

## ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών  
Τομέας Ηλεκτρονικής και Υπολογιστών  
Εργαστήριο Συστημάτων Υπολογιστών

Διπλωματική Εργασία  
του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και  
Τεχνολογίας Υπολογιστών της Πολυτεχνικής Σχολής του  
Πανεπιστημίου Πατρών

ΜΠΟΧΑΛΗΣ ΠΑΝΑΓΙΩΤΗΣ  
Αριθμός Μητρώου: 7858

### Θέμα

**Αναβάθμιση παραδοσιακών εφαρμογών με τη  
χρήση τεχνολογιών διαδικτύου  
Μία προσέγγιση βασισμένη σε IoT και  
συνιστώσες(components)**

### Επιβλέπων

ΘΡΑΜΠΟΥΛΙΔΗΣ ΚΛΕΑΝΘΗΣ

Αριθμός Διπλωματικής Εργασίας:

Πάτρα, Ιούλιος 2018

Πανεπιστήμιο Πατρών - Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών  
Μπόχαλης Παναγιώτης

©2018 - Με την επιφύλαξη παντός δικαιώματος

## **ΠΙΣΤΟΠΟΙΗΣΗ**

Πιστοποιείται ότι η Διπλωματική Εργασία με θέμα

**Αναβάθμιση παραδοσιακών εφαρμογών με την χρήση τεχνολογιών διαδικτύου**

**Μία προσέγγιση βασισμένη σε IoT και συνιστώσες(components)**

Του φοιτητή του Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

Μπόχαλης Παναγιώτης του Κωνσταντίνου

Αριθμός Μητρώου: 7858

Παρουσιάστηκε δημόσια και εξετάστηκε στο Τμήμα Ηλεκτρολόγων Μηχανικών και

Τεχνολογίας Υπολογιστών στις

**5 / 7 / 2018**

Ο Επιβλέπων

Ο Διευθυντής του Τομέα

Καθηγητής

Καθηγητής

Θραμπουλίδης Κλεάνθης

Χούσος Ευθύμιος



## Αριθμός Διπλωματικής Εργασίας:

**Θέμα: Αναβάθμιση παραδοσιακών εφαρμογών με την χρήση τεχνολογιών διαδικτύου. Μία προσέγγιση βασισμένη σε IoT και συνιστώσες(components)**

Φοιτητής

Επιβλέπων

Μπόχαλης Παναγιώτης του Κ.

Καθηγητής Κλεάνθης Θραμπουλίδης

# Περίληψη

Στη σημερινή εποχή, το διαδίκτυο των Αντικειμένων αποτελεί μια εξαιρετικά σημαντική τεχνολογία που αποσκοπεί στην διευκόλυνση της ανθρώπινης ζωής, επιτρέποντας την απρόσκοπτη επικοινωνία μεταξύ αντικειμένων και μηχανών με τον άνθρωπο. Ειδικότερα στο χώρο της βιομηχανίας αντιμετωπίζεται ως η επόμενη Βιομηχανική Επανάσταση. Οι προκλήσεις που δημιουργούνται από την αναμενόμενη ένταξη ενός νέου συνόλου συσκευών στο διαδίκτυο πρέπει να αντιμετωπιστούν, με σκοπό τα οφέλη που θα αποκομίσει η ανθρώπινη κοινωνία να είναι αντάξια των υψηλών προσδοκιών από την τεχνολογία αυτή.

Η παρούσα εργασία αποτελεί μια μελέτη τεχνολογιών που επιτρέπουν την επικοινωνία μεταξύ των συσκευών αλλά και μια μελέτη σχετικά με την αναβάθμιση παραδοσιακών εφαρμογών ακολουθώντας μια προσέγγιση βασισμένη σε συνιστώσες, μέσω της ανάλυσης, του σχεδιασμού και της υλοποίησης του συστήματος ελέγχου ενός IoT-συμβατού κατανεμημένου βιομηχανικού συστήματος. Πιο συγκεκριμένα, το σύστημα ελέγχου που αναπτύχθηκε αξιοποιεί τα πρωτόκολλα CoAP και LwM2M για την επίτευξη της επικοινωνίας μεταξύ των συσκευών.

Η αναβάθμιση της προυπάρχουσας εφαρμογής για τον έλεγχο του βιομηχανικού συστήματος που μελετήθηκε, πραγματοποιήθηκε χρησιμοποιώντας μια προσέγγιση βασισμένη σε συνιστώσες και η υλοποίηση έγινε αξιοποιώντας το OSGi framework. Η αποτελεσματικότητα του προτεινόμενου συστήματος αξιολογήθηκε χρησιμοποιώντας μεταξύ άλλων και εξομοιωτές των μηχανικών μερών του φυσικού συστήματος παραγωγής. Στο τέλος γίνεται μια συνολική αξιολόγηση του συστήματος και προτείνονται κατευθύνσεις προς τις οποίες μπορεί να προσανατολιστεί η περαιτέρω ανάπτυξή του.

**Λέξεις Κλειδιά:** Διαδίκτυο των Αντικειμένων, Συνιστώσα, LwM2M



# Abstract

Internet of Things is an important technology of our times, that promises to simplify everyday tasks, allowing communication between objects and machines with humans. Particularly within the industry domain, IoT has been labeled as the next Industrial Revolution. As a result, in order to unlock the full potential this new concept carries, it is essential to address the challenges that are also raised as a result.

The present thesis consists of a study of some technologies that allow communication between devices as well as a study on the upgrading of traditional applications following a component based approach, by analyzing, designing and implementing the control system of an IOT-compliant distributed industrial system. Specifically, the control system designed utilizes the CoAP and LwM2M protocols to achieve communication between devices.

The upgrade of the existing application used to control the industrial system studied was accomplished using a component-based approach and the implementation was made using OSGi framework. The effectiveness of the system designed is measured using simulators of the mechanical parts the production system consists of. Finally, a review of the system and the development procedure is given, along with suggestions for future development.

**Keywords:** *Internet of Things, Component, LwM2M*



# Ευχαριστίες

*"Scientists dream of great things. Engineers do them."*

James Michener, Author

Αρχικά θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή κύριο Κλεάνθη Θραμπουλίδη τόσο για την ευκαιρία που μου έδωσε να ασχοληθώ με το ιδιαίτερα ενδιαφέρον αντικείμενο της αναβάθμισης ενός παραδοσιακού βιομηχανικού συστήματος χρησιμοποιώντας την προσέγγιση των συνιστωσών και τεχνολογίες διαδίκτυου με σκοπό αυτό το σύστημα να ενταχθεί στο διαδίκτυο των αντικειμένων όσο και για την στήριξη και την πολύτιμη καθοδήγηση που μου πρόσφερε σε όλα τα στάδια της προσπάθειας αυτής. Οι γνώσεις και η εμπειρία που απέκτησα κατά την εκπόνηση της διπλωματικής εργασίας αυτής αποτελούν εξαιρετικά εφόδια για την μετέπειτα πορεία μου.

Επίσης θα ήθελα να ευχαριστήσω θερμά τον Μπουλούμπαση Ιωάννη για την πολύτιμη βοήθεια του σε διάφορα προβλήματα που αντιμετωπίσα καθώς και για την υλοποίηση του LwM2M server που χρησιμοποιήσα για τον έλεγχο της λειτουργικότητας του δικού μου συστήματος.

Τέλος θα ήθελα να ευχαριστήσω θερμά την οικογένεια μου για την αμέριστη συμπαράσταση και κατανόηση που έδειξε σε όλη την διάρκεια της πανεπιστημιακής μου πορείας.





# Περιεχόμενα

<b>Περίληψη</b>	<b>vi</b>
<b>Abstract</b>	<b>viii</b>
<b>Ευχαριστίες</b>	<b>x</b>
<b>1 Εισαγωγή</b>	<b>1</b>
1.1 Ιστορική αναδρομή στο Διαδίκτυο των αντικειμένων . . . . .	1
1.2 Αντικείμενο της εργασίας . . . . .	2
1.3 Δομή της εργασίας . . . . .	3
<b>2 Το διαδίκτυο των αντικειμένων</b>	<b>7</b>
2.1 Γενικά . . . . .	7
2.2 Η κατάσταση μέχρι σήμερα . . . . .	8
2.3 Πρότυπα και πρωτόκολλα IoT . . . . .	12
2.3.1 Η αρχιτεκτονική REST . . . . .	12
2.3.2 Τα πρωτόκολλα HTTP - CoAP . . . . .	13
2.3.3 Το πρωτόκολλο MQTT . . . . .	15
<b>3 Το πρωτόκολλο LwM2M</b>	<b>17</b>
3.1 Γενικά . . . . .	17
3.2 Περιγραφή των μηχανισμών του LwM2M . . . . .	18
3.3 Πλεονεκτήματα του LwM2M . . . . .	21
<b>4 Ανάπτυξη λογισμικού με την προσέγγιση συνιστώσων</b>	<b>23</b>
4.1 Εισαγωγή . . . . .	23
4.2 Η προσέγγιση των συνιστώσων . . . . .	23
4.2.1 Γενικές πληροφορίες και βασικές έννοιες . . . . .	23
4.2.2 Οντότητες Λογισμικού . . . . .	24
4.2.2.1 Component . . . . .	24
4.2.2.2 Interface . . . . .	25
4.2.2.3 Container . . . . .	26
4.2.2.4 Connector . . . . .	26
4.2.3 Στόχοι της CBSD προσέγγισης . . . . .	26
4.2.4 Γιατί ανάπτυξη λογισμικού με συνιστώσες . . . . .	28
4.2.5 Πλεονεκτήματα της CBSD σε σύγκριση με παραδοσιακές τεχνικές ανάπτυξης . . . . .	29
4.3 Open Services Gateway initiative (OSGi) . . . . .	30

---

4.3.1	Γενικά . . . . .	30
4.3.2	Η αρχιτεκτονική του OSGi . . . . .	31
4.3.3	Πλεονεκτήματα του OSGi . . . . .	34
4.3.4	Υλοποιήσεις του OSGi . . . . .	36
<b>5</b>	<b>To Liqueur Plant Σύστημα Παραγωγής</b>	<b>39</b>
<b>6</b>	<b>Σχεδιασμός του συστήματος</b>	<b>41</b>
6.1	Εισαγωγή . . . . .	41
6.2	Σχεδιασμός του API του Industrial Automation Thing . . . . .	41
6.3	Περιγραφή των LwM2M objects και resources . . . . .	42
6.4	Αρχιτεκτονικός σχεδιασμός του συστήματος ελέγχου του σιλό . . . . .	44
6.5	Δομή των σιλό . . . . .	45
6.6	Συμπεριφορά του συστήματος ελέγχου . . . . .	48
<b>7</b>	<b>Υλοποίηση του συστήματος</b>	<b>53</b>
7.1	Γενικά . . . . .	53
7.2	Οι συνιστώσες του συστήματος ελέγχου . . . . .	53
7.3	Ανάλυση των συνιστωσών του συστήματος ελέγχου . . . . .	55
7.3.1	Οι συνιστώσες των drivers του σιλό . . . . .	55
7.3.1.1	Συνιστώσες βαλβίδων του σιλό . . . . .	55
7.3.1.2	Συνιστώσες στοιχείων θέρμανσης και μίξης του σιλό . . . . .	56
7.3.1.3	Συνιστώσα του driver του σιλό . . . . .	57
7.3.1.4	Συνιστώσα του controller του σιλό . . . . .	59
7.3.1.5	Ο LwM2M client του σιλό . . . . .	60
7.3.4	Η πειραματική διάταξη . . . . .	62
<b>8</b>	<b>Συμπεράσματα και μελλοντική εργασία</b>	<b>65</b>
8.1	Συμπεράσματα . . . . .	65
8.2	Προτάσεις για μελλοντική εργασία . . . . .	66
<b>Αναφορές</b>		<b>73</b>
<b>Παράρτημα Α: Πηγαίος Κώδικας</b>		<b>75</b>

# Ευρετήριο Εικόνων

2.1	Το διαδίκτυο των αντικειμένων σε διάφορους τομείς [9] . . . . .	7
2.2	Οι φάσεις των βιομηχανικών επαναστάσεων [3] . . . . .	11
2.3	Η αρχιτεκτονική CoAP [29] . . . . .	15
2.4	Η αρχιτεκτονική MQTT [30] . . . . .	16
3.1	Η LwM2M αρχιτεκτονική [28] . . . . .	19
3.2	LwM2M Information Reporting Interface [32] . . . . .	21
4.1	Διάγραμμα ροής της διαδικασίας CBSD [34] . . . . .	24
4.2	<b>Αριστερά:</b> Components μαζί με τα interfaces τους. <b>Δεξιά:</b> Ένας container μαζί με το component [37] . . . . .	27
4.3	Τα επίπεδα της αρχιτεκτονικής του OSGi [48] . . . . .	31
4.4	Το μητρώο υπηρεσίων του OSGi [50] . . . . .	33
5.1	Σχηματική αναπαράσταση του συστήματος παραγωγής liqueur [59] . . . . .	39
6.1	Η αρχιτεκτονική του σιλό . . . . .	45
6.2	Η δομή του smartSilo1 . . . . .	46
6.3	Η δομή του smartSilo2 . . . . .	47
6.4	Η δομή του smartSilo3 . . . . .	47
6.5	Η δομή του smartSilo4 . . . . .	48
6.6	Το διάγραμμα κλάσεων του state machine του σιλό . . . . .	49
6.7	To state machine του smartSilo1 . . . . .	51
6.8	To state machine του smartSilo2 . . . . .	51
6.9	To state machine του smartSilo3 . . . . .	52
6.10	To state machine του smartSilo4 . . . . .	52
7.1	Το διάγραμμα συνιστωσών ενός σιλό . . . . .	54
7.2	Εξομοιωτής των μηχανικών στοιχείων του σιλό [61] . . . . .	62
7.3	Η πειραματική διάταξη με τα τέσσερα σιλό και τον σωλήνα . . . . .	63
7.4	Η διαδικασία παραγωγής liqueur τύπου B . . . . .	64



# Ευρετήριο Πινάκων

4.1 Διαθέσιμες υλοποιήσεις του OSGi και η έκδοση των modules που υποστηρίζουν	36
6.1 LwM2M objects και Resources	44
7.1 Αντιστοίχηση των φυσικών μερών του σιλό με τη συνιστώσα τους	55



# Ευρετήριο Αλγορίθμων

7.1	Μέρος της υλοποίησης της συνιστώσας την βαλβίδας εξαγωγής υγρού . . . . .	56
7.2	Οι υπηρεσίες που παρέχει η συνιστώσα του στοιχείου θέρμανσης . . . . .	57
7.3	Μέρος της υλοποίησης της συνιστώσας του driver του σιλό . . . . .	58
7.4	Η υλοποίηση των listener του driver του σιλό . . . . .	59
7.5	Οι υπηρεσίες που παρέχει η συνιστώσα του ελεγκτή του σιλό . . . . .	60



# Κεφάλαιο 1

## Εισαγωγή

### 1.1 Ιστορική αναδρομή στο Διαδίκτυο των αντικειμένων

Η ραγδαία εξέλιξη της τεχνολογίας και κατ' επέκταση του διαδικτύου οδηγεί τον κόσμο προς μία “συνεχώς διασυνδεδεμένη” πραγματικότητα. Το διαδίκτυο βρίσκεται σχεδόν παντού, είτε ενσύρματα είτε ασύρματα, και καθημερινώς όλο και περισσότερες συσκευές συνδέονται στο διαδίκτυο. Αυτή η ανάπτυξη της συγκεκριμένης τεχνολογίας, μεγαλώνει σε σημασία και μέσω της εξάπλωσης και των νέων τρόπων αξιοποίησης του δημιουργείται επιπρόσθετη αξία. Η ιστορία του διαδικτύου ξεκινά με το “Διαδίκτυο των Υπολογιστών”, ένα παγκόσμιο δίκτυο που παρείχε υπηρεσίες όπως ο Παγκόσμιος Ιστός. Η ραγδαία εξέλιξη που ακολούθησε μας έφερε στο “Διαδίκτυο των ανθρώπων”, δημιουργώντας έτσι νέες έννοιες όπως το Κοινωνικό δίκτυο (Web 2.0) στο οποίο παράγεται περιεχόμενο από ανθρώπους ώστε να καταναλωθεί από ανθρώπους, με σύνδεση στο διαδίκτυο [1]. Σημαντικά στοιχεία που αποδεικνύουν την ραγδαία εξέλιξη του “διαδικτύου των ανθρώπων” είναι η τεράστια απήχηση που έχουν τα κοινωνικά δίκτυα που φιλοξενούνται στο διαδίκτυο, για παράδειγμα το Facebook με τους 2.2 δισεκατομμύρια χρήστες στο τέλος του 2017 [2].

Τα όρια του Διαδικτύου, διευρύνονται καθημερινά σε συνδυασμό με την τεχνολογική πρόοδο που καθιστά επιτρεπτή την πρόσβαση σε αυτό από όλο και περισσότερα σημεία με όλο και μικρότερο κόστος. Επιπλέον, η επεξεργαστική ισχύς καθώς και η χωρητικότητα των συσκευών συνεχώς αυξάνονται αντιστρόφως ανάλογα με το μέγεθος τους. Όλο αυτό πέρα από το γεγονός ότι αλλάζει την φύση των συσκευών που οι άνθρωποι χρησιμοποιούν για να συνδέονται στο διαδίκτυο, δημιουργεί και σημαντικές νέες ευκαιρίες και εφαρμογές. Οι συσκευές αυτές απαρτίζονται από αισθητήρες και actuators ενώνοντας έτσι τον φυσικό κόσμο με τον κυβερνοχώρο. Ο συνδυασμός όλων των παραπάνω δημιουργεί μια νέα έννοια για το Διαδίκτυο, το “Διαδίκτυο των αντικειμένων” (Internet of Things - IoT).

Η έννοια του Διαδικτύου των Αντικειμένων χρονολογείται από το 1982, όταν ένας αυτόματος πωλητής συνδεόταν στο διαδίκτυο για να αναφέρει τα ποτά που απομένουν καθώς και την θερμοκρασία τους. Το 1991, ένα σύγχρονο όραμα για το IoT διατυπώθηκε από τον Mark Weiser. Ωστόσο, το 1999 ο Bill Joy μας εισήγαγε στον όρο “επικοινωνία συσκευής με συσκευή” (device to device communication) [4] και τον ίδιο χρόνο ο Kevin Ashton πρότεινε τον όρο Internet of Things για να περιγράψει ένα σύστημα διασυνδεδεμένων συσκευών [5]. Τα τελευταία χρόνια, ο όρος “Διαδίκτυο των Αντικειμένων” έχει εξαπλωθεί γρήγορα. Μέχρι το 2005 είχε αρχίσει να

εμφανίζεται σε τίτλους βιβλίων και το 2008 διεξήχθη το πρώτο επιστημονικό συνέδριο με θέμα τον συγκεκριμένο τομέα [6].

Το διαδίκτυο των αντικειμένων θα αλλάξει τα πάντα, ακόμα και εμάς. Το διαδίκτυο έχει αντίκτυπο στην εκπαίδευση, τις επικοινωνίες, την επιστήμη, την κυβέρνηση και την ανθρωπότητα. Είναι ξεκάθαρο ότι το ίντερνετ αποτελεί μια από τις πιο σημαντικές δημιουργίες σε όλη την ανθρώπινη ιστορία και πλέον με την έννοια του Internet of Things μας δίνεται η δυνατότητα να απολαμβάνουμε μία πιο “έξυπνη” ζωή σε όλο το εύρος της. Μέσω της νέας αυτής τεχνολογίας τα αντικείμενα που θα είναι συνδεδεμένα στο διαδίκτυο θα αναγνωρίζονται και θα αποκτούν συμπεριφορά νοημοσύνης. Κάτι τέτοιο θα είναι εφικτό καθώς τα αντικείμενα θα μπορούν να επικοινωνήσουν μεταξύ τους και να ανταλλάξουν δεδομένα για την κατάσταση τους και την λειτουργία τους [7]. Με την εξέλιξη της τεχνολογίας του IoT, η δοκιμή και η ανάπτυξη προϊόντων θα μας φέρει πολύ κοντά στην ανάπτυξη έξυπνων περιβαλλόντων μέχρι το 2020 [8]. Στο εγγύς μέλλον η αποθήκευση και οι υπηρεσίες επικοινωνίας θα είναι πολύ διαδεδομένες. Άνθρωποι, μηχανές, έξυπνα αντικείμενα, ο περιβάλλοντος χώρος και πλατφόρμες διασυνδεδεμένες με ασύρματους ή ενσύρματους αισθητήρες, Machine to Machine (M2M) συσκευές και RFID ετικέτες θα μπορούν να δημιουργήσουν ένα δίκτυο δικτύων (network of networks) [9].

## 1.2 Αντικείμενο της εργασίας

Αντικείμενο της παρούσας εργασία είναι η μελέτη και η αναβάθμιση παραδοσιακών εφαρμογών χρησιμοποιώντας τεχνολογίες IoT και ακολουθώντας μια προσέγγιση βασισμένη σε συνιστώσες για τον σχεδιασμό και την υλοποίηση. Πιο συγκεκριμένα, έγινε μία έρευνα στα προβλήματα που έχει να αντιμετωπίσει η βιομηχανία καθώς και στις λύσεις που παρέχει το διαδίκτυο των αντικειμένων στον συγκεκριμένο τομέα, αλλά και στις προκλήσεις που πρέπει να αντιμετωπιστούν. Στη συνέχεια, μελετήθηκαν διάφορα πρωτόκολλα επικοινωνίας των συσκευών που είναι συμβατά με το IoT και καταλήξαμε στην χρήση των CoAP και LwM2M. Για να επιτευχθεί η αναβάθμιση των παραδοσιακών εφαρμογών ακολουθήθηκε μία προσέγγιση βασισμένη σε συνιστώσες με σκοπό να μελετηθούν τα πλεονεκτήματα που μπορεί να επιφέρει η χρήση της στον συγκεκριμένο τομέα. Πιο αναλυτικά, σύμφωνα με την συγκεκριμένη προσέγγιση ένα παραδοσιακό σύστημα μπορεί να απλοποιηθεί σε επιμέρους τμήματα και κάθε τέτοιο τμήμα να αναπτυχθεί και να υλοποιηθεί ανεξάρτητα.

Η διαδικασία αυτή στην συγκεκριμένη εργασία γίνεται μέσα από την ανάπτυξη ενός IoT-συμβατού βιομηχανικού συστήματος παραγωγής liqueur. Τα επιμέρους συστήματα αυτού δεν είναι χωροταξικά συγκεντρωμένα και επικοινωνούν χρησιμοποιώντας πρωτόκολλα επιπέδου εφαρμογής που είναι κατάλληλα για IoT εφαρμογές. Το σύστημα παραγωγής που χρησιμοποιήθηκε σαν σενάριο μελέτης στην παρούσα εργασία αποτελείται από τέσσερα βασικά σημεία επεξεργασίας της πρώτης ύλης τα οποία αποτελούνται από αισθητήρες και μηχανικά μέρη και

χρησιμοποιούνται ταυτόχρονα από δύο γραμμές παραγωγής διαφορετικών προϊόντων. Κατά την διάρκεια της μελέτης αναπτύχθηκε ένα κατανεμημένο σύστημα ελέγχου του συστήματος παραγωγής ακολουθώντας την προσέγγιση που αναφέρθηκε παραπάνω και χρησιμοποιήθηκαν εξομοιωτές των μηχανικών τμημάτων του, που είχαν σχεδιαστεί και υλοποιηθεί σε προηγούμενη διπλωματική εργασία, ώστε να είναι εφικτό να γίνουν δοκιμές και έλεγχοι της σωστής λειτουργίας του συστήματος ελέγχου που υλοποιήθηκε. Η επικοινωνία μεταξύ των κατανεμημένων μικροϋπολογιστών του συστήματος βασίζεται στα πρωτόκολα CoAP και LwM2M.

Μέσα από αυτή τη μελέτη, εκτιμάται ότι η χρήση της συγκεκριμένης προσέγγισης θα προσφέρει τεράστια πλεονεκτήματα στην ανάπτυξη λογισμικού για τέτοια συστήματα, όπως για παράδειγμα λιγότερος χρόνος ανάπτυξης του λογισμικού, επαναχρησιμοποίηση ήδη υπάρχοντων συνιστωσών για την υλοποίηση ενός νέου συστήματος, αντικατάσταση συνιστωσών χωρίς την απαίτηση για διακοπή των διεργασιών του συστήματος, που περιγράφονται αναλυτικότερα στην συνέχεια.

Από την εργασία αυτή προέκυψαν διάφορα συμπεράσματα τα οποία αφορούν κυριώς την ανάπτυξη λογισμικού χρησιμοποιώντας την προσέγγιση των συνιστωσών. Πέρα από αυτό προέκυψαν διάφορα συμπεράσματα σχετικά με την χρήση του πρωτοκόλου LwM2M για την επικοινωνία μεταξύ των συσκευών καθώς και για τα πλεονεκτήματα που αυτό μπορεί να προσφέρει στον συγκεκριμένο τομέα. Εν τέλει καταλήξαμε στο συμπέρασμα ότι κάθε βιομηχανία μπορεί να επωφεληθεί από την ένταξη της στο διαδίκτυο των αντικειμένων καθώς κάτι τέτοιο θα επιφέρει αρκετές διευκολύνσεις στον τρόπο με τον οποίο γίνεται η παραγωγή αλλά και στον τροπό με τον οποίο οι χειριστές των μηχανών αλλά και οι μηχανικοί παραγωγής θα διαχειρίζονται την παραγωγή.

### 1.3 Δομή της εργασίας

Στο **κεφάλαιο 1** γίνεται μια εισαγωγή στις έννοιες του Διαδικτύου και του Διαδικτύου των αντικειμένων καθώς και στο περιεχόμενο της εργασίας και στο πρόβλημα στο οποίο έπρεπε να δωθεί λύση.

Στο **κεφάλαιο 2** ακολουθεί μια εκτενέστερη ανάλυση του Διαδικτύου των αντικειμένων καθώς και μια εξειδικευμένη ανάλυση της τεχνολογίας αυτής σε ένα συγκεκριμένο τομέα, αυτόν της βιομηχανίας. Στην συνέχεια, γίνεται μία περιγραφή της αρχιτεκτονικής REST και μία ανάλυση των διαφορών πρωτοκόλλων που χρησιμοποιούνται για την επικοινωνία μεταξύ των συσκευών στον τομέα του διαδικτύου των αντικειμένων.

Στο **κεφάλαιο 3** γίνεται μια λεπτομερής περιγραφή του πρωτοκόλου LwM2M που χρησιμοποιήθηκε για την επικοινωνία των διάφορων συσκευών που απαρτίζουν το βιομηχανικό σύστημα παραγωγής. Επίσης γίνεται μία εκτενής περιγραφή των μηχανισμών που παρέχει το

πρωτόκολλο αυτό ώστε να είναι εφικτή η επικοινωνία μεταξύ των συσκευών και ακολουθεί μία αναφορά στα πλεονεκτήματα που προσφέρει η χρήση του.

Στο **κεφάλαιο 4** αναλύεται η χρήση συνιστωσών στην ανάπτυξη λογισμικού, αναφέρονται οι βασικές οντότητες λογισμικού που ορίζονται από την συγκεκριμένη προσέγγιση, οι στόχοι της συγκεκριμένης προσέγγισης καθώς και τα πλεονεκτήματα που προσφέρει. Στην συνέχεια, ακολουθεί μία ανάλυση του OSGi framework, η οποία περιλαμβάνει την αρχιτεκτονική του, τα πλεονεκτήματα που αυτό προσφέρει καθώς και μία ανάλυση των υλοποίησεων αυτού του framework.

Στο **κεφάλαιο 5** γίνεται μια λεπτομερής περιγραφή των μηχανικών τμημάτων του συστήματος που χρησιμοποιήθηκε σαν case study καθώς και της διάταξης τους και στην συνέχεια παρατίθενται οι απαιτήσεις λειτουργίας του συστήματος παραγωγής.

Στο **κεφάλαιο 6** περιγράφεται ο σχεδιασμός του συστήματος ελέγχου των μηχανικών υποσυστημάτων του βιομηχανικού συστήματος παραγωγής. Ο σχεδιασμός του συστήματος ελέγχου ξεκινά από τον σχεδιασμό των υπηρεσιών που πρέπει να παρέχονται στο περιβάλλον, στην συνέχεια αναλύεται ο αρχιτεκτονικός σχεδιασμός του συστήματος, η δομή του κάθε σιλό ξεχωριστά και τέλος η συμπεριφορά του συστήματος ελέγχου.

Στο **κεφάλαιο 7** παρουσιάζεται η διαδικασία ανάπτυξης του απαραίτητου για τη λειτουργία του συστήματος λογισμικού, τα εργαλεία που χρησιμοποιήθηκαν, τα προβλήματα που αντιμετωπίστηκαν και οι λύσεις που επιλέχθηκαν. Επιπλέον, γίνεται αναφορά σε αρκετές τεχνολογίες που παρέχει το OSGi οι οποίες και χρησιμοποιήθηκαν στην υλοποίηση.

Στο **κεφάλαιο 8** περιγράφονται τα συμπεράσματα που προέκυψαν από την παρούσα μελέτη καθώς και μερικές προτάσεις για βελτιώσεις του συστήματος.





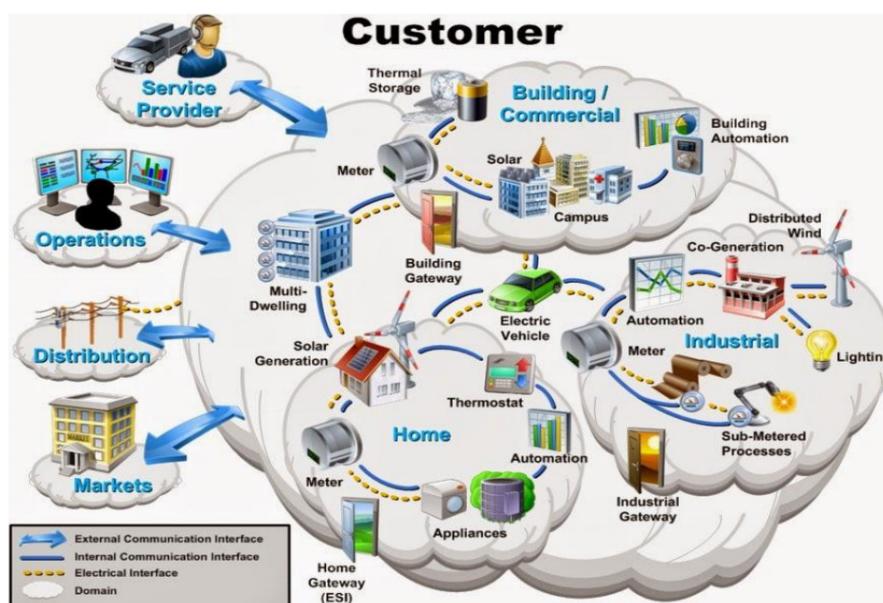
## Κεφάλαιο 2

# Το διαδίκτυο των αντικειμένων

### 2.1 Γενικά

Το διαδίκτυο των αντικειμένων στην βιομηχανία αποτελεί μία νέα πρόταση για την διασύνδεση των βιομηχανικών μηχανημάτων και των αισθητήρων μεταξύ τους, μέσω του διαδικτύου, δίνοντας έτσι στον χειριστή την δυνατότητα να χρησιμοποιήσει πληροφορίες που του παρέχονται από το κάθε αντικείμενο καθώς και να τις επεξεργαστεί ώστε να εξάγει χρήσιμα αποτελέσματα.

Πριν την συγκεκριμένη τεχνολογία, στη βιομηχανία χρησιμοποιούνταν τεχνολογίες Bluetooth και RFID, για τον έλεγχο και την παρακολούθηση βιομηχανικών εφαρμογών. Κάτι τέτοιο όμως περιοριζόταν σε μικρές αποστάσεις ελέγχου. Ο χειριστής θα έπρεπε να βρίσκεται μέσα στο εύρος των Bluetooth ή μέσα στην περιοχή που υπήρχαν οι συχνότητες των RFID. Την λύση σε αυτό το πρόβλημα έφερε η αυτοματοποίηση της βιομηχανίας μέσω του IoT. Χρησιμοποιώντας αυτή την τεχνολογία γίνεται εφικτός ο έλεγχος καθώς και η παρακολούθηση από οπουδήποτε στον κόσμο αρκεί να υπάρχει πρόσβαση στο διαδίκτυο [9]. Η εικόνα 2.1, που εμφανίζεται στο [9] δείχνει πως το IoT μπορεί να παρέχει διευκολύνσεις τόσο στην βιομηχανία όσο και σε διάφορους άλλους τομείς.



Εικόνα 2.1: Το διαδίκτυο των αντικειμένων σε διάφορους τομείς [9]

## 2.2 Η κατάσταση μέχρι σήμερα

Το διαδίκτυο των Αντικειμένων στη βιομηχανία βρίσκεται ακόμα σε πρώτο στάδιο, παρόμοιο με αυτό που το διαδίκτυο βρισκόταν στα τέλη της δεκαετίας του 1990. Ενώ η εξέλιξη του διαδικτύου τις τελευταίες δύο δεκαετίες παρέχει κάποια σημαντικά διδάγματα, είναι αρκετά ασαφές το πως αυτή η γνώση μπορεί να αξιοποιηθεί στο Διαδίκτυο των Αντικειμένων στη βιομηχανία λόγω του μοναδικού πεδίου εφαρμογής του καθώς και λόγω των απαιτήσεων που υπάρχουν. Για παράδειγμα, οι αποκρίσεις σε πραγματικό χρόνο, είναι συχνά κρίσμες στον κατασκευαστικό τομέα, στον τομέα της ενέργειας, της μεταφοράς καθώς και της υγείας. Ο πραγματικός χρόνος για το σημερινό Ίντερνετ συνήθως σημαίνει μερικά δευτερόλεπτα. Εν αντιθέσει, ο πραγματικός χρόνος σε ένα βιομηχανικό σύστημα είναι συχνά στην κλίμακα χιλιοστών του δευτερολέπτου Ο κανόνας του αντίχειρα (rule of thumb) υπαγορεύει ότι μία αλλαγή 10 φορές στην απόδοση απαιτεί μία εντελώς διαφορετική προσέγγιση, για να μην αναφερθεί μια αλλαγή 100x που θα επιφέρει στην βιομηχανία το Διαδίκτυο των Αντικειμένων [10].

Ένας άλλος σημαντικός παράγοντας πέρα από το χρόνο είναι και η αξιοπιστία. Το ίντερνετ μέχρι στιγμής ακολουθεί μία προσέγγιση “καλύτερης προσπάθειας” (best-effort approach) η οποία παρέχει αποδεκτές επιδόσεις για τον άνθρωπο και για την αλληλεπίδραση του με το διαδίκτυο. Οι απροσδόκητες δυσλειτουργίες σε έναν διακομιστή στην Google προκαλούν κάποιες καθυστερήσεις οι οποίες είναι αποδεκτές και δεν επηρεάζουν τόσο την αλληλεπίδραση του χρήστη με τις υπηρεσίες που του παρέχονται. Ωστόσο, η αποτυχία ενός ηλεκτρικού δικτύου, του συστήματος ελέγχου της εναέριας κυκλοφορίας ή ενός αυτοματοποιημένου εργοστασίου για το ίδιο χρονικό διάστημα θα είχε πολύ πιο σοβαρές συνέπειες.

Αυτοί οι παράγοντες της απόκρισης σε πραγματικό χρόνο και της αξιοπιστίας, που συνέβαλαν σε μία συντηρητική κουλτούρα μεταξύ των βιομηχανικών εταιρειών για την ενσωμάτωση των αλλαγών και των νέων τεχνολογιών, μαζί με το κόστος και την διάρκεια ζωής ενός τυπικού βιομηχανικού προϊόντος, είναι όλοι κρίσματα παράγοντες στη διαμόρφωση του τρόπου εξέλιξης του Διαδικτύου των Αντικειμένων στην Βιομηχανία.

Παρά τα εμπόδια αυτά, η υιοθέτηση του Διαδικτύου των Αντικειμένων στην βιομηχανία επιταχύνετε. Κατά τα τελευταία τρία χρόνια για παράδειγμα, ο αριθμός αισθητήρων που μεταφέρθηκαν αυξήθηκε περισσότερο από πέντε φορές. Πιο συγκεκριμένα, από 4,2 δισεκατομμύρια το 2012 αυξήθηκαν σε 23,6 δισεκατομμύρια το 2014 [11]. Μεγάλη προσοχή έχουν τραβήξει επίσης, οι προσπάθειες μεγάλων εταιρειών καθώς και κυβερνητικές πρωτοβουλίες όπως το Industrie 4.0 [10].

Η Industry 4.0 είναι μία πολυετής στρατηγική πρωτοβουλία που πρόσφατα εισήγαγε η Γερμανική κυβέρνηση. Ο στόχος της πρωτοβουλίας αυτής είναι ο μετασχηματισμός της βιομηχανικής παραγωγής μέσω της ψηφιοποίησης και της εκμετάλλευσης των νέων δυνατοτήτων που προσφέρουν οι νέες τεχνολογίες. Η βιομηχανική παραγωγή καθοδηγείται σήμερα από τον

παγκόσμιο ανταγωνισμό και την ανάγκη ταχείας προσαρμογής της παραγωγής στις συνεχώς μεταβαλλόμενες απαιτήσεις της αγοράς. Αυτές οι απαιτήσεις μπορούν να ικανοποιηθούν μόνο με ριζικές προόδους στην τρέχουσα τεχνολογία παραγωγής. Τα τεχνικά ζητήματα αυτών των απαιτήσεων αντιμετωπίζονται με την εφαρμογή των γενικών εννοιών των Cyber-Physical συστημάτων (CPS) και Διαδικτύου των Αντικειμένων στα συστήματα βιομηχανικής παραγωγής [12].

Το Διαδίκτυο των Αντικειμένων στην βιομηχανία παρουσιάζεται σαν μία επανάσταση που υπόσχεται να αλλάξει ριζικά το πρόσωπο της βιομηχανίας. Στην πραγματικότητα, πρόκειται για μία εξέλιξη που έχει τις ρίζες της σε τεχνολογίες και λειτουργίες που έχουν αναπτυχθεί πριν από περισσότερα από 15 χρόνια. Η εμφάνιση του ΠοΤ δημιούργησε τόσο ελπίδα όσο και σύγχυση σε φορείς που είναι υπεύθυνοι για την λειτουργία βιομηχανικών εγκαταστάσεων [13].

Οστόσο, το Διαδίκτυο των Αντικειμένων στην βιομηχανία έχει να αντιμετωπίσει αρκετές προκλήσεις τόσο ερευνητικές όσο και τεχνολογικές. Οι πιο σημαντικές προκλήσεις που πρέπει να αντιμετωπιστούν σύμφωνα με την [14] είναι οι εξής:

- 1. Τεχνολογική διαλειτουργικότητα:** Η διαλειτουργικότητα είναι σημαντικά πιο δύσκολη για το διαδίκτυο των αντικειμένων, καθώς δεν είναι μόνο η σύνδεση των ανθρώπων με τους ανθρώπους, αλλά η απρόσκοπη αλληλεπίδραση μεταξύ συσκευών και ατόμων με συσκευές. Αυτές οι συσκευές μπορεί να διαφέρουν όσον αφορά τις τεχνολογικές τους δυνατότητες.
- 2. Σημασιολογική διαλειτουργικότητα:** Για να επιτευχθεί πλήρης διαλειτουργικότητα, είναι απαραίτητο οι συσκευές να ερμηνεύουν σωστά τις πληροφορίες κοινής χρήσης και να ενεργούν ανάλογα. Μία τέτοιου είδους διαλειτουργικότητα καλύπτεται από την σημασιολογική πτυχή της διαλειτουργικότητας που συνήθως αναφέρεται σαν μοντέλο πληροφοριών (Information Model). Ως εκ τούτου, πρέπει να γίνουν βελτιώσεις όσον αφορά τις κατανεμημένες οντολογίες, τον σημασιολογικό ιστό (semantic web) καθώς και την ανακάλυψη συσκευών μέσω σημασιολογίας.
- 3. Ασφάλεια και προστασία προσωπικών δεδομένων:** Η ακεραιότητα των δεδομένων, η μοναδική αναγνώριση καθώς και η κρυπτογράφηση θεωρούνται βασικές προκλήσεις για το Διαδίκτυο καθώς πολλά από τα δεδομένα που κοινοποιούνται περιέχουν προσωπικές πληροφορίες. Επιπλέον, τα δικαιώματα ιδιοκτησίας δεδομένων, τα νομικά ζητήματα και τα ζητήματα ευθύνης πρέπει να αντιμετωπιστούν αναλόγως. Τέλος, πρέπει να λαμβάνονται υπόψη οι ενεργειακά αποδοτικές τεχνολογίες κρυπτογράφησης και προστασίας δεδομένων.

- 4. Έξυπνα αντικείμενα:** Πρέπει να αναπτυχθούν κυκλώματα και συσκευές εξαιρετικά χαμηλής ισχύος ικανά να αντέχουν σε σκληρά περιβάλλοντα. Επιπλέον, η παράλληλη επεξεργασία σε συστήματα πολλαπλών επεξεργαστών χαμηλής ισχύος, η προσαρμογή, η αυτόνομη συμπεριφορά με ταυτόχρονη εγγύηση της εμπιστοσύνης της ιδιωτικής ζωής και της ασφάλειας συγκαταλέγονται στις βασικές προκλήσεις όσον αφορά τις συσκευές του διαδικτύου.
- 5. Ανθεκτικότητα και αξιοπιστία:** Σε βιομηχανικά περιβάλλοντα ή σε περιπτώσεις έκτακτης ανάγκης, προσωρινές υπολειτουργίες του συστήματος δεν είναι αποδεκτές. Ως εκ τούτου, τα ζητήματα ανθεκτικότητας και αξιοπιστίας στο διαδίκτυο πρέπει να διερευνηθούν από μια συνολική άποψη του συστήματος και επιπλέον περιλαμβάνουν πτυχές όπως η διαθεσιμότητα, η ευρωστία και η ευελιξία της επικοινωνίας και του υλικού στις μεταβαλλόμενες περιβαλλοντικές συνθήκες, η αποφυγή ενιαίων σημείων αποτυχίας ή η ευρωστία δεδομένων επεξεργασίας σε αβέβαιες πληροφορίες.

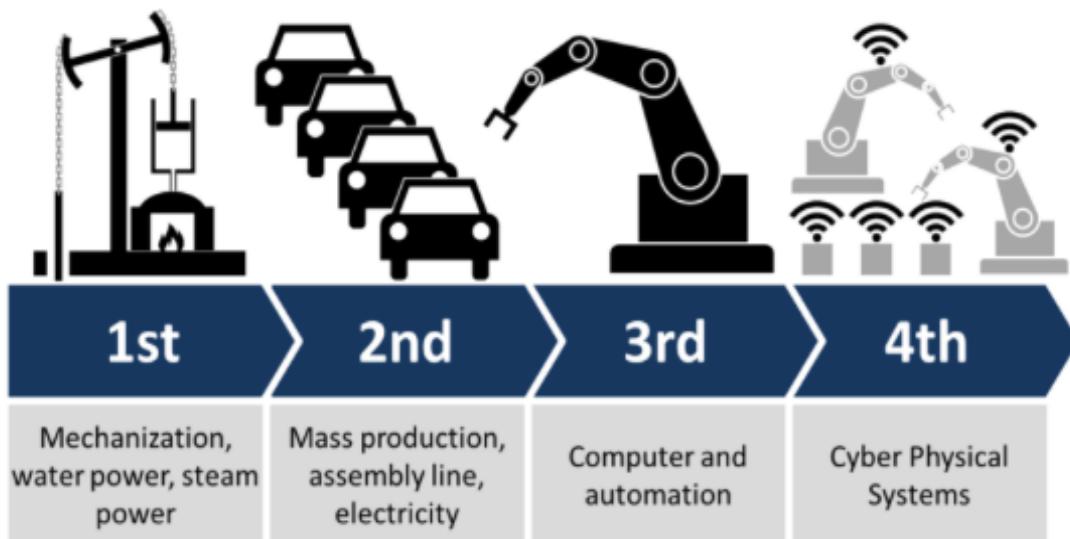
Όταν αυτές οι προκλήσεις αντιμετωπιστούν, η διασύνδεση των βιομηχανικών συστημάτων παραγωγής με διάφορα άλλα συστήματα, συσκευές και καταναλωτές μέσω του διαδικτύου θα δημιουργήσει νέες δυνατότητες όπως αναφέρονται στο [15]:

- 1. Μαζική εξατομίκευση - Δημιουργία εξατομικευμένων προϊόντων με μικρούς χρόνους παράδοσης:** Πάντα υπήρχε ένας συμβιβασμός που έπρεπε να κάνει ένας κατασκευαστής μεταξύ εξατομίκευσης ενός προϊόντος και μαζικής παραγωγής του. Με την εφαρμογή του Διαδικτύου των Αντικειμένων στην βιομηχανία ένα δικτυωμένο - έξυπνο εργοστάσιο καθιστά εφικτή την εξατομίκευση των προϊόντων σε κλίμακα μαζικής παραγωγής.
- 2. Συνεργασία ανθρώπου - μηχανής:** Η συνεργασία ανθρώπου - μηχανής μπορεί να διαχωριστεί σε φορητές διεπαφές ανθρώπου - μηχανής (mobile Human-Machine Interfaces, HMI) και συνεργατικά συστήματα ανθρώπου - μηχανής (Human-Machine Collaborative Systems, HMCS). Οι HMI τεχνολογίες, για παράδειγμα κινητά τηλέφωνα, ταμπλέτες, ηλεκτρονικά που φοριούνται (wearables) σε συνδυασμό με πρόσβαση στο διαδίκτυο θα αλλάξουν ριζικά τον τρόπο με τον οποίο οι χειριστές των μηχανών και οι μηχανικοί θα παρακολουθούν και θα χειρίζονται την παραγωγή. Η φυσική παρουσία του προσωπικού στο χώρο διαχείρισης δεν θα είναι απαραίτητη, γεγονός που αυξάνει την ασφάλεια και οι ικανότητες του χειριστή διευρύνονται με την χρήση νέων τεχνολογιών όπως η εικονική πραγματικότητα για την καλύτερη παρακολούθηση της παραγωγής και τον εντοπισμό των προβλημάτων. Οι HMCS τεχνολογίες στοχεύουν στην κατασκευή ευέλικτων ρομποτικών συστημάτων για κατασκευαστές μικρής κλίμακας οι οποίοι τροποποιούν σχνά τις γραμμές παραγωγής τους. Η δυνατότητα ανάλυσης αλυσίδας ενεργειών (Action

Sequence analysis) θα δίνει τη δυνατότητα στα ρομποτικά συστήματα αυτά να “μαθαίνουν” δυναμικά νέες εργασίες όπως η συναρμολόγηση κάποιου νέου προϊόντος με επαναληπτικές μεθόδους αντί για προγραμματισμό, πράγμα που μπορεί εύκολα να γίνει από ένα εργαζόμενο χωρίς αυτός να έχει ιδιαίτερες γνώσεις προγραμματισμού.

**3. Ενοποίηση της παγκόσμιας αλυσίδας εφοδιασμού:** Η παγκοσμιοποίηση της αγοράς και των αλυσίδων εφοδιασμού, παρόλο που είναι εξαιρετικά κερδοφόρα για τον κατασκευαστή, δημιουργεί ορισμένα προβλήματα. Οι κατασκευαστές οφείλουν να λαμβάνουν υπ' όψη τους πολλούς παράγοντες όπως οι τρέχουσες τιμές, διαθεσιμότητες, χρόνοι παραδοσης, ποιοτικές προδιαγραφές και αποθέματα πρώτων υλών, εργαλείων και μηχανών. Στα πλαίσια του Διαδικτύου των Αντικειμένων στην βιομηχανία αναπτύσσονται εργαλεία τα οποία παρέχουν ενοποιημένη πρόσβαση σε πραγματικό χρόνο στις πληροφορίες αυτές. Κάτι τέτοιο ενισχύει την παραγωγικότητα, την ασφάλεια στο χώρο εργασίας και την ποιότητα της παραγωγής.

Η εφαρμογή της τεχνολογίας του Διαδικτύου των Αντικειμένων σε τομείς της βιομηχανίας αναμένεται να έχει έντονο αντίκτυπο σε όλη την διαδικασία της παραγωγής ώστε τελικά να οδηγήσει στην επονομαζόμενη τέταρτη βιομηχανική επανάσταση. Η εικόνα 2.2 δείχνει τις φάσεις που πέρασε η βιομηχανική επανάσταση.



**Εικόνα 2.2:** Οι φάσεις των βιομηχανικών επαναστάσεων [3]

Οι πιο αισιόδοξες προβλέψεις κάνουν λόγο για παραγωγή επιπρόσθετης αξίας παγκοσμίως από την εφαρμογή τους Διαδικτύου των Αντικειμένων στην βιομηχανίας της τάξης των 15 τρισεκατομμυρίων δολαρίων μέχρι το 2030 [16].

## 2.3 Πρότυπα και πρωτόκολλα IoT

Στο όραμα του Διαδικτύου των Αντικειμένων, το διαδίκτυο εξαπλώνεται πέρα από τον πυρήνα του. Αυτή η τεράστια ανάπτυξη φέρνει τόσο συναρπαστικές δυνατότητες όσο και προκλήσεις στο Διαδίκτυο, όπως το πως θα ενσωματωθούν απρόσκοπτα οι διάφορες συσκευές και οι σένσορες στον παγκόσμιο ιστό. Ένας τρόπος για την ομαλή ενοποίηση του κυβερνο-κόσμου και του φυσικού κόσμου είναι η επαναχρησιμοποίηση των υφιστάμενων τεχνολογιών και προτύπων στον παγκόσμιο ιστό, όσο το δυνατόν περισσότερο. Αυτή την τάση έρευνας την αντιμετωπίζει το Διαδίκτυο των Αντικειμένων ως το Διαδίκτυο των πραγμάτων (Web of Things, WoT). Στο WoT, οι συσκευές δεν έχουν απλά μία IP ώστε να είναι συνδεδεμένες στο διαδίκτυο, αλλά έχουν την δυνατότητα να μιλούν και την ίδια γλώσσα και έτσι είναι σε θέση να επικοινωνούν και να αλληλεπιδρούν ελεύθερα στον Παγκόσμιο Ιστό. Προκειμένου να υλοποιηθεί αυτό το όραμα, πρέπει να διεξαχθεί εκ νέου σχεδιασμός και βελτιώσεις στην κωδικοποίηση ωφέλιμου φορτίου και στα πρωτόκολλα εφαρμογής, ώστε να ικανοποιηθούν οι ειδικές απαιτήσεις των εφαρμογών μηχανής προς μηχανή (Machine to Machine, M2M) σε περιορισμένα περιβάλλοντα του διαδικτύου των αντικειμένων [17].

Ένας τρόπος να εκπληρωθούν τα παραπάνω ζητούμενα είναι η ευρεία τοποθέτηση κοινής αρχιτεκτονικής, τόσο σε επίπεδο δικτύου, το οποίο επιτυγχάνεται με τα 6LoWPAN πρότυπα [19], όσο και σε επίπεδο εφαρμογών μέσω του WoT [20] που αναφέρθηκε παραπάνω. Μία τέτοια προσέγγιση επιτυγχάνει την χαλαρή ζεύξη (loose coupling) μεταξύ των τμημάτων που αποτελούν μια κατανεμημένη εφαρμογή, ιδιότητα - κλειδί για την επίτευξη της ζητούμενης διαλειτουργικότητας [18].

### 2.3.1 Η αρχιτεκτονική REST

Η αρχιτεκτονική αρχή που βρίσκεται στην καρδιά του διαδικτύου, δηλαδή η Representative State Transfer (REST) όπως ορίστηκε από τον Roy Fielding [31], μοιράζεται έναν παρόμοιο στόχο με τις περισσότερες γνωστές τεχνικές ενοποίησης όπως οι υπηρεσίες WS-\* Web services (SOAP, WSDL, κτλ), ο οποίος είναι η αύξηση της διαλειτουργικότητας για μια χαλαρότερη ζεύξη μεταξύ τμημάτων των κατανεμημένων εφαρμογών. Ωστόσο, ο στόχος του REST είναι να το επιτύχει αυτό με έναν πιο ελαφρύ και απλούστερο τρόπο και εστιάζει σε πόρους και όχι σε λειτουργίες, όπως συμβαίνει με τις υπηρεσίες WS-\* Web. Συγκεκριμένα, το REST χρησιμοποιεί το Διαδίκτυο ως μια πλατφόρμα εφαρμογών και εκμεταλλεύεται πλήρως όλες τις λειτουργίες που είναι εγγενείς στο HTTP πρωτόκολλο, όπως ο έλεγχος ταυτότητας, η εξουσιοδότηση, η κρυπτογράφηση, η συμπίεση και η προσωρινή αποθήκευση. Με αυτόν τον τρόπο το REST “φέρνει” υπηρεσίες μέσα στον πρόγραμμα περιήγησης. Οι πόροι μπορούν να συνδεθούν και να ανατεθούν σε σελιδοδείκτες και τα αποτελέσματα να είναι ορατά με οποιοδήποτε πρόγραμμα περιήγησης του ιστού χωρίς να χρειάζεται να παράγουν πολύπλοκο πηγαίο κώδικα από αρχεία WSDL για να μπορούν να αλληλεπιδρούν με μία υπηρεσία [18].

Για να το πετύχει αυτό η REST αρχιτεκτονική απαρτίζεται από δύο βασικούς κανόνες:

- Το μοντέλο εφαρμογής μετασχηματίζεται και αντί να έχει την λειτουργικότητα σαν κύριο άξονα έχει τα δεδομένα. Αυτό σημαίνει ότι κάθε τι που προσφέρει υπηρεσίες γίνεται πλέον ένας πόρος (resource), για παράδειγμα ένας αισθητήρας θερμότητας είναι ένας πόρος, που μπορεί να αναγνωριστεί μέσω ενός συγκεκριμένου URI.
- Οι τέσσερις κύριες λειτουργίες που παρέχονται από το πρωτόκολλο HTTP (PUT, POST, GET, DELETE) είναι οι μόνες διαθέσιμες λειτουργίες που μπορεί να έχει ένας πόρος. Έτσι ορίζεται μια ομοιόμορφη διεπαφή με γνωστή και κοινή σημασιολογία.

Αυτά τα πλεονεκτήματα εξηγούν κυρίως γιατί οι υπηρεσίες που προσφέρει η REST αρχιτεκτονική αποτελούν την τεχνολογική βάση για ένα αυξανόμενο αριθμό υπηρεσιών Web 2.0 όπως αυτές που προσφέρονται από το Flickr, το Twitter, το Google και το Amazon. Παραδοσιακά, η REST αρχιτεκτονική έχει χρησιμοποιηθεί για την ενσωμάτωση διάφορων υπηρεσιών που παρέχονται σε ιστοσελίδες. Ωστόσο, επειδή η αρχιτεκτονική αυτή είναι αρκετά ελαφριά γίνεται άμεσα υποψήφια για ενσωματωμένες συσκευές με περιορισμένους πόρους ώστε να προσφέρουν υπηρεσίες στον παγκόσμιο ιστό. Εφόσον, τέτοιες συσκευές προσφέρουν συνήθως απλές και ατομικές λειτουργίες η μοντελοποίησή τους χρησιμοποιώντας το REST είναι συχνά απλή [18].

Έχει σημασία να επισημανθεί πως το REST δεν περιγράφει κάποιο συγκεκριμένο πρωτόκολλο αλλά έχει μόνο ένα στυλ σχεδιασμού πρωτοκόλλων επικοινωνίας.

### 2.3.2 Τα πρωτόκολλα HTTP - CoAP

Επί του παρόντος, η πρόσβαση στο Διαδίκτυο απαιτεί πρωτόκολλα εφαρμογών μέσω TCP / IP ή UDP / IP. Ένα τέτοιο πρωτόκολλο εφαρμογής αποτελεί το Hyper Text Transfer Protocol (HTTP), το οποίο έχει τυποποιηθεί στο IETF, και έχει εφαρμοστεί για την επικοινωνία στο διαδίκτυο. Ωστόσο, όταν εφαρμόζεται αυτό το πρωτόκολλο στην επικοινωνία μεταξύ συσκευών στο IoT, κατά την οποία μεταφέρεται ένας τεράστιος αριθμός μικροσκοπικών μπλοκ δεδομένων, το overhead που προστίθεται σε συνδυασμό με τις χαμηλές επιδόσεις αποτελούν ένα σημαντικό πρόβλημα. Επιπλέον, η διεύθυνση IP εξαρτάται από την φυσική τοποθεσία της συσκευής, γεγονός που δημιουργεί το πρόβλημα της πολυπλοκότητας του ελέγχου δικτύου.

Επιπλέον ένα μέρος της διαδρομής του δικτύου που θα ακολουθούν τα δεδομένα θα είναι μέσα σε δίκτυα χαμηλής ισχύος και με υψηλό ποσοστό απώλειας δεδομένων (Low power and Lossy Networks, LLN) [21]. Έτσι λοιπόν ένα κατάλληλο για την M2M επικοινωνία πρωτόκολλο πρέπει να μεγιστοποιεί την εξοικονόμηση ενέργειας, να είναι αποδοτικό όταν χρησιμοποιείται σε LLN's και να είναι σχετικά ελαφρύ ώστε να εκτελείται σε μικρών δυνατοτήτων (μνήμης και

επεξεργαστικής ισχύος) ενσωματωμένα συστήματα. Επιπρόσθετα το HTTP δε διαθέτει κάποιο μηχανισμό για αποστολή multi cast και εντοπισμού resources.

Τα προβλήματα αυτά έρχεται να λύσει το πρωτόκολλο Constrained Application Protocol (CoAP). Το CoAP είναι ένα πρωτόκολλο σύγχρονης αίτησης / απόκρισης επιπέδου εφαρμογής που σχεδιάστηκε από την Internet Engineering Task Force (IETF) για να στοχεύσει σε συσκευές περιορισμένων πόρων. Σχεδιάστηκε χρησιμοποιώντας ένα υποσύνολο των μεθόδων του πρωτοκόλλου HTTP που το καθιστούν διαλειτουργικό με το HTTP [23].

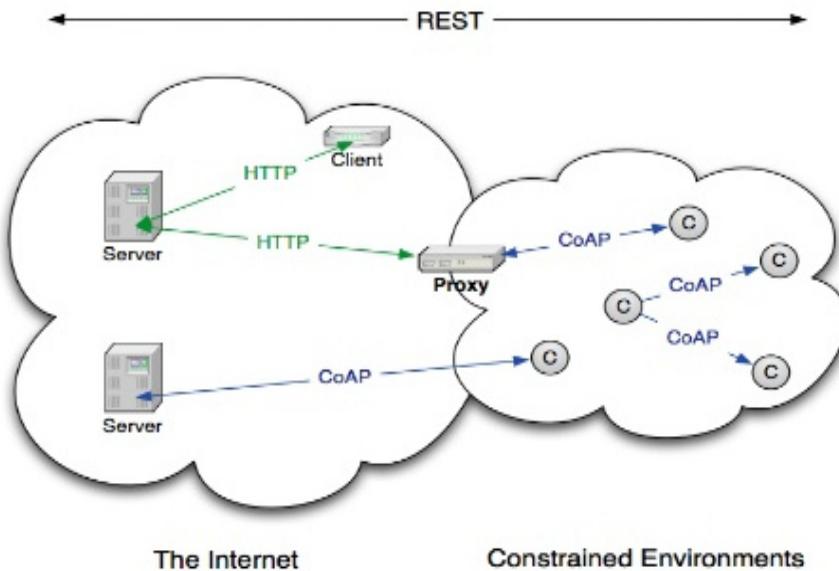
Το CoAP λειτουργεί πάνω από UDP έτοις ώστε να είναι όλη η υλοποίησή του ελαφριά. Χρησιμοποιεί τις εντολές του HTTP, GET, POST, PUT και DELETE για την παροχή αλληλεπιδράσεων προσανατολισμένων σε πόρους σε μια αρχιτεκτονική πελάτη - διακομιστή (client - server). Το CoAP είναι ένα πρωτόκολλο αίτησης / απόκρισης που χρησιμοποιεί τόσο σύγχρονες όσο και ασύγχρονες απαντήσεις. Ο λόγος για το σχεδιασμό ενός πρωτοκόλλου επιπέδου εφαρμογής που βασίζεται σε UDP για την διαχείριση των πόρων είναι η κατάργηση του overhead που προσθέτει το TCP και η μείωση των απαιτήσεων σχετικά με το εύρος ζώνης [24]. Χρησιμοποιώντας το αναξιόπιστο UDP, το CoAP υλοποίησε τους δικούς του μηχανισμούς για την επίτευξη αξιοπιστίας. Δύο bit στην κεφαλίδα κάθε πακέτου αναφέρουν τον τύπο του μηνύματος και το απαιτούμενο επίπεδο ποιότητας υπηρεσίας (QoS). Υπάρχουν τέσσερις τύποι μηνυμάτων:

- 1. Επιβεβαιώσιμο:** Ένα μήνυμα αίτησης που απαιτεί επιβεβαίωση (ACK). Η απόκριση μπορεί να αποσταλεί είτε συγχρόνως (εντός του ACK) είτε εάν χρειάζεται περισσότερος υπολογιστικός χρόνος, μπορεί να σταλεί ασύγχρονα με ένα ξεχωριστό μήνυμα.
- 2. Μη επιβεβαιώσιμο:** Ένα μήνυμα που δεν απαιτεί επιβεβαίωση.
- 3. Αναγνώρισης:** Επιβεβαιώνει τη λήψη ενός επιβεβαίωσιμου μηνύματος.
- 4. Επαναφοράς (Reset):** Επιβεβαιώνει την λήψη ενός μηνύματος που δεν είναι δυνατή η επεξεργασία του.

Υπάρχει επίσης ένας απλός μηχανισμός μετάδοσης Stop - and - Wait για επιβεβαιωμένα μηνύματα και ένα πεδίο κεφαλίδας 16 bit σε κάθε πακέτο CoAP ονομάζεται message ID, το οποίο είναι μοναδικό και χρησιμοποιείται για την ανίχνευση διπλότυπων.

Παρόλο που το CoAP δημιουργήθηκε για τις επικοινωνίες IoT και M2M, δεν περιλαμβάνει ενσωματωμένα χαρακτηριστικά ασφαλείας. Το πρωτόκολλο που προτείνεται για διασφάλιση των συναλλαγών μέσω του CoAP είναι το Datagram Transport Layer Security (DTLS) [25].

Η εικόνα 2.3 αναπαριστά την αρχιτεκτονική του πρωτοκόλλου CoAP:



**Εικόνα 2.3:** Η αρχιτεκτονική CoAP [29]

### 2.3.3 Το πρωτόκολλο MQTT

Το Message Queue Telemetry Transport (MQTT) πρωτόκολλο κυκλοφόρησε από την IBM και στοχεύει σε ελαφρές επικοινωνίες μεταξύ συσκευών (M2M). Πρόκειται για ένα ασύγχρονο πρωτόκολλο δημοσίευσης / εγγραφής (publish / subscribe) που εκτελείται στην κορυφή του TCP. Τα πρωτόκολλα δημοσίευσης / εγγραφής πληρούν καλύτερα τις απαιτήσεις του IoT από τα πρωτόκολλα αιτήματος / απόκρισης (request / response), δεδομένου ότι οι clients δεν χρειάζεται να ζητούν ενημερώσεις και έτσι το εύρος ζώνης του δικτύου μειώνεται και η ανάγκη χρήσης υπολογιστικών πόρων μειώνεται.

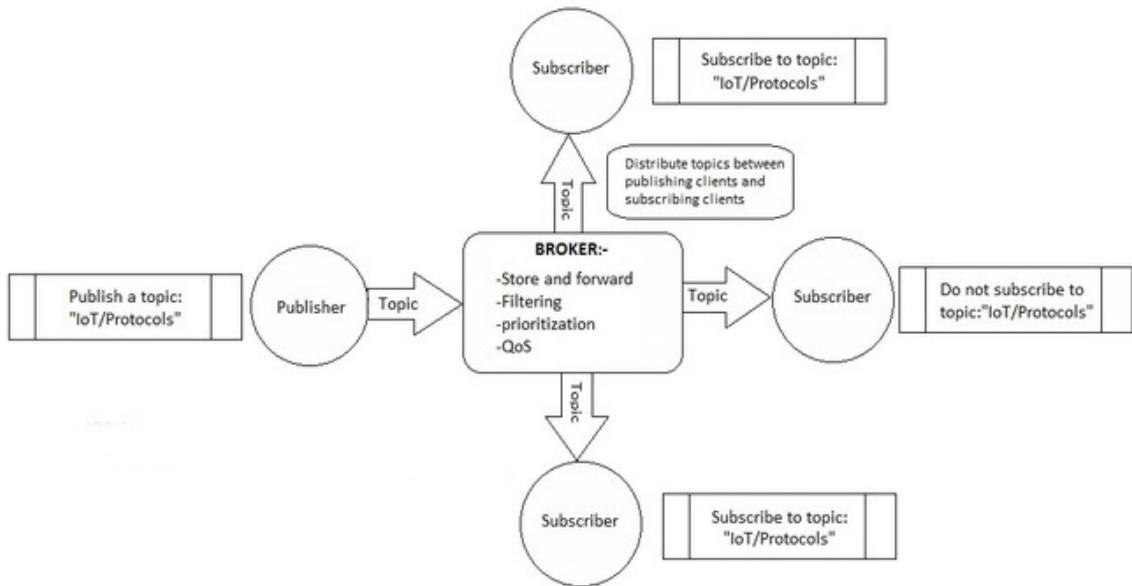
Στο MQTT υπάρχει ένας broker (διακομιστής) που περιέχει ζητήματα. Κάθε πελάτης (client) μπορεί να είναι ένας εκδότης (publisher) που στέλνει πληροφορίες στον broker σχετικά με ένα συγκεκριμένο ζήτημα ή / και έναν συνδρομητή (subscriber) που λαμβάνει αυτόματα μηνύματα κάθε φορά που υφίσταται μία αλλαγή σε ένα ζήτημα στο οποίο έχει εγγραφεί. Το συγκεκριμένο πρωτόκολλο έχει σχεδιαστεί για να χρησιμοποιεί το εύρος ζώνης και τη χρήση μπαταρίας με φειδώ γι' αυτό για παράδειγμα αυτή τη στιγμή χρησιμοποιείται από το Facebook Messenger [26].

Το MQTT εξασφαλίζει αξιοπιστία παρέχοντας την επιλογή τριών επιπέδων QoS:

1. **Fire and forget:** Ένα μήνυμα αποστέλλεται μία φορά και δεν απαιτείται επιβεβαίωση.
2. **Delivered at least one:** Ένα μήνυμα αποστέλλεται τουλάχιστον μία φορά και απαιτείται επιβεβαίωση.

- 3. Delivered exactly one:** Ένας μηχανισμός χειραψίας τεσσάρων κατευθύνσεων χρησιμοποιείται για να εξασφαλίσει ότι το μήνυμα παραδίδεται ακριβώς μία φορά.

Αξίζει να σημειωθεί ότι παρόλο που το MQTT τρέχει πάνω από το TCP, έχει σχεδιαστεί ώστε να προσθέτει χαμηλό overhead σε σύγκριση με άλλες εφαρμογές που βασίζονται στο TCP [27]. Η εικόνα 2.4 δείχνει την αρχιτεκτονική του MQTT πρωτοκόλλου.



**Εικόνα 2.4:** Η αρχιτεκτονική MQTT [30]

Ενδεικτικά, άλλα διαδεδομένα πρωτόκολλα είναι τα XMPP, DDS, SOAP και WebSocket.

## Κεφάλαιο 3

# Το πρωτόκολλο LwM2M

### 3.1 Γενικά

To Lightweight M2M (**LwM2M**) αποτελεί ένα νέο, αναδυόμενο πρότυπο που αναπτύχθηκε από την Open Mobile Alliance (OMA) με μεγάλη συνάφεια με την αναπτυσσόμενη M2M βιομηχανία και το Διαδίκτυο των Αντικειμένων στο σύνολο του. Αυτό το βιομηχανικό πρότυπο παρέχει μέσα για την απομακρυσμένη διαχείριση μιας ευρείας γκάμας απομακρυσμένων ενσωματωμένων συσκευών και συνδεδεμένων συσκευών στο αναδυόμενο Διαδίκτυο των Αντικειμένων για την πραγματοποίηση εξ αποστάσεως εξυπηρέτησης και διαχείρισης απομακρυσμένων εφαρμογών [28].

Η αγορά των διασυνδεδεμένων συσκευών αυξάνεται ραγδαία. Παρόλο που υπάρχουν βιομηχανικά πρότυπα διαθέσιμα που ικανοποιούν τις απαιτήσεις απομακρυσμένης διαχείρισης, για παράδειγμα σταθερών ευρυζωνικών δρομολογητών και smartphones, αυτά τα καθιερωμένα πρότυπα δεν είναι ιδιαίτερα χρήσιμα για την απομακρυσμένη διαχείριση μιας μεγάλης και αναπτυσσόμενης κατηγορίας συνδεδεμένων συσκευών: εκείνων με περιορισμένο εύρος ζώνης δικτύου, υπολογιστικής ισχύος και μνήμης, εκείνων που εξαρτώνται από περιορισμένη διάρκεια ζωής της μπαταρίας και εκείνων που είναι βιώσιμα μόνο με πολύ χαμηλό κόστος παραγωγής. Ως εκ τούτου, αναλήφθηκε αυτή η νέα προσπάθεια για τη δημιουργία ενός μηχανισμού που να καλύπτει επίσης τις ανάγκες των “περιορισμένων” συσκευών. Η βιομηχανία αναζητά έναν απλό, χαμηλού κόστους απομακρυσμένο μηχανισμό διαχείρισης και ενεργοποίησης υπηρεσιών, ο οποίος να περιλαμβάνει σύγχρονες αρχιτεκτονικές αρχές (σύμφωνα με τα πρότυπα του Διαδικτύου) ενώ παράλληλα να λειτουργεί μέσω ασύρματων συνδέσεων και να είναι κατάλληλος για το σκοπό αυτό λόγω των χαμηλών απαιτήσεων σε πόρους. Τη λύση έρχεται να δώσει το πρωτόκολλο LwM2M.

Από τεχνικής άποψης, το LwM2M πρόκειται για ένα πρωτόκολλο επικοινωνίας για χρήση μεταξύ του λογισμικού του client μιας M2M συσκευής και του λογισμικού του server σε μια πλατφόρμα διαχείρισης και ενεργοποίησης υπηρεσιών M2M. Το πρωτόκολλο LwM2M που χρησιμοποιείται για την απομακρυσμένη διαχείριση M2M συσκευών και τις σχετικές δυνατότητες εξυπηρέτησης έχει τέσσερα εξαιρετικά χαρακτηριστικά [28]:

1. Διαθέτει σύγχρονο αρχιτεκτονικό σχεδιασμό βασισμένο στο REST που απευθύνεται σε προγραμματιστές.

2. Ορίζει ένα μοντέλο πόρων και δεδομένων.
3. Έχει σχεδιαστεί με γνώμονα την απόδοση και τους περιορισμένους πόρους των συσκευών M2M.
4. Επαναχρησιμοποιείται και βασίζεται σε ένα αποδοτικό πρότυπο μεταφοράς δεδομένων, το CoAP που περιγράφηκε παραπάνω.

Η διαθεσιμότητα αυτού του ανοικτού κώδικα, τυποποιημένου μηχανισμού απομακρυσμένης διαχείρισης δημιουργεί τις ακόλουθες ευκαιρίες και επιχειρηματικά οφέλη για τους διάφορους “παίκτες” της βιομηχανίας M2M [28]:

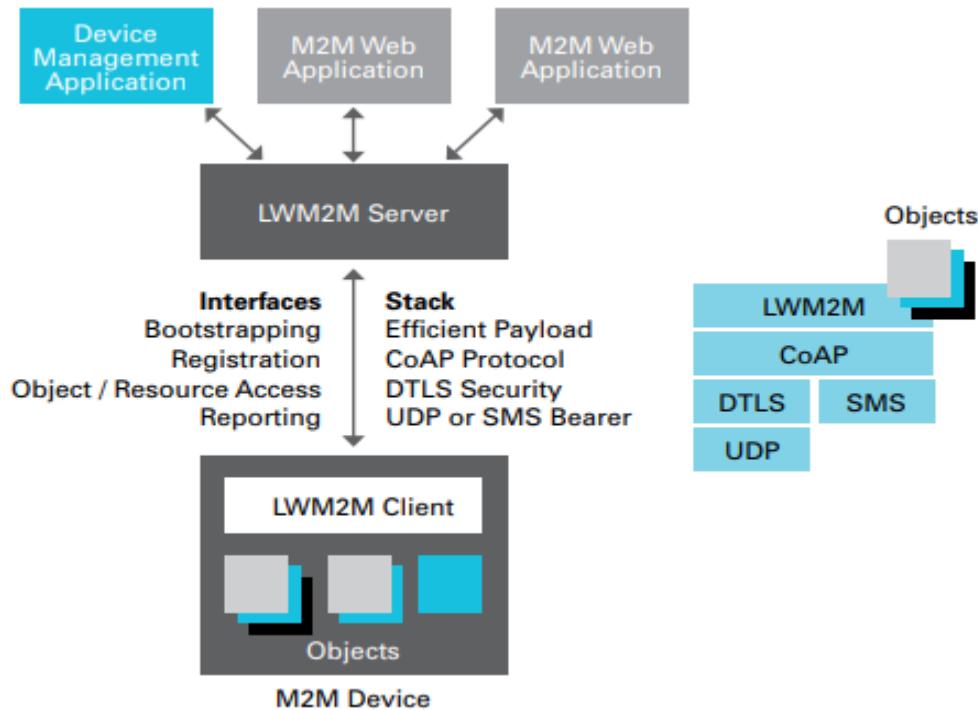
- Θα μειώσει τον βαθμό κατακερματισμού (fragmentation) στον τομέα της απομακρυσμένης διαχείρισης συσκευών M2M, επιτρέποντας έτσι περισσότερες plug-n-play λύσεις μεταξύ μιας αυξανόμενης ποικιλίας συσκευών M2M και των πλατφορμών διαχείρισης τους.
- Δεδομένου του σχεδιασμού του για την κάλυψη των περιορισμένων συσκευών, μπορεί να λειτουργήσει ως ένας παράγοντας που διευκολύνει την ανάπτυξη της αγοράς M2M σε διάφορους τομείς, από την διαχείριση μίας “έξυπνης” πόλης μέχρι την διαχείριση ενέργειας και την παρακολούθηση θέσης. Πιο συγκεκριμένα θα ωφεληθούν τομείς όπου οι συσκευές πρέπει να έχουν χαμηλό κόστος για να καταστήσουν δυνατή την ανάπτυξη βιώσιμων επιχειρηματικών μοντέλων.
- Αυτό το νέο πρότυπο θα μπορούσε να βελτιώσει σημαντικά τον χρόνο time-to-market και την διαχειρισμότητα των συσκευών, παρέχοντας για πρώτη φορά μια λύση που μπορεί να χρησιμοποιηθεί τόσο για την διαχείριση συσκευών όσο και για εφαρμογές δεδομένων και υπηρεσιών, ανεξάρτητα από το πως φιλοξενούνται τα στοιχεία του συστήματος.
- Ως πρωτόκολλο απομακρυσμένης διαχείρισης μεταξύ συσκευών και πλατφορμών M2M οδηγεί σε αποσύνδεση και των δύο πλευρών, επιτρέποντας έτσι μεγαλύτερη ανεξάρτητη καινοτομία των συσκευών M2M και των πλατφορμών M2M.

### 3.2 Περιγραφή των μηχανισμών του LwM2M

Το LwM2M είναι βασισμένο και σχεδιασμένο πάνω στο CoAP και χρησιμοποιεί προαιρετικά το πρωτόκολλο DTLS για την ανάπτυξη εφαρμογών που απαιτούν ασφαλή επικοινωνία. Στην εικόνα 3.1 δίνεται μία οπτικοποίηση της αρχιτεκτονικής του LwM2M πρωτοκόλλου. Μια προς διαχείριση συσκευή αποτελεί έναν LwM2M client ενώ η πλατφόρμα διαχείρισης της συσκευής είναι ένας LwM2M server. Για την επικοινωνία μεταξύ αυτών των δύο το LwM2M ορίζει τέσσερα interfaces που είναι τα εξής:

1. **Bootstrap:** ορίζει την διαδικασία που απαιτείται ώστε ένας LwM2M client να λάβει τις απαιτούμενες πληροφορίες για τον server στον οποίο πρέπει να συνδεθεί (register).
2. **Client Registration:** ορίζει την απαιτούμενη διαδικασία κατά την οποία ένας client εγγράφεται ή απεγγράφεται από έναν LwM2M server.
3. **Device Management and Service Enablement:** περιγράφει τον τρόπο με τον οποίο ένας LwM2M server προσπελαύνει τα LwM2M objects και τα LwM2M resources που έχει ένας LwM2M client.
4. **Information Reporting:** χρησιμοποιείται από τον server για την παρακολούθηση αλλαγών σε κάποιο resource του client.

Ο client αλληλεπιδρά με τον server ακολουθώντας τις αρχές της REST αρχιτεκτονικής. Κάθε υπηρεσία ή πληροφορία που διαθέτει ένας LwM2M client στο περιβάλλον του αποτελεί ένα LwM2M resource. Κάθε resource μεταξύ των υπόλοιπων ιδιοτήτων του, χαρακτηρίζεται και από έναν τύπο (type). Ένα resource μπορεί να είναι τύπου String, Float, Integer, Boolean, Opaque, Time, ObjLink. Τα resources είναι λογικά ομαδοποιημένα σε LwM2M Objects. Ένας LwM2M client μπορεί να έχει οποιοδήποτε αριθμό από resources τα οποία όμως ανήκουν σε κάποιο object. Κάθε object μπορεί να έχει ένα μοναδικό ή πολλαπλά στιγμιότυπα (object instances).



**Εικόνα 3.1:** Η LwM2M αρχιτεκτονική [28]

Κάθε object/ object instance /resource ενός συγκεκριμένου object αναγνωρίζεται μονοσήμαντα από ένα μοναδικό object id/ object instance id / resource id αντίστοιχα. Η αναφορά σε

κάποιο resource γίνεται μέσα από ένα path με τη μορφή “/Object ID/Object Instance ID/Resource ID”.

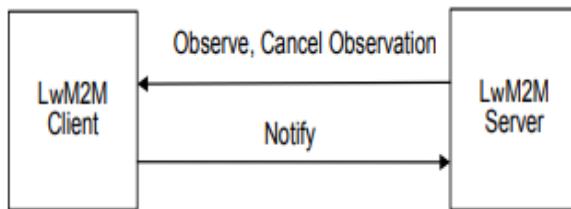
Ο LwM2M server όταν θέλει να ξεκινήσει μία αλληλεπίδραση με έναν LwM2M client του αποστέλλει ένα αίτημα. Ένα τέτοιο αίτημα απαρτίζεται από το path του resource στο οποίο αναφέρεται και την μέθοδο την οποία αιτείται να εφαρμοστεί στο resource αυτό. Το Device Management & Service Enablement Interface (DM&SE) ορίζει το σύνολο των μεθόδων που **πρέπει** να υποστηρίζει τόσο ένας LwM2M client όσο και ο LwM2M server και μπορούν να εφαρμοστούν σε ένα resource [32]. Αυτές είναι:

1. **CREATE:** Δημιουργεί instance
2. **READ:** Ζητείται να επιστραφεί η τιμή ενός resource
3. **WRITE:** Ζητείται η αλλαγή της τιμής ενός resource
4. **DELETE:** Διαγραφεί instance
5. **EXECUTE:** Ζητείται η πραγματοποίηση μίας ενέργειας
6. **WRITE-ATTRIBUTES:** Χρησιμοποιείται για την αλλαγή των τιμών των attributes
7. **DISCOVER:** Χρησιμοποιείται για την ανακάλυψη resources και attributes

Αφού ο LwM2M server αποστέλλει το αίτημα ο LwM2M client στέλνει πίσω μία απάντηση η οποία περιέχει έναν κωδικό απάντησης και σε κάποιες περιπτώσεις ένα περιεχόμενο. Για παράδειγμα μια απάντηση σε ένα αίτημα READ μπορεί να έχει για κωδικό απάντησης το “2.04 Changed” και για περιεχόμενο την τιμή του resource το οποίο αφορούσε το αίτημα. Ανάλογα τον τύπο του αρχικού αιτήματος, ο κωδικός απάντησης έχει διαφορετική σημασία [32].

Μεγάλη σημασία έχει επίσης και ο μηχανισμός observe του CoAP, ο οποίος χρησιμοποιείται από το LwM2M μέσω του Information Reporting Interface. Αυτό το interface χρησιμοποιείται από έναν LwM2M server ώστε να παρατηρούνται οι αλλαγές που γίνονται σε ένα resource ενός καταχωρημένου LwM2M client. Κάθε φορά που αλλάζει η τιμή του resource αποστέλλεται μία ειδοποίηση στον LwM2M server μαζί με τη νέα τιμή του resource. Αυτή η σχέση observe-notify ξεκινά με την αποστολή ενός αιτήματος τύπου “observe” από τον server στον client για ένα object, object instance ή resource που είναι παρακολουθήσιμο (observable). Μετά από την επιτυχή εγκαθίδρυση μιας τέτοιας σχέσης, ο LwM2M client αποστέλλει ένα μήνυμα τύπου “notify” στον server. Το μήνυμα αυτό περιέχει τη νέα τιμή του resource που παρακολουθείται. Η εικόνα 3.2 δείχνει το μοντέλο λειτουργίας για το συγκεκριμένο interface:

Αξίζει να σημειωθεί ότι ο οργανισμός OMA μαζί με τις προδιαγραφές του LwM2M δημοσιοποιεί και ένα μητρώο (registry) το οποίο περιέχει όλα τα LwM2M objects τα οποία έχουν



**Εικόνα 3.2:** LwM2M Information Reporting Interface [32]

καταχωρηθεί σε αυτό από τον οργανισμό καθώς και από τρίτους. Αυτό το μητρώο υλοποιήθηκε με στόχο ένας LwM2M client να μπορεί να ενημερώσει έναν LwM2M server για τα objects τα οποία μπορεί να υποστηρίξει αποστέλλοντας μόνο το object ID. Έτσι όλα τα χαρακτηριστικά καθώς και τα resources που αφορούν τα objects αυτά μπορούν να ανασυρθούν από αυτό το μητρώο που μπορεί να βρίσκεται είτε τοπικά στην μνήμη του LwM2M server είτε στο Διαδίκτυο. Επιπρόσθετα έχει υλοποιηθεί και άλλο ένα μητρώο το Resource Registry το οποίο περιέχει επαναχρησιμοποιούμενα resources τα οποία έχουν κοινά χαρακτηριστικά και σημασία για κάθε LwM2M object που τα χρησιμοποιεί. Στην συγκεκριμένη εργασία δημιουργήθηκε ένα μητρώο που βρίσκεται τοπικά στην μνήμη του LwM2M server.

Επειδή η παρούσα εργασία αφορά την υλοποίηση του Industrial Automation Thing δεν δόθηκε ιδιαίτερη έμφαση στην υλοποίηση του LwM2M server. Εν αντιθέσει υλοποιήθηκε ο LwM2M client ώστε να χρησιμοποιηθούν τα πλεονεκτήματα που δίνονται από το LwM2M. Ο LwM2M server που χρησιμοποιήθηκε υλοποιήθηκε σε άλλη διπλωματική εργασία που εκπονήθηκε ταυτόχρονα με την παρούσα.

### 3.3 Πλεονεκτήματα του LwM2M

Το πρωτόκολλο LwM2M επιλύει ένα σύνολο τεχνολογικών προκλήσεων που σχετίζονται με την διαχείριση συσκευών και την ενεργοποίηση end-to-end υπηρεσιών καθώς η αγορά M2M ωριμάζει και το Διαδίκτυο των Αντικειμένων καθιστούν εφικτή την επικοινωνία μεταξύ τέτοιων συσκευών. Παρακάτω συνοψίζονται τα οφέλη που παρέχει το LwM2M [28]:

- Μεγαλύτερη ανάπτυξη της αγοράς και αποδοτικότητα κόστους για ολόκληρη την βιομηχανία, μέσω των τεχνολογιών που προσφέρει το LwM2M οι οποίες καθιστούν εφικτή την χαλαρή σύνδεση συσκευών, την εύκολη διαχείριση τους και την διαχείριση των υπηρεσιών που αυτές προσφέρουν.
- Οι πάροχοι υπηρεσιών, οι OEMs και οι τελικοί χρήστες επωφελούνται από την ομοιόμορφη διαχείριση των διάφορων συσκευών.
- Σε σύγκριση με τις protocol stacks που χρησιμοποιούνται στις παραδοσιακές συσκευές, το LwM2M μπορεί να παρέχει 10x αύξηση της αποδοτικότητας.

- Το LwM2M συμπληρώνει τις υπάρχουσες λύσεις διαχείρισης συσκευών, όπως το OMA DM και το Broadband Forum TR-69, και επεκτείνει σε μεγάλο βαθμό το εύρος των συσκευών που μπορούν να διαχειριστούν με ασφάλεια.
- Το μοντέλο δεδομένων του LwM2M καθώς και το μητρώο των αντικειμένων παρέχουν προσβάσιμη και επαναχρησιμοποιήσιμη σημασιολογία τόσο για την διαχείριση συσκευών όσο και για την διαχείριση των δεδομένων μιας εφαρμογής για όλο το Διαδίκτυο των Αντικειμένων στην βιομηχανία.
- Παρέχοντας μια ενιαία λύση για την διαχείριση των συσκευών και των δεδομένων τους το LwM2M απλοποιεί τα διάφορα συστήματα και επιτρέπει την δημιουργία νέων και καινοτόμων υπηρεσιών M2M.
- Η πλήρης διαχείριση του κύκλου ζωής και της ασφάλειας που είναι κατάλληλη για περιορισμένες συσκευές επιλύει ένα από τα πιο πιεστικά προβλήματα στη βιομηχανία M2M.
- Το LwM2M ορίζει μόνο της διεπαφή δικτύου της συσκευής, επιτρέποντας έτσι την εύκολη ενσωμάτωση του σε υπάρχουσες υπηρεσίες διαχείρισης συσκευών και υπηρεσιών M2M.

## Κεφάλαιο 4

# Ανάπτυξη λογισμικού με την προσέγγιση συνιστωσών

### 4.1 Εισαγωγή

Για την μείωση της πολυπλοκότητας των συστημάτων λογισμικού, οι μηχανικοί λογισμικού έχουν εφεύρει αρκετές τεχνικές που στοχεύουν στην βελτίωση της επαναχρησιμοποίησης και της συντηρησιμότητας του λογισμικού όπως η μοντελοποίηση λογισμικού σε μονάδες (modules) και η αντικειμενοστρέφεια. Στην [33], αναφέρεται ότι το 53% των συστημάτων λογισμικού δεν πέτυχε να παραδοθεί εντός του χρονοδιαγράμματος και του προϋπολογισμού, καθώς και με την απαιτούμενη λειτουργικότητα και ποιότητα, ενώ το 18% των έργων εγκαταλείφθηκε πλήρως. Αυτά τα στοιχεία καθιστούν σαφή τη σημασία της ακριβής πρόβλεψης της προσπάθειας που απαιτείται για να αναπτυχθεί ένα νέο σύστημα λογισμικού.

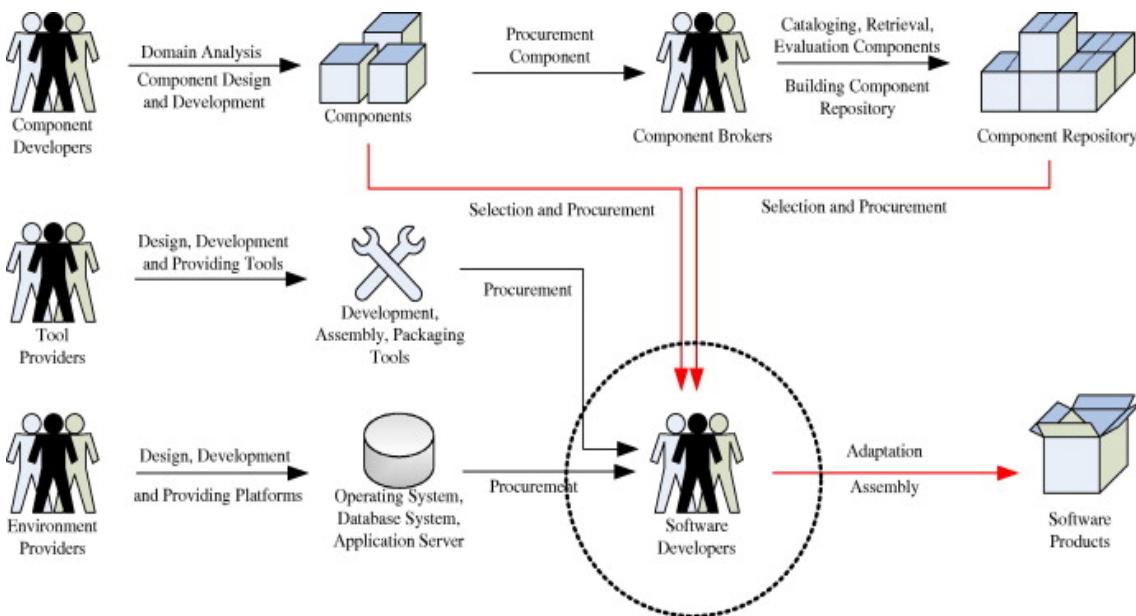
Τα τελευταία χρόνια, μία δημοφιλής προσέγγιση ανάπτυξης λογισμικού, η ανάπτυξη λογισμικού βασισμένη σε components (Component Based Software Development, CBSD) έχει αλλάξει σημαντικά την κατάσταση που έχουν να αντιμετωπίσουν οι μηχανικοί λογισμικού. Στην βιομηχανία έχουν αναφερθεί σημαντικά πλεονεκτήματα που δίνει η χρήση της CBSD προσέγγισης ανάπτυξης λογισμικού έναντι του παραδοσιακού τρόπου ανάπτυξης λογισμικού [33]. Μετά την επιτυχία που είχε η Αντικειμενοστραφής προσέγγιση, η CBSD εμφανίζεται ως η επόμενη επανάσταση στον τρόπο ανάπτυξης λογισμικού [34]. Επιπλέον σύμφωνα με την [33] που αφορά τους τρόπους ανάπτυξης λογισμικού που εφαρμόζονται στην βιομηχανία, διαπιστώθηκε ότι περίπου το 53% από 118 εταιρείες που συμμετείχαν σε έρευνα χρησιμοποιούν την CBSD και η τάση αυτή συνεχώς αυξάνεται.

### 4.2 Η προσέγγιση των συνιστωσών

#### 4.2.1 Γενικές πληροφορίες και βασικές έννοιες

Η ανάπτυξη λογισμικού με βάση τα components αποτελεί μία προσέγγιση βασισμένη στην επαναχρησιμοποίηση για τον ορισμό, την υλοποίηση και την σύνθεση χαλαρά συνδεδεμένων (loosely coupled) ανεξάρτητων components σε συστήματα λογισμικού. Επιδιώκει να επιφέρει ευρύ φάσμα πλεονεκτημάτων τόσο βραχυπρόθεσμα όσο και μακροπρόθεσμα για το ίδιο το λογισμικό και για οργανισμούς που χορηγούν τέτοιες λύσεις. Το σκεπτικό πίσω από αυτή

την προσέγγιση είναι η συναρμολόγηση συστημάτων λογισμικού, χρησιμοποιώντας υπάρχοντα, κατασκευασμένα από τρίτους και δοκιμασμένα components έτσι ώστε να μην είναι απαραίτητη η υλοποίηση όλου του συστήματος από την αρχή. Ο κύκλος ζωής της CBSD μπορεί να χωριστεί σε τέσσερις φάσεις: **Domain analysis, Component Design and Development, Component Cataloging and Retrieval, Component Selection and Application Assembly** [35]. Κατά την ανάπτυξη ενός προϊόντος λογισμικού χρησιμοποιώντας την CBSD τεχνική, τα διαθέσιμα components συναρμολογούνται ή προσαρμόζονται με τη βοήθεια μίας εφαρμογής ανάπτυξης λογισμικού [34]. Στην εικόνα 4.1 δίνεται μία απεικόνιση της διαδικασίας που ακολουθείται κατά την ανάπτυξη ενός συστήματος λογισμικού χρησιμοποιώντας την CBSD προσέγγιση.



**Εικόνα 4.1:** Διάγραμμα ροής της διαδικασίας CBSD [34]

## 4.2.2 Οντότητες Λογισμικού

### 4.2.2.1 Component

Σύμφωνα με το [36], οι χαρακτηριστικές ιδιότητες ενός component είναι:

1. Αποτελεί μία μονάδα που αναπτύσσεται ανεξάρτητα
2. Μπορεί να αναπτυχθεί και από τρίτους
3. Δεν έχει (εξωτερικά) παρατηρήσιμη κατάσταση

Αυτές οι ιδιότητες έχουν πολλές συνέπειες. Για να μπορεί ένα component να αναπτύσσεται ανεξάρτητα, είναι απαραίτητο να διαχωρίζεται από το περιβάλλον στο οποίο θα χρησιμοποιηθεί και από τα άλλα components. Επομένως, ένα component ενθυλακώνει όλα τα χαρακτηριστικά του. Επίσης, καθώς αποτελεί μία μονάδα ανάπτυξης ενός συστήματος δεν θα χρησιμοποιηθεί ποτέ μόνο του. Σε αυτό το πλαίσιο γίνεται σαφές ότι μία ομάδα τρίτων που αναπτύσσει ένα component δεν είναι απαραίτητο να έχει πρόσβαση σε λεπτομέρειες κατασκευής ενός συστήματος καθώς ούτε και σε όλα τα υπόλοιπα components που απαρτίζουν το σύστημα [36].

Για να μπορεί ένα component να συνδυαστεί με άλλα components που μπορεί να έχουν σχεδιαστεί και υλοποιηθεί από τρίτους πρέπει να είναι επαρκώς αυτόνομο [36]. Ένα component παρέχει ένα σύνολο υπηρεσιών (services) και τις εκθέτει στο περιβάλλον του μέσω ενός interface που ονομάζεται “provided interface”. Το component επίσης μπορεί να χρειαστεί κάποιες υπηρεσίες από άλλα components ή από το περιβάλλον στο οποίο λειτουργεί προκειμένου να παρέχει με επιτυχία τις δικές του. Αυτές οι απαιτούμενες υπηρεσίες συγκεντρώνονται στο “required interface” του component. Για το λόγο αυτό σε ένα σύστημα που αποτελείται από components κάθε component συναρμολογείται σε συνεργασία με άλλα components ώστε να μπορούν να καλυφθούν οι απαιτήσεις του. Το component ουσιαστικά ορίζει ένα είδος συμβατικής συμφωνίας με την οποία, αν ικανοποιηθούν όλες οι λειτουργικές ανάγκες της (απαιτούμενες υπηρεσίες), καθώς και άλλες πιθανές υποθέσεις για το περιβάλλον ή την πλατφόρμα εκτέλεσης, τότε είναι σε θέση να παρέχει τις λειτουργικές της υπηρεσίες [37]. Με άλλα λόγια, ένα component πρέπει να “κρύβει” την υλοποίηση του από το περιβάλλον του και να αλληλεπιδρά με αυτό μέσω καλά καθορισμένων interfaces [36].

Τέλος, ένα component δεν πρέπει να έχει οποιαδήποτε εξωτερικά παρατηρήσιμη κατάσταση. Δηλαδή απαιτείται το component να μην διακρίνεται από τα δικά του αντίγραφα. Πιθανές εξαιρέσεις σε αυτό τον κανόνα είναι χαρακτηριστικά που δεν συμβάλουν στην λειτουργικότητα του component, όπως για παράδειγμα οι σειριακοί αριθμοί που χρησιμοποιούνται στην λογιστική. Ο συγκεκριμένος αποκλεισμός της παρατηρήσιμης κατάστασης επιτρέπει τεχνικές χρήσεις της κατάστασης που μπορεί να είναι κρίσιμες για την απόδοση χωρίς να επηρεάζεται η παρατηρήσιμη συμπεριφορά ενός component. Συγκεκριμένα, ένα component μπορεί να χρησιμοποιήσει μια κατάσταση για σκοπούς προσωρινής αποθήκευσης, π.χ. μια μνήμη cache [36].

#### 4.2.2.2 Interface

Τα interfaces αποτελούν σημαντικό στοιχείο των components. Το interface ενός component καθορίζει τις υπηρεσίες που αυτό παρέχει στο περιβάλλον του καθώς και τα σημεία πρόσβασης (access points) του. Κανονικά, ένα component θα απαρτίζεται από πολλά interfaces που θα αντιστοιχούν σε διαφορετικά σημεία πρόσβασης. Κάθε σημείο πρόσβασης μπορεί να παρέχει

διαφορετική υπηρεσία, καλύπτοντας διαφορετικές ανάγκες. Ιδιαίτερη έμφαση πρέπει να δοθεί στον συμβατικό χαρακτήρα των προδιαγραφών ενός interface, καθώς το component και τα components που θα το χρησιμοποιήσουν αναπτύσσονται σε αμοιβαία άγνοια, οπότε η σύμβαση (contract) αυτή αποτελεί το κοινό μέσο για την επιτυχή αλληλεπίδραση των components [36]. Η εικόνα 8 δείχνει δύο components με τα interfaces τους.

#### **4.2.2.3 Container**

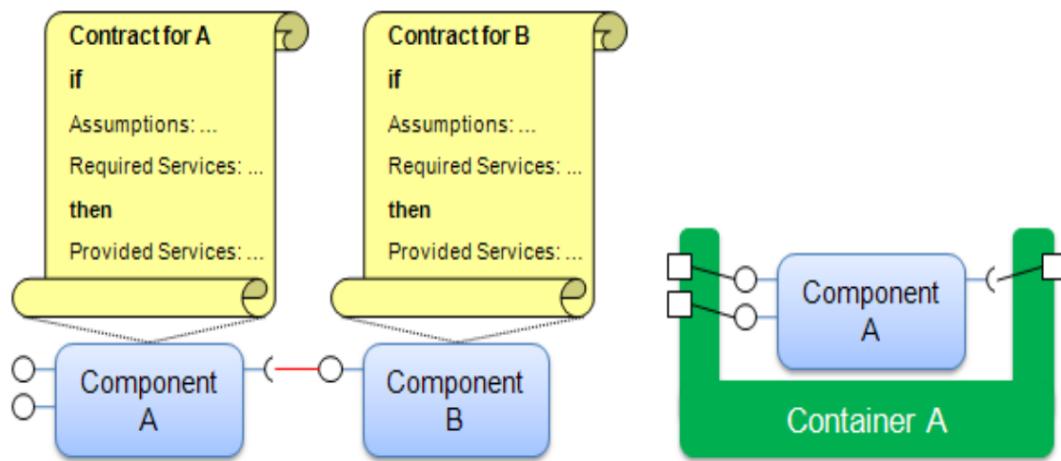
Ο container μπορεί να θεωρηθεί σαν ένας software wrapper γύρω από τα components, ο οποίος είναι υπεύθυνος για την ικανοποίηση των έξω-λειτουργικών απαιτήσεων που τίθενται επί του component. Γενικά, δεν μπορεί να υπάρξει άμεση επικοινωνία μεταξύ των components. Αυτό εν μέρει προκαλείται από τον container μέσα στον οποίο βρίσκεται το component, ο οποίος είναι υπεύθυνος για την απόκτηση και έκθεση των απαιτούμενων και παρεχόμενων interfaces αντίστοιχα. Ο container επίσης, διαμεσολαβεί την πρόσβαση του component σε υπηρεσίες που παρέχονται από την πλατφόρμα εκτέλεσης [37]. Στην εικόνα 4.2 φαίνεται ένα component μαζί με τον container.

#### **4.2.2.4 Connector**

Ο connector αποτελεί μία οντότητα λογισμικού που είναι υπεύθυνη για την αλληλεπίδραση μεταξύ των components, όπως αυτή ορίζεται από τον container. Ο ρόλος του connector είναι να αποσυνδέσει τα προβλήματα αλληλεπίδρασης από τα προβλήματα λειτουργικότητας. Πρακτικά, αυτό σημαίνει ότι ένα component δεν χειρίζεται άμεσα την αλληλεπίδραση του με τα άλλα components του συστήματος. Έτσι γίνεται εύκολα κατανοητό ότι ο κώδικας ενός component μπορεί να προσδιοριστεί ανεξάρτητα από: (1) τα component στα οποία θα συνδεθεί αργότερα και (2) την μορφή επικοινωνίας που θα χρησιμοποιηθεί. Αυτό είναι απαραίτητο καθώς τα components σχεδιάζονται μεμονωμένα και η σύνδεση τους με άλλα components αποτελεί μεταγενέστερη ανησυχία ή μπορεί να ποικίλει και να είναι σχετική με τα περιβάλλοντα στα οποία θα κληθεί να λειτουργήσει το component [37].

### **4.2.3 Στόχοι της CBSD προσέγγισης**

Υπάρχουν πολλές συζητήσεις και διαφωνίες σχετικά με τι το είναι και τι δεν είναι τα components [38]. Μερικοί συγγραφείς προτιμούν να θεωρούν τα components σαν συνεκτικά πακέτα με χρήσιμη συμπεριφορά, ενώ άλλοι τα αντιμετωπίζουν σαν αναπτυσσόμενες μονάδες λογισμικού που εκτελούνται μέσα σε ένα συγκεκριμένο περιβάλλον [39]. Ανεξάρτητα από αυτές τις



**Εικόνα 4.2: Αριστερά:** Components μαζί με τα interfaces τους. **Δεξιά:** Ένας container μαζί με το component [37]

διαφορές, η βασική προσέγγιση της CBSD είναι η κατασκευή συστημάτων από σαφώς ορισμένα, ανεξάρτητα μεταξύ τους κομμάτια. Ωστόσο, οι ενδιαφέρουσες πτυχές της CBSD αφορούν τον τρόπο με τον οποίο επιτυγχάνεται αυτή η προσέγγιση, επιτρέποντας την ανάπτυξη των components ως κατάλληλων συνεκτικών μονάδων λειτουργικότητας και τη διευκόλυνση του σχεδιασμού και της συναρμολόγησης συστημάτων από ένα μείγμα νέων και προηγουμένων ανεπτυγμένων components [40]. Ο στόχος της δημιουργίας συστημάτων από καλά καθορισμένα κομμάτια δεν είναι κάτι καινούργιο.

Το ενδιαφέρον για την CBSD βασίζεται σε μια μακρά ιστορία εργασίας πάνω σε αρθρωτά συστήματα, δομημένο σχεδιασμό και πιο πρόσφατα σε αντικειμενοστραφή συστήματα [41] - [45]. Αυτά αποσκοπούσαν στην ευκολότερη ανάπτυξη και διατήρηση ευρύτερων συστημάτων χρησιμοποιώντας μια προσέγγιση “divide and conquer”. Η CBSD επεκτείνει αυτές τις ιδέες, δίνοντας έμφαση στο σχεδιασμό λύσεων με κομμάτια λειτουργικότητας που παρέχονται σαν components, προσβάσιμα σε άλλους μόνο μέσω σαφώς ορισμένων interfaces και εστιάζοντας στην ελεγχόμενη συναρμολόγηση των components χρησιμοποιώντας τεχνικές σχεδιασμού βασισμένες σε interfaces [40].

Το πιο σημαντικό είναι ότι οι έννοιες αυτές υποστηρίζονται από μια σειρά προϊόντων που εφαρμόζουν ανοικτά πρότυπα που προσφέρουν μια υποδομή υπηρεσιών για τη δημιουργία, τη συναρμολόγηση και την εκτέλεση των components. Κατά συνέπεια, η διαδικασία ανάπτυξης εφαρμογών έχει ανασχεδιαστεί έτσι ώστε η κατασκευή λογισμικού να επιτυγχάνεται σε μεγάλο βαθμό μέσω μιας διαδικασίας επιλογής, αξιολόγησης και συναρμολόγησης των components. Τα components αποκτώνται από ένα διαφορετικό σύνολο πηγών και χρησιμοποιούνται μαζί με τοπικά σχεδιασμένο λογισμικό για την κατασκευή μιας πλήρους εφαρμογής [46].

#### 4.2.4 Γιατί ανάπτυξη λογισμικού με συνιστώσες

Η ανάπτυξη λογισμικού με βάση τα components, έρχεται να δώσει σημαντικές λύσεις σε πολλές προκλήσεις που αντιμετωπίζει η βιομηχανία καθώς και οι επιχειρήσεις στην εποχή του Διαδικτύου. Αυτές οι προκλήσεις αναφέρονται στην επόμενη λίστα [40]:

- **Πολυπλοκότητα:** Σε κάθε περίπλοκη κατάσταση υπάρχουν μερικές βασικές τεχνικές που μπορούν να χρησιμοποιηθούν για την κατανόηση και τη διαχείριση αυτής της πολυπλοκότητας. Αυτές είναι οι τεχνικές αφαίρεσης, αποσύνθεσης και σταδιακής ανάπτυξης. Κάθε λύση ανάπτυξης εφαρμογών πρέπει να παρέχει τρόπους υποστήριξης αυτών των τεχνικών.
- **Μείωση του χρόνου παράδοσης:** Η δυνατότητα έγκαιρης παράδοσης λύσεων αποτελεί ουσιαστική πτυχή κάθε έργου ανάπτυξης λογισμικού. Με τον αυξημένο ρυθμό αλλαγής της τεχνολογίας, αυτή η πτυχή γίνεται πιο σημαντική. Αυτή η ανάγκη για μειωμένο χρόνο παράδοσης συστημάτων λογισμικού αναφέρεται συχνά ως εργασίας σε “ρυθμούς Internet”.
- **Βελτίωση της συνοχής:** Τα περισσότερα συστήματα λογισμικού μοιράζονται σημαντικά χαρακτηριστικά με άλλα συστήματα που είτε έχουν αναπτυχθεί ήδη, είτε βρίσκονται στην φάση της ανάπτυξης ή δεν έχουν κατασκευαστεί ακόμα. Αν χρησιμοποιηθούν αυτά τα χαρακτηριστικά θα είναι εφικτό να βελτιωθεί η συνοχή και να μειωθεί σημαντικά το κόστος ανάπτυξης νέων συστημάτων λογισμικού.
- **Χρήση του καλύτερου στην κατηγορία:** Σε αρκετούς τομείς υπάρχουν ήδη καλά ανεπτυγμένες λύσεις που προσφέρουν ισχυρή λειτουργικότητα και απόδοση. Η αξιοποίηση αυτών των λύσεων ως μέρος μίας μεγαλύτερης προσπάθειας ανάπτυξης ενός συστήματος κρίνεται απαραίτητη.
- **Αύξηση της παραγωγικότητας:** Η έλλειψη δεξιοτήτων στον τομέα της ανάπτυξης λογισμικού έχει ως αποτέλεσμα σημαντικές καθυστερήσεις στους χρήστες συστημάτων. Οποιεσδήποτε νέες προσεγγίσεις πρέπει να βελτιώνουν την παραγωγικότητα των εργαζομένων ώστε να μπορούν να παράγουν ποιοτικά αποτελέσματα με ταχύτερους ρυθμούς.
- **Βελτίωση της ποιότητας:** Καθώς αυξάνεται το οικονομικό και ανθρώπινο αντίκτυπο μίας αποτυχίας ενός συστήματος λογισμικού, πρέπει να δοθεί ιδιαίτερη προσοχή στην ποιότητα των ανεπτυγμένων συστημάτων. Στόχος πρέπει να είναι η σωστή υποστήριξη της κατασκευής συστημάτων εξ' αρχής, χωρίς εκτεταμένες και δαπανηρές δοκιμές και επανεγγραφή του κώδικα.
- **Αύξηση της ορατότητας στην πρόοδο του έργου:** Η διαχείριση μεγάλων έργων αποτελεί ένα εγχείρημα υψηλού κινδύνου. Για να αποφευχθεί αυτός ο κίνδυνος, πρέπει να

είναι δυνατή η μεγαλύτερη ορατότητα καθ' όλη τη διάρκεια του κύκλου ζωής του λογισμικού. Αυτό απαιτεί μια βαθιμαία προσέγγιση στην ανάπτυξη, την παράδοση και την δοκιμή κομματιών λογισμικού.

- **Υποστήριξη παράλληλου και κατανεμημένου τρόπου ανάπτυξης λογισμικού:** Οι κατανεμημένες ομάδες ανάπτυξης λογισμικού απαιτούν προσεγγίσεις που ενθαρρύνουν και επιτρέπουν την παράλληλη ανάπτυξη συστημάτων λογισμικού. Κάτι τέτοιο όμως απαιτεί ιδιαίτερη προσοχή στην διαχείριση της πολυπλοκότητας λόγω της ανάγκης διαιρεσης και επανασυγχρονισμού των αποτελεσμάτων.
- **Μείωση του κόστους συντήρησης:** Το μεγαλύτερο μέρος του κόστους ανάπτυξης ενός συστήματος προκύπτει μετά την αρχική ανάπτυξη του συστήματος. Για να μειωθεί το κόστος συντήρησης, πρέπει να είναι δυνατόν να προσδιοριστεί ευκολότερα η ανάγκη για αλλαγή, να είναι εφικτό να προβλεφθεί το αντίκτυπο που θα φέρει οποιαδήποτε προτεινόμενη αλλαγή και να εφαρμοστεί αυτή η αλλαγή στο υπάρχων σύστημα.

Η παραπάνω λίστα αποτελεί ένα αποθαρρυντικό σύνολο προκλήσεων για οποιαδήποτε προσέγγιση ανάπτυξης λογισμικού. Ωστόσο, τα components και οι προσεγγίσεις που βασίζονται σε αυτά προσφέρουν την πιο ελπιδοφόρα προσπάθεια αντιμετώπισης των προκλήσεων και προσφέρουν τη βάση μιας νέας σειράς τεχνικών που υποστηρίζουν την επόμενη γενιά λύσεων που προσφέρει το λογισμικό.

#### 4.2.5 Πλεονεκτήματα της CBSD σε σύγκριση με παραδοσιακές τεχνικές ανάπτυξης

Αρχικά, η φιλοσοφία ανάπτυξης ενός συστήματος λογισμικού αυτών των δύο προσεγγίσεων είναι εντελώς διαφορετική. Η παραδοσιακή ανάπτυξη λογισμικού είναι ίδια με την κατασκευή από το μηδέν. Αντίθετα η CBSD προσέγγιση στηρίζεται περισσότερο στην “buy, don’t build” φιλοσοφία. Δεύτερον, η δομή του λογισμικού που αναπτύσσεται με παραδοσιακές τεχνολογίες χαρακτηρίζεται από υψηλή σύζευξη μεταξύ των επιμέρους τμημάτων του συστήματος, αλλά η CBSD προσπαθεί να κάνει την δομή του λογισμικού να έχει μία χαμηλότερη σύζευξη με σκοπό την βελτίωση της ποιότητας και της συντηρησιμότητας του κώδικα. Η τρίτη σημαντική διαφορά είναι ότι η διαδικασία ανάπτυξης ενός συστήματος λογισμικού χρησιμοποιώντας CBSD είναι εξελικτική και ταυτόχρονη. Αυτό συμβαίνει γιατί ένα σύστημα υλοποιημένο με βάση τα components στην ουσία οικοδομείται και δεν υλοποιείται από την αρχή. Σε αντίθεση με την παραδοσιακή ανάπτυξη λογισμικού, η CBSD εστιάζει κυρίως σε δραστηριότητες που αφορούν το integration, την αναζήτηση και τον εντοπισμό των υποψήφιων components, την αξιολόγηση και την επιλογή τους βάσει των απαιτήσεων του συστήματος καθώς και των περιορισμών του έργου. Η προσέγγιση CBSD έχει πολλά πλεονεκτήματα, όπως η αποτελεσματική

διαχείριση της πολυπλοκότητας, ο μειωμένος χρόνος προς την αγορά, η αυξημένη παραγωγικότητα και ο μεγαλύτερος βαθμός συνοχής και ένα ευρύτερο φάσμα χρηστικότητας. Επιπλέον, τα επαναχρησιμοποιήσιμα στοιχεία καθιστούν δυνατή τη διάσπαση της διαδικασίας ανάπτυξης εφαρμογών σε διαφορετικά μέρη, έτσι ώστε διαφορετικοί άνθρωποι ή εταιρείες να μπορούν να εκτελούν διαφορετικά μέρη της διαδικασίας.

## 4.3 Open Services Gateway initiative (OSGi)

### 4.3.1 Γενικά

Από την δεκαετία του '70, η οργάνωση του κώδικα σε modules θεωρείται βασική ιδιότητα για την βελτίωση της ευελιξίας, της κατανόησης και της επαναχρησιμοποίησης του [49]. Σε γενικές γραμμές, τα modules αντιστοιχούν σε μονάδες που μπορούν να εκτελέσουν μία διεργασία και μπορούν να υλοποιηθούν ανεξάρτητα, να συγκεντρωθούν και στην συνέχεια να συναρμολογηθούν κατάλληλα ώστε να υλοποιηθεί ένα σύστημα λογισμικού. Για παράδειγμα, οι κλάσεις στην Java ενθυλακώνουν τόσο δεδομένα όσο και υπηρεσίες. Μέσω της τεχνολογίας των πακέτων (packages σε όρους Java), γίνεται ομαδοποίηση και οργάνωση κλάσεων σχετικών μεταξύ τους, με αποτέλεσμα να διευκολύνεται η επαναχρησιμοποίηση. Επιπλέον, τα πακέτα μπορούν να αναπτυχθούν και να διανεμηθούν με την μορφή αρχείων JAR [50].

Ωστόσο το αρθρωτό σύστημα που χρησιμοποιείται στην Java παρουσιάζει τουλάχιστον δύο προβλήματα. Πρώτον, οι μηχανισμοί απόκρυψης πληροφοριών εφαρμόζονται μόνο στο επίπεδο των κλάσεων, αλλά όχι στο επίπεδο των πακέτων και των αρχείων JAR. Για παράδειγμα, δεν είναι δυνατό να περιοριστεί η πρόσβαση σε κλάσεις με ορατότητα public που ορίζονται στα διαθέσιμα πακέτα. Η απουσία τέτοιων κανόνων ελέγχου της ορατότητας μπορεί εύκολα να οδηγήσει σε συστήματα πολύ στενά συνδεδεμένα μεταξύ τους. Από την άλλη πλευρά, το αρθρωτό σύστημα της Java είναι εγγενώς στατικό. Αυτό σημαίνει ότι τα διάφορα modules μπορούν να ενημερωθούν για τυχόν αλλαγές μόνο κατά τον χρόνο ανάπτυξης, γεγονός που απαιτεί διακοπή και επανεκκίνηση του συστήματος [50]. Ένας τέτοιος περιορισμός δημιουργεί τεράστια προβλήματα σε πολλούς τομείς όπως αυτός της βιομηχανίας όπου αναφέρεται και η παρούσα εργασία.

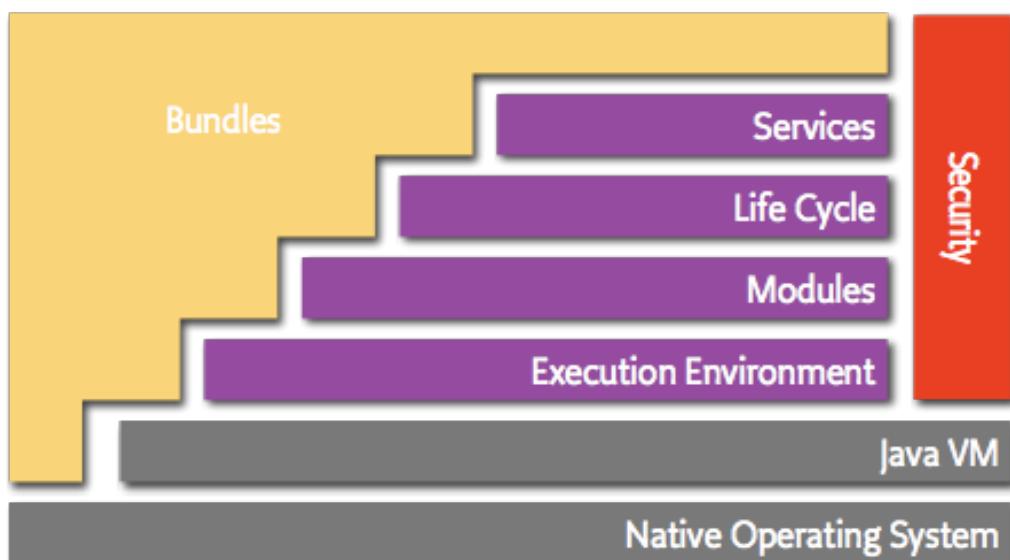
Προκειμένου να αντιμετωπιστούν οι περιορισμοί του τυποποιημένου αρθρωτού συστήματος της Java, η πρωτοβουλία Open Services Gateway (OSGi) πρότεινε το 1999 [48] το OSGi framework και μοντέλο προγραμματισμού, το οποίο παρέχει ένα δυναμικό αρθρωτό σύστημα με γνώμονα την παροχή υπηρεσιών για την γλώσσα προγραμματισμού Java. Σύμφωνα με τις αρχές του OSGi, τα συστήματα λογισμικού πρέπει να είναι δομημένα γύρω από ανεξάρτητα modules, τα οποία σε όρους OSGi ονομάζονται bundles, που παρέχουν σαφώς καθορισμένες υπηρεσίες. Το OSGi, καθορίζει επίσης και μια υποδομή για τον έλεγχο του κύκλου ζωής των

bundles κατά τον χρόνο εκτέλεσης. Αυτή η υποδομή επιτρέπει στους προγραμματιστές να προσθέτουν, να αφαιρούν και να αναβαθμίζουν bundles από το σύστημα δυναμικά [50].

### 4.3.2 Η αρχιτεκτονική του OSGi

Το OSGi παρέχει ένα γενικού σκοπού, ασφαλές και εύκολα διαχειρίσιμο framework για την Java, το οποίο υποστηρίζει την ανάπτυξη επεκτάσιμων εφαρμογών και εφαρμογών που μπορούν να μεταφορτωθούν από το διαδίκτυο, γνωστές και ως bundles. Οι συσκευές που είναι συμβατές με το OSGi μπορούν να λαμβάνουν και να εγκαθιστούν bundles τα οποία θα μπορούν να αφαιρέσουν όταν δεν θα είναι πλέον απαραίτητα. Το framework διαθέτει τεχνολογίες που διαχειρίζονται την εγκατάσταση και την ενημέρωση των bundles σε ένα περιβάλλον συμβατό με το OSGi, με δυναμικό και κλιμακωτό τρόπο. Για να επιτευχθεί αυτό, οι εξαρτήσεις (dependencies) μεταξύ των bundles διαχειρίζονται λεπτομερώς από το framework [51].

Το OSGi έχει ένα πολυεπίπεδο μοντέλο που απεικονίζεται στην εικόνα 4.3:



**Εικόνα 4.3:** Τα επίπεδα της αρχιτεκτονικής του OSGi [48]

**Bundles:** Τα bundles είναι τα OSGi components που φτιάχνονται από τους προγραμματιστές. Πιο συγκεκριμένα, τα bundles είναι κανονικά .jar αρχεία που περιέχουν αρχεία για τις κλάσεις τους, άλλους πόρους όπως εικόνες, απαιτούμενα API κ. λπ., καθώς και ένα αρχείο manifest το οποίο χρησιμοποιείται για την δήλωση στατικών πληροφοριών σχετικά με το bundle, όπως τα πακέτα που εισάγει και εξάγει. Επιπλέον, κάθε bundle μπορεί να παρέχει υπηρεσίες σε άλλα bundles. Στην αρχιτεκτονική του OSGi, μια υπηρεσία είναι ένα τυπικό αντικείμενο Java που έχει καταχωρηθεί χρησιμοποιώντας έναν ή και περισσότερους τύπους διεπαφών και ιδιοτήτων που χρησιμοποιούνται για τον εντοπισμό της αντίστοιχης υπηρεσίας [50].

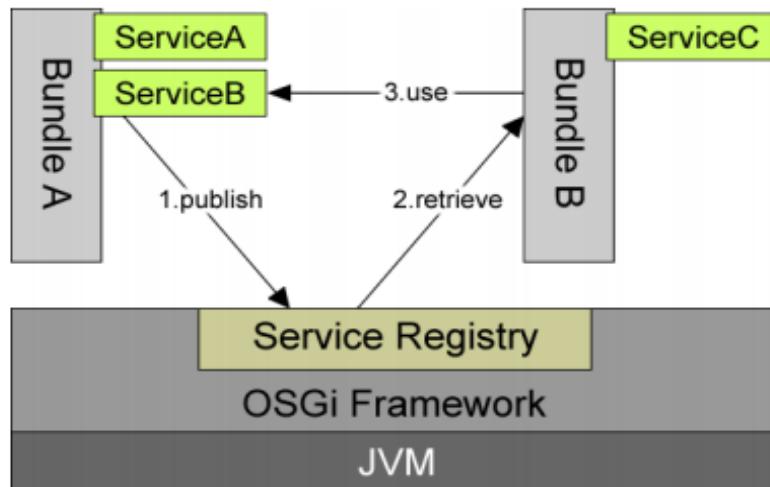
**Services:** Το επίπεδο των services συνδέει τα bundles με ένα δυναμικό τρόπο προσφέροντας ένα publish-find-bind μοντέλο για τα απλά αντικείμενα Java (POJO). Είναι απαραίτητο γιατί στην Java είναι δύσκολη η συνεργατική ανάπτυξη κώδικα με κοινή χρήση μόνο των κλάσεων και των στατικών στοιχείων. Για παράδειγμα αν σε μία εφαρμογή υπάρχει η ανάγκη για την δημιουργία ενός DocumentBuilderFactory θα πρέπει να γίνει κλήση στην στατική μέθοδο του factory DocumentBuilderFactory.newInstance(). Πίσω από αυτή την κλήση όμως, η μέθοδος newInstance() χρησιμοποιεί κάθε τέχνασμα του class loader ώστε να δημιουργήσει ένα νέο instance της κλάσης DocumentBuilderFactory. Αυτό αποτελεί ένα παθητικό μοντέλο. Ο κώδικας της υλοποίησης δεν μπορεί να κάνει τίποτα ώστε να αποδείξει την διαθεσιμότητά του και ο εκάστοτε χρήστης δεν μπορεί να γνωρίζει όλες τις πιθανές υλοποιήσεις ώστε να διαλέξει την καταλληλότερη. Επιπλέον, είναι ένα στατικό μοντέλο. Μόλις μία υλοποίηση δημιουργήσει ένα νέο instance δεν μπορεί να καταστρέψει το αντικείμενο. Το χειρότερο από όλα είναι ότι αυτό ο μηχανισμός χρησιμοποιείται σε πολλά σημεία και κάθε factory έχει τους δικούς του μοναδικούς μηχανισμούς API και ρύθμισης των παραμέτρων του. Δεν υπάρχει συγκεντρωτική επισκόπηση όλων των υλοποιήσεων του εκάστοτε factory που χρησιμοποιείται. Κάτι τέτοιο δημιουργεί πολλά προβλήματα [48].

Τη λύση σε όλα αυτά τα προβλήματα δίνει το μητρώο υπηρεσιών που παρέχει το OSGi. Ένα bundle μπορεί να δημιουργήσει ένα αντικείμενο και να το καταχωρήσει στο μητρώο υπηρεσιών κάτω από μία ή περισσότερες διεπαφές. Άλλα bundles μπορούν να μεταβούν στο μητρώο και να βρουν όλα τα αντικείμενα που είναι καταχωρημένα κάτω από μία διεπαφή ή κλάση. Για παράδειγμα, ένα bundle μπορεί να παρέχει μία συγκεκριμένη υλοποίηση του DocumentBuilderIf. Όταν ξεκινάει, δημιουργεί ένα instance του αντικειμένου DocumentBuilderImpl και το καταχωρεί στο μητρώο υπηρεσιών. Ένα bundle που χρειάζεται ένα αντικείμενο DocumentBuilder μπορεί να μεταβεί στο μητρώο και να ζητήσει όλες τις διαθέσιμες υπηρεσίες που υλοποιούν το DocumentBuilderIf. Ακόμα καλύτερα ένα bundle μπορεί να περιμένει μία υπηρεσία να εμφανιστεί και να πάρει μία κλήση πίσω (call back) [48].

Επομένως, ένα bundle μπορεί να καταχωρίσει μία υπηρεσία στο μητρώο, να λάβει μία υπηρεσία που είναι ήδη καταχωρημένη και να περιμένει μία υπηρεσία να εμφανιστεί ή να εξαφανιστεί. Οποιοσδήποτε αριθμός bundles μπορεί να καταχωρίσει την ίδια υπηρεσία και οποιοσδήποτε αριθμός bundles μπορεί να λάβει την ίδια υπηρεσία [48]. Αυτή η λειτουργικότητα εμφανίζεται στην εικόνα 4.4.

Σύμφωνα με όσα αναφέρθηκαν, είναι εύλογο το ερώτημα πως διαχωρίζονται οι υπηρεσίες αν πολλά bundles έχουν καταχωρήσει την ίδια υπηρεσία. Αρχικά, σε πολλές περιπτώσεις δεν είναι σημαντικό να γίνει διάκριση της υπηρεσίας από ένα συγκεκριμένο bundle. Σε περίπτωση όμως που κάτι τέτοιο κριθεί αναγκαίο το OSGi παρέχει τις ιδιότητες. Κάθε υπηρεσία που καταχωρείται έχει ένα σύνολο από τυπικές και προσαρμοσμένες ιδιότητες. Επιπλέον, υπάρχει μία

συγκεκριμένη γλώσσα που μπορεί να φιλτράρει τις υπηρεσίες που παρέχονται και να διαλέξει μόνο αυτές που χρειάζεται το συγκεκριμένο bundle. Οι ιδιότητες μπορούν να χρησιμοποιηθούν για να επιλεχθεί η κατάλληλη υπηρεσία ή να παίξουν και άλλους ρόλους σε επίπεδο εφαρμογής [48].



**Εικόνα 4.4:** Το μητρώο υπηρεσίων του OSGi [50]

Οι υπηρεσίες είναι δυναμικές. Αυτό σημαίνει ότι ένα bundle μπορεί να αποφασίσει να αποσύρει μία υπηρεσία της από το μητρώο, ενώ άλλα bundles εξακολουθούν να χρησιμοποιούν αυτή την υπηρεσία. Τα bundles που χρησιμοποιούν μια τέτοια υπηρεσία πρέπει να διασφαλίζουν ότι δεν χρησιμοποιούν πλέον το αντικείμενο αυτής της υπηρεσίας και να διαγράψουν οποιοδήποτε reference έχουν σε αυτή την υπηρεσία. Αυτό ίσως ακούγεται σαν να επιφέρει σημαντική πολυπλοκότητα αλλά έχουν υλοποιηθεί αρκετές τεχνολογίες οι οποίες λύνουν αυτό το πρόβλημα και επιφέρουν σημαντικά πλεονεκτήματα. Τέτοιες είναι το Service Tracker και τα Declarative Services τα οποία χρησιμοποιήθηκαν στην παρούσα εργασία. Αυτή η δυναμική των υπηρεσιών καθιστά εφικτή την εγκατάσταση και απεγκατάσταση bundles κατά την διάρκεια του run-time και τα υπόλοιπα bundles του συστήματος να μπορούν να προσαρμοστούν. Δηλαδή στην δική μας υλοποίηση μπορούμε να αλλάξουμε το bundle του Valve κατά την διάρκεια που το liqueur θερμαίνεται και το σύστημα να συνεχίζει να δουλεύει κανονικά.

**Life-Cycle:** Αποτελεί ένα API για την διαχείριση του κύκλου ζωής των bundles, δηλαδή την εγκατάσταση, την έναρξη, την αναβάθμιση, την διακοπή και την απεγκατάσταση των bundles. Επιπρόσθετα, παρέχει ένα περιεκτικό API για τα events για να επιτρέπει σε ένα bundle διαχείρισης να ελέγχει τις λειτουργίες του OSGi framework. Το επίπεδο Life-Cycle απαιτεί το Module επίπεδο, ενώ το επίπεδο Security δεν είναι απαραίτητο [51].

**Modules:** Η πλατφόρμα Java προσφέρει περιορισμένη υποστήριξη για τον διαχωρισμό σε πακέτα, την ανάπτυξη και τον έλεγχο εφαρμογών και components που βασίζονται στην γλώσσα Java. Εξαιτίας αυτού, πολλά έργα που βασίζονται σε Java, όπως τα Jboss και Netbeans,

έχουν καταφύγει στην δημιουργία προσαρμοσμένων αντίστοιχων επιπέδων (module layer) με εξειδικευμένους class loaders ώστε να παρέχουν καλύτερη υποστήριξη στην ανάπτυξη, τον διαχωρισμό σε πακέτα και τον έλεγχο εφαρμογών. Το OSGi παρέχει μια γενική και τυποποιημένη λύση για το modularization εφαρμογών.

Πιο αναλυτικά, το OSGi ορίζει μία μονάδα (module) το bundle. Ένα bundle αποτελείται από κλάσεις Java και άλλους πόρους, οι οποίοι μπορούν να παρέχουν λειτουργίες στους τελικούς χρήστες. Τα bundles μπορούν να μοιράζονται πακέτα με άλλα bundles. Στο OSGi τα bundles είναι οι μοναδικές οντότητες για την ανάπτυξη εφαρμογών. Ένα bundle αναπτύσσεται σαν ένα αρχείο .jar. Τα αρχεία .jar συνήθως χρησιμοποιούνται για την αποθήκευση των εφαρμογών και των πόρων τους σε μία τυπική μορφή αρχείου .zip. Ωστόσο, υπάρχει ένας ειδικός τύπος MIME που προορίζεται για τα OSGi bundles που χρησιμοποιείται για να διακρίνονται τα bundles από κανονικά αρχεία .jar [51].

**Security:** Είναι ένα προαιρετικό επίπεδο που βασίζεται στην αρχιτεκτονική ασφάλειας Java 2. Παρέχει την υποδομή για την ανάπτυξη και την διαχείριση εφαρμογών που πρέπει να εκτελούνται σε ελεγχόμενα και ασφαλή περιβάλλοντα [51].

#### 4.3.3 Πλεονεκτήματα του OSGi

Ο βασικός λόγος που το OSGi είναι τόσο επιτυχημένο είναι ότι παρέχει ένα πολύ ώριμο σύστημα για την ανάπτυξη εφαρμογών με την χρήση component, που λειτουργεί σε έναν εκπληκτικό αριθμό περιβάλλοντων. Το OSGi χρησιμοποιείται σε ιδιαίτερα περίπλοκες εφαρμογές όπως περιβάλλοντα ανάπτυξης κώδικα (IDEs), διακομιστές εφαρμογών, gateways καθώς και σε εφαρμογές βιομηχανικού αυτοματισμού.

Τα πλεονεκτήματα που παρέχει το OSGi σύμφωνα με το [48] αναφέρονται παρακάτω:

- **Το OSGi αρχικά δημιουργήθηκε για το IoT:** Από την αρχή το OSGi είχε σχεδιαστεί με γνώμονα συσκευές με χαμηλή επεξεργαστική ισχύ και μνήμη RAM. Οι συσκευές σήμερα έχουν μεγαλύτερη επεξεργαστική ισχύ και καλύτερους πόρους. Εξαιτίας αυτού δημιουργήθηκαν εξελιγμένες εφαρμογές και λύσεις καθώς και ακμάζοντα οικοσυστήματα οργανισμών που συνεισφέρουν στην ανάπτυξη λογισμικού. Τέτοια οικοσυστήματα μπορούν να βρεθούν σε ποικίλες αγορές όπως την αγορά των “έξυπνων σπιτιών”, των “έξυπνων” αυτοκινήτων και της Industrie 4.0. Το βιομηχανικό σύστημα παραγωγής λικέρ που χρησιμοποιήθηκε ως μελέτη στην παρούσα διπλωματική εργασία υιοθέτησε σαν βάση το OSGi για την ανάπτυξη του συστήματος παραγωγής myLiquerProduction, το οποίο εκμεταλλεύεται τις τεχνολογίες του IoT για να επιτρέπει στους τελικούς χρήστες να παράγουν προσαρμοσμένους τύπους λικέρ. Το smartSilo υλοποιήθηκε χρησιμοποιώντας τον hardware προσομοιωτή Silo που είχε αναπτυχθεί σε προηγούμενη διπλωματική εργασία

και έναν Controller σε επίπεδο λογισμικού που τρέχει σε Raspberry Pi και αναπτύχθηκε κάτω από το Apache Felix Framework το οποίο αποτελεί μία υλοποίηση ανοιχτού κώδικα του OSGi.

- **Μειωμένη πολυπλοκότητα:** Η ανάπτυξη εφαρμογών με το OSGi σημαίνει την ανάπτυξη bundles. Τα bundles κρύβουν τα εσωτερικά στοιχεία τους από τα υπόλοιπα bundles και επικοινωνούν μεταξύ τους μέσω σαφώς καθορισμένων υπηρεσιών. Η απόκρυψη των εσωτερικών στοιχείων προσδίδει μεγαλύτερη ελευθερία στην ανάπτυξη και ευκολία σε οποιαδήποτε αλλαγή που θα επέλθει αργότερα. Κάτι τέτοιο όχι μόνο μειώνει των αριθμό των σφαλμάτων, αλλά κάνει την διαδικασία ανάπτυξης των bundles πιο απλή καθώς κάθε bundle θα υλοποιεί ένα κομμάτι λειτουργικότητας μέσω σαφώς καθορισμένων interfaces. Κάθε component στην συγκεκριμένη εργασία αντιπροσωπεύει αντικείμενα πραγματικού κόσμου του παραδείγματος που υλοποιήθηκε σαν μελέτη, αλλά σε τομέα λογισμικού.
- **Επαναχρησιμοποίηση:** Το μοντέλο για τα components που παρέχει το OSGi καθιστά εύκολη τη χρήση πολλών components, που έχουν αναπτυχθεί από τρίτους, σε μία εφαρμογή. Στην υλοποίηση μας χρησιμοποιήθηκαν πολλά τέτοια bundles όπως το Californium for OSGi που είναι μία υλοποίηση του πρωτοκόλλου CoAP σε Java, ο Pax Logger που χρησιμοποιήθηκε σαν logger, η βιβλιοθήκη Pi4J που είναι υπεύθυνη για τον έλεγχο των General Purpose Inputs Outputs (GPIO) του Raspberry Pi κλπ.
- **Δυναμικότητα:** Το μοντέλο του OSGi είναι δυναμικό. Τα διάφορα bundles μπορούν να εγκαθιστώνται, να αναβαθμίζονται, να σταματάνε και να ξεκινούν χωρίς να πρέπει να σταματήσει να λειτουργεί ολόκληρο το σύστημα. Οι περισσότεροι προγραμματιστές δεν θεωρούν ότι κάτι τέτοιο είναι αξιόπιστο και για αυτό το λόγο δεν το χρησιμοποιούν σε πραγματικά περιβάλλοντα. Επιπλέον, στον κόσμο του OSGi οι υπηρεσίες που παρέχει το κάθε bundle μπορεί ανά πάσα στιγμή να μην υπάρχουν. Ωστόσο και ο πραγματικός κόσμος είναι δυναμικός και οι δυναμικές υπηρεσίες που έρχονται και φεύγουν καθιστούν τις δυναμικές υπηρεσίες του OSGi ένα τέλειο μηχανισμό που δίνει λύσει σε πολλά σενάρια πραγματικού κόσμου. Αυτή η δυναμικότητα όμως εξηγείται καλύτερα μέσω ενός παραδείγματος. Στην μελέτη που έγινε στην παρούσα εργασία μία υπηρεσία θα μπορούσε να είναι το άνοιγμα μίας βάνας ενός σιλό. Έστω ότι παρουσιάζεται ένα πρόβλημα και το λογισμικό του controller για τη συγκεκριμένη λειτουργία του συστήματος παρουσιάζει ένα σφάλμα. Τότε μπορεί να αλλαχθεί το bundle που παρέχει την υπηρεσία για το άνοιγμα της βάνας με κάποιο bundle που διορθώνει το πρόβλημα χωρίς να πρέπει να σταματήσει η λειτουργία ολόκληρου του συστήματος. Μόλις το νέο bundle εγκατασταθεί στο περιβάλλον του OSGi και να καταχωρηθεί η νέα υπηρεσία που αυτό παρέχει. Έτσι ο controller βρίσκει τη νέα υπηρεσία και την επόμενη φορά που θα γίνει μία κλήση σε αυτή την μέθοδο θα χρησιμοποιηθεί αυτή. Το συγκεκριμένο παράδειγμα ελέγχθηκε και στην υλοποίηση μας.

- Απλότητα:** Το OSGi είναι εξαιρετικά απλό παρά την διαχείριση των εξαρτήσεων μεταξύ των bundles και της δυναμικότητας που παρέχει. Ο κώδικας σε ένα bundle συμβατό με το OSGi μοιάζει σχεδόν με τον κλασικό κώδικα Java. Ένας συνδυασμός από annotations (Declarative Services) δηλώνουν στο framework πως να χρησιμοποιήσει τον δυναμικό χαρακτήρα του OSGi, καθώς και το ενημερώνουν για τις διάφορες εξαρτήσεις μεταξύ των bundles. Αυτό το πολύ απλό μοντέλο επιτρέπει τη σταδιακή χρήση πιο προηγμένων λειτουργιών.

#### 4.3.4 Υλοποιήσεις του OSGi

Το OSGi είναι ένα τυποποιημένο σύστημα για modules και μία πλατφόρμα υπηρεσιών για την γλώσσα προγραμματισμού Java. Τα πρότυπα του OSGi ορίζονται από την OSGi Alliance και δημοσιεύονται από την ίδια πρωτοβουλία [48]. Αυτή η ενότητα περιέχει μια επισκόπηση των διαθέσιμων υλοποιήσεων του OSGi, τόσο εμπορικές όσο και ανοιχτού κώδικα. Μία γρήγορη επισκόπηση απεικονίζεται στον πίνακα 4.1, που αποτελεί μία τροποποίηση του πίνακα που βρίσκετε στην [53] .

	Apache Felix (4.4) Open Source	Eclipse Equinox (4.5) Open Source	Knoplerfish (5.1) Open Source	ProSyst (SDK 7.3) Commercial
Security (v1.8)	1.7	1.8	1.7	1.5
Core Framework (v6.0)	6	6	6	6
Package Admin Service (v1.2)	1.2	1.2	1.2	1.2
Start Level Service (v1.1)	1.1	1.1	1.2	1.2
Conditional Admin Service (v1.1)	1.1	1.1	1.1	1.1
Permission Admin Service (v1.2)	1.2	1.2	1.2	1.2
URL Handler Service (v1.0)	1	1	1	1
Log Service (v1.3)	1.3	1.3	1.3	1.3
HTTP Service (v1.2)	1.2	1.2	1.2	1.2
Device Access Service (v1.1)	x	1.1	1.1	1.1
Configuration Admin Service (v1.5)	1.3	1.3	1.5	1.3
Meta-type Service (v1.2)	1.1	1.2	1.2	1.1
Preference Service (v1.1)	1.1	1.1	1.1	1.1
User Admin Service (v1.1)	1.1	1.1	1.1	x
Wire Admin Service (v1.0)	1	x	1	1
IO Connector Service (v1.3)	1.3	x	1.3	1.3
Initial Provisioning Service (v1.2)	1.2	x	1.2	1.2
UPnP Device Service (v1.2)	1.1	x	1.2	1.1
Declarative Service (v1.2)	1.2	1.2	1.2	1.1
Event Admin Service (v1.3)	1.3	1.3	1.3	1.2
Deployment Admin Service (v1.1)	1.1	x	1.1	1.1

**Πίνακας 4.1:** Διαθέσιμες υλοποιήσεις του OSGi και η έκδοση των modules που υποστηρίζουν

Όπως φαίνεται στον πίνακα 4.1 συγκρίνονται τέσσερις υλοποιήσεις του OSGi, οι τρεις εκ των οποίων είναι ανοιχτού κώδικα. Το **Apache Felix** αποτελεί μία κοινοτική προσπάθεια για την υλοποίηση του OSGi Framework και άλλων τεχνολογιών που σχετίζονται με το OSGi υπό την άδεια Apache [52]. Αποτελεί την βασική επιλογή των προγραμματιστών που ασχολούνται

με το OSGi. Αποτελεί την πιο καθαρή υλοποίηση του OSGi γι' αυτό και τα modules του που παρέχουν διάφορες υπηρεσίες των προδιαγραφών του OSGi μπορούν να λειτουργήσουν άφογα ακόμα και αν εγκατασταθούν σε κάποια άλλη υλοποίηση.

Το **Knoplerfish** είναι μία υλοποίηση του OSGi που υποστηρίζεται και αναπτύσσεται από την Makewave [54]. Είναι η πιο απλή υλοποίηση του OSGi καθώς έχει σχεδιαστεί για ενσωματωμένα συστήματα και για περιβάλλοντα με περιορισμένους πόρους. Σύμφωνα με την [53], σε μία σύγκριση των παραπάνω υλοποιήσεων αυτή του Knoplerfish αποδείχτηκε η πιο γρήγορη, όταν οι δοκιμές και οι μετρήσεις έγιναν πάνω σε ένα Raspberry Pi. Η συγκεκριμένη υλοποίηση έχει αναπτυχθεί πάνω στο Apache Felix.

Το **Equinox** είναι μία εφαρμογή των προδιαγραφών που ορίζονται από το OSGi, αλλά παρέχει και ένα σύνολο από bundles που υλοποιούν διάφορες προαιρετικές υπηρεσίες του OSGi καθώς και μία υποδομή για τη λειτουργία συστημάτων που βασίζονται στο OSGi. Υποστηρίζεται και αναπτύσσεται από το Eclipse project [55].

Τέλος, η **Prosyest** είναι μία από τις πρώτες εταιρείες που συμμετείχαν στην πρωτοβουλία OSGi Alliance και έχει συμβάλλει σημαντικά στην ανάπτυξη των προδιαγραφών του OSGi. Πλέον εστιάζει αποκλειστικά στην ανάπτυξη λογισμικού και λύσεων σχετικών με το OSGi, όπως frameworks, bundles, την υλοποίηση του OSGi που μελετήθηκε σε αυτή την ενότητα. Οι λύσεις που παρέχει χρησιμοποιούνται από συσκευές για το Smart Home, κατασκευαστές αυτοκινήτων, κατασκευαστές κινητών τηλεφώνων κ.α [56].

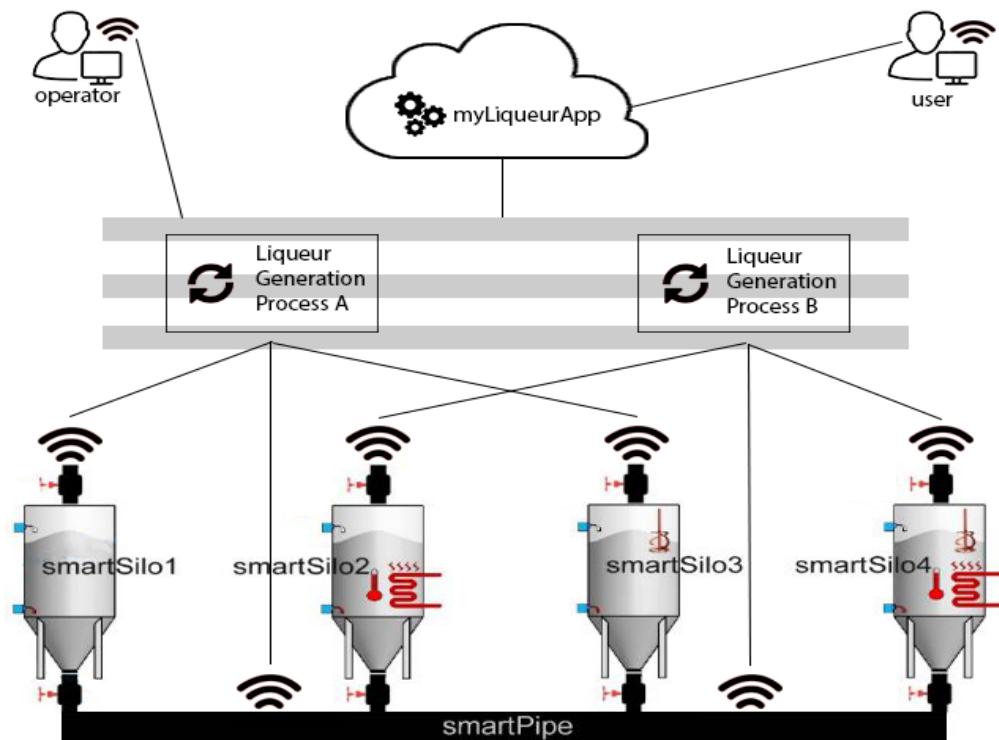
Στην παρούσα μελέτη, χρησιμοποιήθηκε η υλοποίηση που μας παρέχει το Apache Felix. Αυτό έγινε κυρίως γιατί σκοπός ήταν να δημιουργηθεί ένα περιβάλλον OSGi εξ' ολοκλήρου από την αρχή ώστε να υπάρχει έλεγχος σε οποιοδήποτε bundle έτρεχε κατά την διάρκεια του run time. Ένα άλλος λόγος που οδηγηθήκαμε σε αυτή την απόφαση ήταν και η υλοποίηση των Core προδιαγραφών του OSGi καθώς την περίοδο που εκπονήθηκε η εργασία οι υπόλοιπες υλοποιήσεις δεν παρείχαν την τελευταία έκδοση αυτών των προδιαγραφών αλλά παρείχαν κάποια προηγούμενη. Αξίζει να σημειωθεί ότι κατά την διάρκεια της εργασίας χρησιμοποιήθηκε και το **Apache Karaf** το οποίο είναι ένας container βασισμένος σε OSGi χτισμένος πάνω στο Apache Felix. Το Karaf παρέχει πολλές υπηρεσίες χωρίς να χρειάζεται να εγκατασταθούν μία προς μία τις οποίες χρειαστήκαμε στην υλοποίηση μας. Τέτοιες είναι υπηρεσίες όπως οι File Install, Declarative Services, Jetty Server, Felix Webconsole και άλλες που θα αναφερθούν στην συνέχεια. Ωστόσο, στην τελική υλοποίηση χρησιμοποιήσαμε το Apache Felix καθώς ήταν πιο ελαφρύ από το Karaf και μας παρείχε την ίδια λειτουργικότητα.



# Κεφάλαιο 5

## Το Liqueur Plant Σύστημα Παραγωγής

Στο [56] περιγράφεται ένα σύστημα παραγωγής liqueur και ένα σύστημα ελέγχου του βιομηχανικού συστήματος παραγωγής, βασισμένο σε PLC's προγραμματισμένα σε γλώσσες προτύπου IEC61131. Στα πλαίσια της παρούσας διπλωματικής εργασίας αναπτύχθηκε για το ίδιο σύστημα παραγωγής ένα σύστημα ελέγχου βασισμένο σε τεχνολογίες IoT ακολουθώντας την component-based προσέγγιση για την ανάπτυξη του λογισμικού. Στο εξής θα γίνονται αναφορές σε αυτό το βιομηχανικό σύστημα παραγωγής σαν LPPS. Το συγκεκριμένο βιομηχανικό σύστημα έχει μελετηθεί ήδη και στις [57], [58], [59]. Έτσι μελετήθηκε, σχεδιάστηκε και υλοποιήθηκε ένα σύστημα παραγωγής δύο τύπων liqueur βασισμένο στις προδιαγραφές του LPPS, το οποίο φαίνεται στην εικόνα 5.1:



**Εικόνα 5.1:** Σχηματική αναπαράσταση του συστήματος παραγωγής liqueur [59]

Το σύστημα αποτελείται από τέσσερα σιλό συνδεδεμένα μεταξύ τους με ένα σωλήνα. Κάθε ένα σιλό αποτελείται από μία βαλβίδα εισαγωγής και μία βαλβίδα εξαγωγής υγρού, καθώς και από δύο αισθητήρες για τον έλεγχο της παρουσίας υγρού, έναν στο πάνω μέρος ο οποίος ανιχνεύει αν το σιλό έχει γεμίσει και έναν στο κάτω μέρος που ανιχνεύει αν το σιλό έχει αδειάσει. Επιπλέον, δύο από τα σιλό, το smartSilo2 και το smartSilo4, έχουν ένα στοιχείο θέρμανσης του υγρού και έναν αισθητήρα που ανιχνεύει την θερμοκρασία του υγρού. Τα smartSilo3 και smartSilo4 περιλαμβάνουν από ένα στοιχείο μίξης του υγρού.

Στο συγκεκριμένο σύστημα παράγονται δύο τύποι liqueur ακολουθώντας την εξής διαδικασία:

- **Παραγωγή τύπου liqueur A:** Για να ξεκινήσει η παραγωγή του liqueur τύπου A, το σιλό 1 γεμίζει με υγρό. Μόλις γεμίσει θα πρέπει να μεταφερθεί το υγρό αυτό μέσω του σωλήνα στο σιλό 4. Στη συνέχεια μόλις μεταφερθεί όλο το υγρό στο σιλό 4, θερμαίνεται μέχρι να φτάσει σε μία επιθυμητή θερμοκρασία και τέλος αναδεύεται για ένα συγκεκριμένο χρονικό διάστημα με την χρήση του στοιχείου μίξης του σιλό 4. Τέλος το υγρό μεταφέρεται σε άλλο σημείο παραγωγής μέσω της βαλβίδας εξαγωγής του σιλό 4.
- **Παραγωγή τύπου liqueur B:** Για την παραγωγή liqueur τύπου B το σιλό 2 γεμίζει με υγρό. Μόλις γεμίσει το υγρό θερμαίνεται μέχρι να φτάσει σε μία επιθυμητή θερμοκρασία. Στη συνέχεια θα πρέπει να μεταφερθεί μέσω του σωλήνα στο σιλό 3. Αφού μεταφερθεί το υγρό αναδεύεται για ένα συγκεκριμένο χρονικό διάστημα μέσω του στοιχείου μείξης του σιλό 3. Τέλος, το υγρό μεταφέρεται σε άλλο σημείο παραγωγής μέσω της βαλβίδας εξαγωγής του σιλό 3.

Οι δύο τύποι liqueur που μπορούν να παραχθούν καθώς και η διαδικασίες που ακολουθούνται για την παραγωγή τους είναι ανεξάρτητες και μπορούν να γίνουν ταυτόχρονα. Άλλα θα πρέπει να ληφθούν υπόψιν οι εξής περιορισμοί. Ο σωλήνας είναι κοινός πόρος και μπορεί να υποστηρίξει μία μεταφορά κάθε φορά. Δηλαδή μεταφορά υγρού από το σιλό 1 στο σιλό 4 ή μεταφορά υγρού από το σιλό 2 στο σιλό 3. Επιπλέον, υπάρχει και περιορισμός στην κατανάλωση ισχύος και γι' αυτό θα πρέπει να λειτουργεί μόνο ένας αναδευτήρας κάθε φορά και όχι και οι δύο ταυτόχρονα.

Οποιοσδήποτε χρήστης του myLiqueurApp θα μπορεί να ορίζει τον τύπο του liqueur που θέλει καθώς και τις διάφορες ιδιότητες του, δηλαδή σε ποια θερμοκρασία πρέπει να θερμανθεί και για πόση ώρα θα πρέπει να αναδευτεί καθώς και την ποσότητα που θέλει. Επιπλέον, θεωρούμε ότι τα σιλό απέχουν αρκετά μεταξύ τους ώστε να έχει σημασία η χρήση τεχνολογιών IoT για την επικοινωνία τους.

## Κεφάλαιο 6

# Σχεδιασμός του συστήματος

### 6.1 Εισαγωγή

Στο κεφάλαιο αυτό θα γίνει μία περιγραφή του σχεδιασμού του συστήματος ελέγχου του κάθε σιλό του LPPS ξεχωριστά. Για να γίνει αυτό και το κάθε σιλό να είναι σε θέση να αναβαθμιστεί χρησιμοποιώντας τεχνολογίες IoT είναι απαραίτητη η ενσωμάτωση ενός μικρούπολογιστή με δυνατότητες δικτύωσης σε κάθε σιλό. Από εδώ και στο εξής ένα τέτοιο σύστημα, δηλαδή τα μηχανικά στοιχεία ενός σιλό σε συνεργασία με τον μικρούπολογιστή, θα καλείται Industrial Automation Thing (IAT).

Αυτοί οι μικρούπολογιστές, αποτελούν τους LwM2M clients του συστήματος και θα παρέχουν υπηρεσίες στο περιβάλλον τους που μπορεί να τις αξιοποιήσει ένας LwM2M server ώστε να επιτυγχάνεται η παραγωγή liqueur που είναι και το ζητούμενο.

Για τον σχεδιασμό του συστήματος χρησιμοποιήθηκε η γλώσσα UML, που είναι μια γλώσσα γενικής χρήσης, ανάπτυξης και μοντελοποίησης στον τομέα της μηχανικής λογισμικού και παρέχει έναν τυποποιημένο τρόπο απεικόνισης του σχεδιασμού ενός συστήματος. Η υλοποίηση των διαγραμμάτων UML έγινε με την βοήθεια του εργαλείου Papyrus.

### 6.2 Σχεδιασμός του API του Industrial Automation Thing

Για να γίνει ο σχεδιασμός του API του IAT έπρεπε αρχικά να ληφθούν υπόψιν όλα τα κρίσιμα τμήματα των διεργασιών που μπορεί να εκτελέσει ένα σιλό. Τα κρίσιμα τμήματα που σημειώθηκαν και σύμφωνα με τα οποία έγινε ο σχεδιασμός του συστήματος είναι τα παρακάτω:

- Όταν ένα σιλό γεμίζει με υγρό πρέπει οι πάνω βαλβίδα να κλείσει αμέσως μόλις ανιχνευθεί υγρό. Αν δεν γίνει αυτό το σιλό μπορεί να συνεχίζει να γεμίζει πέρα από το επιθυμητό επίπεδο.
- Όταν ένα σιλό αδειάζει από υγρό η κάτω βαλβίδα πρέπει να κλείσει αμέσως μόλις ο κάτω αισθητήρας του σιλό ανιχνεύσει απουσία υγρού.
- Όταν ένα σιλό θερμανθεί μέχρι μία συγκεκριμένη θερμοκρασία τότε το στοιχείο θέρμανσης του σιλό πρέπει να απενεργοποιηθεί άμεσα ώστε να μην ξεπεραστεί η θερμοκρασία αυτή.

- Όταν ένα σιλό έχει αναδευτεί για το επιθυμητό χρονικό διάστημα το στοιχείο μίξης πρέπει να απενεργοποιηθεί αλλιώς μπορεί να μην έχουμε το επιθυμητό αποτέλεσμα στον τελικό προϊόν.

Παρατηρείται ότι οι κρίσιμες λειτουργίες αφορούν τα μηχανικά στοιχεία και τις λειτουργίες μόνο ενός σιλό και έτσι δεν χρειάζεται επιπλέον γνώση οποιασδήποτε πληροφορίας ή αλληλεπίδρασης με το περιβάλλον του. Αυτό σε συνδυασμό με την παραδοχή ότι το κάθε σιλό βρίσκεται σε μεγάλη απόσταση από τα υπόλοιπα καθιστά σαφές ότι θα πρέπει να υπάρχει ενσωματωμένος ένας μικροϋπολογιστής σε κάθε σιλό ξεχωριστά και ότι το κάθε σιλό θα αποτελεί και έναν LwM2M client. Έτσι σύμφωνα με τις προδιαγραφές του OMA LwM2M ορίστηκαν κάποια συγκεκριμένα LwM2M objects καθώς και resources, μέσω των οποίων γίνεται η πρόσβαση στις υπηρεσίες που παρέχει το κάθε σιλό. Αυτά αναλύονται καλύτερα στην επόμενη ενότητα.

### 6.3 Περιγραφή των LwM2M objects και resources

Παρακάτω αναφέρονται και περιγράφονται τα LwM2M objects που ορίστηκαν για την ανάπτυξη του συστήματος ελέγχου του κάθε σιλό:

- Object **Silo**: παρέχει τις υψηλού επιπέδου υπηρεσίες του κάθε σιλό
  - Resource **state**: αναπαριστά ανά πάσα στιγμή την κατάσταση στην οποία βρίσκεται κάθε σιλό. Η κατάσταση ενός σιλό μπορεί να είναι Empty, Empting, Filling, Full, Heating, Heated, Mixing, Mixed.
  - Resource **initialize**: χρησιμοποιείται για διάφορες αρχικοποιήσεις που μπορεί να χρειαστεί το σιλό.
  - Resource **fill**: μέσω αυτού δίνεται η εντολή για να ξεκινήσει η διαδικασία γεμίσματος του σιλό.
  - Resource **empty**: μέσω αυτού δίνεται η εντολή για να ξεκινήσει η διαδικασία αδειάσματος του σιλό.
  - Resource **heat**: μέσω αυτού δίνεται η εντολή για να ξεκινήσει η διαδικασία θέρμανσης του υγρού που υπάρχει στο σιλό.
  - Resource **mix**: μέσω αυτού δίνεται η εντολή για να αρχίσει η διαδικασία ανάδευσης του υγρού.
  - Resource **stop\_filling**: χρησιμοποιείται για να σταματήσει τη διαδικασία γεμίσματος του σιλό χωρίς αυτό να έχει γεμίσει μέχρι το μέγιστο σημείο.
  - Resource **stop\_emptying**: χρησιμοποιείται για να σταματήσει η διαδικασία αδειάσματος του σιλό προτού το υγρό φτάσει στο χαμηλότερο επίπεδο του σιλό.

- Resources **filling\_completed**, **emptying\_completed**, **mixing\_completed**, **heating\_completed**: Μέσω αυτών ενημερώνεται οποιοσδήποτε ενδιαφερόμενος σχετικά με το πέρας μίας λειτουργίας. Ο κάθε ενδιαφερόμενος θα πρέπει να εγκαθιδρύσει μία σχέση observe-notify με το αντίστοιχο resource και θα λαμβάνει true κάθε φορά που μία διαδικασία του σιλό ολοκληρώνεται.
- Resource **target\_temperature**: χρησιμοποιείται για να οριστεί η θερμοκρασία μέχρι την οποία πρέπει να θερμανθεί το υγρό. Μόλις το θερμόμετρο ανιχνεύσει αυτή την θερμοκρασία τότε το υγρό σταματά να θερμαίνεται.
- Resource **event**: είναι ένα json string που περιέχει πληροφορίες για κάθε event του σιλό. Τέτοιες πληροφορίες είναι ο αποστολέας, το event (Fill, Empty, Heat, κ.ο.κ) καθώς και η ημερομηνία και η ώρα της αποστολής.
- Object **Valve**: παρέχει πρόσβαση σε υπηρεσίες χαμηλού επιπέδου μία βαλβίδας ροής υγρού.
  - Resource **on/off**: Όταν έχει την τιμή true τότε η βαλβίδα ροής υγρού είναι ανοιχτή, ειδάλλως είναι κλειστή.
- Object **Heater**: αναφέρεται στο στοιχείο θέρμανσης του σιλό.
  - Resource **on/off**: Όταν έχει την τιμή true τότε η το υγρό θερμαίνεται, ειδάλλως το στοιχείο θέρμανσης είναι κλειστό.
- Object **Mixer**: αναφέρεται στο στοιχείο μίξης του σιλό.
  - Resource **time to operate**: Ορίζει τον χρόνο που πρέπει να αναδευτεί το υγρό.
  - Resource **on/off**: Όταν έχει την τιμή true τότε το στοιχείο μίξης είναι ενεργό, ενώ αν έχει την τιμή false το στοιχείο μίξης δεν λειτουργεί.
- Object **Level Sensor**: παρέχει πρόσβαση σε υπηρεσίες που παρέχει ένας ανιχνευτής παρουσίας υγρού.
  - Resource **sensor type**: επιστρέφει τον τύπο του αισθητήρα.
  - Resource **digital output state**: Γίνεται true όταν ανιχνευθεί υγρό από τον αισθητήρα αλλιώς είναι false.

Ta LwM2M objects και resources που ορίσαμε φαίνονται πιο αναλυτικά και στον πίκανα 6.1. Αξίζει να σημειωθεί ότι μερικά από τα resources, όπως για παράδειγμα τα on/off resources υπάρχουν στο resource registry του OMA LwM2M. Αυτά τα resources φαίνονται στον πίνακα 6.1 στα πράσινα πεδία.

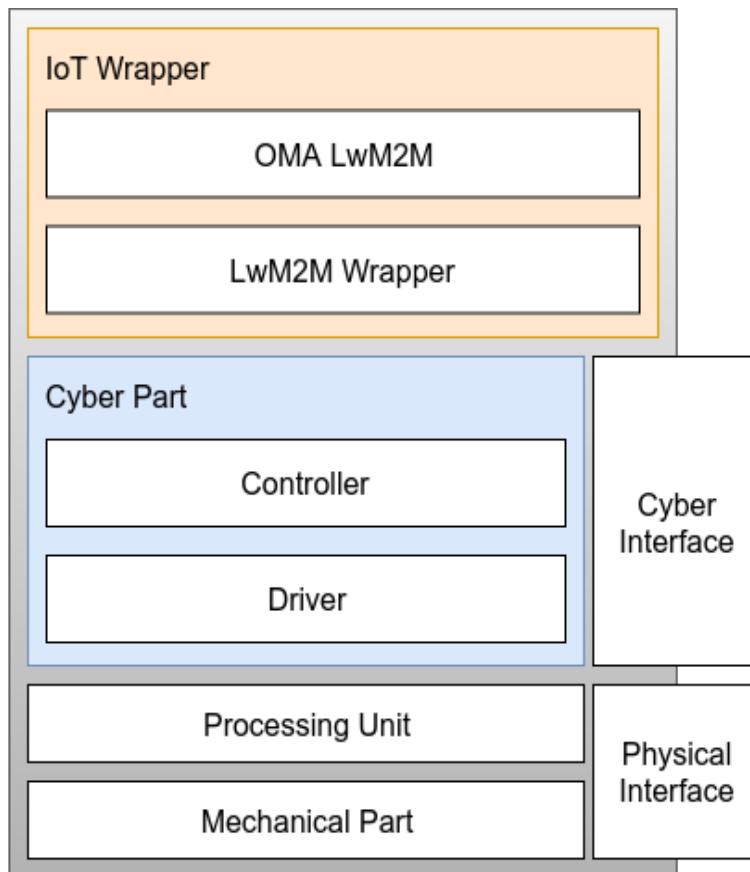
ID	Object	Resource	Resource ID	RWE	Single	Mandatory	Type
20000	Silo				N	Y	
		state	0	R	Y	Y	String
		initialize	1	E	Y	Y	Boolean
		fill	2	E	Y	Y	Boolean
		empty	3	E	Y	Y	Boolean
		heat	4	E	Y	Y	Boolean
		mix	5	E	Y	Y	Boolean
		stop_filling	6	E	Y	Y	Boolean
		stop_emptying	7	E	Y	Y	Boolean
		filling completed	10	R	Y	Y	Boolean
		emptying completed	11	R	Y	Y	Boolean
		heating completed	12	R	Y	N	Boolean
		mixing completed	13	R	Y	N	Boolean
		target temperature	14	RW	Y	N	Float
		Event	15	R	Y	Y	String
20001	Valve				N	N	
		on/off	5850	RW	Y	Y	Boolean
20002	Level Sensor	sensor type digital output state	5751	R	Y	N	String
20004	Mixer				N	N	
		time to operate	0	RW	Y	Y	Integer
		on/off	5850	RW	Y	Y	Boolean
20005	Heater				N	N	
		on/off	5850	RW	Y	Y	Boolean

Πίνακας 6.1: LwM2M objects και Resources

## 6.4 Αρχιτεκτονικός σχεδιασμός του συστήματος ελέγχου του σιλό

Μια αφαιρετική περιγραφή της αρχιτεκτονικής του σιλό φαίνεται στην εικόνα 6.1. Το σιλό αποτελείται από τα μηχανικά του μέρη, που είναι όλοι οι αισθητήρες, οι βαλβίδες, το στοιχείο μίξης και το στοιχείο θέρμανσης. Αυτά τα μέρη πρέπει να επικοινωνούν με κάποιο λογισμικό ώστε να μπορούν να ελεγχθούν. Η επικοινωνία των μηχανικών στοιχείων του σιλό με το λογισμικό γίνεται μέσω του driver. Ο driver ή οι drivers με τη σειρά τους αποστέλλουν πληροφορίες στον Controller του σιλό ο οποίος αποτελεί την καρδιά του συστήματος ελέγχου και με την σειρά του μπορεί να στείλει κάποιο μήνυμα στους drivers ώστε να γίνει μία ενέργεια στα μηχανικά μέρη του σιλό. Για να ενταχθεί όμως το σύστημα αυτό στο διαδίκτυο των αντικειμένων πρέπει

να υπάρχει συνδεσιμότητα με το διαδίκτυο. Αυτό γίνεται μέσω του LwM2M Wrapper ο οποίος δίνει τη δυνατότητα στο σιλό να επικοινωνεί με το δίκτυο χρησιμοποιώντας το πρωτόκολλο LwM2M. Έτσι το σιλό εντάσσεται στο διαδίκτυο των αντικειμένων και παρέχει τις υπηρεσίες του στο περιβάλλον ώστε να μπορέσουν να χρησιμοποιηθούν από έναν LwM2M server για την παραγωγή ενός τύπου liqueur.



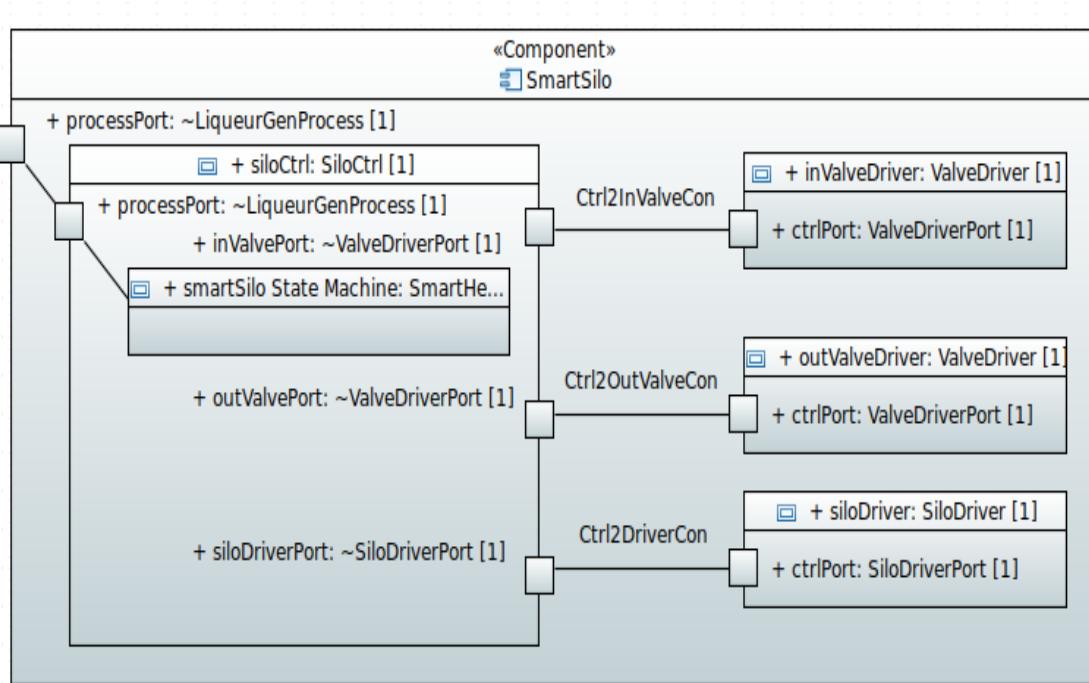
**Εικόνα 6.1:** Η αρχιτεκτονική του σιλό

Η μεταφορά μηνυμάτων από τους drivers προς το σύστημα ελέγχου και το ανάποδο και από το δίκτυο προς το σιλό αλλά και από το σιλό προς το δίκτυο γίνεται με την χρήση Events. Στην συνέχεια θα γίνει μια εκτενέστερη αναφορά στον τρόπο που γίνεται αυτό καθ' όλη την διάρκεια λειτουργίας του συστήματος.

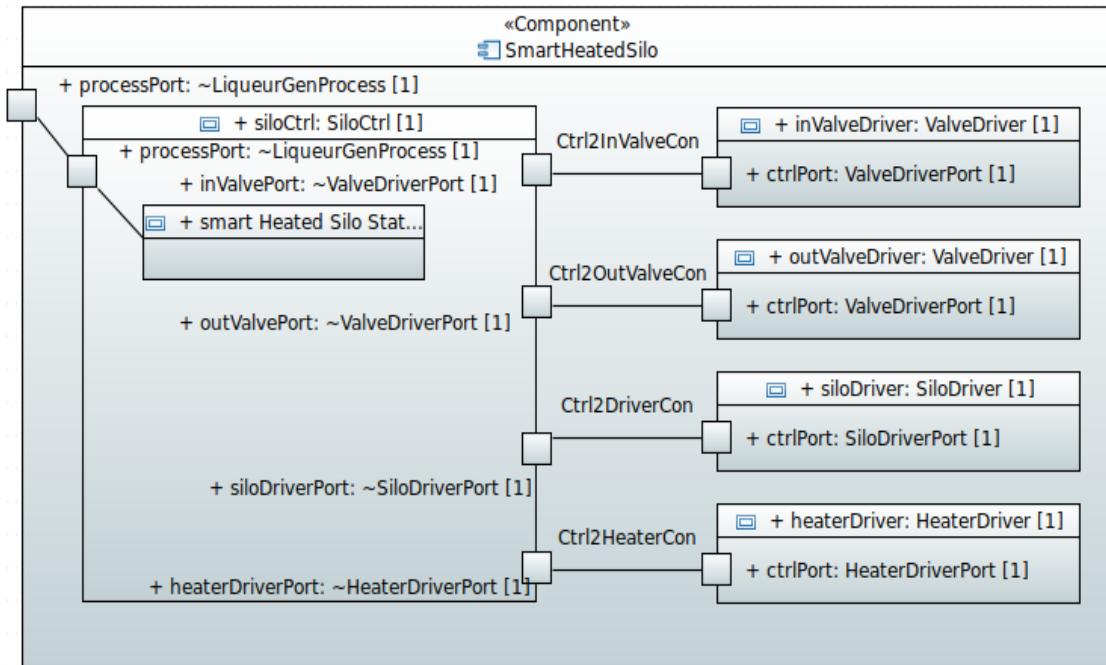
## 6.5 Δομή των σιλό

'Οπως έχει ήδη αναφερθεί, στην παρούσα εργασία η ανάπτυξη του λογισμικού έγινε βασισμένη σε συνιστώσες. Για να γίνει κάτι τέτοιο έπρεπε πρώτα να ξεχωριστούν οι διάφορες συνιστώσες από τις οποίες απαρτίζεται το σύστημα. Όπως φαίνεται και στην εικόνα 6.1 στην αρχιτεκτονική του σιλό το cyber μέρος του αποτελείται από τον driver ή τους drivers αν αυτοί είναι περισσότεροι από έναν και τον controller. Εξαιτίας αυτού έγινε η επιλογή οι διάφοροι

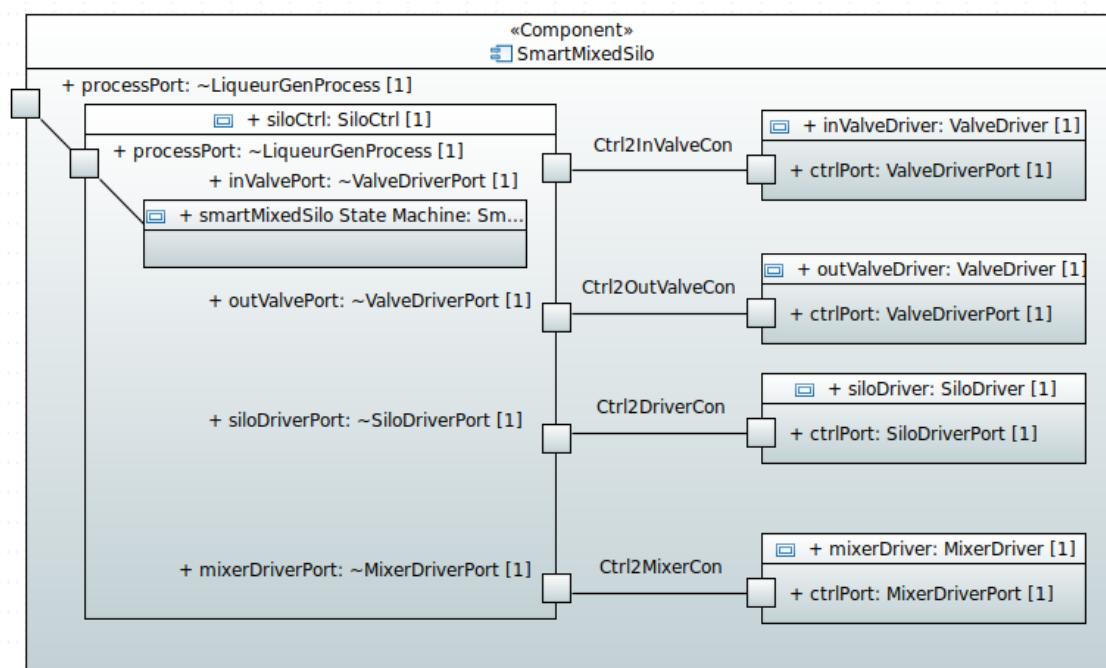
drivers να είναι ξεχωριστά components στο σύστημα μας. Το ίδιο ισχύει και για τον controller του σιλό. Κάτι τέτοιο μας επιτρέπει να επωφεληθούμε από πολλά πλεονεκτήματα που μας δίνει τόσο αυτή η προσέγγιση όσο και το συγκεκριμένο framework (OSGi) που χρησιμοποιήσαμε, καθώς σε περίπτωση που χρειαστεί κάποια αλλαγή στο λογισμικό του συστήματος τότε αυτή μπορεί να γίνει κατά την διάρκεια που το σύστημα λειτουργεί χωρίς να επιφέρει κάποιο πρόβλημα στην λειτουργία όλου του συστήματος ελέγχου. Στις εικόνες 6.2, 6.3, 6.4, 6.5 φαίνεται η εσωτερική δομή του cyber μέρους των σιλό την συμπεριφορά του σιλό και την υποδομή των drivers του σιλό. Μέσω των διάφορων ports ο controller του σιλό επικοινωνεί με τους drivers και το περιβάλλον του.



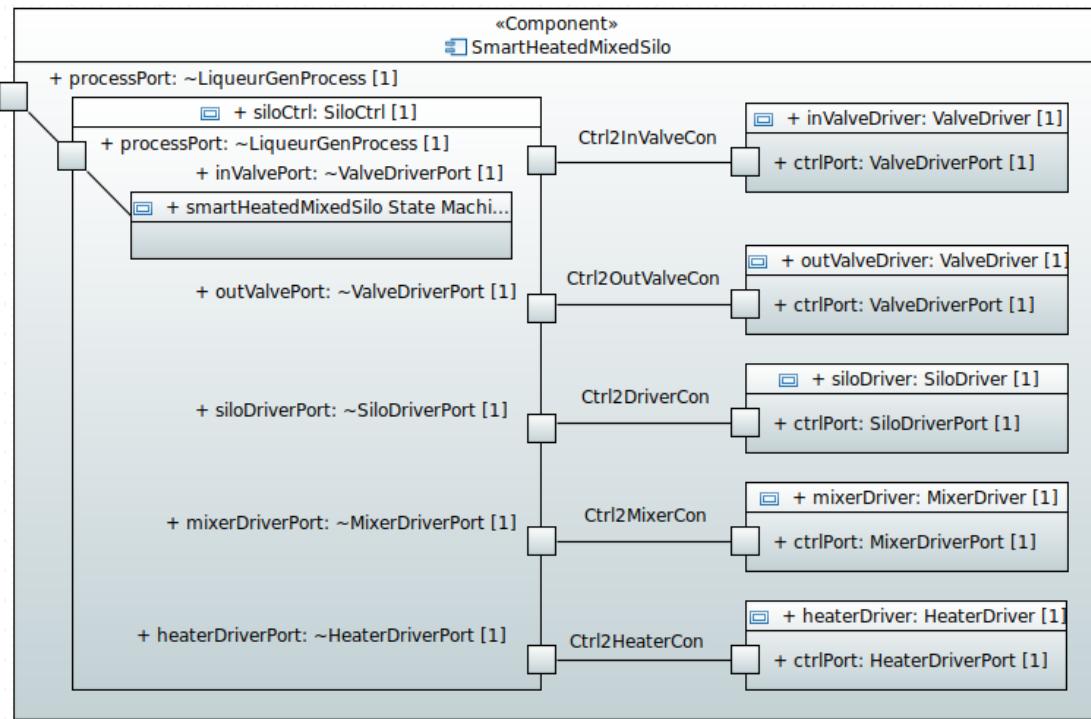
**Εικόνα 6.2:** Η δομή του smartSilo1



Εικόνα 6.3: Η δομή του smartSilo2



Εικόνα 6.4: Η δομή του smartSilo3



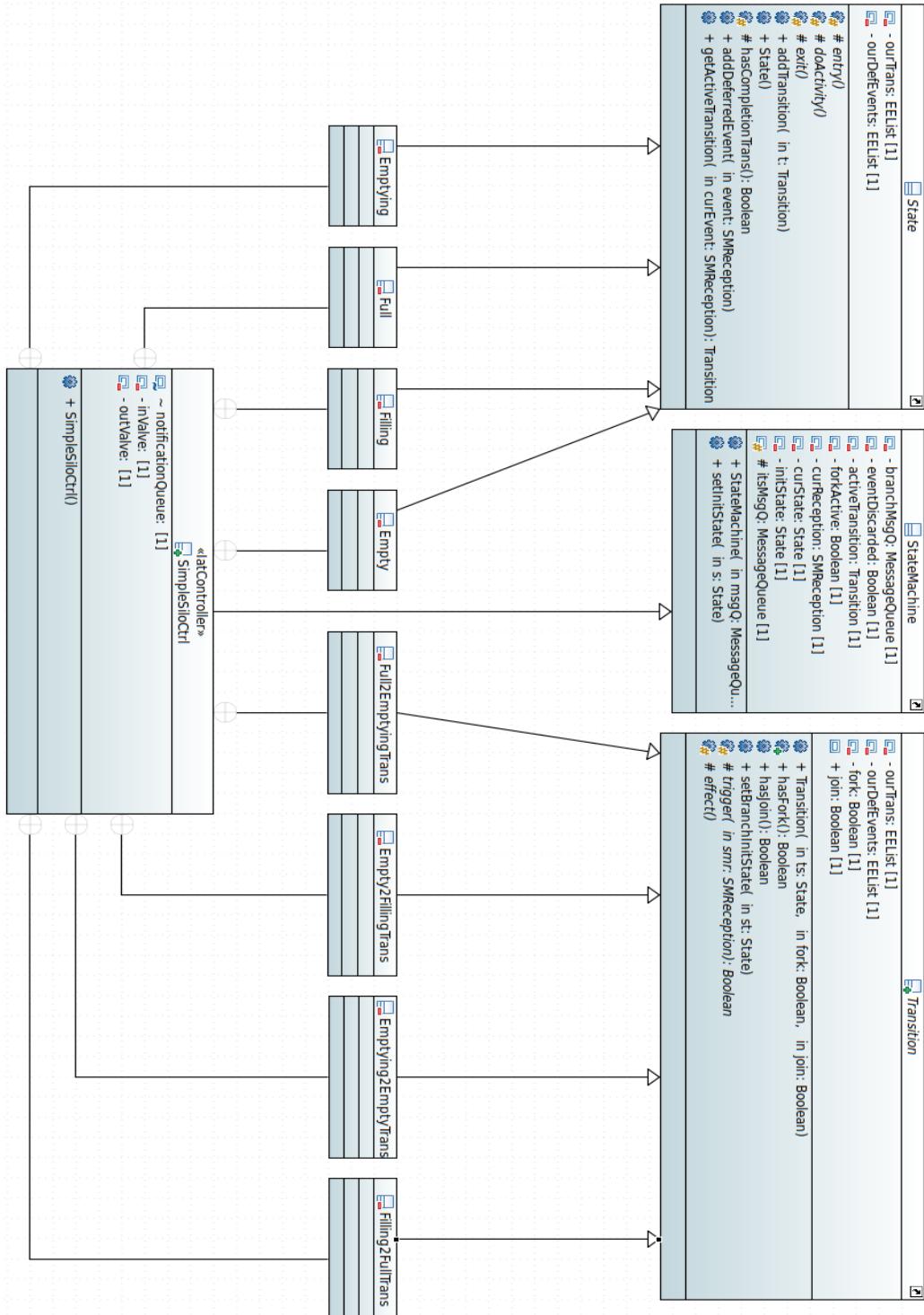
**Εικόνα 6.5:** Η δομή του smartSilo4

Η δομή της συνιστώσας του controller του σιλό φαίνεται στο διάγραμμα κλάσεων της εικόνας 6.6. Η notificationQueue αποτελεί την EventQueue του controller στην οποία έρχονται όλα τα μηνύματα από το περιβάλλον του σιλό. Επιπλέον μέσα από το διάγραμμα μπορούμε να ξεχωρίσουμε τις διάφορες καταστάσεις του σιλό καθώς και τον τρόπο μετάβασης από την μία μετάβαση στην άλλη μέσω των διάφορων Transitions. Παρατηρούμε επίσης ότι έχουμε ορίσει δύο κλάσεις τις State και Transition οι οποίες αποτελούν υπερικλάσεις και κληρονομούνται από τις επιμέρους καταστάσεις του σιλό και από τις μεταβάσεις από την μία κατάσταση στην άλλη αντίστοιχα.

## 6.6 Συμπεριφορά του συστήματος ελέγχου

Κάθε σιλό μπορεί να θεωρηθεί ότι κάθε στιγμή βρίσκεται σε μία συγκεκριμένη κατάσταση μέσα από ένα πεπερασμένο πλήθος καταστάσεων. Με την λήψη ενός μηνύματος είτε από τους drivers είτε από έναν LwM2M server το σιλό εκτελεί τις απαραίτητες λειτουργίες και μεταβαίνει στην επόμενη κατάσταση του.

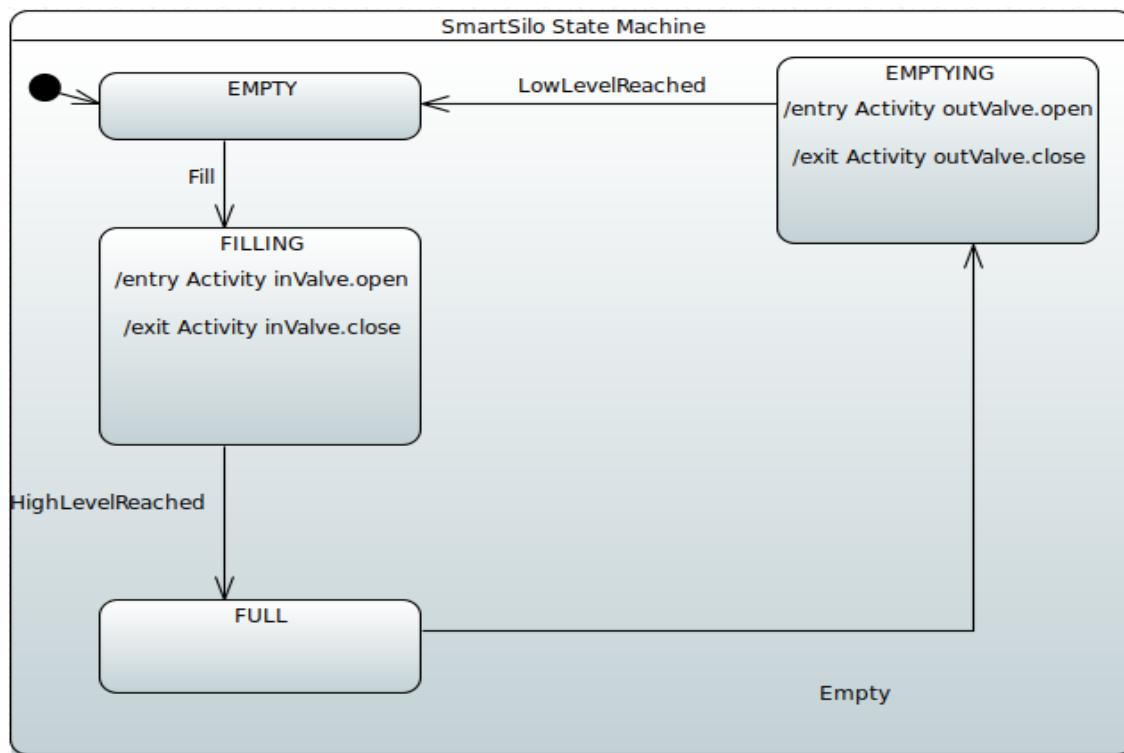
Ας πάρουμε για παράδειγμα, τις καταστάσεις του σιλό που περιλαμβάνει στοιχείο θέρμανσης την στιγμή που ξεκινά η λειτουργία παραγωγής liqueur τύπου B. Αρχικά το σιλό είναι άδειο και βρίσκεται στην κατάσταση **EMPTY**. Όταν ο LwM2M client λάβει το μήνυμα **Fill** από τον LwM2M server τότε θα ανοίξει η πάνω βαλβίδα εισαγωγής υγρού και το σιλό θα αρχίσει να



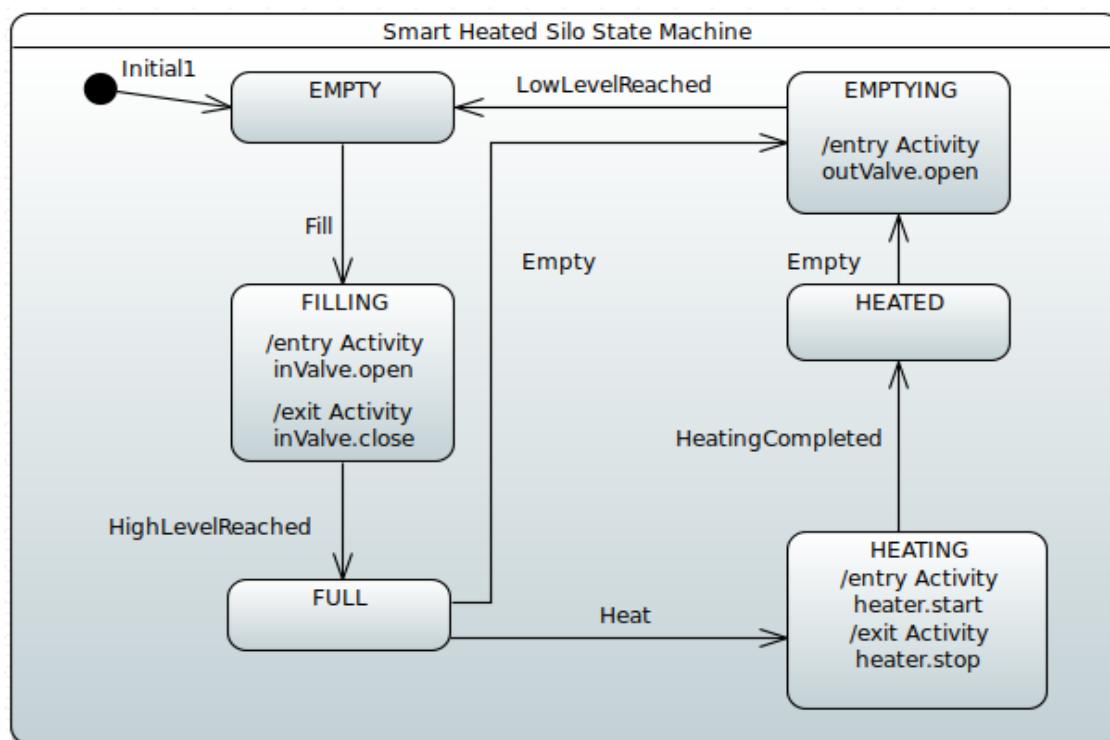
**Εικόνα 6.6:** Το διάγραμμα κλάσεων του state machine του σιλό

γεμίζει με liqueur και θα μεταβεί στην κατάσταση **FILLING**. Στην συνέχεια, αν ο άνω αισθητήρας ανίχνευσης υγρού ενεργοποιηθεί, δηλαδή ανιχνεύσει υγρό τότε ο driver στέλνει το μήνυμα StopFilling και το σιλό μεταβαίνει στην κατάσταση **FULL**. Ένα τέτοιο μήνυμα μπορεί να έρθει και από τον LwM2M server σε περίπτωση που κάποιος θέλει να διακόψει την εισαγωγή υγρού στο σιλό. Στην συνέχεια έρχεται από τον server ένα μήνυμα Heat και αρχίζει η λειτουργία του στοιχείου θέρμανσης του σιλό, το οποίο εισέρχεται στην κατάσταση **HEATING** μέχρις ότου φτάσει το υγρό στην επιθυμητή θερμοκρασία. Όταν γίνει αυτό ο αισθητήρας θερμοκρασίας στέλνει μέσω του driver ένα μήνυμα HeatingCompleted και το σιλό μεταβαίνει στην κατάσταση **HEATED**. Τέλος, με ένα μήνυμα Empty το σιλό μεταβαίνει στην κατάσταση **EMPTYING** και ανοίγει η βαλβίδα εξαγωγής υγρού. Μόλις το υγρό αδειάσει το σιλό μεταβαίνει στην κατάσταση **EMPTY** και η διαδικασία επαναλαμβάνεται. Αυτή η διαδικασία αναπαρίσταται από το state machine της εικόνας 6.7. Παρόμοια συμπεριφορά έχουν και τα υπόλοιπα σιλό όπως φαίνεται και στις εικόνες 6.6, 6.8, 6.9.

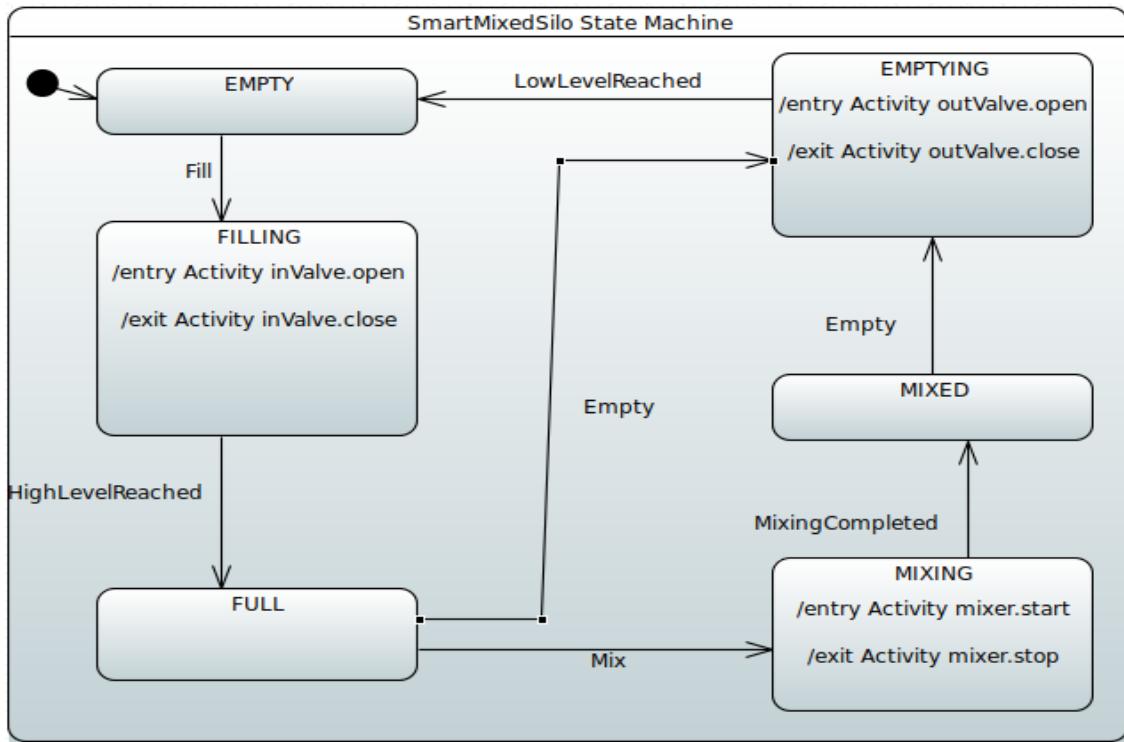
Παρατηρούμε ότι όταν το σιλό βρίσκεται σε μία κατάσταση τότε πρέπει να εκτελεστεί μία λειτουργία από κάποιον driver. Όπως για παράδειγμα όταν το σιλό βρίσκεται στην κατάσταση **FILLING** ο driver πρέπει να στείλει μήνυμα στην άνω βαλβίδα ροής υγρού του σιλό ώστε αυτή να ανοίξει και όταν ο άνω αισθητήρας ανίχνευσης υγρού του σιλού ανιχνεύσει υγρό τότε η βαλβίδα πρέπει να κλείσει. Η επικοινωνία αυτή γίνεται μέσω της EventQueue του controller. Πιο αναλυτικά, όταν ο LwM2M server που εκτελεί μια διεργασία παραγωγής liqueur ενός εκ των δύο τύπων στείλει ένα μήνυμα στον LwM2M client του σιλό, τότε μέσω των υπηρεσίων που παρέχει η συνιστώσα του controller στο περιβάλλον της τα μηνύματα αυτά μεταφέρονται στην EventQueue του controller και από εκεί αποστέλονται τα ανάλογα μηνύματα στους drivers του σιλό μέσα από τις υπηρεσίες που η αντίστοιχη συνιστώσα του εκάστοτε driver παρέχει στο περιβάλλον της.



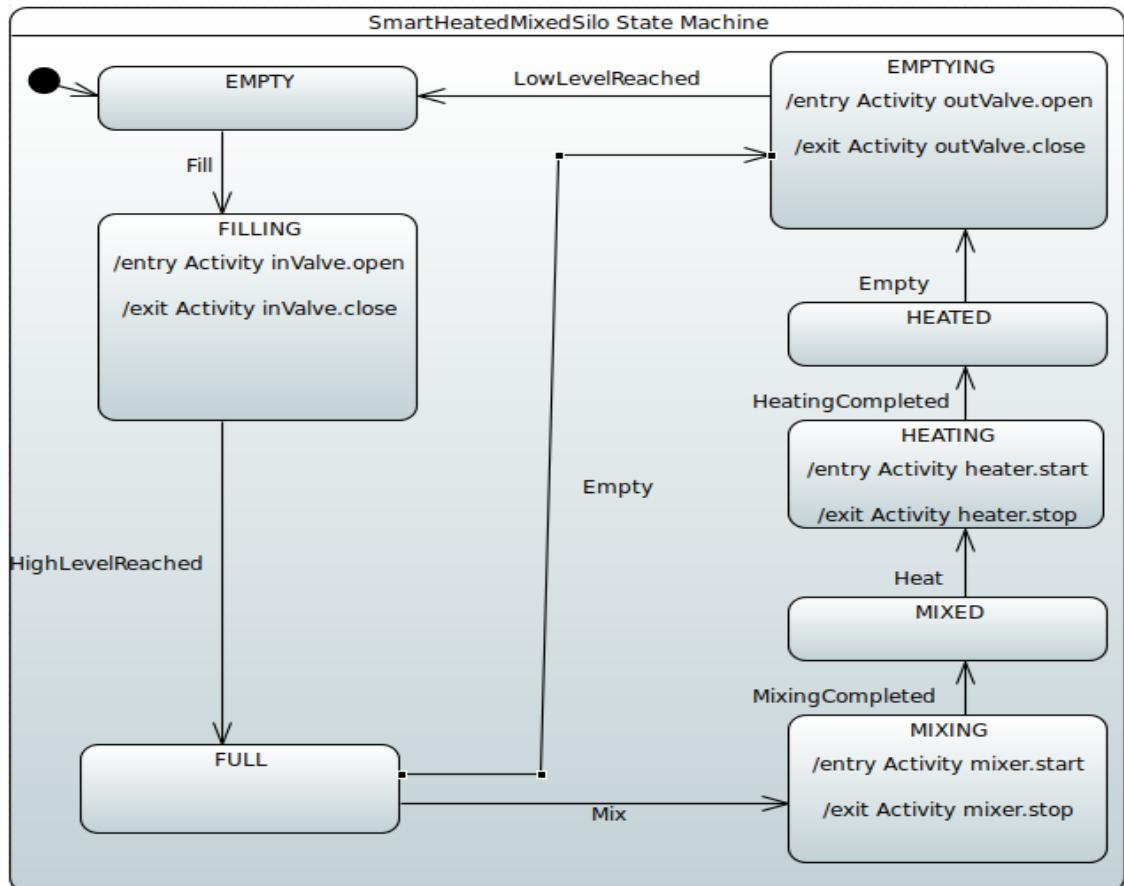
Εικόνα 6.7: To state machine του smartSilo1



Εικόνα 6.8: To state machine του smartSilo2



Εικόνα 6.9: To state machine του smartSilo3



Εικόνα 6.10: To state machine του smartSilo4

## Κεφάλαιο 7

# Υλοποίηση του συστήματος

### 7.1 Γενικά

Σε αυτή την ενότητα θα γίνει μία εκτενής περιγραφή στην ανάπτυξη του συστήματος ελέγχου του σιλό σύμφωνα με τις προδιαγραφές που αναφέρθηκαν παραπάνω. Η υλοποίηση του συστήματος ελέγχου έγινε σε Java χρησιμοποιώντας την component-based προσέγγιση για την ανάπτυξη του λογισμικού. Χρησιμοποιήθηκε το OSGi framework καθώς παρείχε αρκετά πλεονεκτήματα και αρκετές υπηρεσίες που κρίθηκαν πολὺ βοηθητικές. Επιπλέον, χρησιμοποιώντας Java για την ανάπτυξη του συστήματος ελέγχου έχουμε αρκετές επιλογές για το υλικό που μπορεί να χρησιμοποιηθεί καθώς η συγκεκριμένη γλώσσα μπορεί να τρέξει σε μεγάλο εύρος συσκευών. Επιπλέον, επιλέγοντας την συγκεκριμένη γλώσσα προγραμματισμού δίνεται η δυνατότητα χρήσης δύο βιβλιοθηκών που υλοποιούν τα πρωτόκολλα CoAP και LwM2M, τις Californium και Leshan αντίστοιχα, καθώς και της βιβλιοθήκης p4j [60] που μας επιτρέπει να χρησιμοποιήσουμε τα γενικού σκοπού I/O του Raspberry Pi μέσω της Java.

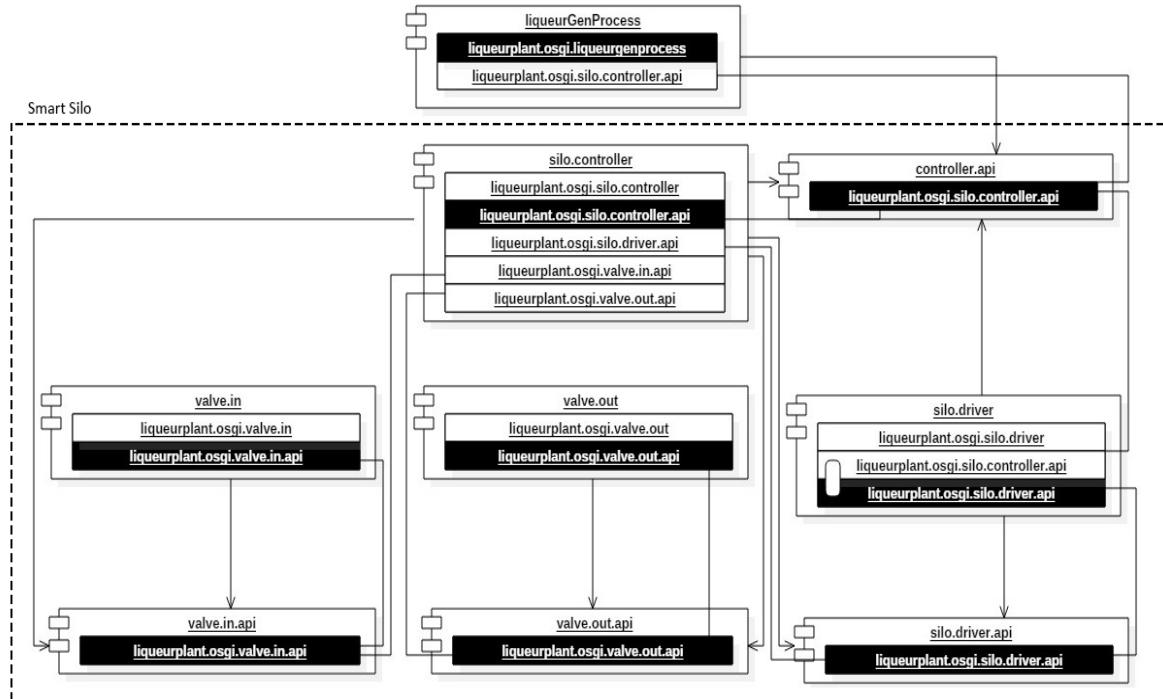
Στην συγκεκριμένη εργασία χρησιμοποιήθηκαν σαν ενσωματωμένοι μικροϋπολογιστές Raspberry Pi 3 Model B και Raspberry Pi 2 Model B. Η υλοποίηση του συστήματος ελέγχου έγινε στο IntelliJ IDEA IDE της Jetbrains καθώς υπήρχε άδεια για την χρησιμοποίηση της έκδοσης Ultimate μέσω του Πανεπιστημίου. Τέλος χρησιμοποιήθηκε το σύστημα ελέγχου πηγαίου κώδικα Git σε συνδυασμό με την υπηρεσία Github.

### 7.2 Οι συνιστώσες του συστήματος ελέγχου

Όπως έχει ήδη αναφερθεί, η υλοποίηση του λογισμικού του συστήματος έγινε χρησιμοποιώντας την προσέγγιση συνιστώσων με την βοήθεια του OSGi framework. Για να γίνει κάτι τέτοιο έπρεπε αρχικά να ξεχωριστούν οι διάφορες συνιστώσες του συστήματος. Φυσικά και το σύστημα ελέγχου θα μπορούσε να υλοποιηθεί σε ένα μόνο component. Κάτι τέτοιο όμως θα σήμαινε ότι το σύστημα ελέγχου θα υλοποιούνταν σαν ένα Plain Old Java Object (POJO) και ως συνέπεια δεν θα μπορούσαμε να επωφεληθούμε από τα πλεονεκτήματα που μας προσφέρει το OSGi. Κατά συνέπεια αποφασίστηκε το σύστημα ελέγχου να αποτελείται από πολλές συνιστώσες, όπου κάθε μία θα είναι υπεύθυνη για μία λειτουργία του σιλό. Στην εικόνα 7.1 φαίνονται οι συνιστώσες από τις οποίες αποτελείται το σύστημα. Το διάγραμμα αυτό είναι ένα component διάγραμμα όπως το προτείνει η OSGi Alliance [48] και δείχνει τη δομή του cyber μερους του

σιλό ως Industrial Automation Thing. Κάθε συνιστώσα σε αυτό το διάγραμμα αντιστοιχεί σε ένα φυσικό μέρος του σιλό το οποίο μπορεί να αντικατασταθεί ανά πάσα στιγμή.

Ένα πολύ σημαντικό χαρακτηριστικό του OSGi είναι ότι οι διάφορες συνιστώσες του συστήματος θα πρέπει να επικοινωνούν μεταξύ τους μέσω καλά καθορισμένων interfaces. Έτσι κάθε συνιστώσα πρέπει να εξάγει τις διεπαφές που είναι απαραίτητες ώστε να επικοινωνήσει με το περιβάλλον της και κάποια άλλη συνιστώσα να μπορεί να επικοινωνήσει με αυτή. Στο διάγραμμα μπορούμε να παρατηρήσουμε ότι κάθε .api συνιστώσα εξάγει το πακέτο που περιέχει τα διάφορα interfaces της. Το πακέτο που εξάγεται εμφανίζεται στο διάγραμμα σαν ένα μαύρο κουτί μέσα στο component, ενώ τα πακέτα που το κάθε component απαιτεί ώστε να μπορέσει να λειτουργήσει μοντελοποιούνται σαν λευκά κουτιά. Για παράδειγμα, η συνιστώσα valve.out υλοποιεί την λειτουργικότητα του driver της βαλβίδας εξαγωγής υγρού του σιλό και η συνιστώσα valve.out.api καθορίζει τις υπηρεσίες που παρέχει η βαλβίδα εξαγωγής υγρού στον περιβάλλον της. Ο controller του σιλό στην εικόνα 7.1 πρέπει να χρησιμοποιήσει τις .api συνιστώσες των τριών drivers, δηλαδή του siloDriver που είναι υπεύθυνος για την λειτουργικότητα των αισθητήρων, του inValveDriver και του outValveDriver που είναι υπεύθυνοι για την λειτουργικότητα της βαλβίδας εισαγωγής και εξαγωγής υγρού αντίστοιχα. Στη συνέχεια θα γίνει μία εκτενέστερη ανάλυση του τρόπου λειτουργίας της κάθε συνιστώσας καθώς και του τρόπου χρήσης του OSGi.



Εικόνα 7.1: Το διάγραμμα συνιστωσών ενός σιλό

## 7.3 Ανάλυση των συνιστώσων του συστήματος ελέγχου

### 7.3.1 Οι συνιστώσες των drivers του σιλό

Όπως περιγράφηκε και στην προηγούμενη ενότητα, κάθε συνιστώσα είναι υπεύθυνη για την λειτουργικότητα ενός φυσικού στοιχείου του σιλό. Αυτό σημαίνει ότι για κάθε στοιχείο που εκτελεί ένα έργο για το σύστημα θα υπάρχει ένα component υπεύθυνο για την υλοποίηση και ένα component υπεύθυνο για τις υπηρεσίες που εξάγει στο περιβάλλον του. Έτσι έχουμε την ακόλουθη αντιστοίχηση φυσικών μερών του συστήματος με συνιστώσες λογισμικού που φαίνεται στον πίνακα 7.1:

Φυσικό Μέρος του Σιλό	Component λογισμικού
Βαλβίδα εισαγωγής υγρού	Valve.in
Βαλβίδα εξαγωγής υγρού	Valve.out
Στοιχείο ανάδευσης του υγρού	Mixer.driver
Στοιχείο θέρμανσης του υγρού	Heater.driver
Αισθητήρες επιπέδου υγρού	Silo.driver

Πίνακας 7.1: Αντιστοίχηση των φυσικών μερών του σιλό με τη συνιστώσα τους

#### 7.3.1.1 Συνιστώσες βαλβίδων του σιλό

Είναι υπεύθυνες για την αντιστοίχηση του μηχανικού μέρους της βαλβίδας του σιλό σε επίπεδου λογισμικού ώστε να μπορεί να χρησιμοποιηθεί από τον ενσωματωμένο μικροϋπολογιστή και να παρέχει τις υπηρεσίες που περιγράφηκαν στο κεφάλαιο 6 στο περιβάλλον του. Γι' αυτό οι συγκεκριμένες συνιστώσες αποτελούνται από μία διεπαφή (.api component) μέσω τις οποίας παρέχουν στον περιβάλλον τους υπηρεσίες για το άνοιγμα και το κλείσμα της βαλβίδας. Αυτές οι διεπαφές υλοποιούνται από τα components valve.in και valve.out όπου με την βοήθεια της βιβλιοθήκης rj4j αποστέλνεται σήμα στην βαλβίδα να ανοίξει ή να κλείσει ανάλογα με την περίπτωση. Η διεπαφή των βαλβίδων αποτελείται από το interface ValveDriverIf το οποίο ορίζει τις μεθόδους open() και close() οι οποίες ανοίγουν και κλείνουν την βαλβίδα αντίστοιχα. Στο component που υλοποιεί αυτή τη λειτουργικότητα έχουμε την υλοποίηση αυτών των μεθόδων, όπου κάθε μέθοδος αλλάζει την τάση σε ένα συγκεκριμένο γενικού σκοπού I/O του Raspberry Pi. Αξίζει επίσης να σημειωθεί ότι χρησιμοποιήθηκαν και όροι OSGi ώστε να μπορεί αυτή η υλοποίηση να είναι component. Αυτό γίνεται χρησιμοποιώντας το annotation **@Component** μέσα στο οποίο ορίζεται το όνομα της συνιστώσας καθώς και το όνομα της υπηρεσίας που απαιτεί για να λειτουργήσει. Αυτά φαίνονται καλύτερα και στον αλγόριθμο 7.1 όπου βλέπουμε την υλοποίηση του component για την βαλβίδα εξαγωγής υγρού από το σιλό.

---

```

@Component(
    name = "liqueurplant.osgi.valve.out",
    service =
        liqueurplant.osgi.valve.out.api.OutValveDriverIf.class
)
public class OutValveDriver implements OutValveDriverIf {

    .
    .
    .

    @Override
    public void open() throws Exception {
        try {
            outValvePin.setState(PinState.LOW);
        } catch (Exception e) {
            LOGGER.error("Exception in open(): " + e.toString());
        }
    }

    @Override
    public void close() throws Exception {
        try {
            outValvePin.setState(PinState.HIGH);
        } catch (Exception e) {
            LOGGER.error("Exception in close(): " + e.toString());
        }
    }
}

```

---

Αλγόριθμος 7.1: Μέρος της υλοποίησης της συνιστώσας την βαλβίδας εξαγωγής υγρού

### 7.3.1.2 Συνιστώσες στοιχείων θέρμανσης και μίξης του σιλό

Είναι υπεύθυνες για την λειτουργικότητα των στοιχείων ανάδευσης και θέρμανσης του υγρού που μπορεί να περιέχει κάποιο σιλό. Σχετικά με το στοιχείο ανάδευσης του υγρού, αποτελείται από ένα .api component μέσω του οποίου παρέχει τις υπηρεσίες έναρξης και τερματισμού λειτουργίας του μηχανικού μέρους για την ανάδευση του υγρού. Αυτές οι υπηρεσίες παρέχονται μέσω interface στον περιβάλλον του σιλό, το MixerDriverIf και υλοποιούνται από

το mixer component στο οποίο όταν έρθει εντολή για έναρξη ανάδευσης το υγρό αναμειγνύεται για συγκεκριμένο χρόνο και η διαδικασία αυτή σταματάει όταν έρθει εντολή τερματισμού. Πάλι και εδώ χρησιμοποιούνται τα αντίστοιχα annotations ώστε το περιβάλλον του OSGi να μπορεί να αναγνωρίζει τη συγκεκριμένη συνιστώσα σαν component. Για το στοιχείο θέρμανσης ακολουθήθηκε παρόμοια λογική, πέρα από το γεγονός ότι υλοποιήθηκε και ένα callback καθώς ήταν απαραίτητη η εισαγωγή πληροφορίας από το μηχανικό μέρος του σιλό. Έτσι μόλις ο αισθητήρας θερμοκρασίας του σιλό επιστρέψει την επιθυμητή τιμή καλείται η μέθοδος heatingCompleted ώστε να σταματήσει η διαδικασία θέρμανσης του υγρού. Στον αλγόριθμο 7.2 φαίνονται οι υπηρεσίες που παρέχει η συνιστώσα του στοιχείου θέρμασης του σιλό στον περιβάλλον της.

---

```

@ProviderType
public interface HeaterDriverIf {
    void start();
    void stop();
    void heat2temp(float temperature);
    void addHeatingCompletedListener(HeatingCompletedListenerIf
        listener);
}

```

---

Αλγόριθμος 7.2: Οι υπηρεσίες που παρέχει η συνιστώσα του στοιχείου θέρμανσης

Με τη χρήση του annotation **@ProviderType** οι υπηρεσίες αυτές εγγράφονται στο μητρώο υπηρεσίων του OSGi και μπορούν να χρησιμοποιηθούν από άλλες συνιστώσες.

### 7.3.1.3 Συνιστώσα του driver του σιλό

Ο συγκεκριμένος driver είναι υπεύθυνος για τους αισθητήρες επιπλέοντος του υγρού. Η συγκεκριμένη συνιστώσα δεν συνοδεύεται από ένα component .api καθώς δεν μπορεί να εξάγει κάποια υπηρεσία. Αυτό έγινε γιατί αντιμετωπίστηκε το πρόβλημα των κυκλικών εξαρτήσεων μεταξύ του συστήματος ελέγχου και της συνιστώσας του driver του σιλό. Γι' αυτό το λόγο ακολουθήθηκε μίλια διαφορετική προσέγγιση κατά την οποία ο driver επικοινωνεί με τον controller μέσω της EventQueue του controller. Για να γίνει αυτό πιο σαφές, ας θεωρήσουμε ότι το σιλό βρίσκεται στην κατάσταση FILLING. Όταν το υγρό φτάσει στο ανώτατο σημείο του σιλό τότε ο άνω αισθητήρας ανίχνευσης υγρού θα ενεργοποιηθεί και ο driver θα εισάγει στην EventQueue του controller ένα μήνυμα ότι το σιλό έχει γεμίσει ώστε το σύστημα ελέγχου να συνεχίσει τις απαραίτητες διαδικασίες ώστε να λειτουργήσει σωστά όλο το σύστημα παραγωγής. Κάτι τέτοιο δημιουργεί την ανάγκη η συνιστώσα του driver να χρειάζεται τις υπηρεσίες που παρέχει η συνιστώσα του controller για να λειτουργήσει. Αυτό φαίνεται και από το reference που υπάρχει στην συνιστώσα του controller στην υλοποίηση του driver που φαίνεται στον αλγόριθμο 7.3.

```

@Component(
    name = "liqueurplant.osgi.silo.driver",
    immediate = true
)
public class SimpleSiloDriver {

    .
    .

    @Reference(
        policy = ReferencePolicy.DYNAMIC,
        cardinality = ReferenceCardinality.OPTIONAL
    )
    protected void setSiloCtrlIf(SiloCtrlIf siloCtrl) {
        this.siloCtrl = siloCtrl;
        LOGGER.info("SILO-CONTROLLER binded.");
    }

    protected void unsetSiloCtrlIf(SiloCtrlIf siloCtrl) {
        this.siloCtrl = null;
        LOGGER.info("SILO-CONTROLLER unbinded.");
    }
}

```

---

Αλγόριθμος 7.3: Μέρος της υλοποίησης της συνιστώσας του driver του σιλό

Από μερίας λογισμικού, ο driver του σιλό υλοποιεί δύο **Listeners** με την βοήθεια της βιβλιοθήκης Pi4j, και συγκεκριμένα του interface GpioPinListenerDigital. Ο listener που αφορά την ακίδα που είναι υπεύθυνη για τον άνω αισθητήρα παρουσίας υγρού του σιλό περιμένει να λάβει σήμα υψηλής τάσης στο γενικού σκοπού I/O του Raspberry Pi από τον εξομοιωτή του σιλό. Όταν έρθει το συγκεκριμένο σήμα τότε εισάγει στην EventQueue του σιλό ένα νέο Signal ότι το υγρό έχει γεμίσει το σιλό. Ομοίως υπάρχει και ένας Listener για τον κάτω αισθητήρα παρουσίας υγρού του σιλό. Η υλοποίηση αυτή φαίνεται και στον αλγόριθμο 7.4. Αξίζει να σημειωθεί ότι το annotation **@Activate** που υπάρχει στο πάνω μέρος της μεθόδου, αποτελεί μέρος του OSGi framework και αφορά τον κύκλο ζωής της συγκεκριμένης συνιστώσας. Έτσι μόλις ενεργοποιηθεί η συγκεκριμένη συνιστώσα θα τρέξει η μέθοδος αυτή ώστε να γίνουν οι απαραίτητες αρχικοποιήσεις.

---

```

@Activate
public void activate() {
    highLevelSensorPin =
        gpioController.provisionDigitalInputPin(RaspiPin.GPIO_02,
        "HIGH-LEVEL-SENSOR", PinPullResistance.PULL_DOWN);
    lowLevelSensorPin =
        gpioController.provisionDigitalInputPin(RaspiPin.GPIO_03,
        "LOW-LEVEL-SENSOR", PinPullResistance.PULL_DOWN);

    highLevelSensorPin.setShutdownOptions(true);
    lowLevelSensorPin.setShutdownOptions(true);

    highLevelSensorPin.addListener((GpioPinListenerDigital)
event -> {
        if (event.getState() == PinState.HIGH) {
            siloCtrl.put2MsgQueue(new
HighLevelReachedSignal());
        }
    });
    lowLevelSensorPin.addListener((GpioPinListenerDigital) event
-> {
        if (event.getState() == PinState.LOW) {
            siloCtrl.put2MsgQueue(new
LowLevelReachedSignal());
        }
    });
}

```

---

Αλγόριθμος 7.4: Η υλοποίηση των listener του driver του συλό

#### 7.3.1.4 Συνιστώσα του controller του συλό

Αποτελεί το βασικό μέρος του συστήματος ελέγχου, καθώς όλα τα μηνύματα που έρχονται τόσο από τον LwM2M server όσο και από τους drivers του σιλό πρέπει να επεξεργαστούν από τον συγκεκριμένο μέρος του συστήματος. Η συγκεκριμένη συνιστώσα του συστήματος περιλαμβάνει μία EventQueue μέσα στην οποία μπορούν να τοποθετηθούν διάφορα μηνύματα που αφορούν τη λειτουργία του σιλό. Ένα άλλο σημαντικό κομμάτι της υλοποίησης είναι το Generic State Machine που χρησιμοποιήθηκε στον controller του σιλό.

Αυτό το state machine έχει υλοποιηθεί σε άλλη διπλωματική εργασία που έγινε παράλληλα με την εκπόνηση της παρούσας και χρησιμοποιείται στο controller του σιλό ώστε να γίνεται η επεξεργασία των μηνυμάτων που λαμβάνει καθώς και οι κατάλληλες ενέργειες που απαιτούνται ώστε να επιτευχθεί σωστή λειτουργία του συστήματος ελέγχου. Έχει υλοποιηθεί με μοντέλο την μηχανή καταστάσεων της UML. Αυτό σημαίνει ότι πρέπει να δηλωθούν οι διαφορετικές καταστάσεις στις οποίες μπορεί να βρεθεί το σιλό καθώς και οι διάφορες μεταβάσεις από την μία κατάσταση στην άλλη. Κάθε κατάσταση περιέχει τρεις μεθόδους τις entry, doActivity, exit όπως ακριβώς ορίζει και η UML. Σε αυτές τις μεθόδους εκτελούνται ενέργειες όταν το σιλό εισέλθει σε μία νέα κατάσταση, κατά την διάρκεια που βρίσκεται σε αυτή και πριν μεταβεί σε κάποια επόμενη κατάσταση αντίστοιχα. Σε αυτές τις μεθόδους βρίσκεται το μεγαλύτερο μέρος της λογικής του συστήματος ελέγχου.

Ο controller του συστήματος παρέχει μέσω της .api συνιστώσας υπηρεσίες υπεύθυνες για την εισαγωγή και εξαγωγή μηνυμάτων από την EventQueue του, οι οποίες φαίνονται στον αλγόριθμο 7.5. Αξίζει να σημειωθεί ότι η EventQueue υλοποιήθηκε με την χρήση της τεχνολογίας ArrayBlockingQueue που μας παρείχε η Java καθώς θα έπρεπε να χρησιμοποιείται από δύο νήματα που θα τρέχουν ταυτόχρονα. Κάτι τέτοιο μας έδωσε την δυνατότητα να μην πέσουμε σε κάποιο deadlock και τα νήματα τόσο του controller όσο και του LwM2M client να λειτουργούν χωρίς να παρουσιαστεί κάποιο πρόβλημα.

Η συγκεκριμένη συνιστώσα πρέπει ανάλογα με τον τύπο του σιλό να περιμένει να λάβει τις υπηρεσίες από τα διάφορα άλλα components όπως για παράδειγμα των βαλβίδων και του driver του σιλό.

---

```
@ProviderType
public interface SiloCtrlIf {
    void put2MsgQueue(BaseSignal signal);
    BaseSignal takeNotification();
}
```

---

Αλγόριθμος 7.5: Οι υπηρεσίες που παρέχει η συνιστώσα του ελεγκτή του σιλό

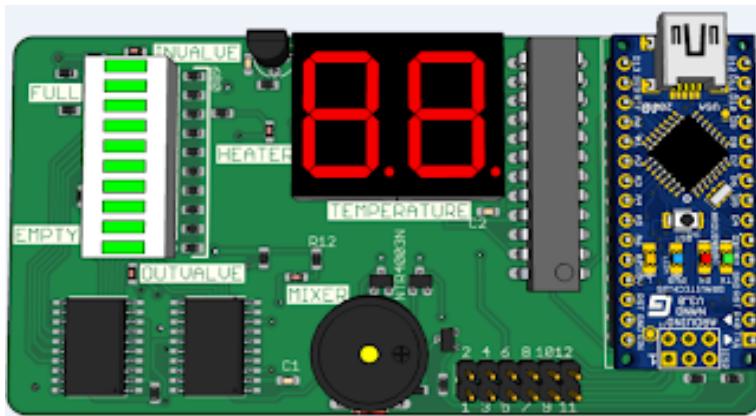
### 7.3.1.5 Ο LwM2M client του σιλό

Η συνιστώσα που υλοποιεί τον LwM2M client του συστήματος ελέγχου είναι υπεύθυνη για την επικοινωνία του σιλό με το εξωτερικό του περιβάλλον. Δημιουργεί κάποια από τα objects και τα resources που αναλύθηκαν στο κεφάλαιο 6 και τα καθιστά προσβάσιμα μέσω του πρωτοκόλλου CoAP ώστε να μπορούν να χρησιμοποιηθούν από έναν LwM2M server. Μόλις ξεκινήσει η λειτουργία του συστήματος η συγκεκριμένη συνιστώσα είναι υπεύθυνη για να κάνει εγγραφή σε έναν LwM2M server, του οποίου γνωρίζει την IP του. Όταν γίνει αυτή η εγγραφή τότε ενημερώνεται τον server για τα object και τα resources που παρέχει ο συγκεκριμένος client. Επιπλέον,

είναι σημαντικό να σημειωθεί ότι ο client περιμένει να λάβει τις υπηρεσίες του controller ώστε να μπορέσουν να μεταφερθούν τα διάφορα μηνύματα που θα λάβει από τον server και το σύστημα να λειτουργήσει σωστά.

## 7.4 Η πειραματική διάταξη

Για να ελεγχθεί η σωστή λειτουργία του συστήματος ελέγχου που υλοποιήθηκε ήταν αναγκαία μία προσομοίωση της διαδικασίας παραγωγής liqueur. Για να γίνει αυτό όπως προαναφέρθηκε χρησιμοποιήθηκαν Raspberry Pi σαν ενσωματωμένοι μικροϋπολογιστές πάνω στους οποίους έτρεχε και το σύστημα ελέγχου. Επίσης, χρησιμοποιήθηκε και ένας προσομοιωτής του μηχανικού σιλό που είχε υλοποιηθεί στην [61]. Ο προσομοιωτής του μηχανικού σιλό όπως φαίνεται στην εικόνα 7.4 αποτελείται από ένα ψηφιακό ηλεκτρονικό κύκλωμα και επικοινωνεί με τον μικροϋπολογιστή Raspberry Pi με την αποστολή ψηφιακών ηλεκτρικών σημάτων.



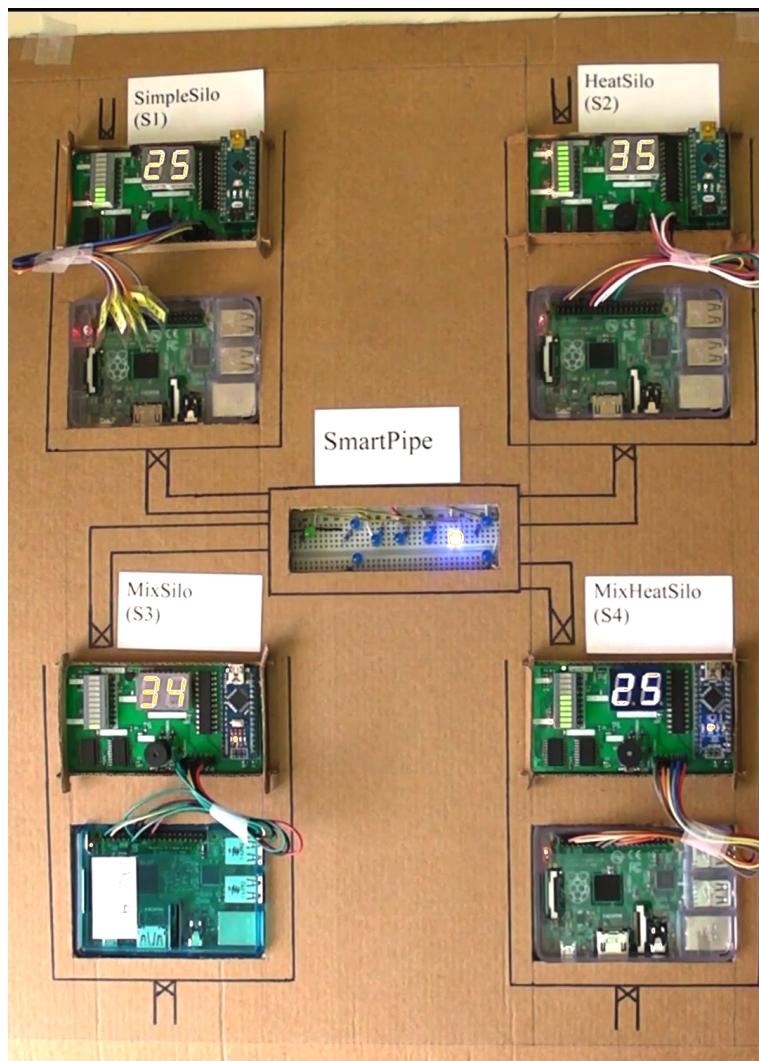
**Εικόνα 7.2:** Εξομοιωτής των μηχανικών στοιχείων του σιλό [61]

Το κύκλωμα διαθέτει, σύμφωνα με την [61]:

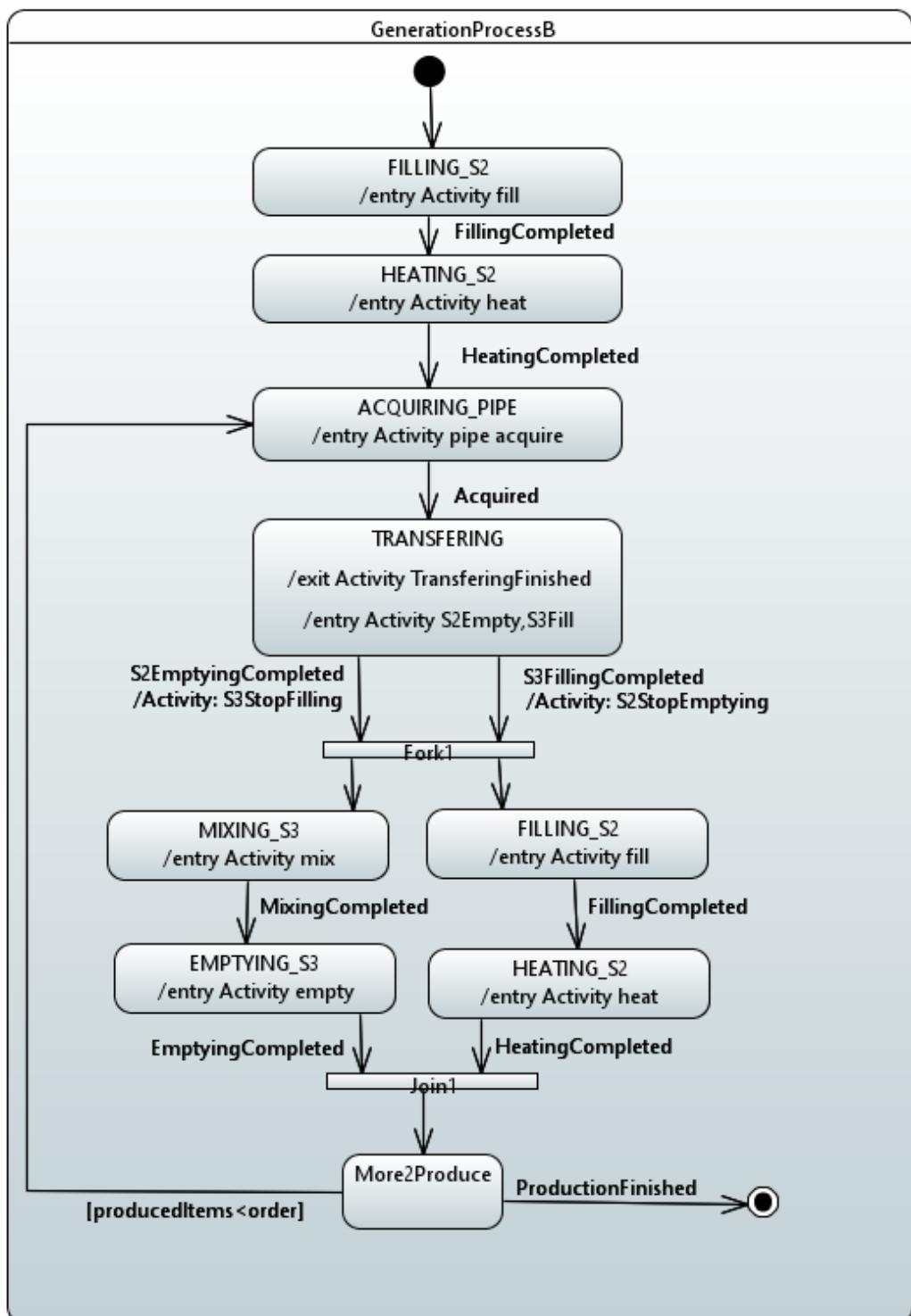
- μία μπάρα LED 10 τμημάτων για την απεικόνιση της στάθμης του υγρού ανά πάσα στιγμή μέσα στο σιλό
- δύο LED για την ένδειξη της κατάστασης των βαλβίδων εισόδου και εξόδου
- δύο LED για την ένδειξη των αισθητήρων παρουσίας υγρού
- ένα LED για την ένδειξη της κατάστασης του στοιχείου θέρμανσης και ένα LED αλλά και ένα buzzer για την ένδειξη της κατάστασης του στοιχείου μίξης
- δύο 7-segment displays για την ένδειξη της τρέχουσας θερμοκρασίας του σιλό.

Για την επικοινωνία με τον μικροϋπολογιστή που χρησιμοποιήθηκε, χρησιμοποιήθηκαν οι διεπαφές που είχε ο εξομοιωτής του μηχανικού σιλό οι οποίες ήταν καλά ορισμένες μέσω της [61]. Από το Raspberry Pi αποστέλλονται ψηφιακά σήματα μέσω της βιβλιοθήκης Pi4J και τα γενικού σκοπού I/O που παρέχει το Raspberry Pi.

Είχαμε στην διάθεση μας τέσσερις εξομοιωτές του σιλό και έναν εξομοιωτή του Pipe που φαίνεται στην εικόνα 7.5. Έτσι αφού εγκαταστάθηκε το Apache Felix Framework στους μικρο-ϋπολογιστές και έγιναν οι κατάλληλες διασυνδέσεις με τους εξομοιωτές των μηχανικών τμημάτων του σιλό το σύστημα ήταν έτοιμο για χρήση. Για την διεξαγωγή των πειραμάτων χρησιμοποιήθηκε ένα LwM2M server ο οποίος υλοποιούσε την λογική της παραγωγής και των δύο τύπων liqueur ταυτόχρονα. Ο LwM2M server είναι υλοποιημένος σε Java σαν ένα POJO και έτρεχε σε ένα Raspberry Pi 2 Model B, το οποίο απαρτίζεται από έναν τετραπύρηνο επεξεργαστή χρονισμένο στα 900MHz και από 1 Gb RAM. Η διαδικασία για την παραγωγή του liqueur τύπου B φαίνεται στην εικόνα 7.6.



**Εικόνα 7.3:** Η πειραματική διάταξη με τα τέσσερα σιλό και τον σωλήνα



**Εικόνα 7.4:** Η διαδικασία παραγωγής liqueur τύπου B

## Κεφάλαιο 8

# Συμπεράσματα και μελλοντική εργασία

Στην εργασία αυτή περιγράφεται ο σχεδιασμός και η ανάπτυξη ενός συστήματος ελέγχου για ένα βιομηχανικό σύστημα παραγωγής χρησιμοποιώντας μία προσέγγιση συνιστωσών καθώς και τεχνολογίες διαδικτύου, ώστε να μετατραπεί το παραδοσιακό βιομηχανικό σύστημα σε ένα σύστημα συμβατό με το διαδίκτυο των αντικειμένων. Η επικοινωνία των επιμέρους υποσυστημάτων έγινε με την χρήση του LwM2M που ακολουθεί της αρχές τις REST αρχιτεκτονικής και η ανάπτυξη του συστήματος ελέγχου βασίστηκε στην ανάπτυξη λογισμικού με την προσέγγιση συνιστωσών.

## 8.1 Συμπεράσματα

Μέσω της εργασίας αυτής μου δόθηκε η δυνατότητα να μελετήσω αρχικά εις βάθος την προσέγγιση ανάπτυξης λογισμικού με συνιστώσες και κυρίως το framework OSGi που παρέχει αυτή τη δυνατότητα στην γλώσσα προγραμματισμού Java. Μέσω αυτής της μελέτης απέκτησα σπουδαία γνώση στον τρόπο με τον οποίο μπορεί να δομηθεί ένα πρόγραμμα ακολουθώντας την συγκεκριμένη προσέγγιση, αφού σε πολλά σημεία χρειάστηκε να ανατρέξω σε πηγαίο κώδικα προκειμένου να συλλεχθούν οι απαραίτητες πληροφορίες που χρειάστηκαν στα πλαίσια της εργασίας μου. Επιπλέον, η ομάδα που είναι υπεύθυνη για το συγκεκριμένο framework έχει κάνει άριστη δουλειά στο documentantion το οποίο ήταν σύμμαχος μου σε οποιοδήποτε πρόβλημα κλήθηκα να αντιμετωπίσω. Επιπλέον, ήρθα σε επαφή με την γλώσσα μοντελοποίησης UML, την οποία σε συνεργασία με τον επιβλέποντα καθηγητή της εργασίας κατανόησα εις βάθος ώστε να μπορέσω να σχεδιάσω το σύστημα που ανέπτυξα με τον καλύτερο δυνατό τρόπο. Ο σχεδιασμός του συστήματος με την χρήση της γλώσσας UML ήταν ίσως το σημαντικότερο μέρος της εργασίας καθώς από την στιγμή που καταλήξαμε σε μερικές παραδοχές τότε η ανάπτυξη του συστήματος ακολουθώντας τις προδιαγραφές που ορίστηκαν ήταν αρκετά δομημένη και απλή. Επίσης, ήρθα σε επαφή και μελέτησα τον τομέα του διαδικτύου των αντικειμένων. Μέσω της εργασίας κατανόησα βασικές έννοιες που τον διέπουν καθώς και προβλήματα που πρέπει να αντιμετωπιστούν ώστε οι τεχνολογίες που παρέχει αυτός ο τομέας να μπορούν να εφαρμοστούν ώστε να κάνουν τον τρόπο ζωής μας πιο απλό και εύκολο. Επιπλέον, κατανόησα βασικές έννοιες και τον τρόπο λειτουργίας των πρωτοκόλλων που χρησιμοποίησα στην παρούσα εργασία.

Από την εργασία προέκυψαν διάφορα συμπεράσματα, τα οποία έχουν να κάνουν κυρίως με την ανάπτυξη λογισμικού χρησιμοποιώντας την προσέγγιση συνιστωσών. Η πρώτη επαφή με την συγκεκριμένη προσέγγιση σίγουρα ήταν δύσκολη, αλλά στην συνέχεια ήταν αρκετή κατανόηση και έρευνα μπορούμε να καταλήξουμε στο συμπέρασμα ότι με την προϋπόθεση ότι αυτή η τεχνολογία θα χρησιμοποιηθεί σωστά τότε μπορεί να αποδώσει σημαντικά πλεονεκτήματα στον τομέα του IoT και την αυτοματοποίησης της βιομηχανικής παραγωγής. Επιπρόσθετα, το OSGi παρέχει πολλούς δωρεάν πόρους στον προγραμματιστή ώστε να κατανοήσει την λειτουργία του, τα πλεονεκτήματα που παρέχει αλλά και να βοηθηθεί κατά την διάρκεια ανάπτυξης ενός συστήματος χρησιμοποιώντας αυτή την τεχνολογία. Αρκετές υλοποιήσεις του framework ήταν ανοιχτού κώδικα και είχαμε πρόσβαση σε αυτές χωρίς κάποια επιβάρυνση. Η κοινότητα που αναπτύσσει την υλοποίηση που εμείς χρησιμοποιήσαμε έχει κάνει σημαντική δουλεία στην υλοποίηση αρκετών υπηρεσιών που παρέχει το OSGi και ήταν αρκετά βοηθητικές για την ανάπτυξη του συστήματος μας. Ωστόσο, ένα σημαντικό πρόβλημα που αντιμετωπίστηκε ήταν η διαχείριση των εξαρτήσεων των διάφορων συνιστωσών που χρησιμοποιήθηκαν στην δική μας υλοποίηση καθώς έπρεπε να ανατρέξουμε στα διάφορα μηνύματα λάθους ώστε να ανακαλύψουμε ποια components λείπουν ώστε το σύστημα να μπορέσει να λειτουργήσει.

Όσον αφορά το πρωτόκολλο LwM2M, αυτό με τη σειρά του προσφέρει αρκετές διευκολύνσεις. Η δύσκολη διαδικασία σχεδιασμού μιας ευέλικτης διεπαφής επάνω σε ένα πρωτόκολλο όπως το HTTP ή το CoAP, ανάγεται στη επιλογή των LwM2M αντικειμένων με τις κατάλληλες ιδιότητες. Παρ' όλα αυτά, κατά το στάδιο του υψηλού επιπέδου σχεδιασμού του συστήματος, έγινε φανερό πως το απλό, αλλά και περιορισμένο, μοντέλο οργάνωσης των resources στο LwM2M παρουσιάζει μια αδυναμία να περιγράψει πιο σύνθετες σχέσεις μεταξύ οντοτήτων όπως για παράδειγμα η πολλαπλών επιπέδων ιεραρχία τους και η κληρονομικότητα.

Εν τέλει, μπορεί να θεωρηθεί σίγουρο πως κάθε βιομηχανία μπορεί να επωφεληθεί σε βάθος χρόνου από την ένταξη της στο διαδίκτυο των αντικειμένων και οι προγραμματιστές στον συγκεκριμένο τομέα μπορούν να επωφεληθούν από την χρήση της προσέγγισης συνιστωσών στην ανάπτυξη του λογισμικού για τα διάφορα συστήματα.

## 8.2 Προτάσεις για μελλοντική εργασία

Σίγουρα το σύστημα που μελετήθηκε και αναπτύχθηκε δεν είναι ικανό να επιδείξει πλήρως τα πλεονεκτήματα που παρέχει η αναβάθμιση που έγινε με την χρήση συνιστωσών. Ωστόσο κάτι τέτοιο μπορεί να γίνει με την μελλοντική μελέτη και εργασία πάνω στο συγκεκριμένο τομέα με τρόπους όπως:

- Εύρεση λύσης στο πρόβλημα των εξαρτήσεων μεταξύ των διάφορων συνιστωσών του συστήματος:** Κατά την διάρκεια εκπόνησης της εργασίας αυτής, ένα πρόβλημα που συναντούσαμε συνέχεια μπροστά μας ήταν οι εξαρτήσεις συνιστωσών που

έπρεπε να ικανοποιηθούν ώστε το σύστημα μας να είναι σε θέση να λειτουργήσει. Κάθε φορά που έπρεπε να εγκαταστήσουμε μία νέα συνιστώσα, όπως για παράδειγμα την συνιστώσα που υλοποιεί τις τεχνολογίες που ορίζει το framework για τα Declarative Services, τότε δημιουργούνταν εξαρτήσεις σε άλλες συνιστώσες και υπηρεσίες που χρειαζόταν η πρώτη συνιστώσα για να μπορέσει να ξεκινήσει η λειτουργία της. Το OSGi δεν παρείχε κάποιο μηχανισμό ώστε να αναγνωρίζονται οι επιπλέον συνιστώσες και να εγκαθίστανται μαζί με την νέα συνιστώσα που θέλαμε εμείς να εισάγουμε. Έτσι έπρεπε να γίνει μία αναζήτηση των απαιτούμενων συνιστωσών μέσα από τα μηνύματα λάθους που μας παρείχε το περιβάλλον του Felix και στην συνέχεια έπρεπε να εκαταστηθούν αυτές οι συνιστώσες που έλειπαν. Μία λύση στο συγκεκριμένο πρόβλημα θα γλύτωνε τους προγραμματιστές τέτοιων συστημάτων από πολύτιμο χρόνο.

- **Ανάπτυξη του LwM2M server χρησιμοποιώντας συνιστώσες:** Ο LwM2M server που χρησιμοποιήθηκε στην εργασία αυτή για τον έλεγχο και την επίδειξη της σωστής λειτουργίας του συστήματος μας έχει υλοποιηθεί σαν ένα POJO. Μία ενδιαφέρουσα προσέγγιση θα ήταν η χρήση συνιστωσών για την ανάπτυξη του LwM2M server. Κάτι τέτοιο θα μπορούσε να παρέχει σημαντικές νέες λειτουργίες αξιοποιώντας τα πλεονεκτήματα που παρέχει αυτή η προσέγγιση. Αν υποθέσουμε ότι η κάθε διεργασία παραγωγής κάποιου τύπου liqueur αποτελούσε ξεχωριστή συνιστώσα του συστήματος, τότε αυτή η συνιστώσα θα μπορούσε να αλλάξει και το σύστημα να είναι σε θέση να παράξει έναν νέο τύπου liqueuer χωρίς να είναι απαραίτητο να τροποποιηθεί όλο το σύστημα από την αρχή.
- **Ανάπτυξη εφαρμογής για φορητές συσκευές ώστε κάποιος να είναι σε θέση να διαχειρίζεται το σύστημα από οπουδήποτε βρίσκεται αρκεί να έχει πρόσβαση στο διαδίκτυο:** Στόχος του διαδικτύου των αντικειμένων είναι να κάνει την ανθρώπινη ζωή πιο εύκολη. Από την στιγμή που ένα τέτοιο σύστημα θα είναι σε θέση να ενταχθεί στον τομέα του διαδικτύου των αντικειμένων και σε συνεργασία με την τεράστια εξέλιξη που έχει ο τομέας των smartphones θα μπορούσε να υλοποιηθεί μία εφαρμογή μέσως της οποίας ο χρήστης θα μπορούσε να ορίσει τις προδιαγραφές του liqueur που θέλει να παραγγείλει και να ενημερώνεται με διάφορα μηνύματα για την διαδικασία παραγωγής του προϊόντος που διάλεξε. Επιπλέον, θα μπορούσε να είναι σε θέση να δει και την κατάσταση του κάθε σιλό κατά την διαδικασία παραγωγής του liqueur.
- **Χρήση της γλώσσας UML με σκοπό την αυτόματη παραγωγή κώδικα του συστήματος μέσω καλά ορισμένων προδιαγραφών:** Κατά την διάρκεια εκπόνησης της εργασίας έγινε μία προσπάθεια αυτόματης παραγωγής του κώδικα της διεργασίας παραγωγής liqueur μέσω των διαγραμμάτων UML. Εν τέλει δεν καταφέραμε να φτάσουμε στο επιθυμητό αποτέλεσμα, αλλά το συγκεκριμένο θέμα θα μπορούσε να αποτελέσει το θέμα κάποιας άλλης έρευνας. Καθώς μέσα από τα διαγράμματα UML είναι εφικτό

να περιγραφεί τόσο η δομή όσο και η συμπεριφορά ενός συστήματος, αν αυτά είναι σωστά ορισμένα και ακολουθούν συγκεκριμένουν κανόνες, τότε θα μπορούσε να παραχθεί κώδικας ο οποίος να είναι σε θέση να εκτελεστεί και να παρέχει την επιθυμητή λειτουργικότητα στο σύστημα.

# Αναφορές

- [1] L. Coetzee and J. Eksteen. "*The Internet of Things - promise for the future? An introduction*". in IST-Africa Conference, 2011.
- [2] "*Number of Facebook users worldwide 2008-2018*". Statista, 2018. [Online] Available: <https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide>.
- [3] B. Marr. "*Why Everyone Must Get Ready For The 4th Industrial Revolution*". Forbes.com, 2016. [Online] Available: <https://www.forbes.com/sites/bernardmarr/2016/04/05/why-everyone-must-get-ready-for-4th-industrial-revolution/#5075c5333f90>.
- [4] J. Pontin. "*Bill Joy's Six Webs*". MIT Technology Review, 2005.
- [5] M. Farooq, M. Waseem, S. Mazhar, A. Khairi and T. Kamal. "*A Review on Internet of Things (IoT)*". International Journal of Computer Applications, vol. 113, no. 1, 2015.
- [6] F. Mattern and C. Floerkemeier. "*From the Internet of Computers to the Internet of Things*". In: Sachs K., Petrov I., Guerrero P. (eds) From Active Data Management to Event-Based Systems and More. Lecture Notes in Computer Science, vol 6462. Springer, Berlin, Heidelberg
- [7] K. Zeinab and S. Elmoustafa. "*Internet of Things Applications, Challenges and Related Future Technologies*". World Scientific News, vol. 67, no. 2, 2017.
- [8] S. Misra et al. "*Security Challenges and Approaches in Internet of Things*". Springer Briefs in Electrical and Computer Engineering, 2016.
- [9] M. Niranjan, N. Madhukar, A. Ashwini, J. Muddsar and M. Saish. "*IOT Based Industrial Automation*". IOSR Journal of Computer Engineering (IOSR-JCE), pp. 36-40, 2017.
- [10] World Economic Forum. "*Unleashing the Potential of Connected Products and Services*". World Economic Forum, 2015.
- [11] W. Elfrink. "*The Internet of Things: Capturing the Accelerated Opportunity*". Cisco Blog, October 15, 2014. Available: <http://blogs.cisco.com/ioe/the-internet-of-things-capturing-the-accelerated-opportunity>
- [12] A. Rojko. "*Industry 4.0 Concept: Background and Overview*". International Journal of Interactive Mobile Technologies (iJIM), vol. 11, no. 5, p. 77, 2017.

- [13] J. Conway. "*The Industrial Internet of things: an evolution to a smart manufacturing enterprise*". Available: [http://www.mcrockcapital.com/uploads/1/o/9/6/10961847/schneider-an\\_evolution\\_to\\_a\\_smart\\_manufacturing\\_enterprise.pdf](http://www.mcrockcapital.com/uploads/1/o/9/6/10961847/schneider-an_evolution_to_a_smart_manufacturing_enterprise.pdf)
- [14] V. Gazis, M. Goertz, M. Huber, and A. Leonardi. "*Short Paper: IoT: Challenges, Projects, Architectures*".
- [15] Industrial Internet Consortium - Smart Factory Task Group, "*Smart Factory Applications in Discrete Manufacturing*", 2017.
- [16] Accenture, "*Driving Unconventional Growth through the Industrial Internet of Things*", 2015.
- [17] M. Laine. "*RESTful Web Services for the Internet of Things*", 2012.
- [18] D. Guinard and V. Trifa. "*Towards the Web of Things: Web Mashups for Embedded Devices*", International World Wide Web Conference, Madrid, Spain, 2009.
- [19] A. Azzarà, D. Alessandrelli, M. Petracca and P. Pagano. "*PyoT, a macroprogramming framework for the IoT*", 13th international symposium on Information processing in sensor networks, Berlin, Germany, 2014.
- [20] D. Guinard, V. Trifa, F. Mattern and E. Wilde. "*From the Internet of Things to the Web of Things: Resource-oriented Architecture and Best Practices*", Architecting the Internet of Things, Springer Berlin Heidelberg, 2011, pp. 97-129.
- [21] W. Colitti, K. Steenhaut, N. D. Caro, B. Buta and V. Dobrota. "*Evaluation of constrained application protocol for wireless sensor networks*", Local & Metropolitan Area Networks, 2011.
- [22] T. Yokotani and Y. Sasaki. "*Comparison with HTTP and MQTT on Required Network Resources for IoT*", The 2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), 2016.
- [23] A. Castellani, M. Gheda, N. Bui, M. Rossi and M. Zorzi. "*Web Services for the Internet of Things through CoAP and EXI*", IEEE International Conference on Communications Workshops (ICC), 5-9 June 2011, pp. 1-6.
- [24] S. L. Keoh, S. S. Kumar, H. Tschofenig. "*Securing the Internet of Things: A Standardization Perspective*", Internet of Things Journal IEEE, vol. 1, no. 3, June 2014, pp. 265-275.
- [25] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego and J. Alonso-Zarate. "*A Survey on Application Layer Protocols for the Internet of Things*", Transaction on IoT and Cloud Computing, 2015.

- [26] <http://mqtt.org/2011/08/mqtt-used-by-facebook-messenger>, cited 28 Jul 2014.
- [27] D. Thangavel, X. Ma, A. Valera, H. Tan, C. Keng-Yan Tan. *"Performance Evaluation of MQTT and CoAP via a Common Middleware"*, IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 21-24 April 2014, pp. 1-6.
- [28] Open Mobile Alliance (OMA). *"Lightweight M2M: Enabling Device Management and Applications for the Internet of Things"*, 2014.
- [29] *"What is CoAP IoT protocol. CoAP Architecture"*, Rfwireless-world.com, 2018. [Online]. Available: <http://www.rfwireless-world.com/IoT/CoAP-protocol.html>.
- [30] R. Gupta. *"5 Things to Know About MQTT - The Protocol for Internet of Things"*, Ibm.com, 2018. [Online]. Available: <https://goo.gl/LaE7se>
- [31] R. Fielding. *"Architectural Styles and the Design of Network-based Software Architectures"*, PhD thesis, University of California, Irvine, USA, 2000.
- [32] Open Mobile Alliance (OMA). *"Lightweight Machine to Machine Technical Specification"*, Candidate Version 1.0, 8 Feb 2017.
- [33] T. Wijayasiriwardhane, R. Lai, K.c. Kang. *"Effort estimation of component-based software development - a survey"*, IET Software, vol. 5, no. 2, pp. 216.
- [34] J. Xie. *"The Advantages and Challenges of Component-based Software Development Compare to Traditional Software Development"*, 2016.
- [35] KS. Ravichandran, Kr. Sekar, P. Suresh."A novel approach for optimal grouping of reusable software components for component based software development systems", International Journal Of Software Engineering And Knowledge Engineering, vol. 23, no. 7, pp.895-912, 2013.
- [36] C. Szyperski. *"Component Software: Beyond Object Oriented Programming"*, 2nd ed. Addison-Wesley Professional, Boston, 2002.
- [37] M. Panunzio and T. Vardanega. *"A Component Model for On-board Software Applications"*, Proc. of the 36th Euromicro Conf. on Software Engineering and Advanced Applications (SEAA), 2010.
- [38] A.W. Brown and K.C. Wallnau. *"The Current State of CBSE"*, IEEE Software, September 1998.
- [39] P. Allen. *"Realizing e-business with Components"*, Addison-Wesley, August 2000.

- [40] A. Brown. "*Large-scale, component-based development*", 1st ed. Upper Saddle River, NJ: Prentice Hall PTR, 2000, pp.70-72.
- [41] D. Garlan, R. Allen, and J. Ockerbloom. "*Architectural Mismatch: Why it's Hard to Build Systems Out of Existing Parts*", Proc. of the International Conference on Software Engineering, April 1995.
- [42] D. Parnas. "*On the Criteria for Decomposing Systems into Modules*", CACM vol. 15, no. 12, pp. 1053–1058, December 1972.
- [43] D. Parnas. "*On the Design and Development of Program Families*", IEEE Transactions on Software Engineering, vol. 2, no. 1, March 1976.
- [44] R. Prieto-Diaz and P. Freeman. "*Classifying Software for Reusability*", IEEE Software, 1987.
- [45] B. Cox. "*Object Oriented Programming - An Evolutionary Approach*", Addison-Wesley, 1986.
- [46] A.W. Brown and K.C. Wallnau. "*Engineering of Component-Based Systems*", Proceedings of the 2nd IEEE International Conference on Complex Computer Systems, Montreal, Canada, October 1996.
- [47] A. Alves. "*OSGi in depth*", Shelter Island, NY: Manning Publications Co., 2012.
- [48] Osgi.org. "*OSGi Alliance - The Dynamic Module System for Java*", 2018. Available at: <https://www.osgi.org/>.
- [49] D. Parnas. "*On the criteria to be used in decomposing systems into modules*", Communications of the ACM, 15(12):1053–1058, 1972.
- [50] A. Tavares and M. Valente. "*A gentle introduction to OSGi*", ACM SIGSOFT Software Engineering Notes, vol. 33, no. 5, 2008, p.1.
- [51] The OSGi Alliance. "*OSGi Core*", Release 6, June 2014.
- [52] Apache Felix. "*OSGi Framework and Service Platform*". Available at: <http://felix.apache.org/>
- [53] M. Stusek, J. Hosek, D. Kovac, P. Masek, P. Cika and F. Kropfl. "*Performance Analysis of the OSGi-based IoT Frameworks on Restricted Devices as Enablers for Connected-Home*", In 2015 7th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). Brno, Czech Republic: 2015. pp. 211-216.
- [54] Knopflerfish. "*Open Source OSGi SDK*". Available at: <http://www.knopflerfish.org/>

- [55] Eclipse Equinox. "*OSGi core framework*". Available at: <http://www.eclipse.org/equinox/>
- [56] F. Basile, P. Chiacchio and D. Gerbasio. "*On the implementation of industrial automation systems based on PLC*", IEEE Transactions on Automation Science and Engineering, vol. 10, no. 4, pp. 990-1003, 2013.
- [57] K. Thramboulidis and F. Christoulakis. "*UML4IoT - A UML-based approach to exploit IoT in cyber-physical manufacturing systems*", Computers in Industry, vol. 82, pp. 259-272, 2016.
- [58] K. Thramboulidis. "*A Cyber-Physical System-based Approach for Industrial Automation Systems*", Computers in Industry, vol. 72, pp. 92-102, 2015.
- [59] K. Thramboulidis, P. Bochalis and J. Bouloumpasis. "*A framework for MDE of IoT-based manufacturing cyber-physical systems*", Proceedings of the Seventh International Conference on the Internet of Things - IoT '17, 2017.
- [60] Pi4J, "*pi4j*". Available at: <http://pi4j.com/>.
- [61] F. Christoulakis. "*IoT in Industrial Automation Systems*", Diploma Thesis, 2017.



# Παράρτημα A

## Πηγαίος Κώδικας

Ο πηγαίος κώδικας της παρούσας εργασίας μπορεί να βρεθεί στην διεύθυνση: <https://github.com/sseg-APT/OSGiBasedLiqueurPlant>

