	AcousticID(enum)	1..1	Ο τύπος αισθητήρα, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO

Το *AcousticID* είναι ένα *enumeration* που δημιουργήθηκε με σκοπό την αναπαράσταση των Ids των αισθητήρων του ρομπότ που αφορούν τη συγκεκριμένη ιδιότητα. Η πλατφόρμα R4A χρειάζεται τη συγκεκριμένη παράμετρο για να εκτελέσει τη συνάρτηση ηχογράφησης. Το συγκεκριμένο *enumeration* παρουσιάζεται στη συνέχεια.

Συσχετίσεις

Πίνακας 16: Συσχετίσεις του AcousticSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Sensor	SuperType – Επέκταση	-	Το στοιχείο AcousticSensor επεκτείνει το στοιχείο Sensor



Σχήμα 22: Enumeration AcousticID

Αρχικοποίηση τιμών (Default Values)

Δεν περιλαμβάνει αρχικές τιμές.

Συμπεριφορικό Περιορισμό (Behavioral Constraints)

AQL Code	aql: self.eContainer() .eContainer() .eAllContents() -> filter(sensorProject::AcousticSensor) ->select(rs rs<>self) ->forAll(rs rs.type <> self.type)
Στοιχείο εφαρμογής	AcousticSensor
Περιγραφή	Κάθε αισθητήρας ήχου του συστήματος πρέπει να έχει ένα μοναδικό τύπο (type)

4.2.10 ElectricSensor

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία του ηλεκτρισμού και συγκεκριμένα αναφέρεται στην μπαταρία του ρομπότ NAO.

Ιδιότητες

Πίνακας 17: Ιδιότητες του ElectricSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
type	ElectricID(enum)	1..1	Ο τύπος αισθητήρα, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO
alarmValue	EDouble	1..1	Συνιστά μία τιμή «συναγερμού» την οποία μπορεί να θέσει ο χρήστης για να ενημερώνεται μόλις ξεπεραστεί. Μετριέται σε (%) και δέχεται τιμές από 0-100. Εάν επιλεχθεί τιμή και η ιδιότητα notifications είναι επιλεγμένη, τότε ο χρήστης μπορεί να δει το συναγερμό στη web εφαρμογή που έχει υλοποιηθεί στη διπλωματική.
notifications	EBoolean	1..1	Ο χρήστης επιλέγει εάν θέλει να ειδοποιηθεί για τις περιπτώσεις που η μπαταρία του ρομπότ θα λάβει τιμές μικρότερες από την alarmValue που θέτει

To *ElectricID* είναι ένα *enumeration* που δημιουργήθηκε με σκοπό την αναπαράσταση των Ids των αισθητήρων του ρομπότ που αφορούν τη συγκεκριμένη ιδιότητα. Η πλατφόρμα R4A χρειάζεται τη συγκεκριμένη παράμετρο για να εκτελέσει τη συνάρτηση συλλογής δεδομένων για το συγκεκριμένο αισθητήρα. Παρόλο που το ρομπότ έχει μόνο έναν αισθητήρα αυτής της κατηγορίας, το *enumeration* δημιουργήθηκε για τους σκοπούς της επεκτασιμότητας και παρουσιάζεται στη συνέχεια.

Συσχετίσεις

Πίνακας 18: Συσχετίσεις του ElectricSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Sensor	SuperType – Επέκταση	-	Το στοιχείο ElectricSensor επεκτείνει το στοιχείο Sensor



Σχήμα 23: Enumeration ElectricID

Αρχικοποίηση τιμών (Default Values)

name	Battery
type	battery
alarmValue	30 (%)
notifications	TRUE

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.alarmValue <> 0.0 implies self.notifications.toString().toUpperCase() = 'TRUE'
Στοιχείο εφαρμογής	ElectricSensor
Περιγραφή	Ένας αισθητήρας ηλεκτρισμού πρέπει να θέσει την τιμή notifications ως TRUE σε περίπτωση που θέσει την τιμή του alarmValue διάφορη του μηδενός.

AQL Code	aql: self.notifications.toString().toUpperCase() = 'TRUE' implies (self.alarmValue >= 0.0) and (self.alarmValue <= 100.0))
Στοιχείο εφαρμογής	ElectricSensor
Περιγραφή	Η τιμή alarmValue ενός αισθητήρα ηλεκτρισμού μπορεί να λάβει τιμές στο εύρος 0 έως 100.

4.2.11 VisionSensor

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της όρασης και συγκεκριμένα αναφέρεται στις δύο RGB κάμερες του ρομπότ NAO.

Ιδιότητες

Πίνακας 19: Ιδιότητες του VisionSensor του μετα-μοντέλου

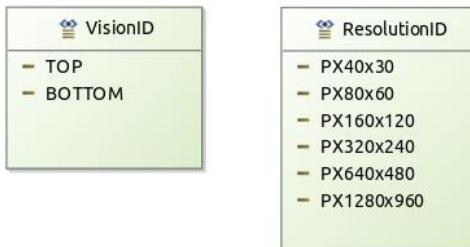
Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
type	VisionID(enum)	1..1	Ο τύπος αισθητήρα, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO
resolution	ResolutionID(enum)	1..1	Η ανάλυση των φωτογραφιών που θα ληφθούν από τις κάμερες. Ο χρήστης επιλέγει από μία τιμή για την ανάλυση από τη λίστα στην οποία συνοψίζονται οι δυνατότητες που παρέχονται.

Το *VisionID* είναι ένα *enumeration* που δημιουργήθηκε με σκοπό την αναπαράσταση των Ids των αισθητήρων του ρομπότ που αφορούν τη συγκεκριμένη ιδιότητα και φαίνεται παρακάτω. Αντίστοιχα, το *ResolutionID* είναι το *enumeration* που περιέχει τις διαφορετικές αναλύσεις για τις φωτογραφίες που θα ληφθούν. Η πλατφόρμα R4A χρειάζεται τις συγκεκριμένες παράμετρους για να εκτελέσει τη συνάρτηση συλλογής δεδομένων για το συγκεκριμένο αισθητήρα.

Συσχετίσεις

Πίνακας 20: Συσχετίσεις του VisionSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Sensor	SuperType – Επέκταση	-	Το στοιχείο VisionSensor επεκτείνει το στοιχείο Sensor



Σχήμα 24: Enumeration VisionID και Enumeration ResolutionID

Αρχικοποίηση τιμών (Default Values)

resolutionID	PX640x480
type	Εάν δεν υπάρχει άλλη κάμερα στο σύστημα, μπαίνει αυτόματα ως τύπος top. Διαφορετικά, όταν υπάρχει κι άλλη κάμερα, ελέγχεται ο τύπος της, αν υπάρχει κάμερα με τύπο top, τότε αν ο χρήστης προσθέσει κι άλλη θα αρχικοποιηθεί ως bottom, και αντίστροφα.

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.eContainer().eContainer().eAllContents() -> filter(sensorProject::DistanceSensor) -> select(rs rs<>self) -> forAll(rs rs.type <> self.type)
Στοιχείο εφαρμογής	VisionSensor
Περιγραφή	Κάθε αισθητήρας όρασης του συστήματος πρέπει να έχει μοναδικό τύπο (type)

4.2.12 DistanceSensor

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της απόστασης και συγκεκριμένα αναφέρεται στους δύο sonar αισθητήρες του ρομπότ NAO.

Ιδιότητες

Πίνακας 21: Ιδιότητες του DistanceSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
type	DistanceID(enum)	1..1	Ο τύπος αισθητήρα, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO
alarmValue	EDouble	1..1	Συνιστά μία τιμή «συναγερμού» την οποία μπορεί να θέσει ο χρήστης για να ενημερώνεται μόλις ξεπεραστεί. Μετριέται σε (m) και δέχεται τιμές ενδιάμεσα από τη MinDistanceAccepted και τη MaxDistanceAccepted. Εάν επιλεχθεί τιμή και η ιδιότητα notifications είναι επιλεγμένη, τότε ο χρήστης μπορεί να δει το συναγερμό στη web εφαρμογή που έχει υλοποιηθεί στη διπλωματική.
notifications	EBoolean	1..1	Ο χρήστης επιλέγει εάν θέλει να ειδοποιηθεί για τις περιπτώσεις που η μπαταρία του ρομπότ θα λάβει τιμές μικρότερες από την alarmValue που θέτει
MinDistanceAccepted	EDouble	1..1	Η ελάχιστη τιμή που θέτει ο χρήστης για το εύρος των τιμών απόστασης που τον ενδιαφέρει να συλλέγει ο αισθητήρας απόστασης του ρομπότ
MaxDistanceAccepted	EDouble	1..1	Η μέγιστη τιμή που θέτει ο χρήστης για το εύρος των τιμών απόστασης που τον ενδιαφέρει να συλλέγει ο αισθητήρας απόστασης του ρομπότ

To *DistanceID* είναι ένα enumeration που δημιουργήθηκε με σκοπό την αναπαράσταση των Ids των αισθητήρων του ρομπότ που αφορούν τη συγκεκριμένη ιδιότητα και φαίνεται παρακάτω. Η πλατφόρμα R4A χρειάζεται τη συγκεκριμένη παράμετρο για να εκτελέσει τη συνάρτηση συλλογής δεδομένων για το συγκεκριμένο αισθητήρα.

Συσχετίσεις

Πίνακας 22: Συσχετίσεις του DistanceSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Sensor	SuperType – Επέκταση	-	Το στοιχείο DistanceSensor επεκτείνει το στοιχείο Sensor



Σχήμα 25: Enumeration DistanceID

Αρχικοποίηση τιμών (Default Values)

type	Εάν δεν υπάρχει άλλο sonar στο σύστημα, μπαίνει αυτόματα ως τύπος right. Διαφορετικά, όταν υπάρχει κι άλλη κάμερα, ελέγχεται ο τύπος της, αν υπάρχει κάμερα με τύπο right, τότε αν ο χρήστης προσθέσει κι άλλο θα αρχικοποιηθεί ως left, και αντίστροφα.
alarmValue	0.0
notifications	FALSE
minDistanceAccepted	0.23 (m)
maxDistanceAccepted	1.80 (m)

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.alarmValue <> 0.0 implies self.notifications.toString().toUpperCase() = 'TRUE'
Στοιχείο εφαρμογής	DistanceSensor
Περιγραφή	Ένας αισθητήρας απόστασης πρέπει να θέσει την τιμή notifications ως TRUE σε περίπτωση που θέσει την τιμή του alarmValue διάφορη του μηδενός.

AQL Code	aql: self.notifications.toString().toUpperCase() = 'TRUE' implies ((self.alarmValue >= self.minDistanceAccepted) and (self.alarmValue <= self.maxDistanceAccepted))
Στοιχείο εφαρμογής	DistanceSensor
Περιγραφή	Η μεταβλητή alarmValue μπορεί να λάβει τιμές εντός του εύρους minDistanceAccepted και maxDistanceAccepted

AQL Code	aql: self.minDistanceAccepted <= self.maxDistanceAccepted
Στοιχείο εφαρμογής	DistanceSensor
Περιγραφή	Η μεταβλητή minDistanceAccepted πρέπει να έχει μικρότερη τιμή από αυτήν της maxDistanceAccepted

AQL Code	aql: self.maxDistanceAccepted >= 0.23 and self.maxDistanceAccepted <= 1.8
Στοιχείο εφαρμογής	DistanceSensor
Περιγραφή	Η μεταβλητή maxDistanceAccepted πρέπει να έχει τιμή εντός του εύρους 0.23 – 1.8

AQL Code	aql: self.minDistanceAccepted >= 0.23 and self.minDistanceAccepted <= 1.80
Στοιχείο εφαρμογής	DistanceSensor
Περιγραφή	Η μεταβλητή minDistanceAccepted πρέπει να έχει τιμή εντός του εύρους 0.23 – 1.8

AQL Code	aql: self.eContainer().eContainer().eAllContents() -> filter(sensorProject::DistanceSensor) -> select(rs rs <> self) -> forAll(rs rs.type <> self.type)
Στοιχείο εφαρμογής	DistanceSensor
Περιγραφή	Κάθε αισθητήρας απόστασης του συστήματος πρέπει να έχει μοναδικό τύπο (type)

4.2.12 SpeedSensor

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της ταχύτητας και συγκεκριμένα αναφέρεται στο αξελερόμετρο(*accelerometer*) του ρομπότ NAO.

Ιδιότητες

Πίνακας 23: Ιδιότητες του SpeedSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
type	SpeedID (enum)	1..1	Ο τύπος αισθητήρα, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO

To *SpeedID* είναι ένα *enumeration* που δημιουργήθηκε με σκοπό την αναπαράσταση των Ids των αισθητήρων του ρομπότ που αφορούν τη συγκεκριμένη ιδιότητα και φαίνεται παρακάτω. Η πλατφόρμα R4A δε χρειάζεται τη συγκεκριμένη παράμετρο για να εκτελέσει τη συνάρτηση συλλογής δεδομένων για το συγκεκριμένο αισθητήρα, ωστόσο η υλοποίηση αυτή εξυπηρετεί τους σκοπούς της επεκτασιμότητας. Παράλληλα, βοηθά το χρήστη να καταλάβει τη θέση του αισθητήρα στο ρομπότ.

Συσχετίσεις

Πίνακας 24: Συσχετίσεις του SpeedSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Sensor	SuperType – Επέκταση	-	Το στοιχείο SpeedSensor επεκτείνει το στοιχείο Sensor



Σχήμα 26: Enumeration SpeedID

Αρχικοποίηση τιμών (Default Values)

name	Torso
type	torso

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

Δεν περιλαμβάνει συμπεριφορικούς περιορισμούς.

4.2.14 PressureSensor

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της πίεσης και συνιστά γενίκευση των αισθητήρων του ρομπότ NAO.

Ιδιότητες

Δεν περιλαμβάνει περαιτέρω ιδιότητες

Συσχετίσεις

Πίνακας 25: Συσχετίσεις του PressureSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Sensor	SuperType – Επέκταση	-	Το στοιχείο PressureSensor επεκτείνει το στοιχείο Sensor

Αρχικοποίηση τιμών (Default Values)

Δεν περιλαμβάνει αρχικές τιμές.

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.eContainer().eContainer().eAllContents() -> filter(sensorProject::PressureSensor) -> select(rs rs<>self) -> forAll(rs rs.type <> self.type)
Στοιχείο εφαρμογής	PressureSensor
Περιγραφή	Κάθε αισθητήρας πίεσης του συστήματος πρέπει να έχει διαφορετικό τύπο (type)

4.2.15 PressureSensorHead

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της πίεσης και συγκεκριμένα αναφέρεται στα τρία κουμπιά(buttons) που είναι τοποθετημένα στο κεφάλι του ρομπότ NAO.

Ιδιότητες

Πίνακας 26: Ιδιότητες του PressureSensorHead του μετα-μοντέλου

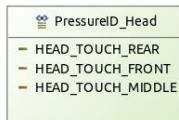
Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
type	PressureID_Head(enum)	1..1	Ο τύπος αισθητήρα, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO

To *PressureID_Head* είναι ένα *enumeration* που δημιουργήθηκε με σκοπό την αναπαράσταση των Ids των αισθητήρων του ρομπότ που αφορούν τη συγκεκριμένη ιδιότητα και φαίνεται παρακάτω. Η πλατφόρμα R4A χρειάζεται τη συγκεκριμένη παράμετρο για να εκτελέσει τη συνάρτηση συλλογής δεδομένων για το συγκεκριμένο είδος αισθητήρα. Παράλληλα, βοηθά το χρήστη να καταλάβει τη θέση τους στο ρομπότ.

Συσχετίσεις

Πίνακας 27: Συσχετίσεις του PressureSensorHead του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
PressureSensor	SuperType – Επέκταση	-	Το στοιχείο PressureSensorHead επεκτείνει το στοιχείο PressureSensor



Σχήμα 27: Enumeration PressureID_Head

Αρχικοποίηση τιμών (Default Values)

Δεν περιλαμβάνει αρχικές τιμές.

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

Δεν περιλαμβάνει συμπεριφορικούς περιορισμούς.

4.2.16 PressureSensorHand

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της πίεσης και συγκεκριμένα αναφέρεται στους αισθητήρες αφής(*tactiles*) που είναι τοποθετημένοι στα χέρια του ρομπότ NAO.

Ιδιότητες

Πίνακας 28: Ιδιότητες του PressureSensorHand του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
type	PressureID_Hand(<i>enum</i>)	1..1	Ο τύπος αισθητήρα, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO

To *PressureID_Hand* είναι ένα *enumeration* που δημιουργήθηκε με σκοπό την αναπαράσταση των Ids των αισθητήρων του ρομπότ που αφορούν τη συγκεκριμένη ιδιότητα και φαίνεται παρακάτω. Η πλατφόρμα R4A χρειάζεται τη συγκεκριμένη παράμετρο για να εκτελέσει τη συνάρτηση συλλογής δεδομένων για το συγκεκριμένο είδος αισθητήρα. Παράλληλα, βοηθά το χρήστη να καταλάβει τη θέση τους στο ρομπότ.

Συσχετίσεις

Πίνακας 29: Συσχετίσεις του PressureSensorHand του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
PressureSensor	SuperType – Επέκταση	-	Το στοιχείο PressureSensorHand επεκτείνει το στοιχείο PressureSensor



Σχήμα 28: Enumeration PressureID_Hand

Αρχικοποίηση τιμών (Default Values)

Δεν περιλαμβάνει αρχικές τιμές.

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

Δεν περιλαμβάνει συμπεριφορικούς περιορισμούς.

4.2.17 PressureSensorFoot

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της πίεσης και συγκεκριμένα αναφέρεται στους αισθητήρες αφής(bumpers) που είναι τοποθετημένοι στα πόδια του ρομπότ NAO.

Ιδιότητες

Πίνακας 30: Ιδιότητες του PressureSensorFoot του μετα-μοντέλου

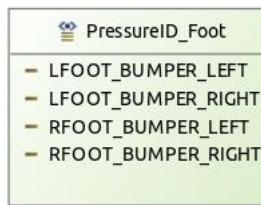
Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
type	PressureID_Foot(<i>enum</i>)	1..1	Ο τύπος αισθητήρα, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO

To *PressureID_Foot* είναι ένα *enumeration* που δημιουργήθηκε με σκοπό την αναπαράσταση των Ids των αισθητήρων του ρομπότ που αφορούν τη συγκεκριμένη ιδιότητα και φαίνεται παρακάτω. Η πλατφόρμα R4A χρειάζεται τη συγκεκριμένη παράμετρο για να εκτελέσει τη συνάρτηση συλλογής δεδομένων για το συγκεκριμένο είδος αισθητήρα. Παράλληλα, βοηθά το χρήστη να καταλάβει τη θέση τους στο ρομπότ.

Συσχετίσεις

Πίνακας 31: Συσχετίσεις του PressureSensorFoot του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
PressureSensor	SuperType – Επέκταση	-	Το στοιχείο PressureSensorFoot επεκτείνει το στοιχείο Pressure Sensor



Σχήμα 29: Enumeration PressureID_Foot

Αρχικοποίηση τιμών (Default Values)

Δεν περιλαμβάνει αρχικές τιμές.

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

Δεν περιλαμβάνει συμπεριφορικούς περιορισμούς.

4.2.18 PositionSensor

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της θέσης και αποτελεί γενίκευση αυτής της κατηγορίας αισθητήρων.

Ιδιότητες

Δεν περιέχει περαιτέρω ιδιότητες.

Συσχετίσεις

Πίνακας 32: Συσχετίσεις του PositionSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Sensor	SuperType – Επέκταση	-	Το στοιχείο PositionSensor επεκτείνει το στοιχείο Sensor

Αρχικοποίηση τιμών (Default Values)

Δεν περιλαμβάνει αρχικές τιμές.

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.eContainer().eContainer().eAllContents() -> filter(sensorProject::PositionSensor) -> select(rs rs<>self) -> forAll(rs rs.type <> self.type)
Στοιχείο εφαρμογής	PositionSensor
Περιγραφή	Κάθε αισθητήρας θέσης του συστήματος πρέπει να έχει μοναδικό τύπο (type)

4.2.19 PostureSensor

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της θέσης και συγκεκριμένα αναφέρεται στη στάση σώματος του ρομπότ NAO. Παρόλο που ανήκει στην κατηγορία της θέσης στην πλατφόρμα R4A, στα πλαίσια της διπλωματικής, επιλέχθηκε να συνιστά ανεξάρτητο αισθητήρα και συνεπώς να επεκτείνει το στοιχείο Sensor και όχι το PositionSensor. Ο λόγος της επιλογής αυτής βασίζεται στο γεγονός πως η συνάρτηση της πλατφόρμας R4A επιστρέφει διαφορετικού είδους δεδομένα για τον αισθητήρα αυτόν, από ότι για τους υπόλοιπους της ίδιας κατηγορίας.

Ιδιότητες

Πίνακας 33: Ιδιότητες του PostureSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
type	PostureID	1..1	Ο τύπος αισθητήρα, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO

To *PostureID* είναι ένα *enumeration* που δημιουργήθηκε με σκοπό την αναπαράσταση των τμημάτων του ρομπότ που αφορούν τη συγκεκριμένη ιδιότητα. Παρόλο που η πλατφόρμα R4A δε χρειάζεται τη συγκεκριμένη παράμετρο για να εκτελέσει κάποια συνάρτηση, η ιδιότητα αυτή εξυπηρετεί την περίπτωση επέκτασης της εφαρμογής. Το συγκεκριμένο *enumeration* παρουσιάζεται στη συνέχεια.

Συσχετίσεις

Πίνακας 34: Συσχετίσεις του PostureSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Sensor	SuperType – Επέκταση	-	Το στοιχείο PostureSensor επεκτείνει το στοιχείο Sensor



Σχήμα 30: Enumeration PostureID

Αρχικοποίηση τιμών (Default Values)

name	RobotPosutre
type	Robot

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.eContainer().eContainer().eAllContents() -> filter(sensorProject::PostureSensor) -> select(rs rs<>self) ->forAll(rs rs.type <> self.type)
Στοιχείο εφαρμογής	PostureSensor
Περιγραφή	Κάθε αισθητήρας θέσης και συγκεκριμένα στάσης σώματος του ρομπότ πρέπει να έχει μοναδικό τύπο (type)

4.2.20 TframeSensor

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της θέσης και συγκεκριμένα αναφέρεται στους μετασχηματισμούς T-frame που επιτρέπει το ρομπότ NAO.

Ιδιότητες

Πίνακας 35: Ιδιότητες του TframeSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
origin	TframeOrigins(enum)	1..1	Ο τύπος του σημείου προς μετασχηματισμό, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO
target	TframeTargets(enum)	1..1	Ο τύπος του είδους του μετασχηματισμού, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO

Τα δύο αυτά enumerations δημιουργήθηκαν με σκοπό την αναπαράσταση των σημείων προς μετασχηματισμό, που αφορούν τους αισθητήρες του ρομπότ NAO, και το είδους του μετασχηματισμού T-frame. Η πλατφόρμα R4A χρειάζεται τις παραμέτρους αυτές για να εκτελέσει τη συνάρτηση μετασχηματισμού. Τα συγκεκριμένα enumerations παρουσιάζεται στη συνέχεια.

Συσχετίσεις

Πίνακας 36: Συσχετίσεις του TframeSensor του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
Sensor	SuperType – Επέκταση	-	Το στοιχείο TframeSensor επεκτείνει το στοιχείο Sensor



Σχήμα 31: Enumeration TframeOrigins και Enumeration TframeTargets

Αρχικοποίηση τιμών (Default Values)

Δεν περιλαμβάνει αρχικές τιμές.

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

AQL Code	aql: self.eContainer().eContainer().eAllContents() -> filter(sensorProject::TframeSensor) -> select(p p <> self) -> forAll(t (t.origin <> self.origin and t.target = self.target) or (t.origin = self.origin and t.target <> self.target))
Στοιχείο εφαρμογής	TframeSensor
Περιγραφή	Κάθε αισθητήρας θέσης και Tframe του συστήματος πρέπει να έχει μοναδικό συνδυασμό origin-target

4.2.21 PositionSensorHead

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της θέσης και συγκεκριμένα αναφέρεται στις αρθρώσεις(*joints*) του κεφαλιού του ρομπότ NAO.

Ιδιότητες

Πίνακας 37: Ιδιότητες του PositionSensorHead του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
type	PositionID_Head	1..1	Ο τύπος αισθητήρα, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO

Το *PositionID_Head* είναι ένα *enumeration* που δημιουργήθηκε με σκοπό την αναπαράσταση των τμημάτων του ρομπότ που αφορούν τη συγκεκριμένη ιδιότητα. Η πλατφόρμα R4A χρειάζεται τη συγκεκριμένη παράμετρο για να εκτελέσει τη συνάρτηση συλλογής δεδομένων από αυτό το είδος αισθητήρων. Το συγκεκριμένο *enumeration* παρουσιάζεται στη συνέχεια.

Συσχετίσεις

Πίνακας 38: Συσχετίσεις του PositionSensorHead του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
PositionSensor	SuperType – Επέκταση	-	Το στοιχείο PositionSensorHead επεκτείνει το στοιχείο PositionSensor



Σχήμα 32: Enumeration PositionID_Head

Αρχικοποίηση τιμών (Default Values)

Δεν περιλαμβάνει αρχικές τιμές.

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

Δεν περιλαμβάνει συμπεριφορικούς περιορισμούς.

4.2.22 PositionSensorHand

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της θέσης και συγκεκριμένα αναφέρεται στις αρθρώσεις(*joints*) του χεριού του ρομπότ NAO.

Ιδιότητες

Πίνακας 39: Ιδιότητες του PositionSensorHand του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
type	PositionID_Hand	1..1	Ο τύπος αισθητήρα, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO

To *PositionID_Hand* είναι ένα *enumeration* που δημιουργήθηκε με σκοπό την αναπαράσταση των τμημάτων του ρομπότ που αφορούν τη συγκεκριμένη ιδιότητα. Η πλατφόρμα R4A χρειάζεται τη συγκεκριμένη παράμετρο για να εκτελέσει τη συνάρτηση συλλογής δεδομένων από αυτό το είδος αισθητήρων. Το συγκεκριμένο *enumeration* παρουσιάζεται στη συνέχεια.

Συσχετίσεις

Πίνακας 40: Συσχετίσεις του PositionSensorHand του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
PositionSensor	SuperType – Επέκταση	-	Το στοιχείο PositionSensorHand επεκτείνει το στοιχείο PositionSensor



Σχήμα 33: Enumeration PositionID_Hand

Αρχικοποίηση τιμών (Default Values)

Δεν περιλαμβάνει αρχικές τιμές.

Συμπεριφορικό Περιορισμό (Behavioral Constraints)

Δεν περιλαμβάνει συμπεριφορικούς περιορισμούς.

4.2.23 PositionSensorFoot

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της θέσης και συγκεκριμένα αναφέρεται στις αρθρώσεις(*joints*) του ποδιού του ρομπότ NAO.

Ιδιότητες

Πίνακας 41: Ιδιότητες του PositionSensorFoot του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
type	PositionID_Foot	1..1	Ο τύπος αισθητήρα, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO

Το *PositionID_Foot* είναι ένα *enumeration* που δημιουργήθηκε με σκοπό την αναπαράσταση των τμημάτων του ρομπότ που αφορούν τη συγκεκριμένη ιδιότητα. Η πλατφόρμα R4A χρειάζεται τη συγκεκριμένη παράμετρο για να εκτελέσει τη συνάρτηση συλλογής δεδομένων από αυτό το είδος αισθητήρων. Το συγκεκριμένο *enumeration* παρουσιάζεται στη συνέχεια.

Συσχετίσεις

Πίνακας 42: Συσχετίσεις του PositionSensorFoot του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
PositionSensor	SuperType – Επέκταση	-	Το στοιχείο PositionSensorFoot επεκτείνει το στοιχείο PositionSensor

 PositionID_Foot
— LHIPYAWPITCH
— LHIPROLL
— LHIPPITCH
— LKNEEPITCH
— LANKLEPITCH
— LANKLEROLL
— RHIPYAWPITCH
— RHIPROLL
— RHIPPITCH
— RKNEEPITCH
— RANKLEROLL
— RANKLEPITCH

Σχήμα 34: Enumeration PositionID_Foot

Αρχικοποίηση τιμών (Default Values)

Δεν περιλαμβάνει αρχικές τιμές.

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

Δεν περιλαμβάνει συμπεριφορικούς περιορισμούς.

4.2.24 PositionSensorRobot

Σύνοψη

Το συγκεκριμένο στοιχείο του μετα-μοντέλου αναπαριστά την έννοια του αισθητήρα που ανήκει στην κατηγορία της θέσης και συγκεκριμένα αναφέρεται στη δυνατότητα τοποθέτησης(*localization*) του ρομπότ στο χώρο.

Ιδιότητες

Πίνακας 43: Ιδιότητες του PositionSensorRobot του μετα-μοντέλου

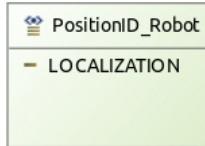
Όνομα	Τύπος	Πολλαπλότητα	Επεξήγηση
type	PositionID_Robot	1..1	Ο τύπος αισθητήρα, όπως αυτός έχει ορισθεί στο documentation της πλατφόρμας, ώστε να γίνεται σωστά η κλήση προς το ρομπότ NAO

Το *PositionID_Robot* είναι ένα *enumeration* που δημιουργήθηκε με σκοπό την αναπαράσταση των τμημάτων του ρομπότ που αφορούν τη συγκεκριμένη ιδιότητα. Η πλατφόρμα R4A χρειάζεται τη συγκεκριμένη παράμετρο για να εκτελέσει τη συνάρτηση συλλογής δεδομένων από αυτό το είδος αισθητήρων, μέσω *SLAM* ααλγορίθμου. Το συγκεκριμένο *enumeration* παρουσιάζεται παρακάτω.

Συσχετίσεις

Πίνακας 44: Συσχετίσεις του PositionSensorRobot του μετα-μοντέλου

Όνομα	Τύπος	Πολλαπλότητα	Δομικοί Περιορισμοί
PositionSensor	SuperType – Επέκταση	-	Το στοιχείο PositionSensorRobot επεκτείνει το στοιχείο PositionSensor



Σχήμα 35: Enumeration PositionID_Robot

Αρχικοποίηση τιμών (Default Values)

name	Localization
type	localization

Συμπεριφορικοί Περιορισμοί (Behavioral Constraints)

Δεν περιλαμβάνει συμπεριφορικούς περιορισμούς.

4.3 Eclipse – Sirius User Interface

Σε αυτό το στάδιο είναι απαραίτητο να δημιουργηθεί ένα γραφικό περιβάλλον, μέσω του οποίου ο χρήστης θα μπορεί εύκολα να δημιουργήσει ένα μοντέλο με γραφικά στοιχεία, το οποίο να είναι σύμφωνο με τους κανόνες του μετα-μοντέλου που ορίστηκε προηγουμένως. Για το σκοπό αυτό, υλοποιήθηκε ένα εργαλείο επεξεργασίας διαγραμμάτων που επιτρέπει στους χρήστες να απεικονίζουν και να επεξεργάζονται τους αισθητήρες που επιθυμούν και τα χαρακτηριστικά τους. Πρόκειται για το αρχείο *SensorProject.odesign*, τύπου *VSM (Viewpoint Specification Model)* το οποίο περιγράφει τη δομή, την εμφάνιση και τη συμπεριφορά των μοντέλων [26].

4.3.1 Διάγραμμα DevelopSystem

To User Interface που υλοποιήθηκε για την παρούσα διπλωματική, ονομάζεται *DevelopSystem* και απαρτίζεται από δύο βασικά στοιχεία, τα *container* που αντιστοιχούν στα υποσυστήματα, και τα *nodes* που αναπαριστούν τους αισθητήρες. Κάθε ένα στοιχείο περιέχει τα χαρακτηριστικά του παραπάνω μετα-μοντέλου, τα οποία φαίνονται στην καρτέλα *properties*. Στη συνέχεια θα παρουσιαστούν όλα τα στοιχεία του εργαλείου αυτού, με τα εικονίδια που αντιστοιχούν στο καθένα, καθώς και ένα στιγμιότυπο από την εισαγωγή του καθενός στο «ταμπλό».

Πίνακας 45: Απεικόνιση AcousticSystem στο Eclipse Sirius UI

Όνομα - Τύπος	Acoustic Sensor System	Container
Εικονίδιο:		
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Category: <input checked="" type="radio"/> DISTANCE <input type="radio"/> ACOUSTIC <input type="radio"/> SPEED <input type="radio"/> PRESSURE <input type="radio"/> POSITION <input type="radio"/> ELECTRIC <input type="radio"/> VISION</p> <p>Refresh rate: <input type="text"/> 10.0</p> <p>Description: <input type="text"/> AudioSystem</p> <p>Record Time: <input type="text"/> 10.0</p> <p>Time Format: <input type="radio"/> SECONDS <input checked="" type="radio"/> MINUTES <input type="radio"/> HOURS <input checked="" type="checkbox"/> Measure Once</p>	

Πίνακας 46: Απεικόνιση DistanceSystem στο Eclipse Sirius UI

Όνομα - Τύπος	Distance Sensor System	Container
Εικονίδιο:		
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Category: <input checked="" type="radio"/> DISTANCE <input type="radio"/> ACOUSTIC <input type="radio"/> SPEED <input type="radio"/> PRESSURE <input type="radio"/> POSITION <input type="radio"/> ELECTRIC <input type="radio"/> VISION</p> <p>Refresh rate: <input type="text"/> 0.25</p> <p>Description: <input type="text"/> Distance Sensor Sub System</p>	

Πίνακας 47: Απεικόνιση ElectricSystem στο Eclipse Sirius UI

Όνομα - Τύπος	Electric Sensor System	Container
Εικονίδιο:		
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Category: <input checked="" type="radio"/> DISTANCE <input type="radio"/> ACOUSTIC <input type="radio"/> SPEED <input type="radio"/> PRESSURE <input type="radio"/> POSITION <input checked="" type="radio"/> ELECTRIC <input type="radio"/> VISION</p> <p>Refresh rate: 300.0</p> <p>Description: Electric Sensor Sub System</p>	

Πίνακας 48: Απεικόνιση VisionSystem στο Eclipse Sirius UI

Όνομα - Τύπος	Electric Sensor System	Container
Εικονίδιο:		
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Category: <input type="radio"/> DISTANCE <input type="radio"/> ACOUSTIC <input type="radio"/> SPEED <input type="radio"/> PRESSURE <input type="radio"/> POSITION <input type="radio"/> ELECTRIC <input checked="" type="radio"/> VISION</p> <p>Refresh rate: 5.0</p> <p>Description: Vision Sensor Sub System</p>	

Πίνακας 49: Απεικόνιση SpeedSystem στο Eclipse Sirius UI

Όνομα - Τύπος	Speed Sensor System	Container
Εικονίδιο:		
Απεικόνιση Χαρακτηριστικών	<p>Properties</p> <p>Category: <input type="radio"/> DISTANCE <input type="radio"/> ACOUSTIC <input checked="" type="radio"/> SPEED <input type="radio"/> PRESSURE <input type="radio"/> POSITION <input type="radio"/> ELECTRIC <input type="radio"/> VISION</p> <p>Refresh rate: 0.1</p> <p>Description: Speed Sensor Sub System</p>	

Πίνακας 50: Απεικόνιση PressureSystem στο Eclipse Sirius UI

Όνομα - Τύπος	Pressure Sensor System	Container
Εικονίδιο:		
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Category: <input checked="" type="radio"/> PRESSURE <input type="radio"/> DISTANCE <input type="radio"/> ACOUSTIC <input type="radio"/> SPEED <input type="radio"/> POSITION <input type="radio"/> ELECTRIC <input type="radio"/> VISION</p> <p>Refresh rate: 0.1</p> <p>Description: Pressure Sensor Sub System</p>	

Πίνακας 51: Απεικόνιση PositionSystem στο Eclipse Sirius UI

Όνομα - Τύπος	Position Sensor System	Container
Εικονίδιο:		
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Category: <input type="radio"/> DISTANCE <input type="radio"/> ACOUSTIC <input type="radio"/> SPEED <input type="radio"/> PRESSURE <input checked="" type="radio"/> POSITION <input type="radio"/> ELECTRIC <input type="radio"/> VISION</p> <p>Refresh rate: 0.2</p> <p>Description: Position Sensor Sub System</p>	

Πίνακας 52: Απεικόνιση Acoustic Sensor στο Eclipse Sirius UI

Όνομα - Τύπος	Acoustic Sensor: Microphone	Node
Εικονίδιο:		
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Name: mic_front</p> <p>Type: <input checked="" type="radio"/> MICFRONT <input type="radio"/> MICREAR <input type="radio"/> MICLEFT <input type="radio"/> MICRIGHT</p>	

Πίνακας 53: Απεικόνιση Distance Sensor στο Eclipse Sirius UI

Όνομα - Τύπος	Distance Sensor: Sonar	Node
Εικονίδιο:		 SONAR
Απεικόνιση Χαρακτηριστικών:	Properties Name: <input type="text" value="Sonar_Right"/> ? Max Distance Accepted: <input type="text" value="1.8"/> ? Min Distance Accepted: <input type="text" value="0.23"/> ? Type: <input checked="" type="radio"/> RIGHT <input type="radio"/> LEFT Alarm Value: <input type="text" value="0.0"/> ? <input type="checkbox"/> Notifications	

Πίνακας 54: Απεικόνιση Electric Sensor στο Eclipse Sirius UI

Όνομα - Τύπος	Electric Sensor: Battery	Node
Εικονίδιο:		 BATTERY
Απεικόνιση Χαρακτηριστικών:	Properties Name: <input type="text" value="Battery"/> ? Type: <input checked="" type="radio"/> BATTERY Alarm Value: <input type="text" value="30.0"/> ? <input checked="" type="checkbox"/> Notifications	

Πίνακας 55: Απεικόνιση Position Sensor Robot στο Eclipse Sirius UI

Όνομα - Τύπος	Position Sensor: Robot	Node
Εικονίδιο:		 LOCALIZATION DATA
Απεικόνιση Χαρακτηριστικών:	Properties Name: <input type="text" value="Joint_Localization"/> ? Type: <input checked="" type="radio"/> LOCALIZATION	

Πίνακας 56: Απεικόνιση Position Sensor Posture στο Eclipse Sirius UI

Όνομα - Τύπος	Position Sensor: Posture	Node
Εικονίδιο:		ROBOT POSTURE
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Name: <input type="text" value="Robot_Posture"/> Type: <input checked="" type="radio"/> ROBOT</p>	

Πίνακας 57: Απεικόνιση Position Sensor Head στο Eclipse Sirius UI

Όνομα - Τύπος	Position Sensor: Head	Node
Εικονίδιο:		HEAD JOINT
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Name: <input type="text" value="Joint_Headyaw"/> Type: <input checked="" type="radio"/> HEADYAW <input type="radio"/> HEADPITCH</p>	

Πίνακας 58: Απεικόνιση Position Sensor tFrame στο Eclipse Sirius UI

Όνομα - Τύπος	Position Sensor: tFrame	Node
Εικονίδιο:		T-FRAME DATA TRANSFORMATION
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Name: <input type="text" value="Tframe_"/> Origin: <input checked="" type="radio"/> BATTERY <input type="radio"/> CAMERATOP <input type="radio"/> HEAD_TOUCH_FRONT <input type="radio"/> HEAD_TOUCH_MIDDLE <input type="radio"/> HEAD_TOUCH_REAR <input type="radio"/> LFOOT_BUMBER_LEFT <input type="radio"/> LFOOT_BUMBER_RIGHT <input type="radio"/> MICFRONT <input type="radio"/> MICREAR <input type="radio"/> MICLEFT <input type="radio"/> MICRIGHT <input type="radio"/> RFOOT_BUMBER_LEFT <input type="radio"/> RFOOT_BUMBER_RIGHT <input type="radio"/> LHAND_TOUCH_BACK <input type="radio"/> LHAND_TOUCH_LEFT <input type="radio"/> LHAND_TOUCH_RIGHT <input type="radio"/> RHAND_TOUCH_BACK <input type="radio"/> RHAND_TOUCH_LEFT <input type="radio"/> RHAND_TOUCH_RIGHT <input type="radio"/> SONAR_LEFT <input type="radio"/> SONAR_RIGHT <input type="radio"/> HEADYAW <input type="radio"/> HEADPITCH <input type="radio"/> LSHOULDERPITCH <input type="radio"/> RSHOULDERPITCH <input type="radio"/> LELBOWYAW <input type="radio"/> RSHOULDERROLL <input type="radio"/> RELBOWYAW <input type="radio"/> RLBOVROLL <input type="radio"/> RWRYSTYAW <input type="radio"/> LHIPWAVPITCH <input type="radio"/> LHIPROLL <input type="radio"/> LHIPPITCH <input type="radio"/> LKNEEPITCH <input type="radio"/> LANKLEPITCH <input type="radio"/> RHIPWAVPITCH <input type="radio"/> RHIPROLL <input type="radio"/> RKNEEPITCH <input type="radio"/> RANKLEPITCH <input type="radio"/> RLANKLEROLL <input type="radio"/> RLBOWROLL <input type="radio"/> LWRYSTYAW Target: <input checked="" type="radio"/> FRAME_ROBOT <input type="radio"/> FRAME_TORSO <input type="radio"/> FRAME_WORLD</p>	

Πίνακας 59: Απεικόνιση Position Sensor Hand στο Eclipse Sirius UI

Όνομα - Τύπος	Position Sensor: Hand	Node
Εικονίδιο:	 HAND JOINT	
Απεικόνιση Χαρακτηριστικών:	Properties Name: <input type="text" value="Joint_Lshoulderpitch"/> Type: <input checked="" type="radio"/> LSHOULDERPITCH <input type="radio"/> LSHOULDERROLL <input type="radio"/> LELBOWYAW <input type="radio"/> LELBOWROLL <input type="radio"/> LWristyaw <input type="radio"/> RShoulderpitch <input type="radio"/> RShoulderroll <input type="radio"/> RLewbowyaw <input type="radio"/> RLewbowroll <input type="radio"/> RWristyaw	

Πίνακας 60: Απεικόνιση Position Sensor Foot στο Eclipse Sirius UI

Όνομα - Τύπος	Position Sensor: Foot	Node
Εικονίδιο:	 FOOT JOINT	
Απεικόνιση Χαρακτηριστικών:	Properties Name: <input type="text" value="Joint_Lhipyawpitch"/> Type: <input checked="" type="radio"/> LHIPAWYPITCH <input type="radio"/> LHIPROLL <input type="radio"/> LHIPPITCH <input type="radio"/> LKNEEPITCH <input type="radio"/> LANKLEPITCH <input type="radio"/> LANKLEROLL <input type="radio"/> RHIPAWYPITCH <input type="radio"/> RHIPROLL <input type="radio"/> RHIPPITCH <input type="radio"/> RKNEEPITCH <input type="radio"/> RANKLEROLL <input type="radio"/> RANKLEPITCH	

Πίνακας 61: Απεικόνιση Pressure Sensor Head στο Eclipse Sirius UI

Όνομα - Τύπος	Pressure Sensor: Head	Node
Εικονίδιο:	 HEAD BUTTON	
Απεικόνιση Χαρακτηριστικών:	Properties Name: <input type="text" value="Touch_Head_touch_rear"/> Type: <input checked="" type="radio"/> HEAD_TOUCH_REAR <input type="radio"/> HEAD_TOUCH_FRONT <input type="radio"/> HEAD_TOUCH_MIDDLE	

Πίνακας 62: Απεικόνιση Pressure Sensor Hand στο Eclipse Sirius UI

Όνομα - Τύπος	Position Sensor: Hand	Node
Εικονίδιο:	 HAND TACTILE	
Απεικόνιση Χαρακτηριστικών:	Properties Name: <input type="text" value="Touch_Lhand_touch_back"/> Type: <input checked="" type="radio"/> LHAND_TOUCH_BACK <input type="radio"/> LHAND_TOUCH_LEFT <input type="radio"/> LHAND_TOUCH_RIGHT <input type="radio"/> RHAND_TOUCH_BACK <input type="radio"/> RHAND_TOUCH_LEFT <input type="radio"/> RHAND_TOUCH_RIGHT	

Πίνακας 63: Απεικόνιση Pressure Sensor Foot στο Eclipse Sirius UI

Όνομα - Τύπος	Position Sensor: Foot	Node
Εικονίδιο:		
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Name: <input type="text" value="Touch_Lfoot_bumper_left"/> Type: <input checked="" type="radio"/> LFOOT_BUMPER_LEFT <input type="radio"/> LFOOT_BUMPER_RIGHT <input type="radio"/> RFOOT_BUMPER_LEFT <input type="radio"/> RFOOT_BUMPER_RIGHT</p>	

Πίνακας 64: Απεικόνιση Speed Sensor στο Eclipse Sirius UI

Όνομα - Τύπος	Speed Sensor: Accelerometer	Node
Εικονίδιο:		
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Name: <input type="text" value="Accelerometer"/> Type: <input checked="" type="radio"/> TORSO</p>	

Πίνακας 65: Απεικόνιση Vision Sensor στο Eclipse Sirius UI

Όνομα - Τύπος	Vision Sensor: RGB Camera	Node
Εικονίδιο:		
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Name: <input type="text" value="Camera_Top"/> Type: <input checked="" type="radio"/> TOP <input type="radio"/> BOTTOM</p> <p>Resolution: <input type="radio"/> PX40x30 <input type="radio"/> PX80x60 <input type="radio"/> PX160x120 <input type="radio"/> PX320x240 <input checked="" type="radio"/> PX640x480 <input type="radio"/> PX1280x960</p>	

Πίνακας 66: Απεικόνιση SensorSystem στο Eclipse Sirius UI

Όνομα - Τύπος	Sensor System	-
Εικονίδιο:	Δεν υπάρχει. Το SensorSystem αναπαρίσταται από το λευκό ταμπλό του διαγράμματος	
Απεικόνιση Χαρακτηριστικών:	<p>Properties</p> <p>Name: <input type="text" value="MySensorSystem"/> Duration: <input type="text" value="1.0"/> Time Format: <input type="radio"/> SECONDS <input type="radio"/> MINUTES <input checked="" type="radio"/> HOURS</p>	

Έχουν προστεθεί στο User Interface ορισμένες δυνατότητες που διευκολύνουν το χρήστη να δημιουργήσει γρήγορα το σύστημα της επιλογής του. Αρχικά, όπως έχει αναφερθεί σε προηγούμενη ενότητα, έχουν προκαθοριστεί ορισμένες αρχικές τιμές για τα περισσότερα χαρακτηριστικά των αισθητήρων. Σκοπός των προκαθορισμένων τιμών είναι να καθιστούν ακόμη γρηγορότερη τη διαδικασία σχεδίασης του συστήματος. Αξιοσημείωτο είναι εδώ, πως οι τιμές αυτές έχουν οριστεί έπειτα από την περίοδο δοκιμών, κατά την οποία έγινε φανερή η προτίμηση σε ορισμένες τιμές. Ακόμη, μία επιπλέον δυνατότητα που παρέχεται στο χρήστη, είναι κάνοντας διπλό κλικ σε ένα εικονίδιο που αντιστοιχεί σε αισθητήρα ή, να θέτει σε αυτόν αυτόματα ένα όνομα ανάλογα με το είδος του και το id που έχει ορισθεί για αυτόν στην πλατφόρμα R4A. Το ίδιο ισχύει και για το υποσύστημα της κατηγορίας ήχου, όπου η συμπλήρωση της περιγραφής του στοιχείου καθίσταται απαραίτητη. Στα πλαίσια της λογικής της εύκολης και γρήγορης σχεδίασης, προσθέτοντας τα υποσυστήματα που αφορούν τις κατηγορίες *Acoustic, Distance, Electric, Speed, Vision*, το διάγραμμα προσθέτει αυτόματα έναν αισθητήρα εντός του υποσυστήματος, με τις προκαθορισμένες τιμές. Επιπλέον, κάθε χαρακτηριστικό όλων των στοιχείων που εμφανίζονται στο διάγραμμα έχει ειδικά διαμορφωμένο μήνυμα επεξήγησης, ώστε να μπορεί οποιοσδήποτε να χρησιμοποιεί το σύστημα, ανεξαρτήτως των γνώσεων που χρειάζονται αναφορικά με τα ρομπότ, τους αισθητήρες τους, ή ακόμη και την πλατφόρμα R4A, για την υλοποίηση του.

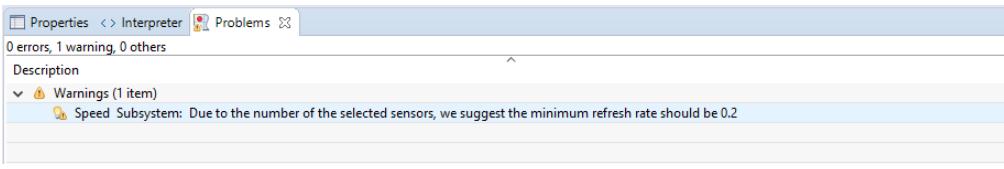
4.3.2 Κανόνες επικύρωσης διαγράμματος

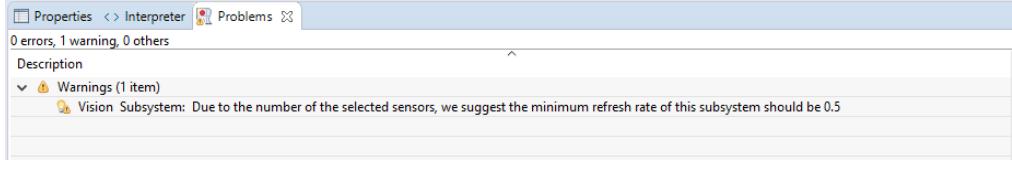
Η ύπαρξη του User Interface παίζει διπλό ρόλο. Αρχικά, όπως αναφέρθηκε στην προηγούμενη ενότητα, αποτελεί έναν εύκολο τρόπο σχεδίασης του συστήματος μέσω γραφικών αναπαραστάσεων των στοιχείων. Επιπλέον όμως, συνιστά έναν τρόπο διασφάλισης πως το τελικό μοντέλο θα είναι σωστό και σύμφωνο με το μετα-μοντέλο. Κατά συνέπεια, οι αλγόριθμοι που θα παραχθούν από τη γεννήτρια *Accelero* θα ανταποκρίνονται σωστά και θα λειτουργούν με αξιοπιστία. Για το σκοπό αυτό, το διάγραμμα εμπειριέχει μία σειρά κανόνων επικύρωσης (*Validation Rules*). Πιο αναλυτικά, ο χρήστης ανά πάσα στιγμή μπορεί να αποθηκεύσει το διάγραμμα στο οποίο σχεδιάζει το σύστημα του και να το ελέγξει. Σε περίπτωση που κάποιος από τους κανόνες παραβιάζεται, τότε εμφανίζεται ένα μήνυμα λάθους, το οποίο έχει γραφτεί με τέτοιο τρόπο, ώστε ο χρήστης να το ερμηνεύει εύκολα. Αυτή η διαδικασία επαναλαμβάνεται έως ότου εξαλειφθούν όλα τα λάθη και το μοντέλο είναι έγκυρο.

Οι περισσότεροι κανόνες αφορούν λογικούς περιορισμούς. Για παράδειγμα, να μην μπορεί ο χρήστης να παραμετροποίησει τον ίδιο αισθητήρα περισσότερες από μία φορά ή να πρέπει να συμπληρώσει όλα τα χαρακτηριστικά που χρειάζονται για να λειτουργεί σωστά το σύστημα. Οι συμπεριφορικοί αυτοί περιορισμοί

έχουν παρουσιαστεί αναλυτικά στην ενότητα 4.2 του κεφαλαίο. Στο σημείο αυτό, πρέπει να επισημανθούν οι κανόνες ελέγχου που αφορούν τον έλεγχο των refresh rate και παρουσιάζονται στο χρήστη ως warning.

Όπως έχει ήδη αναφερθεί, χρησιμοποιούνται συναρτήσεις της πλατφόρμας R4A προκειμένου να γίνει η επικοινωνία με το ρομπότ, να ληφθούν οι μετρήσεις και να αποθηκευτούν τα δεδομένα στην τοπική βάση δεδομένων του ρομπότ. Αυτή η διαδικασία εισάγει στο σύστημα επιπλέον σημαντικούς παράγοντες που επηρεάζουν την απόδοσή του, όπως είναι οι περιορισμοί που έχουν τεθεί από την πλατφόρμα R4A, καθώς και το δίκτυο που είναι συνδεδεμένο το ρομπότ. Υπάρχουν πολλοί εξωγενείς παράγοντες που επηρεάζουν την ποιότητα του δικτύου, ενώ παράλληλα, η πλατφόρμα θέτει περιορισμούς στον τρόπο που γίνονται οι κλήσεις. Αυτοί οι παράγοντες όμως ξεπερνούν τα πλαίσια αυτής της διπλωματικής εργασίας και για το λόγο αυτό δε θα αναλυθούν. Ωστόσο, για να είναι το σύστημα μας όσο το δυνατόν αξιόπιστο και να εκτελεί σωστά τις κλήσεις για τον εκάστοτε αισθητήρα, χρειάζεται να τεθούν περιορισμοί που αφορούν το πλήθος των αισθητήρων σε σχέση με το refresh rate που έχουν. Οι περιορισμοί αυτοί προέκυψαν έπειτα από την περίοδο των δοκιμών για συστήματα με διαφορετικά είδη και πλήθος αισθητήρων και εμφανίζονται στο χρήστη με τη μορφή warning, καθώς δεν επηρεάζουν την αυτόματη παραγωγή του API. Βάσει των συνθηκών του δικτύου του *Εργαστήριο Επεξεργασίας Πληροφορία και Υπολογισμών*, έπειτα από την πειραματική περίοδο, ορίστηκαν οι περιορισμοί για το refresh rate. Στη συνέχεια παρουσιάζονται ενδεικτικά δύο από αυτούς. Ο αναγνώστης μπορεί να βρει αυτούς τους κανόνες στο αρχείο *SensorProject.odesign*. Το αρχείο αυτό είναι διαθέσιμο στο github, στον σύνδεσμο: https://github.com/dimitrantz/Thesis/tree/master.ecore/SensorProject_SiriusUI/SensorProject.design.

Warning	
AQL Code	aql: if (not(self.oclIsKindOf(sensorProject::VisionSystem)) then ((self.eContainer().eAllContents ()-> filter(sensorProject::Sensor)-> size ()>0) and (self.eContainer().eAllContents ()-> filter(sensorProject::Sensor)-> size () < 4)) implies self.refresh_rate >=0.2 else ' ' endif
Στοιχείο εφαρμογής	SubSystem
Περιγραφή	Ο κανόνας αυτός ελέγχει αρχικά το πλήθος των αισθητήρων στο σύστημα, εκτός από τους αισθητήρες όρασης. Εάν το πλήθος είναι έως και 3, τότε σε περίπτωση που ένα υποσύστημα έχει τιμή Refresh Rate μικρότερη από 0.2, εμφανίζει ένα μήνυμα warning, ειδοποιώντας το χρήστης πως η ελάχιστη τιμή για τη βέλτιστη λειτουργία του συστήματος είναι 0.2.
Απεικόνιση Warning	 A screenshot of the Sirius UI interface. At the top, there are tabs for 'Properties', 'Interpreter', and 'Problems'. Below the tabs, it says '0 errors, 1 warning, 0 others'. Under the 'Description' section, there is a dropdown menu. In the dropdown, under 'Warnings (1 item)', it shows a single warning: 'Speed Subsystem: Due to the number of the selected sensors, we suggest the minimum refresh rate should be 0.2'.

Warning	
AQL Code	aql: ((self.eContainer().eAllContents() -> filter(sensorProject::VisionSensor) -> size() > 0) and (self.eContainer().eAllContents() -> filter(sensorProject::Sensor) -> size() < 4)) implies self.refresh_rate >= 0.5
Στοιχείο εφαρμογής	VisionSystem
Περιγραφή	Ο κανόνας αυτός ελέγχει αρχικά πως υπάρχει αισθητήρας όρασης στο σύστημα, και πως το συνολικό πλήθος τους είναι έως και 3. Τότε σε περίπτωση που το υποσύστημα όρασης έχει τιμή Refresh Rate μικρότερή από 0.5, εμφανίζει ένα μήνυμα warning, ειδοποιώντας το χρήστης πως η ελάχιστη τιμή για τη βέλτιστη λειτουργία του συστήματος και αυτό το υποσύστημα είναι 0.5.
Απεικόνιση Warning	

4.4 Acceleo Project

Σε αυτήν την ενότητα, θα εξεταστεί ο κώδικας που παράγεται αυτόματα στην παρούσα διπλωματική. Ο κώδικας αυτός αποτελείται από *modules*, καθένα από τα οποία δέχεται μία είσοδο και παράγει ένα αρχείο. Στον παρακάτω πίνακα παρουσιάζονται συνοπτικά τα *modules* που αναπτύχθηκαν, η είσοδος που δέχεται το καθένα και μία συνοπτική περιγραφή.

Πίνακας 67: Επισκόπηση των *modules* του μετασχηματισμού

Όνομα	Είσοδος	Έξοδος	Περιγραφή
Main	SensorSystem	Python 2.7	Το main module του μετασχηματισμού. Είναι υπεύθυνο για την παραγωγή του αρχείου <i>main.py</i> το οποίο είναι και το εκτελέσιμο αρχείο της εφαρμογής. Καλεί τα <i>modules</i> , <i>SensorUtilities</i> και το <i>RedisAPI</i>
RedisAPI	SensorSystem	Python 2.7	Υπεύθυνο για την παραγωγή του αρχείου <i>RedisAPI.py</i> που αφορά την αποθήκευση όλης της απαραίτητης πληροφορίας στη βάση

SensorUtilities	SensorSystem	Python 2.7	Υπεύθυνο για την παραγωγή του αρχείου <i>sensors.py</i> που συγκεντρώνει όλες τις κλάσεις των αισθητήρων. Καλεί όλα τα υπόλοιπα <i>modules</i>
SensorBase	SensorSystem	Python 2.7	Υπεύθυνο για την παραγωγή του αρχείου <i>sensor.py</i> που καθιστά την κύρια κλάση, την οποία κληρονομούν όλες οι υπόλοιπες κλάσεις αισθητήρων
DistanceSensor	SensorSystem	Python 2.7	Υπεύθυνο για την παραγωγή του αρχείου <i>distanceSensor.py</i> που υλοποιεί την κλάση του αισθητήρα απόστασης
SpeedSensor	SensorSystem	Python 2.7	Υπεύθυνο για την παραγωγή του αρχείου <i>speedSensor.py</i> που υλοποιεί την κλάση του αισθητήρα ταχύτητας
PressureSensor	SensorSystem	Python 2.7	Υπεύθυνο για την παραγωγή του αρχείου <i>pressureSensor.py</i> που υλοποιεί την κλάση του αισθητήρα πίεσης
VisionSensor	SensorSystem	Python 2.7	Υπεύθυνο για την παραγωγή του αρχείου <i>visionSensor.py</i> που υλοποιεί την κλάση του αισθητήρα όρασης
ElectricSensor	SensorSystem	Python 2.7	Υπεύθυνο για την παραγωγή του αρχείου <i>electricSensor.py</i> που υλοποιεί την κλάση του αισθητήρα ηλεκτρισμού
PositionSensor	SensorSystem	Python 2.7	Υπεύθυνο για την παραγωγή του αρχείου <i>positionSensor.py</i> που υλοποιεί την κλάση του αισθητήρα θέσης
TframeSensor	SensorSystem	Python 2.7	Υπεύθυνο για την παραγωγή του αρχείου <i>tframeSensor.py</i> που υλοποιεί την κλάση του αισθητήρα μετασχηματισμού tframe
PostureSensor	SensorSystem	Python 2.7	Υπεύθυνο για την παραγωγή του αρχείου <i>postureSensor.py</i> που υλοποιεί την κλάση του αισθητήρα απόστασης

AcousticSystemBase	SensorSystem	Python 2.7	Υπεύθυνο για την παραγωγή του αρχείου <i>acousticSystem.py</i> που υλοποιεί την κλάση του αισθητήρα απόστασης
SensorSystemBase	SensorSystem	Python 2.7	Είναι το πιο σημαντικό module ολόκληρου του συστήματος. Είναι υπεύθυνο για την παραγωγή του αρχείου <i>sensorSystem.py</i> που υλοποιεί ολόκληρο το σύστημα

Παρακάτω παρουσιάζονται ενδεικτικά ορισμένα τμήματα του κώδικα σε *Acceleo Template Language*. Τα τμήματα αυτά αφορούν το SensorSystemBase module.

Αλγόριθμος 1: Υλοποίηση συστήματος αισθητήρων βάσει των παραμέτρων από το μοντέλο του χρήστη

```
class SensorSystem:
    '''Master class of the system. Common class for a sensor system. '''

    def __init__(self):
        self.controller = api.RobotApi()
        self.name = '[aSensorSystem.name/]'
        tempTime = [aSensorSystem.duration/]
        [if (aSensorSystem.timeFormat.toString() = 'SECONDS')]
        self.duration = tempTime
        [elseif (aSensorSystem.timeFormat.toString() = 'MINUTES')]
        self.duration = tempTime * 60
        [elseif (aSensorSystem.timeFormat.toString() = 'HOURS')]
        self.duration = tempTime * 3600
        [/if]
        self.mySensors = list()
        self.myCategories = list()
        self.mySensorsIds = list() # Data needed for API
```

Στον Αλγόριθμο 1 ελέγχεται η μονάδα μέτρησης που ορίζει ο χρήστης για τη διάρκεια λειτουργίας του συστήματος και γίνεται σωστά η καταχώρηση της τιμής αυτής.

Στον Αλγόριθμο 2 ελέγχονται όλα τα στοιχεία που ανήκουν στο *hasSubsystem* του *SensorSystem*. Εάν κάποιο από αυτά είναι τύπου *DistanceSystem*, τότε γίνεται περαιτέρω έλεγχος στους αισθητήρες που ανήκουν σε αυτό. Με αυτόν τον τρόπο εξασφαλίζεται πως μόνο σε περίπτωση που το μοντέλο περιλαμβάνει στοιχείο τύπου *DistanceSensor* θα δημιουργηθεί το κατάλληλο instance και θα προστεθεί στο σύστημα ένας αισθητήρας αυτής της κατηγορίας.

Αλγόριθμος 2: Δημιουργία ενός instance της κλάσης DistanceSensor βάσει των παραμέτρων που θέτει ο χρήστης στο μοντέλο και καταχώρηση αυτού στο σύστημα

```

[for (subsystem: SubSystem | aSensorSystem.hasSubsystem)]
    [if (subsystem.oclIsTypeOf(DistanceSystem))]
        #Creating Distance Sensors
        [for (sensor: Sensor | subsystem.hasSensor)]
            [if (sensor.oclIsTypeOf(DistanceSensor))]
                temp = sensors.DistanceSensor()
                temp.category = '[subsystem.category.toString().toUpperCase() /]'
                temp.frequency = '[subsystem.refresh_rate /]'
                temp.duration = self.duration
                temp.type = '[sensor.oclAsType(DistanceSensor).type.toString().toLowerCase() /]'
                temp.name = '[sensor.oclAsType(DistanceSensor).name /]'
                temp.maxDistance = '[sensor.oclAsType(DistanceSensor).maxDistanceAccepted /]'
                temp.minDistance = '[sensor.oclAsType(DistanceSensor).minDistanceAccepted /]'
                [if (sensor.oclAsType(DistanceSensor).notifications = true)]
                    temp.alarmValue = '[sensor.oclAsType(DistanceSensor).alarmValue /]'
                [else]
                    temp.alarmValue = 'None'
                [/if]
                self.mySensors.append(temp)
                self.myCategories.append('[subsystem.category.toString().toUpperCase() /]')

                # Data needed for API
                redisAPI.delete_from_redis(temp.name)
                self.mySensorsIds.append(str({'category': temp.category, 'index': temp.type,
                                              'sensor': temp.name, 'frequency': temp.frequency,
                                              'maxDistanceAccepted': temp.maxDistance,
                                              'minDistanceAccepted': temp.minDistance,
                                              'alarmValue': temp.alarmValue}))
                temp.get_distance_sensor()
            [/if]
        [/for]
    ]

```

Επιπλέον, σε αυτό το τμήμα κώδικα (αλγ.2), προκειμένου να αποθηκεύονται σωστά οι τιμές των αισθητήρων και να μην με άλλες υπάρχουσες στη βάση δεδομένων, διαγράφεται εφόσον υπάρχει κλειδί με το ίδιο όνομα, όπως ορίστηκε για αυτόν τον αισθητήρα. Τέλος, στο τμήμα αυτό κώδικα, παράγονται ορισμένα δεδομένα που είναι απαραίτητα για την παραγωγή του API.

Στη συνέχεια, στον Αλγόριθμο 3, παρουσιάζεται η περίπτωση που το SubSystem είναι τύπου AcousticSystem. Σε αυτήν την περίπτωση, για λειτουργεί σύμφωνα με την πλατφόρμα R4A, ορίζεται ως υποσύστημα αισθητήρων το οποίο θα περιέχει όλα τα μικρόφωνα που θα ενεργοποιηθούν για να εκτελέσουν ηχογράφηση. Ενδιαφερόμενοι για το υποσύστημα και για τον κάθε ένα αισθητήρα χωριστά, στο τμήμα αυτό κώδικα αφού καθοριστούν τα μικρόφωνα για την ηχογράφηση, γίνεται κλήση στο ρομπότ μέσω της κατάλληλης συνάρτησης της R4A για να ανακτηθούν τα τεχνικά χαρακτηριστικά του εκάστοτε μικροφώνου. Στη συνέχεια αποθηκεύονται στη βάση δεδομένων χρησιμοποιώντας τις συναρτήσεις που περιέχονται στο αρχείο RedisAPI.py, ώστε να γίνει η πληροφορία διαθέσιμη μέσω του API.

Αλγόριθμος 3: Δημιουργία ενός *instance* της κλάσης *AcousticSystem* βάσει των παραμέτρων
που θέτει ο χρήστης στο μοντέλο και καταχώρηση αυτού στο σύστημα

```

[elseif subsystem.oclIsTypeOf(AcousticSystem)]
    #Creating Acoustic Sensors
    temp = sensors.AcousticSystem()

    temp.category = '[subsystem.oclAsType(AcousticSystem).category.toString().toUpperCase() /]'
    temp.duration = self.duration
    temp.frequency = '[subsystem.oclAsType(AcousticSystem).refresh_rate /]'
    temp.description = '[subsystem.oclAsType(AcousticSystem).description /]'
    tempTime = '[subsystem.oclAsType(AcousticSystem).recordTime /]'

    [if (subsystem.oclAsType(AcousticSystem).timeFormat.toString() = 'SECONDS')]
        temp.recordTime = tempTime
    [elseif (subsystem.oclAsType(AcousticSystem).timeFormat.toString() = 'MINUTES')]
        temp.recordTime = tempTime * 60
    [elseif (subsystem.oclAsType(AcousticSystem).timeFormat.toString() = 'HOURS')]
        temp.recordTime = tempTime * 3600
    [/if]
    temp.myMicrophones = list()
    [for (sensor: Sensor | subsystem.hasSensor)]
        [if (sensor.oclIsTypeOf(AcousticSensor))]
            [if (sensor.oclAsType(AcousticSensor).type.toString() = 'MICFRONT')]
                temp.myMicrophones.append('micFront')
            temp.get_acoustic_sensor('[sensor.oclAsType(AcousticSensor).name.toString()/]')
            [elseif (sensor.oclAsType(AcousticSensor).type.toString() = 'MICREAR')]
                temp.myMicrophones.append('micRear')
            temp.get_acoustic_sensor('[sensor.oclAsType(AcousticSensor).name.toString()/]')
            [elseif (sensor.oclAsType(AcousticSensor).type.toString() = 'MICLEFT')]
                temp.myMicrophones.append('micLeft')
            temp.get_acoustic_sensor('[sensor.oclAsType(AcousticSensor).name.toString()/]')
            [elseif (sensor.oclAsType(AcousticSensor).type.toString() = 'MICRIGHT')]
                temp.myMicrophones.append('micRight')
            temp.get_acoustic_sensor('[sensor.oclAsType(AcousticSensor).name.toString()/]')
        [/if]
    [/for]
    self.mySensors.append(temp)
    self.myCategories.append('[subsystem.category.toString().toUpperCase() /]')
    mics = ''
    if len(temp.myMicrophones) == 1:
        mics = temp.myMicrophones['[/]0[/]']
    else:
        for name in temp.myMicrophones:
            if name == myMicrophones['[/]-1[/]']:
                mics = mics + str(name)
            else:
                mics = mics + str(name) + ', '
    # Data needed for API
    redisAPI.delete_from_redis(temp.description)
    self.mySensorsIds.append(str({'category': temp.category, 'sensor': temp.description,
                                   'index': mics, 'frequency': temp.frequency,
                                   'recordTime': str(temp.recordTime)}))
[/if]

```

Με αντίστοιχο τρόπο έχουν δημιουργηθεί κώδικες για όλα τα στοιχεία που μπορεί να έχει ένα μοντέλο.
Περισσότερες πληροφορίες για τον κώδικα της γεννήτριας *Accelero* που γράφτηκε για την εφαρμογή, μπορείτε να βρείτε στο [github](https://github.com/dimitrantz/Thesis/tree/master.ecore/SensorProjectGenerator) στο σύνδεσμο: <https://github.com/dimitrantz/Thesis/tree/master.ecore/SensorProjectGenerator>

4.5 Redis

Στην ενότητα αυτή θα παρουσιαστούν λακωνικά οι συναρτήσεις που σχετίζονται με την αποθήκευση των πληροφοριών που αναφέρονται στους αισθητήρες, αλλά και με την αποθήκευση των δεδομένων που συλλέχθηκαν από αυτούς. Όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο, ο τρόπος αποθήκευσης γίνεται μέσω του Redis, με μη σχεσιακό τρόπο, παρέχοντας αρκετή ευελιξία στην αποθήκευση της απαραίτητης πληροφορίας. Επιπλέον, η χρήση της γλώσσας Python, προσδίδει με τη σειρά της μεγαλύτερη ευκολία στη διαχείριση αυτών των δεδομένων, τα οποία είναι ποικίλων τύπων. Όπως είναι λογικό τα είδη των συναρτήσεων που υλοποιήθηκαν χωρίζονται σε δύο κατηγορίες, αυτές που χειρίζονται τα δεδομένα και τα αποθηκεύουν και αυτές που τα ανακτούν από τη βάση.

4.5.1 Συναρτήσεις Αποθήκευσης

Το αρχείο που αφορά την αποθήκευση των δεδομένων ονομάζεται *RedisAPI.py* και εμπεριέχεται στον κώδικα που παράγεται αυτόματα από το *Accelero*, μέσω του *RedisAPI.mtl*. Η επιλογή να παράγεται αυτός ο κώδικας αυτόματα γίνεται για να αποφευχθούν τυχόν λάθη που θα μπορούσαν να υπάρξουν σε άλλες περιπτώσεις, ενώ παράλληλα διευκολύνεται ο χρήστης της εφαρμογής, αφού δε χρειάζεται να κάνει καμία άλλη ενέργεια.

Το αρχείο αυτό περιλαμβάνει τις ακόλουθες συναρτήσεις:

delete_from_redis(key)

Περιγραφή

Εκτελεί διαγραφή ενός δεδομένου κλειδιού, εφόσον αυτό υπάρχει στη βάση. Με αυτόν τον τρόπο, θα αποθηκευτούν στη βάση τα δεδομένα του αισθητήρα στο σωστό κλειδί, χωρίς προηγούμενη πληροφορία.

Παράμετροι εισόδου

key: String που περιέχει το όνομα του κλειδιού προς διαγραφή

save_sensor_info(key)

Περιγραφή

Εκτελεί αποθήκευση των τεχνικών χαρακτηριστικών ενός αισθητήρα σε ένα συγκεκριμένο κλειδί, το οποίο προκύπτει ως συνδυασμός της λέξης *info_* και του κλειδιού του οποίου την τιμή επιθυμούμε να ανακτήσουμε.

Παράμετροι εισόδου

key: *String* που περιέχει το όνομα του κλειδιού του οποίου το περιεχόμενο επιθυμούμε να ανακτηθεί

save_categories(value)

Περιγραφή

Εκτελεί αποθήκευση των κατηγοριών του συστήματος, βάσει του μοντέλου που δημιούργησε ο χρήστης.

Παράμετροι εισόδου

value: Λίστα από *String* που περιέχουν τα όνομα των κατηγοριών προς αποθήκευση

save_sensors_parameters(sensors)

Περιγραφή

Εκτελεί αποθήκευση όλων των παραμέτρων των αισθητήρων του συστήματος, βάσει του μοντέλου που δημιούργησε ο χρήστης.

Παράμετροι εισόδου

sensors: Λίστα από *Tuples* που περιέχουν όλα τα χαρακτηριστικά των αισθητήρων που θέτει ο χρήστης στο μοντέλο του.

save_measurement(key, value)

Περιγραφή

Εκτελεί αποθήκευση μίας μέτρησης ενός αισθητήρα.

Παράμετροι εισόδου

key: *String* που περιέχει το όνομα του αισθητήρα, βάσει του οποίου θα γίνει η αποθήκευση των μετρήσεων του στη βάση

value: *Tuple* το οποίο περιέχει τη μέτρηση και το timestamp αυτής

save_audio(key, name, timestamp)

Περιγραφή

Εκτελεί αποθήκευση του ονόματος αρχείου ήχου. Βάσει αυτού του ονόματος, γίνεται η αναζήτηση στο φάκελο όπου αποθηκεύονται τα αρχεία για να ανακτηθεί.

Παράμετροι εισόδου

key: *String* που περιέχει το όνομα του αισθητήρα, βάσει του οποίου θα γίνει η αποθήκευση των μετρήσεων του στη βάση

name: Ένα όνομα για το αρχείο ήχου, το οποίο δημιουργείται αυτόματα από το σύστημα

timestamp: Ένα μοναδικό χαρακτηριστικό που προσδιορίζει τη χρονική στιγμή που δημιουργήθηκε το αρχείο

save_audio(key, name, timestamp)

Περιγραφή

Εκτελεί αποθήκευση του ονόματος μίας εικόνας. Βάσει αυτού του ονόματος, γίνεται η αναζήτηση στο φάκελο όπου αποθηκεύονται τα αρχεία για να ανακτηθεί.

Παράμετροι εισόδου

key: *String* που περιέχει το όνομα του αισθητήρα, βάσει του οποίου θα γίνει η αποθήκευση των μετρήσεων του στη βάση

name: Ένα όνομα για το αρχείο ήχου, το οποίο δημιουργείται αυτόματα από το σύστημα

timestamp: Ένα μοναδικό χαρακτηριστικό που προσδιορίζει τη χρονική στιγμή που δημιουργήθηκε το αρχείο

4.5.2 Συναρτήσεις Ανάκτησης

Το αρχείο που περιέχει την ανάκτηση των δεδομένων ονομάζεται *flaskApp.py*. Πρόκειται για το αρχείο βάσει του οποίου παράγεται το *API* και το οποίο περιγράφεται στην αμέσως επόμενη ενότητα. Το τμήμα αυτό του κώδικα δεν παράγεται αυτόματα, ωστόσο, κάθε φορά δημιουργεί διαφορετικά *endpoints* προσαρμοσμένα στο μοντέλο που σχεδιάζει κάθε φορά ο χρήστης. Καθώς λοιπόν, είναι απαραίτητο να ανακτηθούν τα δεδομένα από τη βάση προκειμένου να γίνουν διαθέσιμα στο χρήστη. μέσω του *API*, αποφασίστηκε το συγκεκριμένο τμήμα κώδικα ανάκτησης των δεδομένων να ενοποιηθεί στο συγκεκριμένο αρχείο. Το αρχείο αυτό περιλαμβάνει τις ακόλουθες συναρτήσεις:

retrieve_sensors_parameters()

Περιγραφή

Επιστρέφει όλα τα χαρακτηριστικά όλων των αισθητήρων του συστήματος.

Παράμετροι εισόδου

Δε δέχεται ορίσματα

retrieve_categories()

Περιγραφή

Επιστρέφει τις κατηγορίες των αισθητήρων του συστήματος.

Παράμετροι εισόδου

Δε δέχεται ορίσματα

retrieve_sensor_infor(key)

Περιγραφή

Επιστρέφει τις τεχνικές πληροφορίες ενός συγκεκριμένου αισθητήρα.

Παράμετροι εισόδου

key: *String* που περιέχει το όνομα του αισθητήρα, βάσει του οποίου θα γίνει η αναζήτηση του στη βάση

retrieve_last_measurement(key)

Περιγραφή

Επιστρέφει την τελευταία μέτρηση ενός συγκεκριμένου αισθητήρα.

Παράμετροι εισόδου

key: *String* που περιέχει το όνομα του αισθητήρα, βάσει του οποίου θα γίνει η αναζήτηση των μετρήσεων του στη βάση

retrieve_all_measurements(key)

Περιγραφή

Επιστρέφει όλες τις μετρήσεις ενός συγκεκριμένου αισθητήρα.

Παράμετροι εισόδου

key: *String* που περιέχει το όνομα του αισθητήρα, βάσει του οποίου θα γίνει η αναζήτηση των μετρήσεων του στη βάση

retrieve_last_measurements(key, number)

Περιγραφή

Επιστρέφει τις τελευταίες μετρήσεις ενός συγκεκριμένου αισθητήρα. Ο αριθμός των μετρήσεων που θα επιστραφούν ορίζεται από την τιμή *number*.

Παράμετροι εισόδου

key: *String* που περιέχει το όνομα του αισθητήρα, βάσει του οποίου θα γίνει η αναζήτηση των μετρήσεων του στη βάση

number: Ο αριθμός που αντιστοιχεί στο πλήθος των μετρήσεων που θα επιστραφούν

4.6 Flask REST API

Στην παράγραφο αυτή παρουσιάζεται εκτενώς η διεπαφή υψηλού επιπέδου που παράγεται από το σύστημα. Θα παρουσιαστούν όλες οι επιτρεπτές κλήσεις και τα endpoints που μπορούν να δημιουργηθούν. Σκοπός της παραγωγής της διεπαφής είναι να μπορεί ο χρήστης αφού, δημιουργεί το σύστημα των αισθητήρων που επιθυμεί, να ανακτήσει πληροφορίες σχετικά με αυτό ανά πάσα στιγμή από οπουδήποτε. Κατά τη σχεδίαση του API, γεννήθηκε η ιδέα να υλοποιηθεί με τέτοιο τρόπο, ώστε ο χρήστης να έχει άμεση επέμβαση και σε αυτό κατά το μέγιστο δυνατό. Για το λόγο αυτό, πολλά endpoints δεν έχουν ένα σταθερό συγκεκριμένο όνομα, αλλά αυτό τροποποιείται, ανάλογα με τα ονόματα των αισθητήρων που θέτει ο χρήστης. Πρόκειται συνεπώς για την παραγωγή ενός εξατομικευμένου API, πιο προσωπικού και πιο κοντά στις επιθυμίες του χρήστη. Με αυτόν τον τρόπο, ο χρήστης σχεδιάζει ένα API το οποίο έχει άμεση επιρροή και το οποίο θα χρησιμοποιεί πιθανώς πιο εύκολα, καθώς πλέον δε χρειάζεται να απομνημονεύει μακροσκελή και περίπλοκα ονόματα endpoints. Αντιθέτως, προσαρμόζει τα ονόματα σύμφωνα με τη δική του λογική κι επιθυμία και δημιουργεί ένα πρωτότυπο και μοναδικό API.

Σημαντικό είναι να μπορέσει ο αναγνώστης να κατανοήσει σε βάθος τον τρόπο αυτής της διαδικασίας αυτόματης παραγωγής της διεπαφής. Για το λόγο αυτό περιλαμβάνονται περιγραφές και σχόλια σε όποιο σημείο θεωρείται

αξιοσημείωτο. Όλος ο κώδικας που αφορά το API βρίσκεται στο αρχείο *genAPI.py*, το οποίο είναι διαθέσιμο στο github, στο σύνδεσμο: <https://github.com/dimitrantz/Thesis/tree/master/web-app/public>.

Για να γίνει πιο εύκολο στον ανάγνωση, η συγκεκριμένη παράγραφος χωρίζεται σε δύο ενότητες. Στην πρώτη θα παρουσιαστεί το τμήμα του παραγόμενου API που αφορά γενικές πληροφορίες του συστήματος, ενώ στην επόμενη αυτό που αφορά μεμονωμένους αισθητήρες.

4.6.1 Endpoints γενικού περιεχομένου του συστήματος

Πίνακας 68: Αίτημα GET για την ανάκτηση της λίστας categories των αισθητήρων.

Request Method:	GET
Request URL:	http://localhost:3001/categories
Content-Type:	application/JSON
Status Code:	200: Οι κατηγορίες των αισθητήρων
Response Body: (Status 200)	{ "categories": [{ "category": "string" }] }

Ο παραπάνω πίνακας περιγράφει τον τρόπο που γίνεται η ανάκτηση των κατηγοριών του συστήματος μέσω της παραγόμενης διεπαφής.

Πίνακας 69: Αίτημα GET για την ανάκτηση της λίστας χαρακτηριστικών όλων των αισθητήρων.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters
Content-Type:	application/JSON
Status Code:	200: Τα χαρακτηριστικά όλων των αισθητήρων
Response Body: (Status 200)	{ "sensors_parameters": [{}] }

Κατά τη δημιουργία του συστήματος, στον εκτελέσιμο κώδικα, δημιουργείται ένας πίνακας που περιλαμβάνει όλα τα χαρακτηριστικά του κάθε αισθητήρα που έχει ορίσει ο χρήστης. Η παραπάνω κλήση, έχει ως στόχο την ανάκτηση αυτών των πληροφοριών, η οποία θα επιστραφεί σε μορφή json όπως ορίζεται προηγουμένως.

Πίνακας 70: Αίτημα GET για την ανάκτηση της λίστας ονομάτων όλων των αισθητήρων.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/names
Content-Type:	application/JSON
Status Code:	200: Τα ονόματα όλων των αισθητήρων
Response Body: (Status 200)	{ "sensors_names": ["string"] }

Η κλήση του Πίνακα 70 εξυπηρετεί το σκοπό που ο χρήστης επιθυμεί να γνωρίζει τα ονόματα των αισθητήρων, όπως τα έχει ορίσει στο σύστημα του. Το αποτέλεσμα που θα επιστραφεί είναι η λίστα των ονομάτων σε αλφαριθμητική σειρά.

Πίνακας 71: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Acoustic.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/category/ACOUSTIC
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες των αισθητήρων που έχει η κατηγορία Acoustic
Response Body: (Status 200)	{ "category_sensors": [{}] }

Όταν ο χρήστης έχει συμπεριλάβει στο σύστημα του αισθητήρες της κατηγορίας ήχου, μέσω της κλήσης αυτής να ενημερώνεται για όλους εκείνους τους αισθητήρες αυτής της κατηγορίας. Με άλλα λόγια, η κλήση του Πίνακα 71 εξυπηρετεί και την περίπτωση που ο χρήστης επιθυμεί να κάνει αναζήτηση αισθητήρων βάσει κατηγορίας ήχου.

Πίνακας 72: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Distance.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/category/DISTANCE
Content-Type:	application/JSON

Status Code:	200: Οι ιδιότητες των αισθητήρων που έχει η κατηγορία Distance.
Response Body: (Status 200)	{ "category_sensors": [{}] }

Ομοίως, σε περίπτωση που ο χρήστης έχει συμπεριλάβει στο σύστημα του αισθητήρες της κατηγορίας απόστασης, μπορεί μέσω της κλήσης αυτής να ενημερωθεί για όλους εκείνους τους αισθητήρες αυτής της κατηγορίας, εξυπηρετώντας και την αναζήτηση αισθητήρων βάσει κατηγορίας απόστασης.

Πίνακας 73: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Electric.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/category/ELECTRIC
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες των αισθητήρων που έχει η κατηγορία Electric.
Response Body: (Status 200)	{ "category_sensors": [{}] }

Ομοίως, όταν ο χρήστης συμπεριλαμβάνει στο σύστημα του αισθητήρες της κατηγορίας ηλεκτρισμού, μέσω της κλήσης αυτής μπορεί να ενημερώνεται για όλους εκείνους τους αισθητήρες αυτής της κατηγορίας, εξυπηρετώντας και την αναζήτηση αισθητήρων βάσει κατηγορίας αισθητήρων βάσει κατηγορίας ηλεκτρισμού.

Ακολουθώντας την παραπάνω λογική, όταν το σύστημα περιέχει αισθητήρες της κατηγορίας πίεσης, μπορεί μέσω της ακόλουθης κλήσης να ανακτηθούν τα χαρακτηριστικά αυτών, εξυπηρετώντας και την αναζήτηση αισθητήρων βάσει αυτής κατηγορίας.

Πίνακας 74: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Pressure.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/category/PRESSURE
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες των αισθητήρων που έχει η κατηγορία Pressure.
Response Body: (Status 200)	{ "category_sensors": [{}] }

Συνεχίζοντας, όταν το σύστημα περιέχει αισθητήρες της κατηγορίας θέσης, μπορεί μέσω της ακόλουθης κλήσης, στον *Πίνακα 75*, να ενημερώνεται για όλους εκείνους τους αισθητήρες αυτής της κατηγορίας, εξυπηρετώντας και την περίπτωση που ο χρήστης επιθυμεί να κάνει αναζήτηση αισθητήρων βάσει κατηγορίας θέσης.

Πίνακας 75: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Position.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/category/POSITION
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες των αισθητήρων που έχει η κατηγορία Position.
Response Body: (Status 200)	{ "category_sensors": [{}] }

Πίνακας 76: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Speed.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/category/SPEED
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες των αισθητήρων που έχει η κατηγορία Speed.
Response Body: (Status 200)	{ "category_sensors": [{}] }

Ομοίως, όταν το σύστημα περιλαμβάνει αισθητήρες της κατηγορίας ταχύτητας, μπορεί μέσω της κλήσης που αναφέρεται στον *Πίνακα 76* να ανακτηθούν οι πληροφορίες των αισθητήρων αυτών. Η παραπάνω κλήση εξυπηρετεί και την περίπτωση που ο χρήστης επιθυμεί να κάνει αναζήτηση αισθητήρων βάσει κατηγορίας ταχύτητας.

Πίνακας 77: Αίτημα GET για την ανάκτηση των χαρακτηριστικών των αισθητήρων της κατηγορίας Vision.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/category/VISION
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες των αισθητήρων που έχει η κατηγορία Vision.

Response Body: (Status 200)	<pre>{ "category_sensors": [{}] }</pre>
------------------------------------	---

Αντίστοιχα, όταν στο σύστημα συμπεριλαμβάνονται αισθητήρες της κατηγορίας όρασης, μπορεί μέσω της κλήσης αυτής να ανακτηθούν τα χαρακτηριστικά των αισθητήρων αυτών. Η παραπάνω κλήση εξυπηρετεί και την περίπτωση που ο χρήστης επιθυμεί να κάνει αναζήτηση αισθητήρων βάσει κατηγορίας όρασης.

4.6.2 Endpoints μεμονωμένων αισθητήρων

Πίνακας 78: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Distance.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/nameDistanceSensor*
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες του αισθητήρα τύπου Distance.
Response Body: (Status 200)	<pre>{ "sensor_parameters": { "category": "string", "sensor": "string", "index": "string", "frequency": "string", "alarmValue": "string", "minDistanceAccepted": "string", "maxDistanceAccepted": "string" } }</pre>

Για κάθε αισθητήρα τύπου απόστασης, δημιουργείται ένα endpoint το οποίο επιστρέφει τα χαρακτηριστικά του, σε μορφή json, όπως παρουσιάζεται στον Πίνακα 78. Στη συνέχεια παρουσιάζονται ορισμένα επιπρόσθετα χαρακτηριστικά των αισθητήρων τύπου απόστασης

category:	κατηγορία στην οποία ανήκει ο αισθητήρας
sensor:	όνομα του αισθητήρα, όπως ορίστηκε στο μοντέλο
index:	id του αισθητήρα, όπως ορίζεται στην R4A
frequency:	το refresh rate που όρισε ο χρήστης για τον αισθητήρα
minDistanceAccepted:	κατώτατο όριο του εύρους αποδεκτών τιμών
maxDistanceAccepted:	ανώτερο όριο του εύρους αποδεκτών τιμών
alarmValue:	όριο απόστασης βάσει του οποίου ο χρήστης θα ενημερώνεται

Πίνακας 79: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Acoustic.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/ <i>nameAcousticSystem</i> * * <i>nameAcousticSystem</i> : όνομα που δίνει ο χρήστης σε ένα υποσύστημα Acoustic
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες του αισθητήρα τύπου Acoustic.
Response Body: (Status 200)	{ "sensor_parameters": { "category": "string", "sensor": "string", "index": "string", "frequency": "string", "recordTime": "string" } }

Για κάθε υποσύστημα τύπου ήχου, δημιουργείται ένα endpoint το οποίο επιστρέφει τα χαρακτηριστικά του, σε μορφή json, όπως παρουσιάζεται στον Πίνακα 79. Τα παραπάνω χαρακτηριστικά επεξηγούνται στην αρχή της παραγράφου 4.6.2. Παρακάτω παρουσιάζεται ένα επιπλέον χαρακτηριστικό του στοιχείου αυτού:

recordTime:	χρόνος ηχογράφησης του υποσυστήματος
-------------	--------------------------------------

Πίνακας 80: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Electric.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/ <i>nameElectricSensor</i> * * <i>nameElectricSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Electric
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες του αισθητήρα τύπου Electric.
Response Body: (Status 200)	{ "sensor_parameters": { "category": "string", "sensor": "string", "index": "string", "frequency": "string", "alarmValue": "string" } }

Για κάθε αισθητήρα τύπου ηλεκτρισμού, δημιουργείται ένα *endpoint* το οποίο επιστρέφει τα χαρακτηριστικά του, σε μορφή *json*, όπως παρουσιάζεται στον *Πίνακα 80*. Τα παραπάνω χαρακτηριστικά επεξηγούνται στην αρχή της παραγράφου 4.6.2. Στην προκειμένη περίπτωση, η ιδιότητα *alarmValue*, ορίζεται ως εξής:

alarmValue: όριο (%) βάσει του οποίου ο χρήστης θα ενημερώνεται, πχ, επίπεδα μπαταρίας

Πίνακας 81: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Position.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/ <i>namePositionSensor</i> * * <i>namePositionSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες του αισθητήρα τύπου Position.
Response Body: (Status 200)	{ "sensor_parameters": { "category": "string", "sensor": "string", "index": "string", "frequency": "string" } }

Για κάθε αισθητήρα τύπου θέσης, δημιουργείται ένα *endpoint* το οποίο επιστρέφει τα χαρακτηριστικά του, σε μορφή *json*, όπως παρουσιάζεται στον *Πίνακα 81*. Συνεχίζοντας, Για κάθε αισθητήρα τύπου θέσης και συγκεκριμένα, όσον αφορά τη στάση σώματος του ρομπότ, δημιουργείται ένα *endpoint* το οποίο επιστρέφει τα χαρακτηριστικά του, σε μορφή *json*, όπως παρουσιάζεται παραπάνω (πίν.82). Τα χαρακτηριστικά που περιέχουν οι δύο αυτές κλήσεις επεξηγούνται στην αρχή της παραγράφου 4.6.2.

Πίνακας 82: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Position - Posture.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/ <i>namePositionPostureSensor</i> * * <i>namePositionPostureSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position Posture
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες του αισθητήρα τύπου Position Posture.

Response Body: (Status 200)	<pre>{ "sensor_parameters": { "category": "string", "sensor": "string", "index": "string", "frequency": "string" } }</pre>
------------------------------------	--

Πίνακας 83: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Position - Tframe.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/ <i>namePositionTframeSensor</i> *
	* <i>namePositionTframeSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position Tframe
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες του αισθητήρα τύπου Position Tframe.
Response Body: (Status 200)	<pre>{ "sensor_parameters": { "category": "string", "sensor": "string", "index": "string", "frequency": "string", "origin": "string", "target": "string" } }</pre>

Για κάθε αισθητήρα τύπου θέσης και συγκεκριμένα, όσον αφορά το μετασχηματισμό από ένα tframe σε ένα άλλο, δημιουργείται ένα endpoint το οποίο επιστρέφει τα χαρακτηριστικά του, σε μορφή json, όπως παρουσιάζεται στον Πίνακα 83. Τα παραπάνω χαρακτηριστικά επεξηγούνται στην αρχή της παραγράφου 4.6.2, ενώ προστίθενται δύο ακόμη:

origin:	tframe του ρομπότ που θα μετασχηματιστεί στο target
target	tframe στο οποίο θα μετασχηματιστεί το origin

Ομοίως, παρουσιάζεται η κλήση που αφορά την κατηγορία θέσης και αφορά την τοποθέτηση του ρομπότ στο χώρο. Εάν συμπεριληφθεί ο συγκεκριμένος αισθητήρας, τότε δημιουργείται αυτόματα το αντίστοιχο endpoint, που επιστρέφει τα χαρακτηριστικά του αισθητήρα αυτού. Τα δεδομένα μετατρέπονται σε json μορφή, όπως εμφανίζονται στον Πίνακα 84. Τα χαρακτηριστικά επεξηγούνται στην αρχή της παραγράφου 4.6.2

Πίνακας 84: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Position - Robot.

Request Method:	GET
Request URL:	<code>http://localhost:3001/sensors_parameters/namePositionRobotSensor*</code> * <code>namePositionRobotSensor</code> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position Robot
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες του αισθητήρα τύπου Poision Robot.
Response Body: (Status 200)	<pre>{ "sensor_parameters": { "category": "string", "sensor": "string", "index": "string", "frequency": "string" } }</pre>

Πίνακας 85: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Pressure.

Request Method:	GET
Request URL:	<code>http://localhost:3001/sensors_parameters/namePressureSensor*</code> * <code>namePressureSensor</code> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Pressure
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες του αισθητήρα τύπου Pressure.
Response Body: (Status 200)	<pre>{ "sensor_parameters": { "category": "string", "sensor": "string", "index": "string", "frequency": "string" } }</pre>

Για κάθε αισθητήρα τύπου πίεσης, παράγεται ένα endpoint το οποίο επιστρέφει τα χαρακτηριστικά του, σε μορφή json, όπως παρουσιάζεται στον Πίνακα 85, τα οποία επεξηγούνται στην αρχή της παραγράφου 4.6.2.

Πίνακας 86: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Speed.

Request Method:	GET
Request URL:	<code>http://localhost:3001/sensors_parameters/nameSpeedSensor*</code> * <code>nameSpeedSensor</code> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Speed

Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες του αισθητήρα τύπου Speed.
Response Body: (Status 200)	<pre>{ "sensor_parameters": { "category": "string", "sensor": "string", "index": "string", "frequency": "string" } }</pre>

Αντίστοιχα, για κάθε αισθητήρα τύπου ταχύτητας, παράγεται ένα *endpoint* το οποίο επιστρέφει τα χαρακτηριστικά του, σε μορφή *json*, όπως παρουσιάζεται στον *Πίνακα 86*. Τα παραπάνω χαρακτηριστικά επεξηγούνται στην αρχή της παραγράφου 4.6.2.

Πίνακας 87: Αίτημα GET για την ανάκτηση των χαρακτηριστικών ενός αισθητήρα τύπου Vision.

Request Method:	GET
Request URL:	http://localhost:3001/sensors_parameters/ <i>nameVisionSensor</i> *
	* <i>nameVisionSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Vision
Content-Type:	application/JSON
Status Code:	200: Οι ιδιότητες του αισθητήρα τύπου Vision.
Response Body: (Status 200)	<pre>{ "sensor_parameters": { "category": "string", "sensor": "string", "index": "string", "frequency": "string", "resolution": "string" } }</pre>

Αντίστοιχα, για κάθε αισθητήρα τύπου πίεσης, παράγεται ένα *endpoint* το οποίο επιστρέφει τα χαρακτηριστικά του, σε μορφή *json*, όπως παρουσιάζεται στον *Πίνακα 87*. Τα παραπάνω χαρακτηριστικά επεξηγούνται στην αρχή της παραγράφου 4.6.2. Στη συνέχεια παρουσιάζεται ένα επιπλέον χαρακτηριστικό του αισθητήρα όρασης:

resolution: η ανάλυση της φωτογραφίας

Στη συνέχεια παρουσιάζονται όλες οι κλίσεις που αφορούν την ανάκτηση των τεχνικών χαρακτηριστικών των αισθητήρων, όπως αυτά έχουν οριστεί στην πλατφόρμα R4A.

Αρχικά, πρόκειται για την κλήση, *Πίνακας 88*, που έχει ως σκοπό την επιστροφή πληροφοριών σχετικά με το ρομπότ, μία σύντομη περιγραφή αυτού και του λειτουργικού του συστήματος, καθώς και τον τύπο του. Επιπλέον, η κλήση προσδιορίζεται μοναδικά από ένα *timestamp*.

Πίνακας 88: Αίτημα GET για την ανάκτηση των ορισμένων χαρακτηριστικών του ρομπότ.

Request Method:	GET
Request URL:	http://localhost:3001/info/Robot
Content-Type:	application/JSON
Status Code:	200: Χαρακτηριστικά του ρομπότ.
Response Body: (Status 200)	<pre>{ "info": { "information": { "description": "string", "os": "string", "rtype": "string" }, "timestamp": 0 } }</pre>

Η κλήση που ακολουθεί αφορά την επιστροφή των τεχνικών χαρακτηριστικών ενός αισθητήρα απόστασης. Πιο συγκεκριμένα, περιγράφονται ορισμένα στοιχεία όπως το field of view, οι γωνίες που είναι τοποθετημένος ο αισθητήρας στο ρομπότ, η ελάχιστη και μέγιστη απόσταση που μπορεί να αναγνωρίζει το ρομπότ, η ανάλυση του αισθητήρα, καθώς και ορισμένες τεχνικές πληροφορίες.

Η κλήση αυτή, περιέχει ακόμη το χαρακτηριστικό *timestamp* το οποίο αναφέρεται στη χρονική στιγμή της κλήσης.

Πίνακας 89: Αίτημα GET για την ανάκτηση των τεχνικών χαρακτηριστικών ενός αισθητήρα τύπου Distance.

Request Method:	GET
Request URL:	http://localhost:3001/info/nameDistanceSensor* * nameDistanceSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Distance
Content-Type:	application/JSON
Status Code:	200: Τα τεχνικά χαρακτηριστικά του αισθητήρα τύπου Distance.
Response Body: (Status 200)	<pre>{ "info": { "angles": [0], "fov": { "horizontal": 0, "vertical": 0 } } }</pre>

	<pre> "information": { "brand": "string", "description": "string", "url": "string" }, "max_distance": 0, "max_hz": 0, "min_distance": 0, "resolution": 0, "timestamp": 0 } } </pre>
--	---

Ακολουθεί η κλήση αναφορικά με τον αισθητήρα ήχου. Πιο αναλυτικά, η κλήση αυτή έχει ως σκοπό να επιστρέψει τις πληροφορίες που έχουν οριστεί στην πλατφόρμα για το συγκεκριμένο αισθητήρα. Περιλαμβάνει πληροφορίες για τη μάρκα του αισθητήρα και μία σύντομη περιγραφή του. Περιλαμβάνει ακόμη ένα timestamp, το οποίο δηλώνει την ώρα της κλήσης.

Πίνακας 90: Αίτημα GET για την ανάκτηση των τεχνικών χαρακτηριστικών ενός αισθητήρα τύπου Acoustic.

Request Method:	GET
Request URL:	http://localhost:3001/info/nameAcousticSystem* * nameAcousticSystem: όνομα που δίνει ο χρήστης σε ένα υποσύστημα Acoustic
Content-Type:	application/JSON
Status Code:	200: Τα τεχνικά χαρακτηριστικά του αισθητήρα τύπου Acoustic.
Response Body: (Status 200)	<pre> { "info": { "information": { "brand": "string", "description": "string", "url": "string" }, "timestamp": 0 } } </pre>

Μία παρόμοια κλήση περιγράφεται για την περίπτωση του αισθητήρα ηλεκτρισμού. Η κλήση αυτή επιστρέφει σε μορφή json, ορισμένες πληροφορίες αναφορικά με τα τεχνικά χαρακτηριστικά του αισθητήρα, όπως αυτά ορίζονται στην πλατφόρμα R4A. Η κλήση περιέχει ακόμη ένα timestamp, υποδηλώνοντας τη χρονική στιγμή που εκτελείται.

Πίνακας 91: Αίτημα GET για την ανάκτηση των τεχνικών χαρακτηριστικών ενός αισθητήρα τύπου Electric.

Request Method:	GET
Request URL:	http://localhost:3001/info/nameElectricSensor*
	* nameElectricSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Electric
Content-Type:	application/JSON
Status Code:	200: Τα τεχνικά χαρακτηριστικά του αισθητήρα τύπου Electric.
Response Body: (Status 200)	<pre>{ "info": { "information": { "brand": "string", "description": "string", "url": "string" }, "timestamp": 0 } }</pre>

Η κλήση που ακολουθεί αφορά την επιστροφή πληροφοριών για τον αισθητήρα θέσης. Όπως και οι προηγούμενες κλήσεις αυτού του σκοπού, περιέχει τεχνικές πληροφορίες για τον αισθητήρα και ένα timestamp μέσω του οποίου δηλώνεται η στιγμή της εκτέλεσης αυτής της κλήσης.

Πίνακας 92: Αίτημα GET για την ανάκτηση των τεχνικών χαρακτηριστικών ενός αισθητήρα τύπου Position.

Request Method:	GET
Request URL:	http://localhost:3001/info/namePositionSensor*
	* namePositionSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position
Content-Type:	application/JSON
Status Code:	200: Τα τεχνικά χαρακτηριστικά του αισθητήρα τύπου Position.
Response Body: (Status 200)	<pre>{ "info": { "information": { "brand": "string", "description": "string", "url": "string" }, "timestamp": 0 } }</pre>

Η κλήση που ακολουθεί αφορά την επιστροφή πληροφοριών για τον αισθητήρα πίεσης. Όπως έχει ήδη αναφερθεί, οι κλήσεις αυτού του σκοπού, περιέχουν τεχνικές πληροφορίες για τον αισθητήρα και ένα *timestamp* μέσω του οποίου δηλώνεται η στιγμή της εκτέλεσης αυτής της κλήσης.

Πίνακας 93: Αίτημα GET για την ανάκτηση των τεχνικών χαρακτηριστικών ενός αισθητήρα τύπου Pressure.

Request Method:	GET
Request URL:	http://localhost:3001/info/namePressureSensor* * namePressureSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Pressure
Content-Type:	application/JSON
Status Code:	200: Τα τεχνικά χαρακτηριστικά του αισθητήρα τύπου Pressure.
Response Body: (Status 200)	{ "info": { "information": { "brand": "string", "description": "string", "url": "string" }, "timestamp": 0 } }

Αυτή η κατηγορίας κλήσεων ολοκληρώνεται με την κλήση που αφορά τον αισθητήρα όρασης. Αφορά την επιστροφή των τεχνικών χαρακτηριστικών ενός αισθητήρα όρασης. Πιο αναλυτικά, περιγράφεται το *field of view* του αισθητήρα, το εύρος της, το ύψος στο οποίο βρίσκεται και αν είναι κάμερα τύπου *depth* ή όχι. Ακόμη, παρουσιάζονται ορισμένες πληροφορίες για τη μάρκα της κάμερα, αλλά και το μέγιστο όριο του *refresh rate* του αισθητήρα. Τέλος,, όπως κάθε άλλη κλήση της κατηγορίας, περιλαμβάνει ένα χαρακτηριστικό *timestamp* που δηλώνει τη χρονική στιγμή της κλήσης.

Πίνακας 94: Αίτημα GET για την ανάκτηση των τεχνικών χαρακτηριστικών ενός αισθητήρα τύπου Vision.

Request Method:	GET
Request URL:	http://localhost:3001/info/nameVisionSensor* * nameVisionSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Vision
Content-Type:	application/JSON
Status Code:	200: Τα τεχνικά χαρακτηριστικά του αισθητήρα τύπου Vision.

Response Body: (Status 200)	<pre>{ "info": { { "fov": { "horizontal": 0, "vertical": 0 }, "height": 0, "information": { "brand": "string", "description": "string", "url": "string" }, "is_depth": 0, "max_hz": 0, "timestamp": 0, "width": 0 } } }</pre>
------------------------------------	---

Στη συνέχεια παρουσιάζονται όλες οι κλήσεις που έχουν υλοποιηθεί για την επιστροφή των μετρήσεων των αισθητήρων, όλων των τύπων και των κατηγοριών. Αρχικά, στον πίνακα 95, παρουσιάζεται η κλήση αναφορικά με την επιστροφή όλων των μετρήσεων ενός αισθητήρα απόστασης. Η κλήση αυτή, επιστρέφει σε μορφή *json*, έναν πίνακα από μία δομή που έχει ως στοιχεία τα εξής:

data:	δεδομένα της μέτρησης σε μορφή αριθμού
timestamp:	χρονική στιγμή της μέτρησης

Πίνακας 95: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Distance.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/all/nameDistanceSensor* <i>* nameDistanceSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Distance</i>
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Distance.
Response Body: (Status 200)	<pre>{ "measurements": [{ "data": 0, "timestamp": 0 }] }</pre>

Στη συνέχεια, παρουσιάζεται η κλήση που αφορά το υποσύστημα ήχου (πιν. 96) και επιστρέφει όλες τις μετρήσεις αυτού του αισθητήρα. Η κλήση αυτή, επιστρέφει σε μορφή *json*, έναν πίνακα από μία δομή που έχει ως στοιχεία τα εξής:

data:	όνομα του αρχείου ήχου που δημιουργήθηκε
timestamp:	χρονική στιγμή της μέτρησης

Πίνακας 96: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Acoustic.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/all/ <i>nameAcousticSystem</i> *
	* <i>nameAcousticSystem</i> : όνομα που δίνει ο χρήστης σε ένα υποσύστημα Acoustic
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Acoustic.
Response Body: (Status 200)	{ "measurements": [{ "data": "string", "timestamp": 0 }]} }

Ακολουθεί η κλήση που αφορά έναν αισθητήρα ηλεκτρισμού (πιν. 97) και επιστρέφει όλες τις μετρήσεις αυτού του αισθητήρα. Η κλήση αυτή, επιστρέφει σε μορφή json, έναν πίνακα από μία δομή που έχει ως στοιχεία τα εξής:

data:	επίπεδα μπαταρίας (%)
timestamp:	χρονική στιγμή της μέτρησης

Πίνακας 97: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Electric.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/all/ <i>nameElectricSensor</i> * <i>nameElectricSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Electric
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Electric.
Response Body: (Status 200)	{ "measurements": [{ "data": 0, "timestamp": 0 }]} }

Στη συνέχεια παρουσιάζεται η κλήση αναφορικά με τον αισθητήρα θέσης (πιν. 98) που επιστρέφει όλες τις μετρήσεις αυτού του αισθητήρα. Η κλήση αυτή, επιστρέφει σε μορφή *json*, έναν πίνακα από μία δομή που έχει ως στοιχεία τα εξής:

data:	"pose": { "pitch": 0, "roll": 0, "x": 0, "y": 0, "yaw": 0, "z": 0 }
timestamp:	χρονική στιγμή της μέτρησης

Πίνακας 98: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Position.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/all/ <i>namePositionSensor</i> * * <i>namePositionSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Position.
Response Body: (Status 200)	{ "measurements": [{ "data": {}, "timestamp": 0 }] }

Στη συνέχεια παρουσιάζεται η κλήση αναφορικά με τον αισθητήρα θέσης (πιν. 99) που επιστρέφει όλες τις μετρήσεις αυτού του αισθητήρα. Η κλήση αυτή, επιστρέφει σε μορφή *json*, έναν πίνακα από μία δομή που έχει ως στοιχεία τα εξής:

data:	στάση σώματος του ρομπότ
timestamp:	χρονική στιγμή της μέτρησης

Πίνακας 99: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Position - Posture.

Request Method:	GET
Request URL:	<code>http://localhost:3001/measurements/all/namePositionPostureSensor*</code> * <code>namePositionPostureSensor</code> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position Posture
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Position Posture.
Response Body: (Status 200)	{ "measurements": ["data": "string", "timestamp": 0] }

Ακολουθεί η κλήση αναφορικά με τον αισθητήρα μετασχηματισμού από ένα tframe σε ένα άλλο (πιν. 100) που επιστρέφει όλες τις μετρήσεις αυτού του αισθητήρα. Έπειτα, γίνεται αναφορά στην κλήση του αισθητήρα θέσης και συγκεκριμένα στην τοποθέτηση του ρομπότ στο χώρο (πιν. 101) και η οποία επιστρέφει όλες τις μετρήσεις του αισθητήρα. Και οι δύο αυτές κλήσεις, επιστρέφουν σε μορφή json, έναν πίνακα από μία δομή που έχει ως στοιχεία τα εξής:

```
data:      "pose": {  
    "pitch": 0,  
    "roll": 0,  
    "x": 0,  
    "y": 0,  
    "yaw": 0,  
    "z": 0  
}  
timestamp: χρονική στιγμή της μέτρησης
```

Πίνακας 100: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Position - Tframe.

Request Method:	GET
Request URL:	<code>http://localhost:3001/measurements/all/namePositionTframeSensor*</code> * <code>namePositionTframeSensor</code> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position Tframe
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Position Tframe.

Response Body: (Status 200)	<pre>{ "measurements": [{ "data": {}, "timestamp": 0 }] }</pre>
------------------------------------	---

Πίνακας 101: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Position - Robot.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/all/namePositionRobotSensor*
	* namePositionRobotSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position Robot
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Poisition Robot.
Response Body: (Status 200)	<pre>{ "measurements": [{ "data": {}, "timestamp": 0 }] }</pre>

Στη συνέχεια παρουσιάζεται η κλήση αναφορικά με τον αισθητήρα πίεσης (πιν. 102) που επιστρέφει όλες τις μετρήσεις αυτού του αισθητήρα. Η κλήση αυτή, επιστρέφει σε μορφή json, έναν πίνακα από μία δομή που έχει ως στοιχεία τα εξής:

data:	0 εάν δεν ανιχνεύεται πάτημα ή αφή, 1 σε αντίθετη περίπτωση
timestamp:	χρονική στιγμή της μέτρησης

Πίνακας 102: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Pressure.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/all/namePressureSensor*
	* namePressureSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Pressure
Content-Type:	application/JSON

Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Pressure.
Response Body: (Status 200)	<pre>{ "measurements": [{ "data": 0, "timestamp": 0 }] }</pre>

Έπειτα, γίνεται αναφορά στην κλήση του αισθητήρα ταχύτητας (πιν. 103), η οποία επιστρέφει όλες τις μετρήσεις του αισθητήρα, δηλαδή τη γραμμική και γωνιακή ταχύτητα στους 3 άξονες. Η κλήση επιστρέφει σε μορφή *json*, έναν πίνακα από μία δομή που έχει ως στοιχεία τα εξής:

data:	<pre>"twist": { "ang_x": 0, "ang_y": 0, "ang_z": 0, "lin_x": 0, "lin_y": 0, "lin_z": 0 }</pre>
timestamp:	χρονική στιγμή της μέτρησης

Πίνακας 103: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Speed.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/all/ <i>nameSpeedSensor</i> * <small>* <i>nameSpeedSensor</i>: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Speed</small>
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Speed.
Response Body: (Status 200)	<pre>{ "measurements": [{ "data": {}, "timestamp": 0 }] }</pre>

Η κατηγορία αυτή των κλήσεων, ολοκληρώνεται με την παρουσίαση της κλήσης που αφορά τον αισθητήρα ταχύτητας(πιν. 104) και επιστρέφει όλες τις μετρήσεις αυτού του αισθητήρα. Η κλήση αυτή, επιστρέφει σε μορφή *json*, έναν πίνακα από μία δομή που έχει ως στοιχεία τα εξής:

data:	όνομα του αρχείου ήχου που δημιουργήθηκε
timestamp:	χρονική στιγμή της μέτρησης

Πίνακας 104: Αίτημα GET για την ανάκτηση όλων των μετρήσεων ενός αισθητήρα τύπου Vision.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/all/ <i>nameVisionSensor</i> * <small>* <i>nameVisionSensor</i>: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Vision</small>
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Vision.
Response Body: (Status 200)	{ "measurements": ["data": "string", "timestamp": 0] }

Στη συνέχεια παρουσιάζονται όλες οι κλήσεις που αφορούν την ανάκτηση της τελευταίας μόνο μέτρησης ενός αισθητήρα οποιουδήποτε τύπου και κατηγορίας. Παρακάτω, περιγράφεται η κλήση ενός αισθητήρα απόστασης(πιν. 105). Τα χαρακτηριστικά παραμένουν ίδια, όπως αναφέρθηκαν προηγουμένως.

Πίνακας 105: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Distance.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/last/ <i>nameDistanceSensor</i> * <small>* <i>nameDistanceSensor</i>: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Distance</small>
Content-Type:	application/JSON
Status Code:	200: Η τελευταία μέτρηση του αισθητήρα τύπου Distance.
Response Body: (Status 200)	{ "measurements": { "data": 0, "timestamp": 0 } }

Στο σημείο αυτό, περιγράφεται η κλήση ενός υποσυστήματος ήχου(πιν. 106). Τα χαρακτηριστικά παραμένουν ίδια, όπως αναφέρθηκαν προηγουμένως.

Πίνακας 106: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Acoustic.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/last/ <i>nameAcousticSystem</i> * * <i>nameAcousticSystem</i> : όνομα που δίνει ο χρήστης σε ένα υποσύστημα Acoustic
Content-Type:	application/JSON
Status Code:	200: Η τελευταία μέτρηση του αισθητήρα τύπου Acoustic.
Response Body: (Status 200)	{ "measurements": { "data": "string", "timestamp": 0 } }

Στη συνέχεια, περιγράφεται η κλήση ενός αισθητήρα ηλεκτρισμού(πιν. 107). Τα χαρακτηριστικά παραμένουν ίδια, όπως αναφέρθηκαν προηγουμένως.

Πίνακας 107: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Electric.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/last/ <i>nameElectricSensor</i> * * <i>nameElectricSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Electric
Content-Type:	application/JSON
Status Code:	200: Η τελευταία μέτρηση του αισθητήρα τύπου Electric.
Response Body: (Status 200)	{ "measurements": { "data": 0, "timestamp": 0 } }

Παρακάτω, περιγράφεται η κλήση ενός αισθητήρα θέσης(πιν. 108). Τα χαρακτηριστικά παραμένουν ίδια, όπως αναφέρθηκαν προηγουμένως.

Πίνακας 108: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Position.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/last/ <i>namePositionSensor</i> * * <i>namePositionSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position
Content-Type:	application/JSON
Status Code:	200: Η τελευταία μέτρηση του αισθητήρα τύπου Position.
Response Body: (Status 200)	{ "measurements": { "data": {}, "timestamp": 0 } }

Ακολουθούν οι κλήσεις που αντιστοιχούν στους αισθητήρες θέσης και πιο συγκεκριμένα στη σάση σώματος του ρομπότ(πιν. 109), στο μετασχηματισμό από ένα tframe σε ένα άλλο (πιν. 110), και στην τοποθέτηση του ρομπότ στο χώρο(πιν. 111). Τα χαρακτηριστικά παραμένουν ίδια, όπως αναφέρθηκαν προηγουμένως.

Πίνακας 109: Αίτημα GET για την ανάκτηση τελευταίας μέτρησης ενός αισθητήρα τύπου PositionPosture.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/last/ <i>namePositionPostureSensor</i> * * <i>namePositionPostureSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position Posture
Content-Type:	application/JSON
Status Code:	200: Η τελευταία μέτρηση του αισθητήρα τύπου Position Posture.
Response Body: (Status 200)	{ "measurements": { "data": "string", "timestamp": 0 } }

Πίνακας 110: Αίτημα GET για την ανάκτηση τελευταίας μέτρησης ενός αισθητήρα τύπου PositionTframe.

Request Method:	GET
Request URL:	<code>http://localhost:3001/measurements/last/namePositionTframeSensor*</code> * <code>namePositionTframeSensor</code> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position Tframe
Content-Type:	application/JSON
Status Code:	200: Η τελευταία μέτρηση του αισθητήρα τύπου Position Tframe.
Response Body: (Status 200)	{ "measurements": { "data": {}, "timestamp": 0 } }

Πίνακας 111: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου PositionRobot.

Request Method:	GET
Request URL:	<code>http://localhost:3001/measurements/last/namePositionRobotSensor*</code> * <code>namePositionRobotSensor</code> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position Robot
Content-Type:	application/JSON
Status Code:	200: Η τελευταία μέτρηση του αισθητήρα τύπου Poisition Robot.
Response Body: (Status 200)	{ "measurements": { "data": {}, "timestamp": 0 } }

Παρακάτω, περιγράφεται η κλήση ενός αισθητήρα πίεσης(πιν. 112) και ακολουθούν οι κλήσεις για επιστροφή της τελευταίας μόνο μέτρησης ενός αισθητήρα, στις περιπτώσεις που ο αισθητήρας είναι ταχύτητας(πιν. 113) ή όρασης(πιν. 114). Τα χαρακτηριστικά σε κάθε περίπτωση επιστρέφονται και πάλι σε μορφή json και παραμένουν ίδια, όπως αναφέρθηκαν προηγουμένως για την αντίστοιχη κατηγορία.

Πίνακας 112: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Pressure.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/last/ <i>namePressureSensor</i> * * <i>namePressureSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Pressure
Content-Type:	application/JSON
Status Code:	200: Η τελευταία μέτρηση του αισθητήρα τύπου Pressure.
Response Body: (Status 200)	{ "measurements": { "data": 0, "timestamp": 0 } }

Πίνακας 113: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Speed.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/last/ <i>nameSpeedSensor</i> * * <i>nameSpeedSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Speed
Content-Type:	application/JSON
Status Code:	200: Η τελευταία μέτρηση του αισθητήρα τύπου Speed.
Response Body: (Status 200)	{ "measurements": { "data": {}, "timestamp": 0 } }

Πίνακας 114: Αίτημα GET για την ανάκτηση της τελευταίας μέτρησης ενός αισθητήρα τύπου Vision.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/last/ <i>nameVisionSensor</i> * * <i>nameVisionSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Vision
Content-Type:	application/JSON
Status Code:	200: Η τελευταία μέτρηση του αισθητήρα τύπου Vision.
Response Body: (Status 200)	{ "measurements": { "data": "string", "timestamp": 0 } }

Στο σημείο αυτό παρουσιάζονται όλες οι κλήσεις μέσω των οποίων ο χρήστης μπορεί να ορίζει τον ακριβή αριθμό των μετρήσεων που επιθυμεί να ανακτήσει. Για το λόγο αυτό, οι κλήσεις που περιγράφονται παρακάτω μπορούν να δημιουργούν πολλά διαφορετικά endpoints, ανάλογα κάθε φορά με την επιλογή του χρήστη.

Τα δεδομένα που επιστρέφει η κάθε κατηγορία παραμένουν ίδια όπως έχουν επεξηγηθεί προηγουμένως σε αυτήν την ενότητα.

Πίνακας 115: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Distance.

Request Method:	GET
Request URL:	<pre>http://localhost:3001/measurements/last/nameDistanceSensor*/number**</pre> <p>* nameDistanceSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Distance ** number: αριθμός μετρήσεων προς ανάκτηση</p>
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Distance.
Response Body: (Status 200)	<pre>{ "measurements": [{ "data": 0, "timestamp": 0 }] }</pre>

Πίνακας 116: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Acoustic.

Request Method:	GET
Request URL:	<pre>http://localhost:3001/measurements/last/nameAcousticSystem*/number**</pre> <p>* nameAcousticSystem: όνομα που δίνει ο χρήστης σε ένα υποσύστημα Acoustic ** number: αριθμός μετρήσεων προς ανάκτηση</p>
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Acoustic.
Response Body: (Status 200)	<pre>{ "measurements": [{ "data": "string", "timestamp": 0 }] }</pre>

Πίνακας 117: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Electric.

Request Method:	GET
Request URL:	<pre>http://localhost:3001/measurements/last/nameElectricSensor*/number**</pre> <p>* nameElectricSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Electric ** number: αριθμός μετρήσεων προς ανάκτηση</p>
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Electric.
Response Body: (Status 200)	<pre>{ "measurements": [{ "data": 0, "timestamp": 0 }] }</pre>

Πίνακας 118: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Position.

Request Method:	GET
Request URL:	<pre>http://localhost:3001/measurements/last/namePositionSensor*/number**</pre> <p>* namePositionSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position ** number: αριθμός μετρήσεων προς ανάκτηση</p>
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Position.
Response Body: (Status 200)	<pre>{ "measurements": [{ "data": {}, "timestamp": 0 }] }</pre>

Πίνακας 119: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου PositionPosture.

Request Method:	GET
Request URL:	<pre>http://localhost:3001/measurements/last/namePositionPostureSensor*/number**</pre> <p>* namePositionPostureSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position Posture ** number: αριθμός μετρήσεων προς ανάκτηση</p>
Content-Type:	application/JSON

Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Position Posture.
Response Body: (Status 200)	<pre>{ "measurements": [{ "data": "string", "timestamp": 0 }] }</pre>

Πίνακας 120: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Position Tframe.

Request Method:	GET
Request URL:	<code>http://localhost:3001/measurements/last/namePositionTframeSensor*/number**</code> * namePositionTframeSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position Tframe ** number: αριθμός μετρήσεων προς ανάκτηση
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Position Tframe.
Response Body: (Status 200)	<pre>{ "measurements": [{ "data": {}, "timestamp": 0 }] }</pre>

Πίνακας 121: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Position Robot.

Request Method:	GET
Request URL:	<code>http://localhost:3001/measurements/last/namePositionRobotSensor*/number**</code> * namePositionRobotSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Position Robot ** number: αριθμός μετρήσεων προς ανάκτηση
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Poision Robot.
Response Body: (Status 200)	<pre>{ "measurements": [{ "data": {}, "timestamp": 0 }] }</pre>

Πίνακας 122: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Pressure.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/last/ <i>namePressureSensor</i> */ <i>number</i> ** * <i>namePressureSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Pressure ** <i>number</i> : αριθμός μετρήσεων προς ανάκτηση
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Pressure.
Response Body: (Status 200)	{ "measurements": [{ "data": 0, "timestamp": 0 }] }

Πίνακας 123: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Speed.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/last/ <i>nameSpeedSensor</i> */ <i>number</i> ** * <i>nameSpeedSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Speed ** <i>number</i> : αριθμός μετρήσεων προς ανάκτηση
Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Speed.
Response Body: (Status 200)	{ "measurements": [{ "data": {}, "timestamp": 0 }] }

Πίνακας 124: Αίτημα GET για την ανάκτηση ορισμένων μετρήσεων ενός αισθητήρα τύπου Vision.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/last/ <i>nameVisionSensor</i> */ <i>number</i> ** * <i>nameVisionSensor</i> : όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Vision ** <i>number</i> : αριθμός μετρήσεων προς ανάκτηση

Content-Type:	application/JSON
Status Code:	200: Όλες οι μετρήσεις του αισθητήρα τύπου Vision.
Response Body: (Status 200)	<pre>{ "measurements": [{ "data": "string", "timestamp": 0 }] }</pre>

Τέλος, έχουν υλοποιηθεί δύο διαφορετικές κλήσεις, μέσω των οποίων είναι δυνατή η ανάκτηση του αρχείου, εικόνας και ήχου. Αφορούν δηλαδή τις περιπτώσεις που το σύστημα περιλαμβάνει αισθητήρες ήχου ή όρασης. Για να μπορεί ο χρήστης να τις καλεί, πρέπει να γνωρίζει το όνομα του αισθητήρα και το όνομα του αρχείου. Είναι εύκολα κατανοητό πως στην περίπτωση των αισθητήρων όρασης, το αποτέλεσμα της κλήσης είναι το αρχείο της εικόνας, ενώ στην περίπτωση των αισθητήρων ήχου, το αρχείο αυτό.

Πίνακας 125: Αίτημα GET για την ανάκτηση αρχείου εικόνας ενός αισθητήρα τύπου Vision.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/one/nameVisionSensor*/filename** * nameVisionSensor: όνομα που δίνει ο χρήστης σε έναν αισθητήρα τύπου Vision ** filename: όνομα αρχείου
Content-Type:	image/png
Status Code:	200: Αρχείο εικόνας.
Response Body: (Status 200)	image

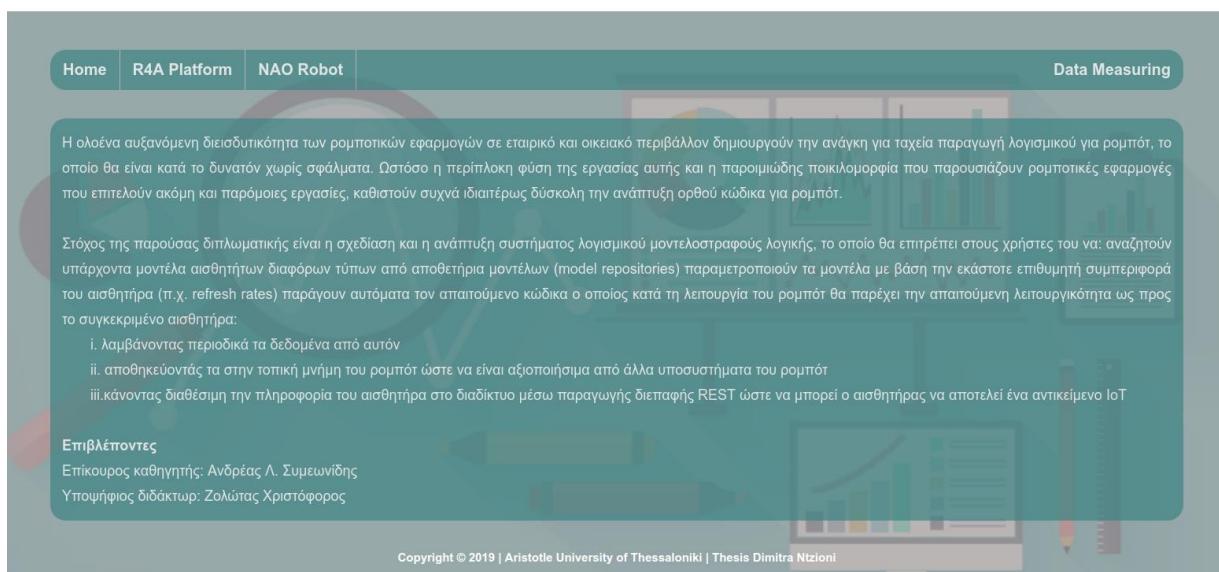
Πίνακας 126: Αίτημα GET για την ανάκτηση αρχείου ήχου ενός αισθητήρα τύπου Acoustic.

Request Method:	GET
Request URL:	http://localhost:3001/measurements/one/nameAcousticSystem*/filename** * nameAcousticSystem: όνομα που δίνει ο χρήστης σε ένα υποσύστημα τύπου Acoustic ** filename: όνομα αρχείου
Content-Type:	audio/wav
Status Code:	200: Αρχείο ήχου.
Response Body: (Status 200)	audio

4.7 Web Application

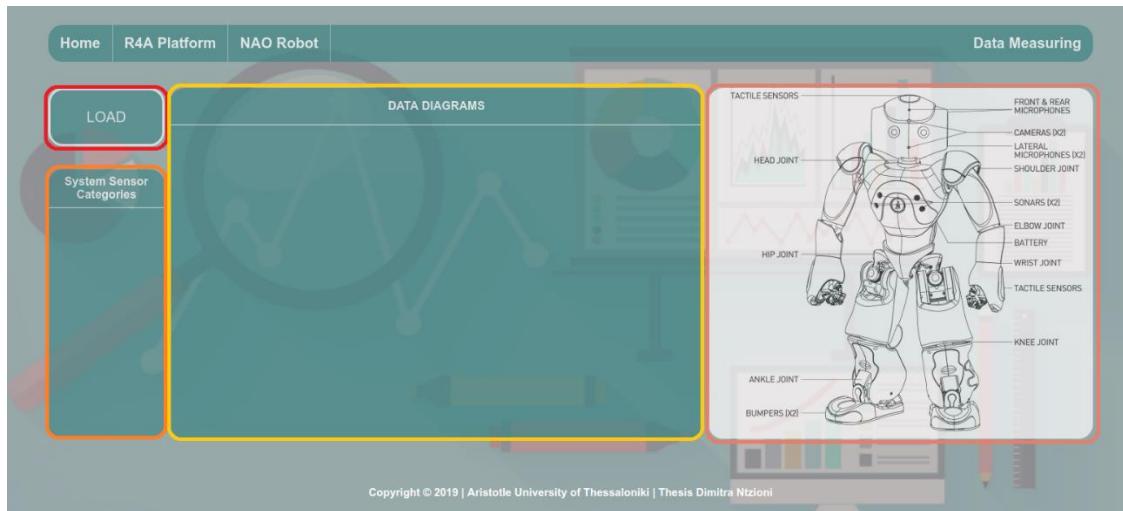
Στα πλαίσια της παρούσας διπλωματικής υλοποιήθηκε ακόμη μία λειτουργική web εφαρμογή, η οποία μπορεί να αξιοποιηθεί για την παρουσίαση της συλλογής των δεδομένων από τους αισθητήρες. Η παρούσα εφαρμογή έχει γραφθεί σε *REACT* [23] και *JavaScript*, ενώ για την ενημέρωση της συνδέεται με το *API* του συστήματος. Τα οφέλη της εφαρμογής αυτής είναι πολλαπλά. Αρχικά, μέσω αυτής της εφαρμογής, έγινε ευκολότερη η δοκιμή του παραγόμενου *API* και διορθώθηκαν σφάλματα. Παράλληλα, διερευνήθηκαν ορισμένες ιδέες αναφορικά με την πληροφορία που γίνεται διαθέσιμη, οδηγώντας στον άρτιο σχεδιασμό ενός *API* το οποίο θα προσφέρει τις πληροφορίες όλου του συστήματος. Επιπλέον, μέσω αυτής της εφαρμογής, δίνεται η ευκαιρία στο χρήστη να παρακολουθεί τη ροή πληροφορίας που παράγει το ρομπότ μέσω των αισθητήρων του, σε ένα φιλικό προς το χρήστη περιβάλλον. Παρακάτω, γίνεται παρουσίαση της εφαρμογής μέσα από μία σειρά στιγμιότυπων κατά τη λειτουργία ολόκληρου του συστήματος [25].

Η εφαρμογή περιλαμβάνει ουσιαστικά δύο σελίδες, την αρχική (*Home*) και τη σελίδα παρακολούθησης του συστήματος (*Data Measuring*). Ωστόσο, μπορεί κανείς από το *menu* να περιηγηθεί και στις επίσημες ιστοσελίδες που αφορούν το ρομπότ *NAO*, αλλά και την πλατφόρμα *R4A* και το *Εργαστήριο Επεξεργασίας Πληροφοριών και Υπολογισμών*.



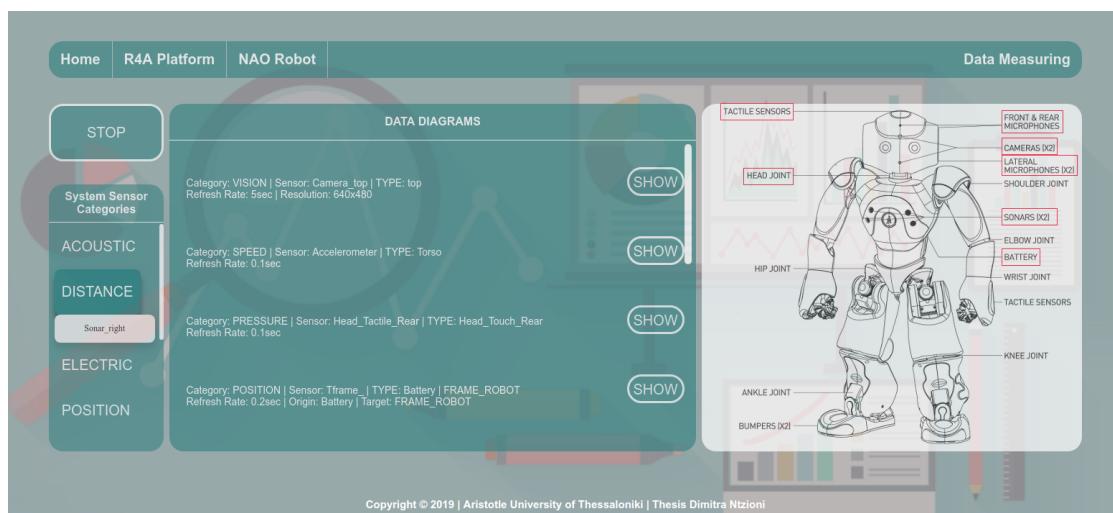
Σχήμα 36: Στιγμιότυπο της αρχικής σελίδας της web εφαρμογής

Στην αρχική σελίδα, Σχήμα 36, ο χρήστης μπορεί να μάθει το γνωστικό αντικείμενο της συγκεκριμένης εφαρμογής και να ενημερωθεί για τους στόχους που τέθηκαν.



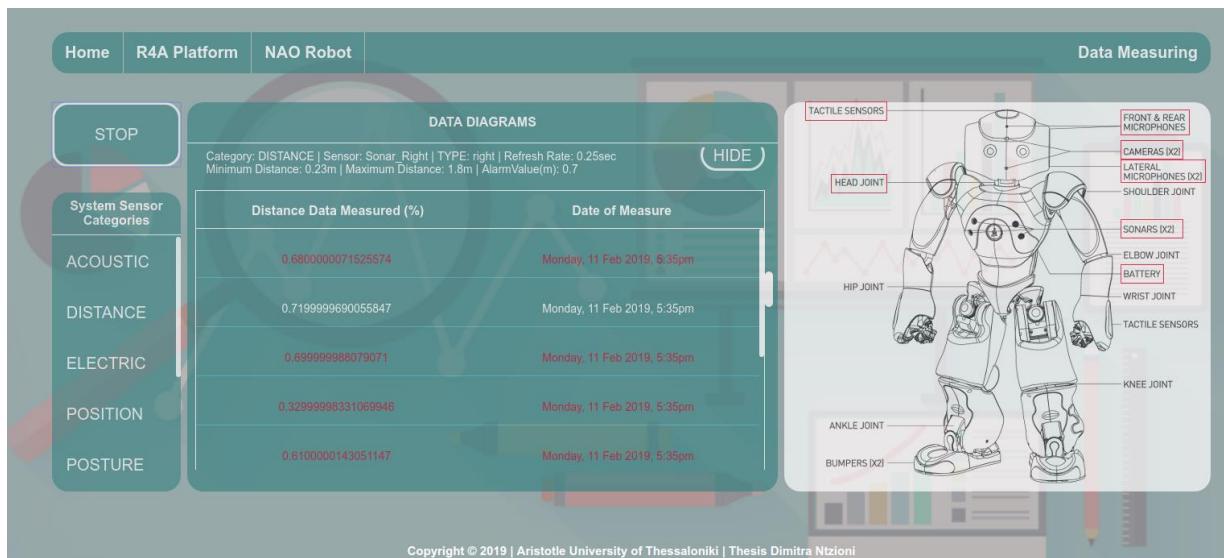
Σχήμα 37: Στιγμιότυπο της σελίδας Data Measuring της εφαρμογής

Επιλέγοντας να μεταφερθεί στη σελίδα παρακολούθησης του συστήματος, ο χρήστης θα δει ένα παράθυρο, όπως αυτό στο Σχήμα 37. Όπως αναφέρεται στην αριστερή στήλη, με κόκκινο πλαίσιο, εμφανίζεται το κουμπί *load*, μέσω του οποίου ο χρήστης μπορεί να φορτώσει στη σελίδα όλα τα δεδομένα της παρακολούθησης. Ακριβώς από κάτω, μέσα σε πορτοκαλί πλαίσιο, υπάρχει μία στήλη που θα περιέχει τις κατηγορίες αισθητήρων του συστήματος που επέλεξε νωρίτερα ο χρήστης στο μοντέλο του. Το σημείο στο οποίο θα φαίνονται όλα τα δεδομένα βρίσκεται στη μεσαία στήλη, που παρουσιάζεται με κίτρινο πλαίσιο. Στη δεξιά πλευρά του παραθύρου, υπάρχει μία εικόνα του ρομπότ *NAO*, σε σομόν πλαίσιο, η οποία παρουσιάζει τους αισθητήρες που φέρει το ρομπότ αλλά και τη θέση τους πάνω σε αυτό. Με αυτόν τον τρόπο, ο χρήστης μπορεί να κατανοεί καλύτερα την όλη διαδικασία και να αποκτά μία ακόμη πιο άρτια εικόνα για το σύστημα που φτιάχνει.



Σχήμα 38: Στιγμιότυπο της εφαρμογής μία στιγμή, έπειτα από το πάτημα του κουμπιού *load*

Στο Σχήμα 38 απεικονίζονται τα δεδομένα ενός συστήματος που υλοποιήθηκε σε πειραματικό στάδιο της διπλωματικής. Τα πλαίσια της οθόνης περιέχουν τιμές που όπως αναφέρθηκε και προηγουμένως αναφέρονται στις κατηγορίες και τους αισθητήρες του συστήματος. Μάλιστα, ο χρήστης περνώντας πάνω από την κάθε κατηγορία μπορεί να δει ποιους αισθητήρες περιέχει, τους οποίους ο ίδιος έχει ονοματίσει νωρίτερα στο μοντέλο του. Για να είναι πιο εύκολο στο χρήστη να ξεχωρίζει τα δεδομένα που λαμβάνει από κάθε έναν αισθητήρα, μόνο για την εφαρμογή, ο αισθητήρας που αφορά τη στάση σώματος του ρομπότ, εμφανίζεται ως ξεχωριστή κατηγορία, από τος υπόλοιπους της θέσης, η οποία ονομάζεται *Posture*. Ακόμη, για τη διευκόλυνση του χρήστη έχει προστεθεί η δυνατότητα να εμφανίζει και να κρύβει τον πίνακα που αφορά ένα συγκεκριμένο αισθητήρα, επικεντρώνοντας την προσοχή του σε όσους επιθυμεί. Πάνω από κάθε πίνακα εμφανίζονται σε μορφή τίτλου, όλα τα χαρακτηριστικά που αφορούν το συγκεκριμένο αισθητήρα, τα οποία έχει θέσει ο χρήστης κατά τη δημιουργία του συστήματος του. Παράλληλα, μπορεί ανά πάσα στιγμή να δει στη διπλανή εικόνα τους αισθητήρες που έχουν περιτριγυριστεί από κόκκινο πλαίσιο, γεγονός που σημαίνει ότι συμπεριλαμβάνονται στο σύστημα.



Σχήμα 39: Στιγμιότυπο της εφαρμογής που παρουσιάζονται μετρήσεις του αισθητήρα τάξε

Στο Σχήμα 39 παρουσιάζεται ενδεικτικά ο τρόπος εμφάνισης των μετρήσεων για έναν αισθητήρα τύπου απόστασης. Με κόκκινο χρώμα εμφανίζονται οι τιμές οι οποίες ξεπερνούν την τιμή που έθεσε ο χρήστης ως *alarmValue* νωρίτερα στο σύστημα του.

Αξίζει να αναφερθεί στο σημείο αυτό πως η εφαρμογή χρησιμοποιεί το παραγόμενο API που υλοποιείται στην παρούσα διπλωματική εργασία, για την ανάκτηση όλης της απαραίτητης πληροφορίας αναφορικά με τους αισθητήρες και τις παραμέτρους τους, τις κατηγορίες του συστήματος και τέλος τις ίδιες τις μετρήσεις για τον

κάθε αισθητήρα. Ακόμη, για να μπορεί ο χρήστης να έχει μία πιο σφαιρική εικόνα του συστήματός του, η εφαρμογή δεν παρουσιάζει μόνο την τελευταία τιμή της μέτρησης, αλλά ένα ιστορικό των τελευταίων δέκα μετρήσεων που υλοποίησε ο αισθητήρας, συνοδευόμενες από τη χρονική στιγμή που έγινε η μέτρηση. Οι κλήσεις για τη σύνδεση με το API γίνονται μέσω της βιβλιοθήκης *axios*⁽¹³⁾ η οποία επιτρέπει την εκτέλεση *XMLHttpRequests* σε γλώσσα *JavaScript*. Περισσότερες πληροφορίες για τη συγκεκριμένη εφαρμογή μπορεί να βρει ο αναγνώστης στο *github*, στον παρακάτω σύνδεσμο: <https://github.com/dimitrantz/Thesis/tree/master/web-app/src>

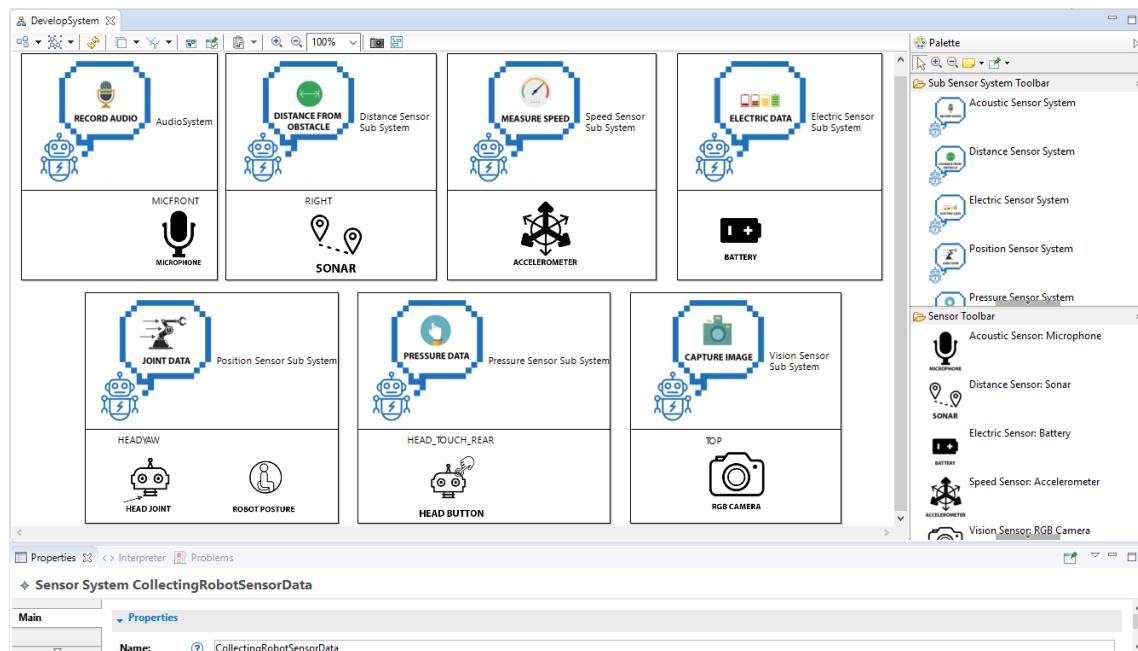
⁽¹³⁾ Περισσότερες πληροφορίες για τη βιβλιοθήκη *axios* μπορείτε να βρείτε εδώ: <https://github.com/axios/axios>

5 Παράδειγμα Χρήσης Συστήματος

Το σύστημα δοκιμάστηκε, υλοποιώντας πολλές εφαρμογές οι οποίες περιείχαν διαφορετικό συνδυασμό αισθητήρων, προκειμένου να ελεγχθεί η ομαλή λειτουργία του υπό διαφορετικές συνθήκες. Πιο αναλυτικά, υλοποιήθηκε σύστημα το οποίο περιείχε αρχικά έναν αισθητήρα, ενώ ο αριθμούς αυτός, έφτασε τελικά τους πενήντα, δημιουργώντας κάθε φορά διαφορετικούς συνδυασμούς. Παρακάτω, για να είναι εύκολα κατανοητό στον αναγνώστη, θα παρουσιαστεί ενδεικτικά ένα παράδειγμα, το οποίο περιλαμβάνει ένα σύστημα με έναν αισθητήρα σε κάθε κατηγορία, εκτός της κατηγορίας θέσης, που περιλαμβάνονται δύο.

5.1 Παρουσίαση επιλεγμένων στοιχείων

Συγκεκριμένα, το σύστημα περιλαμβάνει συνολικά εφτά υποσυστήματα διαφορετικής κατηγορίας μεταξύ τους και οχτώ αισθητήρες. Αναλυτικά, πρόκειται για ένα μικρόφωνο, μία RGB κάμερα, ένα Sonar, την μπαταρία και το αξελερόμετρο του ρομπότ, το πίσω κουμπί του κεφαλιού και μία άρθρωση του κεφαλιού του ρομπότ, καθώς ακόμη και τη στάση σώματος του ρομπότ. Σκοπίμως, δε θα παρουσιαστεί κανένα χαρακτηριστικό των αισθητήρων, αντιθέτως όλη η πληροφορία του συστήματος, μαζί με τις μετρήσεις, θα ανακτηθούν από το παραγόμενο API. Στο Σχήμα 40 που ακολουθεί εμφανίζεται ο σχεδιασμός του συστήματος στο *Sirius UI*.



Σχήμα 40: Αναπαράσταση συστήματος στο γραφικό περιβάλλον *Eclipse Sirius*

5.2 Αποτελέσματα

Στο σημείο αυτό, ο χρήστης πρέπει να προχωρήσει στην επιλογή αυτόματης παραγωγής του εκτελέσιμου κώδικα σε γλώσσα Python 2.7, καθώς επίσης να συνδεθεί στο ρομπότ και να τρέξει τον παραγόμενο κώδικα. Αμέσως μετά, μπορεί να εκτελέσει το αρχείο του API, μέσω του οποίου θα γίνουν διαθέσιμα όλα τα σχετικά δεδομένα. Παρακάτω παρουσιάζονται όλες οι κλήσεις του API που παράχθηκαν. Τα αρχεία ήχου και εικόνας θα αποθηκεύονται στον αντίστοιχο φάκελο που έχει καθοριστεί και ο χρήστης μπορεί να τα βρει αναζητώντας τα με το όνομα τους.

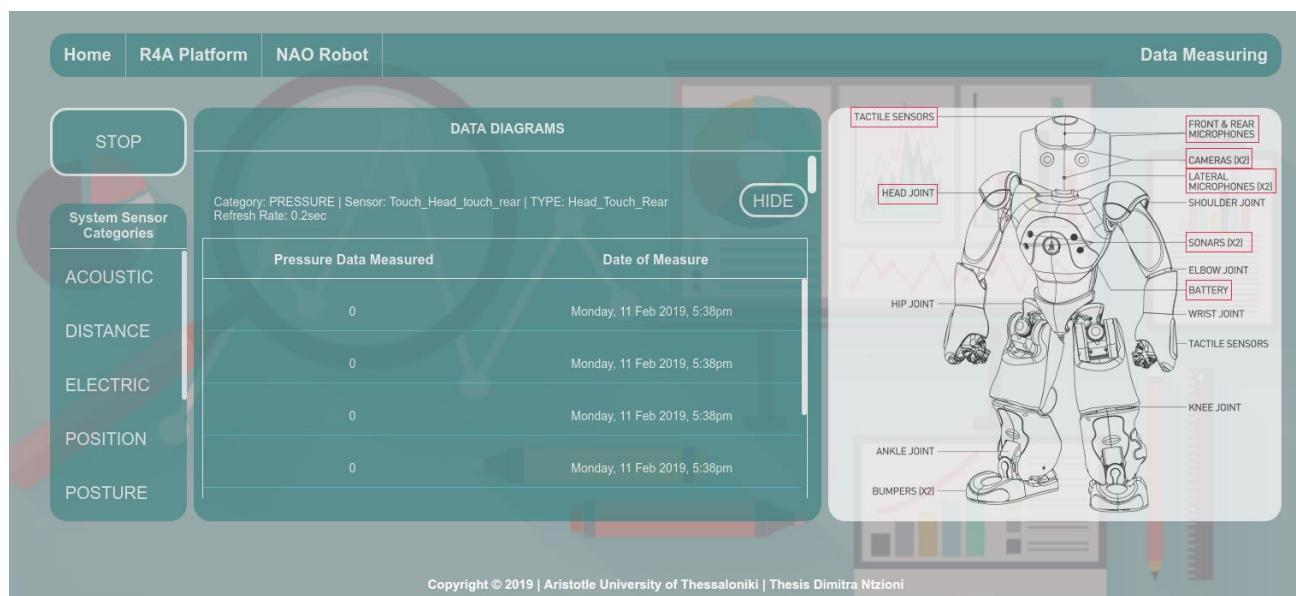
Πίνακας 127: Κλήσεις του API που παράχθηκαν μέσω του συστήματος

- | | |
|---|--|
| 1. GET/ categories | 29. GET/ sensors_parameters/Camera |
| 2. GET/ sensors_parameters | 30. GET/ measurements/all/Camera |
| 3. GET/ sensors_parameters/names | 31. GET/ measurements/last/Camera |
| 4. GET/ sensors_parameters/category/ACOUSTIC | 32. GET/ measurements/last/Camera/1..* |
| 5. GET/ sensors_parameters/category/DISTANCE | 33. GET/ measurements/one/Camera /filename |
| 6. GET/ sensors_parameters/category/ELECTRIC | 34. GET/ sensors_parameters/Accelerometer |
| 7. GET/ sensors_parameters/category/POSITION | 35. GET/ measurements/all/Accelerometer |
| 8. GET/ sensors_parameters/category/PRESSURE | 36. GET/ measurements/last/Accelerometer |
| 9. GET/ sensors_parameters/category/SPEED | 37. GET/ measurements/last/Accelerometer/1..* |
| 10. GET/ sensors_parameters/category/VISION | 38. GET/ sensors_parameters/Battery |
| 11. GET/ info/micFront | 39. GET/ measurements/all/Battery |
| 12. GET/ info/Sonar_right | 40. GET/ measurements/last/Battery |
| 13. GET/ info/Camera | 41. GET/ measurements/last/Battery/1..* |
| 14. GET/ info/Accelerometer | 42. GET/ sensors_parameters/RobotPosture |
| 15. GET/ info/Battery | 43. GET/ measurements/all/RobotPosture |
| 16. GET/ info/RobotPosture | 44. GET/ measurements/last/RobotPosture |
| 17. GET/ info/Robot | 45. GET/ measurements/last/RobotPosture/1..* |
| 18. GET/ info/Joint_Headyaw | 46. GET/ sensors_parameters/Joint_Headyaw |
| 19. GET/ info/Button_HeadTouchRear | 47. GET/ measurements/all/Joint_Headyaw |
| 20. GET/ sensors_parameters/AudioSystem | 48. GET/ measurements/last/Joint_Headyaw |
| 21. GET/ measurements/all/AudioSystem | 49. GET/ measurements/last/Joint_Headyaw/1..* |
| 22. GET/ measurements/last/AudioSystem | 50. GET/ sensors_parameters/Sonar_right |
| 23. GET/ measurements/last/AudioSystem/1..* | 51. GET/ measurements/all/Sonar_right |
| 24. GET/ measurements/one/AudioSystem/filename | 52. GET/ measurements/last/Sonar_right |
| 24. GET/ sensors_parameters/Button_HeadTouchRear | 53. GET/ measurements/last/Sonar_right/1..* |
| 26. GET/ measurements/all/Button_HeadTouchRear | |
| 27. GET/ measurements/last/Button_HeadTouchRear | |
| 28. GET/ measurements/last/Button_HeadTouchRear/1..* | |

Οι φράσεις micFront, Sonar_right, Camera, Accelerometer, Battery, RobotPosture, Joint_Headyaw, Button_HeadTouchRear αποτελούν τα ονόματα των αισθητήρων, ενώ το AudioSystem είναι το όνομα του υποσυστήματος κατηγορίας ήχου, όπως αυτά ορίστηκαν από το χρήστη.

Όπως είναι φανερό, βάσει των ονομάτων αυτών μπορεί να ανακτήσει πληροφορίες σχετικά με τα χαρακτηριστικά που όρισε, αλλά και τα τεχνικά χαρακτηριστικά αυτών. Μπορεί ακόμη να δει και να επεξεργαστεί τα δεδομένα των μετρήσεων, λαμβάνοντας την τελευταία, ένα συγκεκριμένο αριθμό μετρήσεων, ή όλες τις μετρήσεις που αφορούν έναν αισθητήρα. Επιπλέον, του δίνεται η δυνατότητα να αναζητήσει τους αισθητήρες μίας κατηγορίας ή τα ονόματα που έχει ορίσει ο ίδιος. Τέλος, μπορεί να ενημερωθεί για τα τεχνικά χαρακτηριστικά του ρομπότ εκτελώντας την αντίστοιχη κλήση. Στο σημείο αυτό, πρέπει να επισημανθεί πως οι κλήσεις 23, 28, 32, 37, 45, 49 και 53, παράγουν διαφορετικό endpoint για κάθε αριθμό μετρήσεων που επιλέγει να ανακτήσει ο χρήστης. Επιπρόσθετα, οι κλήσεις 24 και 33 δημιουργούν διαφορετικό endpoint για κάθε αρχείο εικόνας και ήχου που δημιουργείται από τους εκάστοτε αισθητήρες.

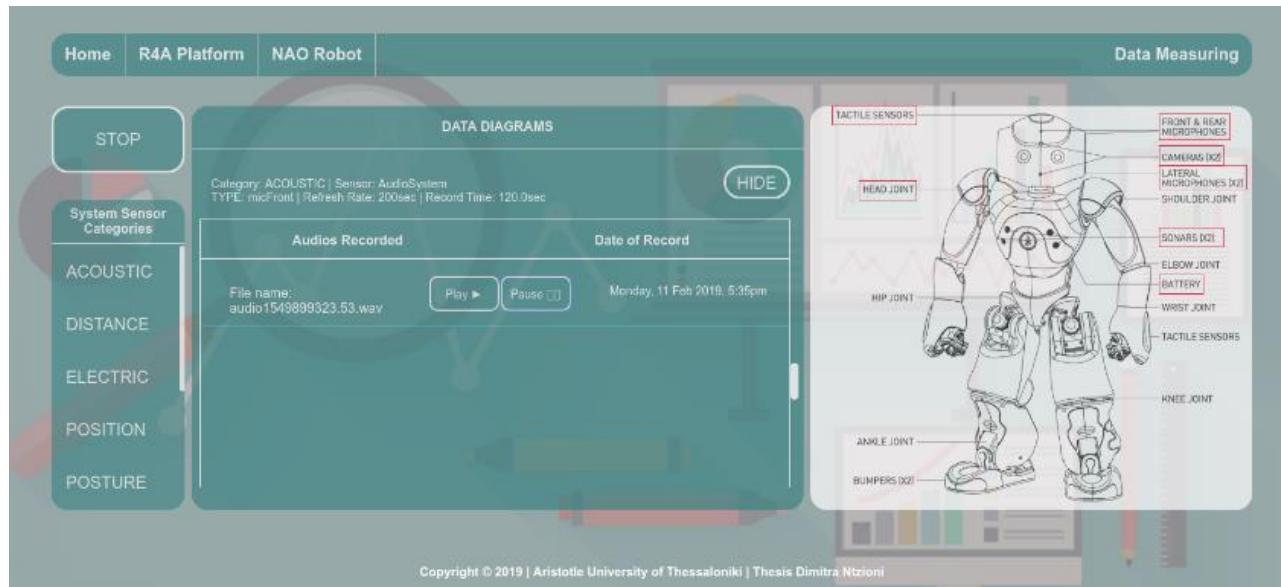
Παρακάτω, στο **Σχήμα 41**, φαίνεται ένα στιγμιότυπο από την web εφαρμογή που κατασκευάστηκε. Συγκεκριμένα, παρουσιάζεται η περισσότερη πληροφορία του συστήματος, φαίνονται οι κατηγορίες αισθητήρων, οι αισθητήρες που υπάρχουν στο σύστημα και η θέση τους στο ρομπότ, καθώς και ορισμένες μετρήσεις αυτών. Με αυτόν τον τρόπο αποδεικνύεται πως το API που δημιουργείται είναι πλήρως λειτουργικό και αξιόπιστο, ως προς την ανάκτηση των ζητούμενων πόρων.



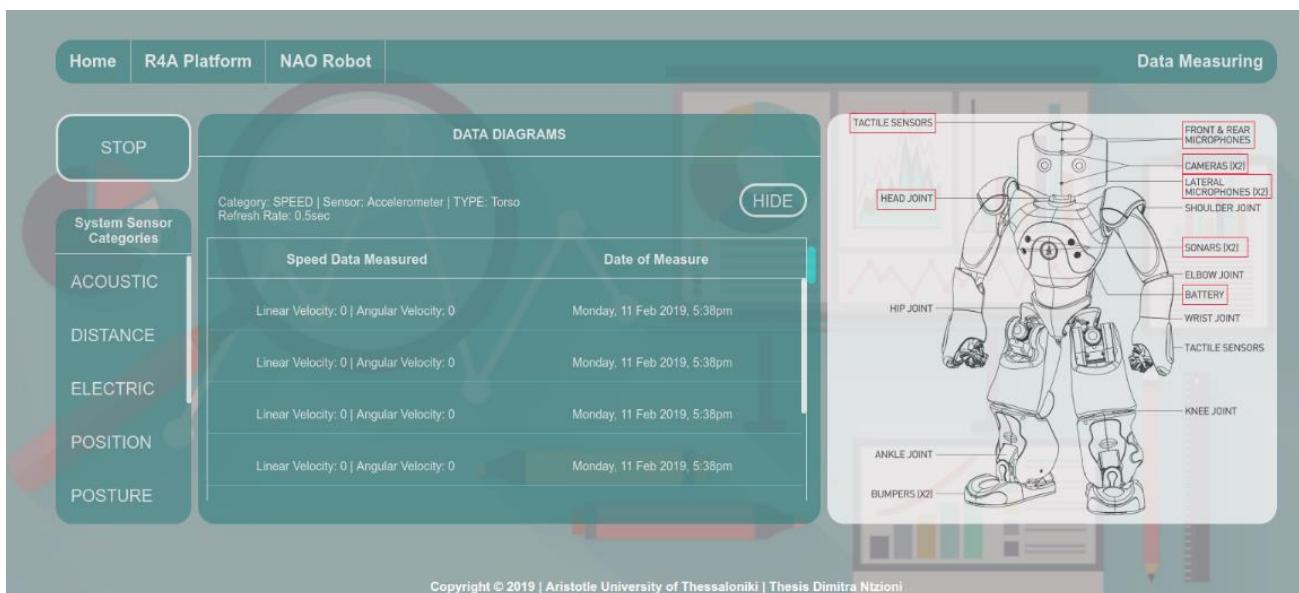
Σχήμα 41: Στιγμιότυπο της web εφαρμογής κατά τη συλλογή δεδομένων από αισθητήρα πίεσης του ρομπότ

Συμπερασματικά, η εφαρμογή καλύπτει επαρκώς την ανάγκη για ένα front-end περιβάλλον για το υλοποιημένο Rest API. Μέσα από το παράδειγμα που παρουσιάστηκε γίνεται αντιληπτό πως η μηχανή που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματική εργασίας μπορεί να ανταπεξέλθει σε πλήθος διαφορετικών συστημάτων αισθητήρων, ενώ παράλληλα μειώνει σημαντικά το χρόνο και τον κόπο συγγραφής και παραγωγής κώδικα.

Παρακάτω, στα Σχήματα 42 και 43 παρουσιάζονται δύο στιγμιότυπα της εφαρμογής, τα οποία αναδεικνύουν τον τρόπο εμφάνισης των μετρήσεων που λαμβάνονται από τους αισθητήρες.



Σχήμα 42: Στιγμιότυπο της web εφαρμογής κατά τη συλλογή δεδομένων από αισθητήρα ύχου του ρομπότ



Σχήμα 43: Στιγμιότυπο της web εφαρμογής κατά τη συλλογή δεδομένων από αισθητήρα ύχου του ρομπότ

6 Συμπεράσματα & Μελλοντικές Επεκτάσεις

6.1 Συμπεράσματα

Στα πλαίσια της παρούσας διπλωματικής εργασίας διερευνήθηκε η δυνατότητα δημιουργίας ενός συστήματος αυτόματης παραγωγής διεπαφής, ώστε να συλλέγονται δεδομένα από αισθητήρες που εμπεριέχονται σε ρομπότ. Συγκεκριμένα, επιλέχθηκε η μεθοδολογία *MDE* που αξιοποιεί την έννοια του μετα-μοντέλου, μέσω της οποίας δημιουργούνται μοντέλα όλων των θεμάτων που σχετίζονται με το πρόβλημα, στοχεύοντας σε αφηρημένες αναπαραστάσεις.

Το μετα-μοντέλο που παράχθηκε αξιοποιεί το σύνολο των δυνατοτήτων της πλατφόρμας *R4A* που αφορούν το ρομπότ *NAO*. Ο χρήστης δημιουργεί το μοντέλο των αισθητήρων του στο *Sirius UI* που σχεδιάστηκε, ορίζοντας έτσι τις παραμέτρους τους. Έπειτα, η προσαρμοσμένη στο διάγραμμα γεννήτρια κώδικα, γραμμένη σε *Accelleo Template Language*, παράγει τον τελικό εκτελέσιμο κώδικα σε γλώσσα *Python 2.7*. Ο χρήστης, τέλος, μπορεί να συνδεθεί στο ρομπότ *NAO* για να τρέξει τον κώδικα που δημιούργησε και να τρέξει την *flask* εφαρμογή που παράγει το *Restful API*. Με αυτόν τον τρόπο έχει έτοιμη στη διάθεση του την αυτόματη διεπαφή που επιθυμεί. Παράλληλα, μπορεί να αξιοποιήσει *web* εφαρμογή που κατασκευάστηκε, ώστε να δει τις πληροφορίες των αισθητήρων που επέλεξε, τη θέση αυτών πάνω στο ρομπότ και τα δεδομένα που συλλέγονται από αυτούς σε πραγματικό χρόνο.

Τα πειράματα πραγματοποιήθηκαν σε πραγματικό ρομπότ, και όχι σε κάποιο μέσο προσομοίωσης και έδειξαν ότι η υλοποίηση με τους κατάλληλους περιορισμούς των παραμέτρων του συστήματος, όπως έχουν τεθεί, λειτουργεί δίχως κανένα πρόβλημα.

6.2 Μελλοντικές Επεκτάσεις

Το μοντέλο που δημιουργείται επιτυγχάνει σε μεγάλο βαθμό τους ερευνητικούς στόχους που τέθηκαν εξαρχής για την παρούσα διπλωματική. Παρόλα αυτά πρέπει να επισημανθεί πως συνιστά μία απλοποιημένη προσέγγιση στο πρόβλημα του *MDE* και σίγουρα επιδέχεται βελτίωσης. Στη συνέχεια, αναφέρονται λακωνικά ορισμένες προτάσεις για επέκταση του συστήματος.

Αρχικά, όπως έχει επισημανθεί το μοντέλο προσαρμόστηκε στα χαρακτηριστικά των αισθητήρων που έχει το ρομπότ NAO. Ένα πρώτο βήμα που θα μπορούσε να γίνει είναι αναπτυχθεί το μοντέλο που να περιλαμβάνει ακόμη περισσότερες κατηγορίες αισθητήρων, καθώς ακόμη και για μεγαλύτερο πλήθος ρομπότ, όπως το *TurtleBot*, το *ARDrone* κ.α. Με αυτόν τον τρόπο, το λογισμικό θα μπορεί να χρησιμοποιηθεί σε περισσότερες εφαρμογές καλύπτοντας μεγαλύτερο φάσμα αναγκών. Ακόμη, μία ενδιαφέρουσα προσθήκη θα ήταν να μπορεί ο χρήστης να μοντελοποιεί εκτός των αισθητήρων και τη βάση δεδομένων όπου θα αποθηκεύεται όλη απαραίτητη πληροφορία που παράγεται από το σύστημα. Συγκεκριμένα, να μπορεί να επιλέξει τύπο, αλλά και να την παραμετροποιεί, επιλέγοντας για παράδειγμα τη χωρητικότητά της. Ο χρήστης μπορεί έτσι να ορίζει περισσότερες παραμέτρους και να επεμβαίνει σε μεγαλύτερο βαθμό στη δημιουργία της διεπαφής που τον ενδιαφέρει. Επίσης, παρόλο που η συγκεκριμένη εργασία είναι βασισμένη στην πλατφόρμα R4A, θα μπορούσαν να γίνουν κινήσεις για επέκταση της, ώστε να καλύπτει μεγαλύτερο φάσμα εφαρμογών. Τέλος, ένας σημαντικός παράγοντας που επηρεάζει την ποιότητα και την απόδοση της εφαρμογής είναι και η ποιότητα του δικτύου. Για το λόγο αυτό, μία βελτίωση που προτείνεται ακόμη είναι να μοντελοποιηθεί και το δίκτυο του χρήστη και να ορίζει παραμέτρους για αυτό, προκειμένου να ελέγχονται καλύτερα και οι περιορισμοί που θέτει για τους αισθητήρες του συστήματος.

7 Βιβλιογραφία

- [1] International Federation of Robotics [Ηλεκτρονικό].Available: <https://ifr.org/>.[Πρόσβαση 11 02 2019]
- [2] IEEE Standards Board «IEEE Standard Glossary of Software Engineering Terminology» 1990
- [3] «Wiki: Documentation» [Ηλεκτρονικό].Available: <http://wiki.ros.org/Documentation>.[Πρόσβαση 11 02 2019]
- [4] Terminology J. Bézivin, «In search of a basic principle for Model Driven Engineering» Special Novatica Issue - UML and Model Engineering, τόμ. 5, αρ. 2, pp. 21-24, 2004.
- [5] F. Truyen, «The Fast Guide to Model Driven Architecture The Basics of Model Driven Architecture» Practice, p. 12, 2006.
- [6] T. Mens και P. Van Gorp, «A taxonomy of model transformation» Electronic Notes in Theoretical Computer Science, τόμ. 152, αρ. 1-2, pp. 125-142, 2006.
- [7] Marco Torchiano, Federico Tomassetti, Relevance, Filippo Riccab, Alessandro Tisob και Gianna Reggio «Relevance, benefits, and problems of software modelling and model driven techniques—A survey in the Italian industry», 3 2012.
- [8] K. Czarnecki και S. Helsen, «Feature-Based Survey of Model Transformation Approaches» IBM Systems Journal, τόμ. 45, αρ. 3, pp. 621-645, 2006.
- [9] C. Zolotas και T. Diamantopoulos, «D2.3.2 Ontology to MOF Models Transformation Mechanism» 2015.
- [10] C. Zolotas και K. Chatzidimitriou, «D2.2 Definition of Platform Independent and Platform Specific Service Models» 2015.
- [11] M. D. Engineering, «On S-CASE automation engine, Model Driven Engineering and queen Victoria Era buildings (Part 1)!» αρ. Part 1, pp. 2-5, 2000
- [12] I. Sacevski και J. Veseli, «Introduction to Model Driven Architecture (MDA)» Department of Computer Science, University of Salzburg, pp. 1-15, 2007.
- [13] Jose Evora, Jose Juan Hernandez και Mario Hernandez «Advantages of Model Driven Engineering for studying complex systems» Springer Science+Business Media Dordrecht 2014
- [14] «Wikipedia, the free encyclopedia» [Ηλεκτρονικό].Available: https://en.wikipedia.org/wiki/Representational_state_transfer [Πρόσβαση 11 02 2019]
- [15] «Rest API Tutorial» [Ηλεκτρονικό].Available: <https://restfulapi.net/http-methods/>. [Πρόσβαση 11 02 2019]
- [16] «Ecore - Eclipsepedia - Eclipse Wiki» [Ηλεκτρονικό].Available: <https://wiki.eclipse.org/Ecore> [Πρόσβαση 11 02 2019]
- [17] «redislabs, Home of Redis» [Ηλεκτρονικό].Available: <https://redislabs.com/ebook/part-1-getting-started> [Πρόσβαση 11 02 2019]

- [18] «Flask web development one drop at a time» [Ηλεκτρονικό].Available:
<http://flask.pocoo.org/>. [Πρόσβαση 11 02 2019]
- [19] «Robotics 4 All groups» [Ηλεκτρονικό].Available: <https://r4a.issel.ee.auth.gr/>. [Πρόσβαση 11 02 2019]
- [20] Craig Schlenoff, Edson Prestes, Raj Madhavan, Paulo Goncalves, Howard Li, Stephen Balakirsky, Thomas Kramer και Emilio Miguelanez «An IEEE Standard Ontology for Robotics and Automation», 2 2011
- [21] Cyril Cecchine, Sebastien Mosser, Philippe Collet «Automated Deployment of Data Collection Policies over Heterogeneous Shared Sensing Infrastructures», 2016
- [22] «A better way to design» [Ηλεκτρονικό].Available: <https://www.figma.com/>. [Πρόσβαση 11 02 2019]
- [23] «React, a javascript library for building user interfaces» [Ηλεκτρονικό].Available:
<https://reactjs.org/>. [Πρόσβαση 11 02 2019]
- [24] «Interneting is hard» [Ηλεκτρονικό].Available: <https://internetingishard.com/>. [Πρόσβαση 11 02 2019]
- [25] «Wikipedia, the free encyclopedia» [Ηλεκτρονικό].Available:
[https://en.wikipedia.org/wiki/Web_application/](https://en.wikipedia.org/wiki/Web_application). [Πρόσβαση 11 02 2019]
- [26] «Eclipse Sirius» [Ηλεκτρονικό].Available: <https://www.eclipse.org/sirius/>. [Πρόσβαση 11 02 2019]
- [27] Obeo, «Eclipse Acceleo» [Ηλεκτρονικό].Available: <https://www.eclipse.org/acceleo/>. [Πρόσβαση 11 02 2019]
- [28] Emmanouil Tsardoulas, Konstantinos Panayiotou, Andreas Symeonidis «Formalizing the robot application development via robot agnostic APIs»