



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Εφαρμογές IoT σε βιομηχανικά συστήματα
παραγωγής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΛΕΩΝΙΔΑ ΤΣΑΝΤΑΡΛΙΩΤΗ
Α.Μ. 1059642

Επιβλέπων: Ανδρέας Κομνηνός
Επίκουρος Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την Ημερομηνία Εξέτασης.

(Υπογραφή)

(Υπογραφή)

(Υπογραφή)

.....
Ανδρέας Κομνηνός
Επίκουρος Καθηγητής

.....
Ιωάννης Γαροφαλάκης
Καθηγητής

.....
Ονοματεπώνυμο
Τίτλος

Πάτρα, Ιούνιος 2024



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ
ΠΛΗΡΟΦΟΡΙΚΗΣ

© Copyright συγγραφής Λεωνίδας Τσανταρλιώτης, 2024.

© Copyright θέματος Ανδρέας Κομνηνός, 2024.

Με την επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα.

Το περιεχόμενο αυτής της εργασίας δεν απηχεί απαραίτητα τις απόψεις του Τμήματος, του Επιβλέποντα, ή της επιτροπής που την ενέχρινε.

Τπεύθυνη Δήλωση

Βεβαιώνω ότι είμαι συγγραφέας αυτής της διπλωματικής εργασίας, και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στη διπλωματική εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η διπλωματική εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του τμήματος Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής του Πανεπιστημίου Πατρών.

Λεωνίδας Τσανταρλιώτης

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επίκουρο καθηγητή, κ. Ανδρέα Κομνηνό και τον Υπ. Διδάκτορα, κ. Γεώργιο Γεωργιάδη, για την εξαιρετική συνεργασία και καθοδήγηση καθ' όλη τη διάρκεια εκπόνησης της διπλωματικής εργασίας.

Περίληψη

Η Τέταρτη Βιομηχανική Επανάσταση μετασχηματίζει με ταχείς ρυθμούς τη βιομηχανία, καθιδηγούμενη από τεχνολογίες όπως το βιομηχανικό διαδίκτυο των πραγμάτων (IIoT) και τα Κυβερνο-Φυσικά Συστήματα (CPS). Η ενσωμάτωση αυτών των τεχνολογιών σε υφιστάμενα περιβάλλοντα παραγωγής θέτει προκλήσεις λόγω του ετερογενούς εξοπλισμού, των παλαιών συστημάτων και της ανάγκης για διαλειτουργικότητας μεταξύ διαφορετικών συσκευών IoT. Η παρούσα Διπλωματική Εργασία προτείνει μια καινοτόμο προσέγγιση για την αντιμετώπιση αυτών των προκλήσεων με την αξιοποίηση της AutomationML και της πλατφόρμας ThingsBoard για απλοποιημένη ενσωμάτωση και διαχείριση.

Η AutomationML χρησιμοποιείται για τη μοντελοποίηση του βιομηχανικού περιβάλλοντος παραγωγής, παρέχοντας μια τυποποιημένη αναπαράσταση. Χρησιμοποιώντας το LEGO SPIKE Prime κιτ, κατασκευάστηκε ένα ρομπότ το οποίο προσομειώνει μια πραγματική IoT συσκευή, παρουσιάζοντας την παραπάνω μοντελοποίηση. Το ρομπότ ενσωματώνει δύο Q-Learning σενάρια για προσαρμοστική συμπεριφορά: ένα που βασίζεται στην κατανομή Boltzmann και έναν άλλο που θέτει δυναμικά κατώφλια ανταμοιβής και χρησιμοποιεί έναν μηχανισμό σταθμισμένης επιλογής δράσης. Αυτά τα σενάρια σχεδιάστηκαν για τη βελτιστοποίηση των ενεργειών του ρομπότ υπό διαφορετικές συνθήκες και παρουσιάζεται μια ολοκληρωμένη αξιολόγηση των επιδόσεών τους, με έμφαση στην ταχύτητα σύγκλισης και την προσαρμοστικότητα.

Επιπλέον, με τη χρήση ενός Python Script, επιτεύχθηκε η μετάφραση του AutomationML μοντέλου σε οντότητες εντός της πλατφόρμας ThingsBoard. Ακόμα, επιτεύχθηκε η αμφίδρομη επικοινωνία μεταξύ του ρομπότ και του ThingsBoard χρησιμοποιώντας μια ποικιλία τεχνολογιών επικοινωνίας, όπως είναι το MQTT και τα REST APIs, επιτρέποντας την απρόσκοπτη παρακολούθηση και έλεγχο της συσκευής.

Η παρούσα ΔΕ αποδεικνύει την αποτελεσματικότητα της μοντελοποίησης μέσω AutomationML σε συνδυασμό με πλατφόρμες IoT για την ενίσχυση της ενσωμάτωσης εφαρμογών IoT σε βιομηχανικά συστήματα παραγωγής. Η προτεινόμενη προσέγγιση συμβάλλει σε μια πιο ευέλικτη, αποτελεσματική και κλιμακούμενη στρατηγική ανάπτυξης του IIoT στη βιομηχανία.

Λέξεις Κλειδιά

Τέταρτη Βιομηχανική Επανάσταση, Διαλειτουργικότητα, Κυβερνο-Φυσικά Συστήματα (CPS) IoT, IIoT, AutomationML, ThingsBoard, Q-Learning.

Abstract

The Fourth Industrial Revolution is rapidly transforming industry, driven by technologies like the Industrial Internet of Things (IIoT) and Cyber-Physical Systems (CPS). Integrating these technologies into existing production environments poses challenges due to heterogeneous equipment, legacy systems, and the need for interoperability between diverse IoT devices. This thesis proposes an innovative approach to address these challenges by leveraging AutomationML and IoT platforms for streamlined integration and management.

AutomationML is utilized to model the industrial production environment, providing a standardized representation. Using the LEGO SPIKE Prime kit, a robot was built that simulates a real IoT device, presenting the above modeling. This robot incorporates two Q-learning algorithms for adaptive behavior: one based on Boltzmann's distribution and another which sets dynamic reward thresholds and employs a weighted action choice mechanism. These algorithms were designed to optimize the robot's actions under varying conditions, and a comprehensive evaluation of their performance is presented, focusing on convergence speed and adaptability.

Furthermore, a Python script facilitated translation of the AutomationML model into entities within the ThingsBoard IoT platform. Bidirectional communication between the robot and ThingsBoard was established using a variety of communication technologies like MQTT, and REST APIs, allowing for seamless device monitoring and control.

This thesis demonstrates the effectiveness of using AutomationML modeling in conjunction with IoT platforms to enhance the integration of IoT applications in industrial production systems. The proposed approach contributes to a more flexible, efficient, and scalable IIoT deployment strategy within industry.

Keywords

Fourth Industrial Revolution, Interoperability, Cyber-Physical Systems (CPS), IoT, IIoT, AutomationML, ThingsBoard, Q-Learning

Περιεχόμενα

Ευχαριστίες	i
Περίληψη	iii
Abstract	v
Περιεχόμενα	viii
Κατάλογος Σχημάτων	x
Κατάλογος Πινάκων	xi
1 Εισαγωγή	1
1.1 Σημασία του προβλήματος	1
1.2 Σκοπός της Διπλωματικής Εργασίας	1
1.3 Διάρθρωση της Διπλωματικής Εργασίας	2
2 Βιβλιογραφική Ανασκόπηση	3
2.1 Η Τέταρτη Βιομηχανική Επανάσταση και ο αντίκτυπός της στη βιομηχανική παραγωγή	3
2.2 Industrial Internet of Things (IIoT)	7
2.2.1 Τεχνολογίες κλειδιά	7
2.2.2 Εφαρμογές και οφέλη	8
2.2.3 Πρότυπα Διαλειτουργικότητας και MQTT	9
2.2.4 Ασφάλεια	11
2.3 Cyber-Physical Systems	11
2.3.1 Ορισμός	12
2.3.2 Εφαρμογές στο Industry 4.0	13
2.3.3 Προκλήσεις	13
2.4 Προκλήσεις της ενσωμάτωσης του IoT στα συστήματα παραγωγής	14
2.4.1 Ποικιλία εξοπλισμού και αισθητήρων	14
2.4.2 Ενσωμάτωση παλαιών συστημάτων	15
2.4.3 Διαλειτουργικότητα και ανταλλαγή δεδομένων	16

2.5	Q-Learning και παρόμοιες μελέτες	16
2.5.1	Εξερεύνηση Boltzmann (Boltzmann Exploration)	17
2.5.2	Δυναμικά κατώφλια ανταμοιβής και προσαρμοστική μάθηση στο Q-Learning	18
3	Επισκόπηση Τεχνολογιών	21
3.1	AutomationML	21
3.1.1	Σκοπός	21
3.1.2	Γενική αρχιτεκτονική	22
3.2	ThingsBoard	26
3.2.1	ThingsBoard Transports	27
3.2.2	ThingsBoard Core	28
3.2.3	ThingsBoard Rule Engine	29
3.2.4	ThingsBoard WEB UI	30
4	Περιγραφή Υλοποίησης	33
4.1	LEGO SPIKE Prime	33
4.1.1	Περιγραφή	33
4.1.2	Περιβάλλον προγραμματισμού	35
4.1.3	Επεξήγηση βασικής λειτουργίας	35
4.1.4	Ανάλυση Q-Learning σεναρίων	36
4.2	AutomationML σε ThingsBoard οντότητες	43
4.3	Αμφίδρομη επικοινωνία μεταξύ ThingsBoard και LEGO SPIKE Prime	48
4.3.1	Επικοινωνία LEGO SPIKE Prime με PC μέσω Bluetooth	48
4.3.2	Επικοινωνία ThingsBoard με PC μέσω REST API και MQTT	50
4.3.3	Οπτικοποίηση δεδομένων στο ThingsBoard και παράδειγμα λειτουργίας	55
4.3.4	Αποτελέσματα	62
5	Επίλογος	65
5.1	Συμπεράσματα	65
5.2	Μελλοντικές κατευθύνσεις και πιθανές επεκτάσεις	66
A'	Pybricks Threshold-Guided Weighted Q-Learning	67

Κατάλογος Σχημάτων

2.1	Εξέλιξη των Βιομηχανιών (Πηγή: https://www.sentegroup.com/keeping-up-with-the-4th-industrial-revolution/)	4
2.2	Χαρακτηριστικά του Industry 4.0 (Πηγή: https://www.calsoft.com/what-is-industry-4-0/)	6
2.3	Industrial Internet of Things (Πηγή: https://www.rfpage.com/applications-of-industrial-internet-of-things/)	7
2.4	Διάγραμμα μιας βασικής ροής μηνυμάτων MQTT publish-subscribe. Ένας client στέλνει το μήνυμα $20^{\circ}C$ σε έναν MQTT Broker, ο οποίος το προωθεί στον subscriber που ενδιαφέρεται για το θέμα θερμοκρασία. (Πηγή: https://www.paessler.com/it-explained/mqtt/)	9
2.5	Σχέση μεταξύ IoT, IIoT, Industry 4.0 και CPS	11
2.6	Διάγραμμα ενός Cyber-Physical System (CPS) στο Industry 4.0 (Πηγή: https://www.geeksforgeeks.org/introduction-to-cyber-physical-system/)	12
3.1	Δομή AutomationML project [16]	23
3.2	Παράδειγμα Instance Hierarchy	24
3.3	Παράδειγμα System Unit Class	25
3.4	AutomationML BaseRoleClass Library	25
3.5	AutomationML InterfaceClass Library	26
3.6	Αρχιτεκτονική του ThingsBoard (Πηγή: https://2021.desosa.nl/print/projects/thingsboard/posts/architecture/)	27
3.7	Παράδειγμα ενός Rule Chain (Πηγή: https://thingsboard.io/docs/user-guide/rule-engine-2-0/overview/)	30
3.8	ThingsBoard Tenant Administrator WEB UI	30
3.9	Παράδειγμα Dashboard με διάφορα widgets που παρέχουν πληροφορίες σχετικά με τα υπάρχοντα Devices	32
4.1	LEGO SPIKE Prime BallLifter Robot	34
4.2	Energy Reward vs Motor Speed	36
4.3	Time Reward vs Execution Time	37
4.4	Boltzmann Distribution Q-Learning	40
4.5	Threshold-Guided Weighted Q-Learning	42

4.6	Instance Hierarchy of BallLifter Robot	44
4.7	System Unit Class BallLifter Robot	45
4.8	Interface Class BallLifter Robot	45
4.9	Αποτέλεσμα Python Script	46
4.10	Devices	47
4.11	Assets	47
4.12	Device Profiles	47
4.13	Asset Profiles	48
4.14	Αυφίδρομη επικοινωνία μεταξύ LEGO SPIKE Prime και ThingsBoard	48
4.15	Control Widgets	51
4.16	RPC set value method για Knob Control	52
4.17	RPC set value method για Switch Control	52
4.18	Rule Chain που κλείνει το Switch Control Widget αυτόματα μετά από 40 επαναλήψεις του BallLifter	54
4.19	Σύνδεση του Root Rule Chain με το custom Rule Chain	54
4.20	Telemetry Keys &Values	55
4.21	Widgets	56
4.22	Εκτέλεση Python Script	56
4.23	BallLifter αφού πατήσουμε το Hub Button	57
4.24	BallLifter αφού πατήσουμε το Left Button	57
4.25	Εκκίνηση BallLifter από το Dashboard	58
4.26	Αναμονή για ζύγισμα μπάλας	59
4.27	Zύγισμα μπάλας	59
4.28	Σετάρισμα BallLifter Robot	60
4.29	1η Επανάληψη	60
4.30	1η Επανάληψη	61
4.31	Μετά από 8 επαναλήψεις	61
4.32	Μετά από 39 επαναλήψεις	62
4.33	Μετά από 39 επαναλήψεις	62
4.34	Αποτελέσματα για Boltzmann Exploration with Epsilon-Greedy Q-Learning	63
4.35	Αποτελέσματα για Threshold-Guided Weighted Q-Learning	63

Κατάλογος Πινάκων

4.1	Βασικές διαφορές ανάμεσα στα δύο σενάρια	43
4.2	Κανόνες μετατροπής ανάμεσα σε AutomationML και ThingsBoard	43

Κεφάλαιο 1

Εισαγωγή

1.1 Σημασία του προβλήματος

Τα τελευταία χρόνια στα βιομηχανικά συστήματα παραγωγής υπάρχει μία ενδιαφέρουσα εξέλιξη λόγω της 4ης Βιομηχανικής Επανάστασης που περιλαμβάνει την ενσωμάτωση του Industrial Internet of Things (IIoT) καθώς και των τεχνολογιών Cyber Physical Systems (CPS). Αυτή η εξέλιξη ισχύει και για άλλους τομείς που περιλαμβάνει παρόμοιες απαιτήσεις, που αφορούν μεγάλους αριθμούς συσκευών (αισθητήρες) που πρέπει να διαλειτουργούν (interoperability), να ανταλλάσσουν τα δεδομένα τους και να είναι ελεγχόμενα. Η ενσωμάτωση στο περιβάλλον παραγωγής παραμένει μια πρόκληση λαμβάνοντας υπόψη την ποικιλομορφία του εξοπλισμού, συσκευές και αισθητήρες, την ύπαρξη παλαιών συστημάτων και την ανάγκη να ενσωματώνουν σύγχρονες συσκευές IoT που συμμετέχουν στο σύστημα παραγωγής. Η πολυπλοκότητα αυξάνεται από την ανάγκη διατήρησης αξιόπιστης και ασφαλούς συνδεσιμότητας σε όλο το δίκτυο.

Για να αντιμετωπιστούν αυτές οι ιδιαιτερότητες, η Ενισχυτική Μάθηση (Reinforcement Learning), ένας κλάδος της μηχανικής μάθησης που επιτρέπει στα συστήματα να μαθαίνουν βέλτιστες συμπεριφορές μέσω της αλληλεπίδρασης με το περιβάλλον τους, αναδύεται ως μια πολλά υποσχόμενη λύση. Αλγόριθμοι της Ενισχυτικής Μάθησης, όπως είναι το Q-Learning, μπορούν να δώσουν τη δυνατότητα στις συσκευές IoT να προσαρμόζουν τις ενέργειές τους, βελτιστοποιώντας την απόδοση και βελτιώνοντας τη συνολική αποδοτικότητα των βιομηχανικών διαδικασιών.

1.2 Σκοπός της Διπλωματικής Εργασίας

Η παρούσα διατριβή εμβαθύνει σε αυτές τις προκλήσεις και διερευνά καινοτόμες λύσεις για την αξιοποίηση της δύναμης του IIoT στα βιομηχανικά συστήματα παραγωγής. Σκοπός μας είναι:

- Να εξετάσουμε τις δυνατότητες της AutomationML (AML) ως ένα κρίσιμο εργαλείο για την υπέρβαση των εμποδίων ενσωμάτωσης ποικίλου εξοπλισμού, συσκευών και αισθητήρων σε σύγχρονα συστήματα παραγωγής

- Να αναπτύξουμε μια IoT εφαρμογή που θα προσομοιώνει μια βιομηχανική παραγωγική διαδικασία κάνοντας χρήση ενός ρομποτικού συστήματος και της πλατφόρμας Things-Board
- Να σχεδιάσουμε Q-Learning άλγορίθμους που επιτρέπουν την προσαρμοστική συμπεριφορά του ρομποτικού συστήματος, επιτρέποντάς του να βελτιστοποιεί την απόδοσή του ανάλογα με τις μεταβαλλόμενες συνθήκες
- Να προτείνουμε ιδέες για περαιτέρω πρόοδο στην ενσωμάτωση των τεχνολογιών IoT στα βιομηχανικά συστήματα παραγωγής, με βάση τα ευρήματα και τις εμπειρίες που αποκτήθηκαν από τη μελέτη

Με την εκπλήρωση των παραπάνω στόχων, η μελέτη στοχεύει να συμβάλει στην πρόοδο των εφαρμογών IoT σε βιομηχανικά συστήματα παραγωγής, ιδίως όσον αφορά τη μοντελοποίηση, την εξομοίωση και την προσαρμογή των διαδικασιών παραγωγής σε πραγματικό χρόνο.

1.3 Διάρθρωση της Διπλωματικής Εργασίας

Η παρούσα Διπλωματική Εργασία οργανώνεται σε 5 κεφάλαια. Το Κεφάλαιο 1 είναι εισαγωγικό και αναφέρει τους στόχους της εργασίας. Το Κεφάλαιο 2 παρουσιάζει μια εκτενή βιβλιογραφική ανασκόπηση της υπάρχουσας βιβλιογραφίας για το Industry 4.0, θέτοντας τις βάσεις για τη διερεύνηση των εφαρμογών του IoT σε βιομηχανικά περιβάλλοντα. Το Κεφάλαιο 3, εμβαθύνει στις βασικές τεχνολογίες που χρησιμοποιήθηκαν στο κομμάτι της υλοποίησης, δηλαδή την AutomationML και το ThingsBoard. Στο κεφάλαιο 4 παρουσιάζεται, μια αναλυτική περιγραφή της υλοποίησης που πραγματοποιήθηκε στα πλαίσια αυτής της ΔΕ, όπου ουσιαστικά παρουσιάζονται η πρακτική εφαρμογή των τεχνολογιών που αναφέρθηκαν στο Κεφάλαιο 3 και η υλοποίηση ενός ρομποτικού συστήματος. Τέλος, στο Κεφάλαιο 5, παρουσιάζονται τα συμπεράσματα και πιθανές επεκτάσεις της ΔΕ.

Κεφάλαιο 2

Βιβλιογραφική Ανασκόπηση

2.1 Η Τέταρτη Βιομηχανική Επανάσταση και ο αντίκτυπός της στη βιομηχανική παραγωγή

Πρωτού εμβαθύνουμε στο τι πραγματεύεται και ποιος είναι ο αντίκτυπος της Τέταρτης Βιομηχανικής Επανάστασης, είναι απαραίτητο να κατανοήσουμε την εξέλιξη της βιομηχανικής προόδου (βλ. Σχήμα 2.1) που διαμόρφωσε τον σύγχρονο κόσμο μας.

1η Βιομηχανική Επανάσταση

Η Πρώτη Βιομηχανική Επανάσταση αντιπροσώπευε τη μετάβαση από τη χειρωνακτική παραγωγή στη χρήση μηχανών, καθώς εισήγαγε τη χρήση ατμοηλεκτρικής και υδροηλεκτρικής ενέργειας. Αναφέρεται στην περίοδο μεταξύ του 1760 έως το 1840. Οι επιδράσεις της επανάστασης αυτής επηρέασαν την κλωστοϋφαντουργία, τη βιομηχανία του σιδήρου, τη γεωργία και την εξόρυξη, ενώ είχε επίσης κοινωνικές επιπτώσεις με τη δημιουργία μιας αναπτυσσόμενης μεσαίας τάξης [7].

2η Βιομηχανική Επανάσταση

Η Δεύτερη Βιομηχανική Επανάσταση, γνωστή και ως Τεχνολογική Επανάσταση, είναι η περίοδος μεταξύ 1871 και 1914 και προέκυψε ως αποτέλεσμα της κατασκευής εκτεταμένων σιδηροδρομικών και τηλεγραφικών δικτύων, τα οποία διευκόλυναν την ταχύτερη μετακίνηση ανθρώπων και ιδεών, καθώς και ηλεκτρικής ενέργειας. Ο αυξανόμενος εξηλεκτρισμός επέτρεψε στα εργοστάσια να αναπτύξουν τη σύγχρονη γραμμή παραγωγής. Ήταν μια περίοδος μεγάλης οικονομικής ανάπτυξης, με αύξηση της παραγωγικότητας, η οποία προχάλεσε επίσης αύξηση της ανεργίας, καθώς πολλοί εργάτες εργοστασίων αντικαταστάθηκαν από μηχανές [19].

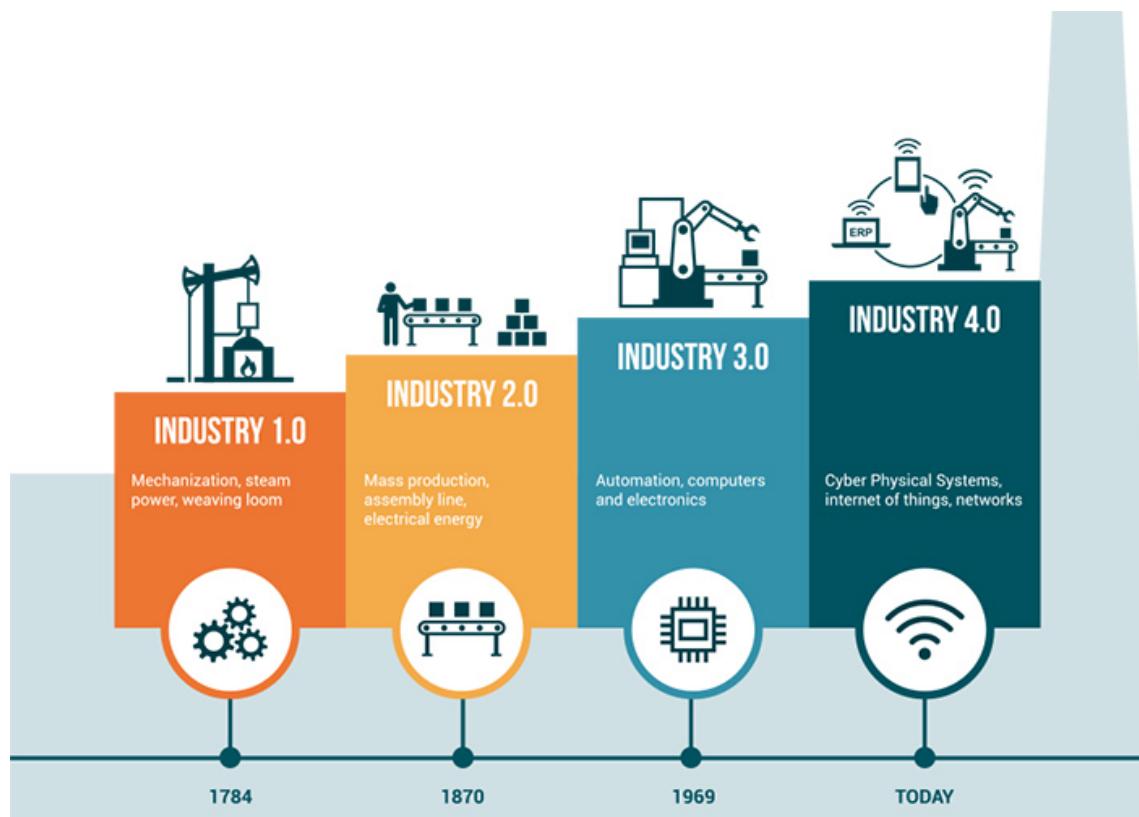
3η Βιομηχανική Επανάσταση

Η Τρίτη Βιομηχανική Επανάσταση, έλαβε χώρα στα τέλη του 20ού αιώνα, μετά το πέρας των δύο παγκοσμίων πολέμων, και ήταν αποτέλεσμα της επιβράδυνσης της εκβιομηχάνισης και

της τεχνολογικής προόδου σε σύγκριση με τις προηγούμενες περιόδους. Η παραγωγή του υπολογιστή Z1, ο οποίος χρησιμοποιούσε δυαδικούς αριθμούς κινητής υποδιαστολής και λογική Boole, μια δεκαετία αργότερα, αποτέλεσε την απαρχή πιο προηγμένων ψηφιακών εξελίξεων. Η επόμενη σημαντική εξέλιξη στις τεχνολογίες επικοινωνίας ήταν ο υπερυπολογιστής, με εκτεταμένη χρήση των τεχνολογιών υπολογιστών και επικοινωνιών στην παραγωγική διαδικασία.

4η Βιομηχανική Επανάσταση

Η Τέταρτη Βιομηχανική Επανάσταση (4th Industrial Revolution ή 4IR ή Industry 4.0), θεωρείται ότι ξεκίνησε στις αρχές του 21ου αιώνα. Αντιπροσωπεύει τη συγχώνευση του ψηφιακού, του φυσικού και του βιολογικού κόσμου, οδηγώντας σε πρωτοφανείς προόδους στην τεχνολογία και αναδιαμορφώνει διάφορες πτυχές της κοινωνίας, της οικονομίας και της διοικούσας ηγεσίας. Χαρακτηρίζομενη από την ταχεία ενσωμάτωση αναδυόμενων τεχνολογιών, όπως είναι η τεχνητή νοημοσύνη, η ρομποτική, το Internet of Things, το 3D Printing, η γενετική μηχανική και η κβαντική πληροφορική, το Industry 4.0 φέρνει επανάσταση στις βιομηχανίες και αμφισβητεί τα παραδοσιακά πρότυπα.



Σχήμα 2.1: Εξέλιξη των Βιομηχανιών (Πηγή: <https://www.sentegroup.com/keeping-up-with-the-4th-industrial-revolution/>)

Καθώς το Industry 4.0 συνεχίζει να εξελίσσεται, ο τομέας της βιομηχανικής παραγωγής βρίσκεται στην πρώτη γραμμή αυτής της τεχνολογικής επανάστασης, η οποία μετασχηματίζει

2.1 Η Τέταρτη Βιομηχανική Επανάσταση και ο αντίκτυπός της στη βιομηχανική παραγωγή

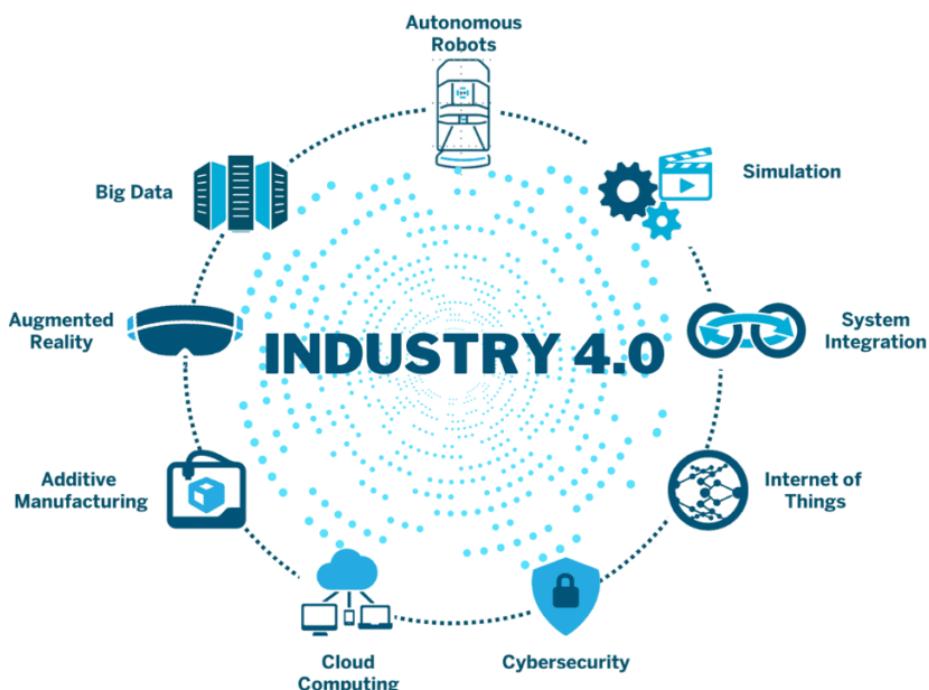
σημαντικά τις δραστηριότητές της, παρουσιάζοντας αρκετές θετικές επιδράσεις αλλά και καινούργιες προκλήσεις [20]. Ας ξεκινήσουμε απαριθμώντας τα θετικά:

- **Αυξημένη παραγωγικότητα και αποδοτικότητα:** Η αυτοματοποίηση, οι αποφάσεις βάσει δεδομένων και η βελτιστοποίηση σε πραγματικό χρόνο οδηγούν σε υψηλότερη απόδοση και αποτελεσματικότερη χρήση των πόρων
- **Βελτιωμένη ευελιξία και προσαρμογή:** Τα έξυπνα εργοστάσια προσαρμόζονται στις μεταβαλλόμενες απαιτήσεις, επιτρέποντας τη μαζική προσαρμογή και τα εξατομικευμένα προϊόντα
- **Προληπτική συντήρηση και μειωμένος χρόνος διακοπής λειτουργίας:** Η παρακολούθηση μέσω αισθητήρων και οι αναλύσεις από AI προβλέπουν βλάβες στον εξοπλισμό, αποτρέποντας δαπανηρές διακοπές
- **Ευκαιρίες για νέα προϊόντα και υπηρεσίες:** Το 3D Printing ξεκλειδώνει καινοτόμα σχέδια, ενώ η τεχνητή νοημοσύνη επιτρέπει την ανάπτυξη προϊόντων με βάση τα δεδομένα (data-driven)
- **Βελτίωση ασφάλειας και αξιοπιστίας:** Μειωμένη εξάρτηση στην ανθρώπινη παρέμβαση σε επικίνδυνες καταστάσεις και δυνατότητες αέναης λειτουργίας
- **Αυξημένη ανάπτυξη των εσόδων:** Η ζήτηση βελτιωμένου εξοπλισμού, εφαρμογές δεδομένων και εξατομικευμένων προϊόντων, οδηγεί στην αύξηση των εσόδων
- **Μαζική παραγωγή υψηλής ευελιξίας:** Ερευνώνται αυτόνομα συστήματα για ακόμη μεγαλύτερη ευελιξία
- **Βελτιστοποίηση λήψης αποφάσεων:** Η διαφάνεια σε πραγματικό χρόνο επιτρέπει ταχύτερες και πιο τεκμηριωμένες αποφάσεις σχεδιασμού και παραγωγής
- **Βελτιστοποίηση διαδικασίας προμηθειών:** Το λογισμικό επιτρέπει την ευέλικτη και αυτόνομη επιλογή προμηθευτών, μειώνοντας ενδεχομένως το κόστος αγοράς
- **Ακριβής διαχείριση κινδύνων:** Η ανάλυση δεδομένων βοηθά στον έγκαιρο εντοπισμό και αντιμετόπιση των κινδύνων
- **Μειωμένο κόστος:** Το κόστος μεταφοράς, αποθήκευσης και παραγωγής μπορεί να μειωθεί με διάφορους τρόπους
- **Αποδοτικότητα πόρων και βιωσιμότητα:** Το Industry 4.0 προωθεί την αποδοτική χρήση των πόρων και ελαχιστοποιεί τα απόβλητα
- **Βελτιωμένη ισορροπία μεταξύ εργασίας και προσωπικής ζωής:** Οι ευέλικτοι οργανισμοί εργασίας καλύπτουν τις ανάγκες των εργαζομένων για προσωπική και επαγγελματική ανάπτυξη

Ωστόσο, όπως αναφέραμε, δημιουργούνται και νέες προκλήσεις μερικές εκ των οποίων είναι:

- **Μείωση εργασιακών θέσεων:** Η αυτοματοποίηση μπορεί να οδηγήσει σε απώλεια θέσεων εργασίας σε ορισμένους τομείς, απαιτώντας την επανεκπαίδευση και την προσαρμογή του εργατικού δυναμικού
- **Έλλειμμα δεξιοτήτων:** Απαιτούνται νέες δεξιότητες για τη λειτουργία και τη συντήρηση προηγμένων τεχνολογιών
- **Απειλές για την ασφάλεια στον κυberνοχώρο (cybersecurity):** Η αυξημένη συνδεσιμότητα εκθέτει τα βιομηχανικά συστήματα σε κυβερνοεπιθέσεις (cyberattacks)
- **Δεοντολογικά ζητήματα:** Θέματα όπως η προστασία της ιδιωτικής ζωής των δεδομένων, η υπεύθυνη ανάπτυξη της TN και η ισότιμη πρόσβαση στην τεχνολογία χρήζουν προσεκτικής εξέτασης

Στο σύνολό της, η Τέταρτη Βιομηχανική Επανάσταση παρουσιάζει νέες ευκαιρίες αλλά και καινούργιες προκλήσεις στον τομέα της βιομηχανίας. Η κατανόηση και η προσαρμογή στα νέα δεδομένα που εισήγαγε το Industry 4.0, χρήζει κομβικής σημασίας για την ανάπτυξη και την ευημερία των επιχειρήσεων αλλά και της κοινωνίας γενικότερα.

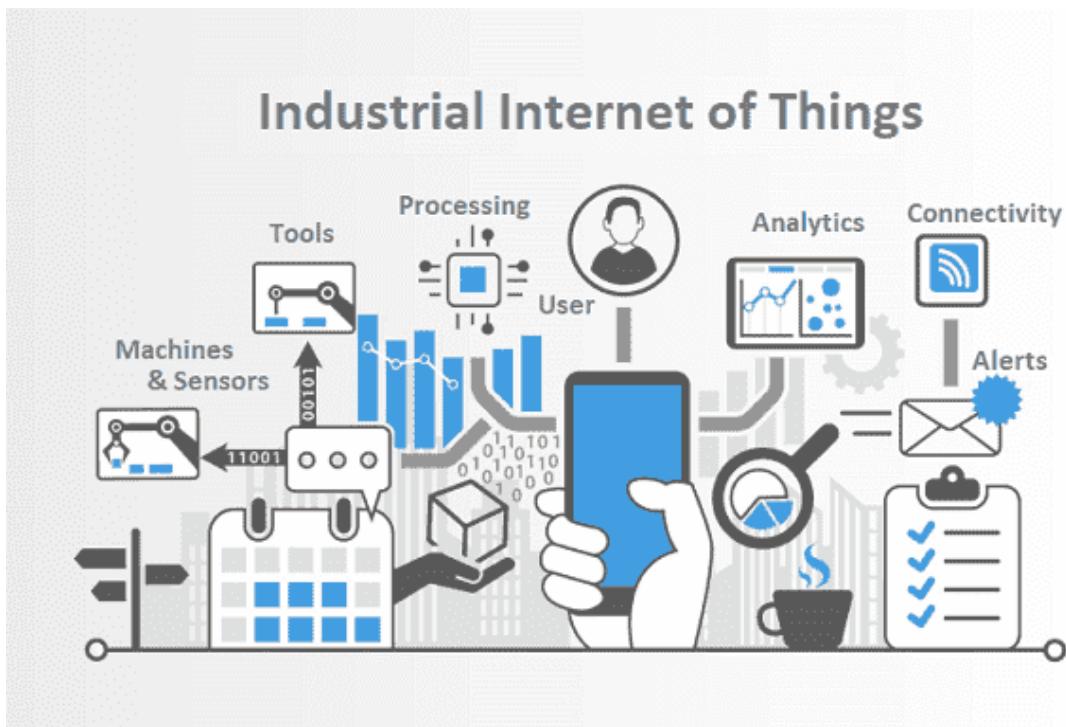


Σχήμα 2.2: Χαρακτηριστικά του Industry 4.0 (Πηγή: <https://www.calsoft.com/what-is-industry-4-0/>)

2.2 Industrial Internet of Things (IIoT)

Στο προηγούμενο κεφάλαιο έγινε αναφόρα στον όρο Industrial Internet of Things (IIoT). Σε αυτή την ενότητα θα εμβαθύνουμε λίγο παραπάνω στη σημασία του IIoT και το ρόλο που έχει στο Industry 4.0.

Για να κατανοήσουμε καλύτερα τον όρο IIoT, ας αναφερθούμε αρχικά στο IoT (Internet of Things). Τα τελευταία χρόνια, το IoT έχει κερδίσει τεράστια προσοχή από πολλές διαφορτικές βιομηχανίες γιατί επιτρέπει την εύκολη διασύνδεση πολλών συσκευών, ανεξαρτήτου τόπου και χρόνου [15]. Στην καθημερινή ζωή, το IoT έχει πολλές εφαρμογές (έξυπνο σπίτι, έξυπνη πόλη, έξυπνο δίκτυο και έξυπνες μεταφορές), οι οποίες προσφέρουν στους πολίτες άνεση, ευκολία και αποτελεσματικότερη αξιοποιήση του χρόνου τους. Επεκτείνοντας λοιπόν αυτή την τεχνολογία, το IIoT, όπως φαίνεται και από το όνομα του, είναι η εφαρμογή των τεχνολογιών του IoT, στον βιομηχανικό τομέα. Μέσω της διασύνδεσης του «οιδιήποτε» (αισθητήρες, ενεργοποιητές, ελεγκτές, γραμμές παραγωγής, εξοπλισμό κ.λπ.), αποσκοπεί στη βελτίωση της παραγωγικότητας, της αποδοτικότητας και της ασφάλειας των βιομηχανικών μονάδων [23].



Σχήμα 2.3: Industrial Internet of Things (Πηγή: <https://www.rfpage.com/applications-of-industrial-internet-of-things/>)

2.2.1 Τεχνολογίες κλειδιά

Ένα σύστημα IIoT αποτελέται από δύο βασικά στοιχεία, συστήματα κυβερνοχώρου (cyber-systems) και υλικά συστήματα (physical systems) [23]. Οι τεχνολογίες-κλειδιά ενός τέτοιου συστήματος είναι οι εξής:

- 1. Αισθητήρες και ενεργοποιητές:** Οι αισθητήρες συλλέγουν δεδομένα από τον φυσικό κόσμο, όπως θερμοκρασία, πίεση, κραδασμούς κ.λπ. ενώ οι ενεργοποιητές εκτελούν ενέργειες βάσει δεδομένων (π.χ. πάτημα ενός κουμπιού)
- 2. Πρωτόκολλα επικοινωνίας:** Δυνατότητα ανταλλαγής δεδομένων μεταξύ συσκευών και συστημάτων
- 3. Edge Computing:** Επεξεργασία δεδομένων τοπικά, με αποτέλεσμα τη μείωση της καθυστέρησης και του φορτίου δικτύου
- 4. Υπολογιστικό νέφος (Cloud Computing):** Αποθήκευση-Ανάλυση μεγάλου όγκου δεδομένων με σκοπό την εξαγωγή συμπερασμάτων
- 5. Ανάλυση δεδομένων:** Εξαγωγή πληροφορίας από τα συλλεχθέντα δεδομένα
- 6. Ασφάλεια:** Προστασία δεδομένων και συστημάτων από απειλές στον κυβερνοχώρο (cyber-threats)
- 7. Διαλειτουργικότητα:** Επιτρέπει την επικοινωνία και συνεργασία μεταξύ διαφορετικών συσκευών και συστημάτων

2.2.2 Εφαρμογές και οφέλη

Λόγω της ραγδαίας ανάπτυξης του βιομηχανικού τομέα, το ΙΙoT έχει βρει εφαρμογή σε πολλούς τομείς του σύγχρονου κοινωνικοοικονομικού κόσμου και αποτελεί την κινητήρια δύναμη του Industry 4.0 [4]. Μερικές από αυτές τις εφαρμογές, παρουσιάζονται στη συνέχεια:

- 1. Ψηφιακό εργοστάσιο:** Μέσω της αυτοματοποίησης, τα μηχανήματα και τα gadgets μπορούν να παρέχουν στους μηχανικούς και στους κατασκευαστές χρήσιμες πληροφορίες που μπορούν να χρησιμοποιήθουν για την εξ αποστάσεως διαχείριση των μονάδων παραγωγής
- 2. Διαχείριση αποθήκης:** Οι ΙΙoT εφαρμογές επιτρέπουν συνεχή παρακολούθηση κατά μήκος ολόχληρης της αλυσίδας εφοδιασμού
- 3. Έλεγχος ποιότητας:** Οι αισθητήρες παίζουν καθοριστικό ρόλο στη συλλογή και στην ανάλυση πληροφοριών οι οποίες στη συνέχεια αξιοποιούνται για την διαμόρφωση και την τελειοποίηση του τελικού προϊόντος. Ως εκ τούτου, ένας σωστός μηχανισμός ελέγχου ποιότητας παίζει καθοριστικό ρόλο στην σωστή και αποτελεσματική ανάλυση της πληροφορίας
- 4. Διαχείριση των εγκαταστάσεων:** Η χρήση IoT αισθητήρων, επιτρέπει την ανίχνευση πιθανών βλαβών και παρέχει ειδοποιήσεις για την συντήρηση του εξοπλισμού. Επίσης, επιτρέπει το δυναμικό έλεγχο της σωστής κατάστασης λειτουργίας των μηχανημάτων, με αποτέλεσμα την αύξηση της αποδοτικότητας

5. Εξοικονόμηση ενέργειας: Το IIoT μπορεί να χρησιμοποιηθεί για τη βελτίωση της ενεργειακής απόδοσης. Για παράδειγμα, μπορούν να χρησιμοποιηθούν αισθητήρες για την παρακολούθηση της ενεργειακής κατανάλωσης ενός εργοστασίου.

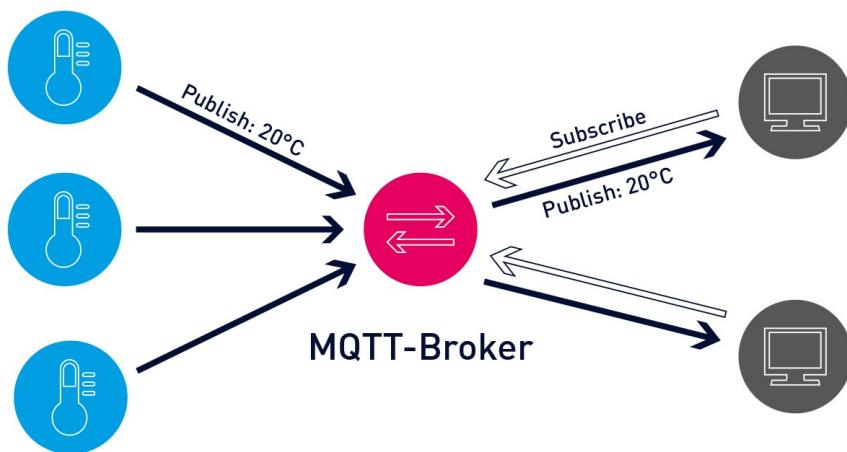
Αυτά είναι μερικά μόνο παραδείγματα από τις πολλές εφαρμογές του IIoT. Η επεκτασιμότητα, η σύνδεση, η αξιοπιστία, η εξοικονόμηση χρόνου και κόστους μέσω της προληπτικής συντήρησης και τα ενισχυμένα μέτρα ασφάλειας και προστασίας, καθιστούν το IIoT εξαιρετικά κερδοφόρο για τις σύγχρονες βιομηχανίες [17].

2.2.3 Πρότυπα Διαλειτουργικότητας και MQTT

Η δημιουργία προτύπων διαλειτουργικότητας έχει σημαντικό ρόλο στον κόσμο του IIoT γιατί εξασφαλίζουν την ομαλή συνεργασία μεταξύ συσκευών και συστημάτων από διαφορετικούς κατασκευαστές. Ενώ υπάρχουν πολλά πρότυπα για την κάλυψη αυτής της ανάγκης, όπως το OPC UA, το CoAP και το DDS, σε αυτή την εργασία θα επικεντρωθούμε στις δυνατότητες και τη σημασία του MQTT (Message Queuing Telemetry Transport), του οποίου έγινε και χρήση στο κομμάτι της υλοποίησης.

Ορισμός

Το MQTT είναι ένα ελαφρύ πρωτόκολλο ανταλλαγής μηνυμάτων, το οποίο έχει σχεδιαστεί με σκοπό την αποτελεσματική επικοινωνία μεταξύ συσκευών με περιορισμένους πόρους, κάνοντάς το κατάλληλο για τον τομέα του IIoT. Στο πλαίσιο του MQTT, οι συσκευές χωρίζονται σε δύο κύριους ρόλους: τον Publisher και τον Subscriber, που επικοινωνούν μέσω ενός Broker. Όπως φαίνεται και στο Σχήμα 2.4, ο Publisher αποστέλλει δεδομένα στον Broker, καθορίζοντας ένα συγκεκριμένο θέμα στο μήνυμα, ενώ οι Subscribers λαμβάνουν δεδομένα από διάφορους Publishers, ανάλογα με τη συνδρομή τους σε αντίστοιχα θέματα [18].



Σχήμα 2.4: Διάγραμμα μιας βασικής ροής μηνυμάτων MQTT publish-subscribe. Ένας client στέλνει το μήνυμα 20°C σε έναν MQTT Broker, ο οποίος το προωθεί στον subscriber που ενδιαφέρεται για το θέμα θερμοκρασία. (Πηγή: <https://www.paessler.com/it-explained/mqtt/>)

Βασικά Χαρακτηριστικά

Παρακάτω, περιγράφονται τα βασικά χαρακτηριστικά του MQTT που αναδεικνύουν την αποτελεσματικότητά του στο IoT:

- Ασύγχρονη επικοινωνία:** Οι συσκευές μπορούν να δημοσιεύουν μηνύματα στον Broker χωρίς να περιμένουν επιβεβαίωσεις
- Last Will and Testament (LWT):** Οι MQTT Clients μπορούν να καθορίσουν ένα μήνυμα «τελευταίας βιούλησης» που θα δημοσιευτεί από τον Broker σε περίπτωση απροσδόκητης αποσύνδεσης, ενισχύοντας με αυτόν τον τρόπο την αξιοπιστία
- Επίπεδα QoS (Quality of Service):** Το MQTT υποστηρίζει διαφορετικά επίπεδα QoS, επιτρέποντας τη διασφάλιση της παράδοσης μηνυμάτων σύμφωνα με τις απαιτήσεις αξιοπιστίας της εκάστοτε εφαρμογής
- Διατήρηση Μηνυμάτων:** Ο Broker έχει τη δυνατότητα να κρατήσει το τελευταίο μήνυμα ενός θέματος (topic), διασφαλίζοντας ότι οι Subscribers λαμβάνουν τις πιο πρόσφατες πληροφορίες κατά τη σύνδεση τους

Πλεονεκτήματα

Στη συνέχεια παρουσιάζονται τα πλεονεκτήματα της χρήσης του πρωτοκόλλου MQTT:

- Επεκτασιμότητα:** Ο ελαφρύς χαρακτήρας του MQTT και η αποδοτική δρομολόγηση μηνυμάτων, επιτρέπουν την εύκολη χλιμάκωση και την δημιουργία ενός δικτύου που αποτελείται από χιλιάδες ή ακόμη και εκατομμύρια διασυνδεδεμένες συσκευές
- Διαλειτουργικότητα:** Η ευρεία υποστήριξη του MQTT από διάφορες πλατφόρμες, γλώσσες προγραμματισμού και λειτουργικά συστήματα, διασφαλίζει τη διαλειτουργικότητα μεταξύ διαφορετικών συσκευών και IoT συστημάτων
- Αξιοπιστία:** Τα επίπεδα QoS του MQTT και χαρακτηριστικά όπως η διατήρηση μηνυμάτων και το LWT, ενισχύουν την αξιοπιστία παράδοσης της πληροφορίας, που είναι κρίσιμη για την ομαλή λειτουργία IoT εφαρμογών
- Ευελιξία:** Το ευέλικτο σύστημα δρομολόγησης μηνυμάτων που περιγράφηκε νωρίτερα, επιτρέπει τη δυναμική ανταλλαγή δεδομένων και την ενσωμάτωση σε υπάρχωντα συστήματα, επιτρέποντας στις βιομηχανίες να προσαρμόζονται στις μεταβαλλόμενες επιχειρηματικές απαιτήσεις

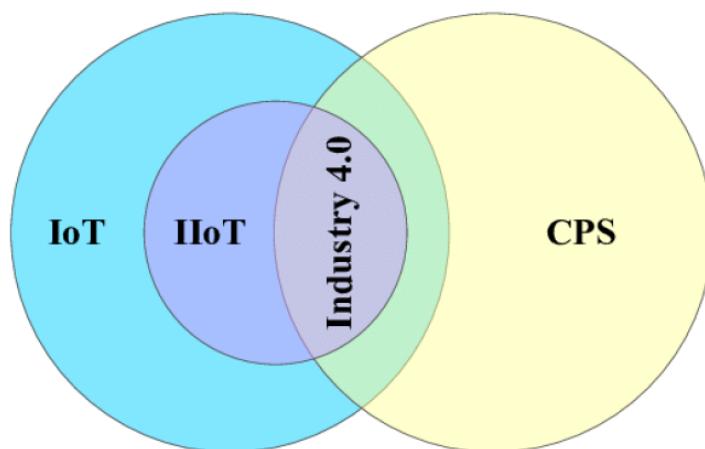
Εν κατακλείδι, το MQTT αναδεικνύει τη σημαντικότητα της διαλειτουργικότητας και της ανάγκης ομαλής επικοινωνίας στα IoT συστήματα. Ο ελαφρύς, επεκτάσιμος και αξιόπιστος χαρακτήρας του, σε συνδυασμό με τις πολυάριθμες εφαρμογές του, το καθιστούν μια εξαιρετική επιλογή για την ανάπτυξη IoT συστημάτων σε διάφορους χλάδους.

2.2.4 Ασφάλεια

Η ασφάλεια στο IIoT είναι υψηλής σημασίας λόγω της διασυνδεδεμένης φύσης των συσκευών. Η προστασία ευαίσθητων δεδομένων, η αποτροπή μη εξουσιοδοτημένης πρόσβασης και η διασφάλιση της ακεραιότητας των συστημάτων πρέπει να αποτελούν προτεραιότητα χάθε βιομηχανίας [24]. Μερικά από τα πιο σημαντικά ζητήματα ασφάλειας στο IIoT είναι:

- **Απόρρητο/Ιδιωτικότητα Δεδομένων:** Διασφάλιση ότι η μετάδοση και η αποθήκευση ευαίσθητων δεδομένων προστατεύεται από μη εξουσιοδοτημένη πρόσβαση ή παραβιάσεις
- **Ακεραιότητα CPS (Cyber-Physical Systems):** Οι περισσότερες IIoT συσκευές είναι CPS, οπότε πρέπει να υποστηρίζεται η ακεραιότητα. Ένας βασικός μηχανισμός για την επαλήθευση της ακεραιότητας του configuration ενός συστήματος λογισμικού είναι πιστοποίηση (attestation), η οποία επιτρέπει την ανίχνευση κακόβουλων τροποποιήσεων [24]
- **Ασφάλεια διακτύου:** Ασφάλεια των καναλιών επικοινωνίας μεταξύ των IIoT συσκευών, των πυλών (gateways) και του cloud για την αποτροπή υποκλοπής ή μη εξουσιοδοτημένης πρόσβασης
- **Αντιμετώπιση και αποκατάσταση προβλημάτων:** Ανάπτυξη στρατηγικών και πρωτοκόλλων για την έγκαιρη ανίχνευση και αντιμετώπιση παραβιάσεων ασφαλέιας ώστε να ελαχιστοποιούνται οι επιπτώσεις στις διεργασίες

2.3 Cyber-Physical Systems

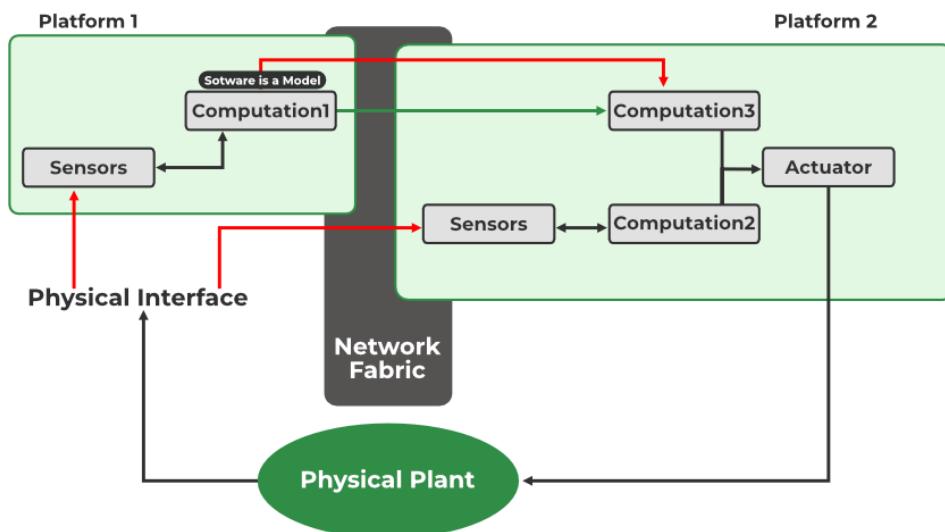


Σχήμα 2.5: Σχέση μεταξύ IoT, IIoT, Industry 4.0 και CPS

Η τέταρτη βιομηχανική επανάσταση αντιπροσωπεύεται από τη σύγκλιση του ψηφιακού και του φυσικού κόσμου. Στο επίκεντρο αυτής της μετεξέλιξης βρίσκονται τα Κυβερνο-Φυσικά Συστήματα (CPS). Αντιπροσωπεύουν την απόλυτη συνένωση της ψηφιακής επεξεργασίας με τον πραγματικό κόσμο, δημιουργώντας έναν νέο τρόπο αλληλεπίδρασης ανάμεσα σε ψηφιακούς αλγορίθμους και φυσικά συστήματα [12]. Από τις βιομηχανίες και τον υγειονομικό τομέα έως τις μεταφορές και τη διαχείριση της ενέργειας, τα CPS αναδύονται ως το θεμέλιο της σύγχρονης τεχνολογικής εξέλιξης, επιδιώκοντας την απόλυτη αυτοματοποίηση, την ευφυή λειτουργία και την αποτελεσματικότητα. Σε αυτή την ενότητα, εξετάζουμε το ευρύ φάσμα των CPS, αναλύοντας το σκοπό, τις εφαρμογές και τις προκλήσεις που προσφέρουν στον σύγχρονο τεχνολογικό κόσμο.

2.3.1 Ορισμός

Όπως αναφέρθηκε και προηγουμένως, τα CPS είναι αυτοματοποιημένα συστήματα που επιτρέπουν τη διασύνδεση λειτουργιών της φυσικής πραγματικότητας με υπολογιστικές και επικοινωνιακές υποδομές. Πιο συγκεκριμένα, είναι πολύπλοκα συστήματα που συνδυάζουν φυσικά στοιχεία, όπως είναι οι μηχανές και τα ρομπότ, με υπολογιστικά στοιχεία, όπως είναι οι αισθητήρες, οι ενεργοποιητές και διάφορα είδη λογισμικού. Στο Σχήμα 2.6 παρουσιάζεται ένα διάγραμμα μιας φυσικής εγκατάστασης (μηχανής ή διεργασίας) συνδεδεμένης με αισθητήρες, ενεργοποιητές και υπολογιστικές μονάδες. Τα μοντέλα λογισμικού της εγκατάστασης χρησιμοποιούνται για την ανάλυση των δεδομένων των αισθητήρων και την αποστολή εντολών ελέγχου στους ενεργοποιητές, επιτρέποντας τη βελτιστοποίηση και την αυτοματοποίηση σε πραγματικό χρόνο.



Σχήμα 2.6: Διάγραμμα ενός Cyber-Physical System (CPS) στο Industry 4.0 (Πηγή: <https://www.geeksforgeeks.org/introduction-to-cyber-physical-system/>)

2.3.2 Εφαρμογές στο Industry 4.0

Τα CPS έχουν βρει εφαρμογή σε πολλούς διαφορετικούς τομείς, ιδίως στο πλαίσιο του Industry 4.0. Η χρήση τους οδηγεί στην αύξηση της αποδοτικότητας, της παραγωγικότητας και της ανταγωνιστικότητας καθιστώντας τα ένα αναπόσπαστο κομμάτι των σύγχρονων επιχειρήσεων. Μερικές από τις πιο σημαντικές εφαρμογές τους μπορούν να κατηγοριοποιηθούν ως εξής:

Βιομηχανία

- **Έξυπνα εργοστάσια:** Περιλαμβάνουν διασυνδεδεμένα μηχανήματα, ρομπότ και αισθητήρες, τα οποία παρακολουθούν και βελτιστοποιούν τις διαδικασίες παραγωγής σε πραγματικό χρόνο
- **Προβλεπτική συντήρηση:** Οι αισθητήρες στον εξοπλισμό ανιχνεύουν πιθανές βλάβες πριν συμβούν, ελαχιστοποιώντας το χρόνο διακοπής λειτουργίας και το κόστος συντήρησης

Επιμελητεία (Logistics)

- **Έξυπνες αποθήκες:** Αυτόματα συστήματα αποθήκευσης βελτιστοποιούν τη χρήση του χώρου και την ολοκλήρωση των παραγγελιών
- **Παρακολούθηση - Εντοπισμός (Track - Trace):** Η παρακολούθηση των εμπορευμάτων σε πραγματικό χρόνο κατά μήκος όλης της αλυσίδας εφοδιασμού, πιστοποιεί την ασφάλεια και την αποτελεσματικότητα
- **Αυτόνομα οχήματα:** Αυτοκινούμενα φορτηγά και drones παραδίδουν αγαθά, μειώνοντας το κόστος μεταφοράς και τις εκπομπές ρύπων

Ενέργεια

- **Έξυπνα δίκτυα:** Εξοικονόμηση ενέργειας μέσω της βελτιστοποίησης της διανομής ενέργειας, μείωσης της κατανάλωσης της και της αποτελεσματικής ενσωμάτωσης ανανεώσιμων πηγών
- **Έξυπνοι μετρητές:** Επιτρέπουν την παρακολούθηση της κατανάλωσης της ενέργειας και βελτιστοποιούν τα πρότυπα κατανάλωσης

2.3.3 Προκλήσεις

Παρά τις τεράστιες δυνατότητές τους, τα CPS στο Industry 4.0 αντιμετωπίζουν αρκετές προκλήσεις οι οποίες πρέπει να αντιμετωπιστούν για την επίτευξη της ομαλής λειτουργίας τους [6].

- **Διαλειτουργικότητα:** Η μεγάλη ποικιλομορφία των στοιχείων που αποτελούν ένα CPS, δυσκολεύει την ανταλλαγή δεδομένων, λόγω της χρήσης διαφορετικών πρωτοκόλλων επικοινωνίας
- **Ασφάλεια:** Τα διασυνδεδεμένα δίκτυα CPS είναι ευάλωτα σε κυβερνοεπιθέσεις που μπορούν να θέσουν σε κίνδυνο την ακεραιότητα των δεδομένων, την αξιοπιστία του συστήματος, ακόμη και τη φυσική ασφάλεια
- **Ιδιωτικότητα δεδομένων:** Η συλλογή και επεξεργασία μεγάλου όγκου δεδομένων, εγείρει ανησυχίες σχετικά με την προστασία της ιδιωτικής ζωής και την πιθανή κατάχρηση πληροφορίας
- **Επεκτασιμότητα:** Τα CPS πρέπει να προσαρμόζονται σε διαφορετικά μεγέθη συστημάτων, απαιτώντας ευέλικτες αρχιτεκτονικές και τυποποιημένα interfaces
- **Συνεχής συντήρηση:** Η ταχύρρυθμη εξέλιξη της τεχνολογίας, απαιτεί συνεχή έρευνα και προσαρμογή στις νέες τάσεις για την εξασφάλιση της συμβατότητας και μακροζωίας του συστήματος

2.4 Προκλήσεις της ενσωμάτωσης του IoT στα συστήματα παραγωγής

Τα βιομηχανικά περιβάλλοντα παραγωγής αποτελούν μια ξεχωριστή πρόκληση για την εφαρμογή του IoT λόγω της εγγενούς πολυπλοκότητάς τους. Τα δυνητικά οφέλη από την απόκτηση πληροφοριών σε πραγματικό χρόνο και τη βελτιστοποίηση των διαδικασιών, συχνά έρχονται αντιμέτωπα με προκλήσεις που προκύπτουν από την ενσωμάτωση συσκευών IoT σε ένα τοπίο που χαρακτηρίζεται από ποικίλο εξοπλισμό, παλαιά συστήματα και την ανάγκη για ισχυρή διαλειτουργικότητα ώστε να είναι δυνατή η ανταλλαγή δεδομένων μεταξύ συσκευών.

2.4.1 Ποικιλία εξοπλισμού και αισθητήρων

Οι βιομηχανικοί χώροι παραγωγής διαθέτουν συχνά ένα πολύπλοκο συνδυασμό εξοπλισμού και μηχανημάτων από διάφορους κατασκευαστές, που ενδεχομένως προέρχονται από διαφορετικές γενιές. Αυτή η ετερογένεια αποτελεί σημαντικό εμπόδιο για την ενσωμάτωση του IoT:

- **Ιδιόκτητα πρωτόκολλα και μορφές δεδομένων:** Οι συσκευές ενδέχεται να επικοινωνούν χρησιμοποιώντας πρωτόκολλα τα οποία είναι εξειδικευμένα από τον προμηθευτή (vendor-specific) και δεν έχουν σχεδιαστεί με σκοπό τη διαλειτουργικότητα. Επιπλέον, τα δεδομένα που παράγουν μπορεί να είναι δομημένα σε ασύμβατες μορφές [10]. Αυτό καθιστά δύσκολη, αν όχι αδύνατη, τη «συνεννόηση» των διαφορετικών συσκευών μεταξύ τους

- **Διαφορετικοί κύκλοι ζωής:** Ο εξοπλισμός σε ένα χώρο παραγωγής μπορεί να έχει εγκατασταθεί με διαφορά ετών ή και δεκαετιών. Τα παλαιότερα μηχανήματα μπορεί να μην διαθέτουν τις διεπαφές δικτύου ή τις υπολογιστικές δυνατότητες για να ενσωματωθούν άμεσα σε ένα σύστημα IoT [14]
- **Προκλήσεις προσαρμογής:** Η εκ των υστέρων τοποθέτηση ή προσθήκη αισθητήρων σε παλαιότερο εξοπλισμό μπορεί να είναι δαπανηρή και πολύπλοκη. Μπορεί να χρειαστούν εξειδικευμένες (custom) λύσεις, αυξάνοντας τόσο την οικονομική όσο και τη χρονική δαπάνη για την ενσωμάτωση του IoT
- **Πολυπλοκότητα της ενσωμάτωσης δεδομένων:** Ακόμα και αν η συνδεσιμότητα έχει δημιουργηθεί, η κατανόηση των διαφορετικών ροών δεδομένων που προέρχονται από διαφορετικές συσκευές απαιτεί εξελιγμένες τεχνικές κανονικοποίησης και μετάφρασης δεδομένων. Αυτό προσθέτει άλλο ένα επίπεδο πολυπλοκότητας στην εξαγωγή πολύτιμων πληροφοριών μεταξύ των συσκευών

2.4.2 Ενσωμάτωση παλαιών συστημάτων

Οι βιομηχανικές εγκαταστάσεις παραγωγής συχνά βασίζονται σε συστήματα παλαιάς τεχνολογίας - εξοπλισμό, λογισμικό και μηχανήματα που μπορεί να λειτουργούν εδώ και χρόνια ή ακόμη και δεκαετίες. Αυτά τα συστήματα αποτελούν ιδιαίτερη πρόκληση κατά την ενσωμάτωση νέων συσκευών και τεχνολογιών IoT:

- **Περιορισμένη συνδεσιμότητα:** Συχνά τα παλαιά συστήματα σχεδιάστηκαν πριν από τις σύγχρονες έννοιες της διασύνδεσης. Μπορεί να μην διαθέτουν ενσωματωμένες συνδέσεις Ethernet, σύγχρονες ασύρματες δυνατότητες ή τυποποιημένες διεπαφές επικοινωνίας [5]
- **Ζητήματα συμβατότητας:** Ακόμη και αν μπορεί να δημιουργηθεί διασυνδεσιμότητα, τα παλαιά συστήματα ενδέχεται να χρησιμοποιούν ξεπερασμένα πρωτόκολλα ή μορφές δεδομένων που δεν είναι συμβατές με τις σύγχρονες συσκευές και πλατφόρμες IoT [22]
- **Εξάρτηση σε προμηθευτή:** Ορισμένα παλαιά συστήματα σχεδιάστηκαν με ιδιόκτητες τεχνολογίες, καθιστώντας δύσκολη ή δαπανηρή τη διασύνδεση με λύσεις εκτός του οικοσυστήματος του αρχικού προμηθευτή
- **Λειτουργικοί κίνδυνοι:** Ο εκσυγχρονισμός ή η αλλοίωση κρίσιμων παλαιών συστημάτων ενέχει τον κίνδυνο να διαταραχθούν οι διαδικασίες παραγωγής που βασίζονται στη σταθερότητά τους
- **Επενδυτικοί προβληματισμοί:** Ανάλογα με την ηλικία και την πολυπλοκότητα ενός παλαιού συστήματος, η πλήρης αντικατάσταση μπορεί να είναι απαγορευτικά δαπανηρή [5]. Η εξεύρεση οικονομικά αποδοτικών τρόπων για να μπορέσουν τα παλαιά συστήματα να συνυπάρξουν με την υποδομή IoT είναι ζωτικής σημασίας

2.4.3 Διαλειτουργικότητα και ανταλλαγή δεδομένων

Ακόμα και αν είναι δυνατή η σύνδεση διαφορετικών τύπων εξοπλισμού και παλαιών συστημάτων, η ουσιαστική αξία από την ανάπτυξη του βιομηχανικού IoT προϋποθέτει ότι οι συσκευές μπορούν να «συνομιλούν» απρόσκοπτα μεταξύ τους και ότι τα δεδομένα τους μπορούν να συνδυαστούν με νόημα. Σε αυτό το σημείο προκύπτουν προκλήσεις όσον αφορά τη διαλειτουργικότητα και την ανταλλαγή δεδομένων:

- **Έλλειψη τυποποίησης:** Το πλήθος των πρωτοκόλλων επικοινωνίας και των μορφοτύπων δεδομένων που χρησιμοποιούνται από διαφορετικές συσκευές και προμηθευτές IoT λειτουργεί ως εμπόδιο της απρόσκοπτης επικοινωνίας. Χωρίς συμφωνημένα πρότυπα, οι συσκευές ενδέχεται να μην καταλαβαίνουν η μία την άλλη, εμποδίζοντας τη ροή πληροφοριών [8]
- **Σημασιολογικές αναντιστοιχίες:** Ακόμα και αν οι συσκευές μπορούν τεχνικά να ανταλλάσσουν δεδομένα, η σωστή ερμηνεία αυτών των δεδομένων είναι μια άλλη πρόκληση. Διαφορετικές συσκευές ενδέχεται να χρησιμοποιούν διαφορετική ορολογία ή μονάδες μέτρησης για την ίδια έννοια (π.χ. θερμοκρασία). Η επίλυση αυτών των σημασιολογικών αναντιστοιχιών είναι ζωτικής σημασίας για την αποτελεσματική χρήση των δεδομένων
- **Προβλήματα ποιότητας δεδομένων:** Η ακρίβεια και η αξιοπιστία των δεδομένων που προέρχονται από συσκευές IoT μπορεί να ποικίλλει. Η διασφάλιση της ποιότητας των δεδομένων είναι απαραίτητη για αξιόπιστες πληροφορίες και τη λήψη αποφάσεων. Αυτό θα μπορούσε να περιλαμβάνει την αντιμετώπιση της εγκυρότητας των αισθητήρων, τη βαθμονόμηση και τον χειρισμό ακραίων δεδομένων
- **Περιεχομενοποίηση δεδομένων:** Για την αξιοποίηση δεδομένων από πολλαπλές συσκευές για μεγαλύτερες γνώσεις, είναι σημαντική η κατανόηση του πλαισίου των δεδομένων (πού παράγονται, τι αντιπροσωπεύουν). Η προσθήκη μεταδεδομένων με βάση το πλαίσιο στις ροές δεδομένων IoT είναι απαραίτητη για να καταστούν πραγματικά αξιοποιήσιμα [26]

2.5 Q-Learning και παρόμοιες μελέτες

Στο κεφάλαιο αυτό παρουσιάζονται υπάρχουσες μελέτες σχετικά με την ενισχυτική μάθηση, με ιδιαίτερη έμφαση στο Q-Learning, μια τεχνική ενισχυτικής μάθησης που αποτελεί τη βάση για την εκπαίδευση του ρομποτικού συστήματος που αναπτύχθηκε στα πλαίσια της υλοποίησης της παρούσας ΔΕ και περιγράφεται λεπτομερώς στο κεφάλαιο 4.

Αρχικά ας πούμε λίγα πράγματα για το Q-Learning:

- **Τύπος μάθησης:** Είναι ένα είδος μηχανικής μάθησης (machine learning) και πιο συγκεκριμένα ενισχυτικής μάθησης (reinforcement learning). Είναι model-free το οποίο

σημαίνει ότι δεν είναι απαραίτητη η ύπαρξη ενός προκατασκευασμένου μοντέλου για το πώς λειτουργεί το περιβάλλον. Μαθαίνει απευθείας από τις αλληλεπιδράσεις

- Ο παράγοντας "Q": Το γράμμα "Q" είναι από την αγγλική λέξη quality δηλαδή ποιότητα. Στο Q-Learning, ένας πράκτορας (agent) μαθαίνει ένα Q-value για κάθε πιθανό συνδυασμό κατάστασης (state) και δράσης (action)
- Πώς μαθαίνει:
 1. Περιβάλλον: Ο πράκτορας αλληλεπιδρά με ένα περιβάλλον (π.χ. ένα παιχνίδι, μια προσωπικότητα)
 2. Κατάσταση: Ο πράκτορας παρατηρεί την τρέχουσα κατάστασή του
 3. Δράση: Επιλέγει μια ενέργεια με βάση την τρέχουσα πολιτική του
 4. Ανταμοιβή: Το περιβάλλον παρέχει μια ανταμοιβή (θετική ή αρνητική) για την ενέργεια
 5. Ενημέρωση: Ο πράκτορας ενημερώνει το Q-value του για το ζεύγος κατάστασης-δράσης χρησιμοποιώντας την ανταμοιβή και πληροφορίες για την επόμενη κατάσταση
 6. Επανάληψη: Η διαδικασία συνεχίζεται, με τον πράκτορα να μαθαίνει σταδιακά την καλύτερη ενέργεια που πρέπει να λάβει σε κάθε κατάσταση για να μεγιστοποιήσει τη συνολική ανταμοιβή του

Βασικά σημεία:

- Off-Policy: Το Q-Learning μπορεί να μάθει τον βέλτιστο τρόπο συμπεριφοράς ακόμα και αν οι ενέργειές του κατά τη διάρκεια της διαδικασίας εκμάθησης είναι διερευνητικές και όχι πάντα οι βέλτιστες
- Εξίσωση Bellman: Η βασική ενημέρωση του Q-value βασίζεται στην εξίσωση Bellman, μια θεμελιώδη αρχή στον τομέα της ενισχυτικής μάθησης

2.5.1 Εξερεύνηση Boltzmann (Boltzmann Exploration)

Η εξερεύνηση Boltzmann, εμπνευσμένη από την κατανομή Boltzmann στη στατιστική μηχανική, αποτελεί θεμελιώδη λίθο της έρευνας γύρω από το Q-Learning για την εξισορρόπηση της εξερεύνησης και της εκμετάλλευσης σε χώρους συνεχούς δράσης. Αποδίδει πιθανότητες σε δράσεις με βάση τα εκτιμώμενα Q-values, επιτρέποντας ένα βαθύ εξερεύνησης, ενώ παράλληλα ευνοεί ενέργειες με υψηλότερες πιθανές ανταμοιβές.

Αξιοσημείωτες Έρευνες

- Οι Groot Kormelink κ.α.[9], εξέτασαν μεθόδους εξερεύνησης σχετικά με το Q-Learning στο παιχνίδι Bomberman, διαπιστώνοντας ότι η εξερεύνηση Max-Boltzmann υπερτερεί

έναντι άλλων στρατηγικών λόγω της ισορροπημένης προσέγγισής της μεταξύ εξερεύνησης και εκμετάλλευσης. Η μελέτη έδειξε ότι η προσαρμοστικότητα της εξερεύνησης Boltzmann, μέσω μιας παραμέτρου θερμοκρασίας, επέτρεψε την αποτελεσματική εκμάθηση με τη σταδιακή μείωση της εξερεύνησης με την πάροδο του χρόνου. Τα ευρήματά τους υποδηλώνουν ότι η εξερεύνηση Boltzmann μπορεί να αποτελέσει πολύτιμο εργαλείο για το Q-Learning σε πολύπλοκα περιβάλλοντα με μεγάλους χώρους καταστάσεων ή όταν χρησιμοποιούνται νευρωνικά δίκτυα για την προσέγγιση συναρτήσεων

- Οι Gu κ.α.[11], εισήγαγαν τις κανονικοποιημένες συναρτήσεις πλεονεκτήματος (Normalized Advantage Functions - NAF), έναν αλγόριθμο βαθιάς ενισχυτικής μάθησης (DRL) που έχει σχεδιαστεί ειδικά για εργασίες συνεχούς ελέγχου. Σε αυτό το πλαίσιο, η εξερεύνηση Boltzmann έπαιξε καθοριστικό ρόλο στην επιλογή δράσης. Ο αλγόριθμος δειγματολήπτησε δράσεις από μια Γκαουσιανή κατανομή, της οποίας οι παράμετροι καθορίζονταν από το δίκτυο πρακτόρων (actors). Η πιθανότητα επιλογής μιας δράσης ήταν ανάλογη με το εκτιμώμενο πλεονέκτημά της, όπως υπολογίστηκε από το δίκτυο κριτών, κλιμακωμένο με μια παράμετρο θερμοκρασίας. Αυτή η προσέγγιση επέτρεψε στον ρομποτικό βραχίονα να εξερευνήσει ένα ευρύ φάσμα πιθανών ενεργειών, ενώ σταδιακά επικεντρωνόταν σε εκείνες που απέφεραν υψηλότερες ανταμοιβές. Πειράματα σε προσομοιωμένα περιβάλλοντα έδειξαν ότι ο συνδυασμός NAF με εξερεύνηση Boltzmann υπερείχε έναντι άλλων αλγορίθμων συνεχούς ελέγχου όσον αφορά την ταχύτητα εκμάθησης και την τελική απόδοση σε εργασίες όπως η επίτευξη θέσεων στόχων και ο χειρισμός αντικειμένων

2.5.2 Δυναμικά κατώφλια ανταμοιβής και προσαρμοστική μάθηση στο Q-Learning

Στα πλαίσια του Q-Learning, ένα κατώτατο όριο ανταμοιβής (κατώφλι) χρησιμεύει ως σημείο αναφοράς για την αξιολόγηση της απόδοσης του πράκτορα και την καθοδήγηση της διαδικασίας μάθησής του. Παραδοσιακά, οι Q-learning αλγόριθμοι χρησιμοποιούν σταθερά κατώφλια ανταμοιβής, όπου η αναμενόμενη ανταμοιβή για μια δεδομένη ενέργεια παραμένει σταθερή καθ' όλη τη διάρκεια της διαδικασίας εκμάθησης. Ωστόσο, μια πιο προσαρμοστική προσέγγιση, που περιλαμβάνει δυναμικά κατώφλια ανταμοιβής, έχει αναδειχθεί ως μια πολλά υποσχόμενη τεχνική για την ενίσχυση της αποδοτικότητας και της απόδοσης τέτοιων αλγορίθμων.

Οφέλη των δυναμικών κατωφλίων ανταμοιβής στο Q-Learning

- **Επιτάχυνση της μάθησης:** Τα δυναμικά κατώφλια δημιουργούν ένα «πρόγραμμα εκπαίδευσης» αυξανόμενης δυσκολίας, ξεκινώντας με ένα σχετικά επιεικές κατώτατο όριο και αυξάνοντάς το σταδιακά καθώς ο πράκτορας μαθαίνει. Αυτό επιτρέπει στον πράκτορα να επικεντρωθεί στην εκμάθηση ευκολότερων πτυχών μιας εργασίας προτού αντιμετωπίσει πιο δύσκολες, οδηγώντας σε ταχύτερη σύγκλιση σε βέλτιστες πολιτικές

- **Βελτιωμένη ισορροπία εξερεύνησης-εκμετάλλευσης:** Τα δυναμικά κατώφλια μπορούν να ενσωματωθούν με στρατηγικές εξερεύνησης όπως η εξερεύνηση Boltzmann για τη λεπτομερή ρύθμιση της ισορροπίας μεταξύ εξερεύνησης (δοκιμή νέων δράσεων) και εκμετάλλευσης (επιλογή των καλύτερων γνωστών δράσεων). Καθώς η απόδοση του πράκτορα βελτιώνεται, το κατώφλι μπορεί να προσαρμοστεί ώστε να ενισχαρρύνει την καλύτερη εκμετάλλευση των επιτυχημένων ενεργειών, επιτρέποντας παράλληλα την εξερεύνηση δυνητικά καλύτερων επιλογών
- **Ενισχυμένη ευρωστία:** Σε πολύπλοκα περιβάλλοντα, τα σταθερά κατώφλια ανταμοιβής ενδέχεται να μην είναι κατάλληλα για όλα τα στάδια της μάθησης. Τα δυναμικά κατώφλια μπορούν να προσαρμόζονται στην πρόοδο του πράκτορα, καθιστώντας τον αλγόριθμο πιο ανθεκτικό στις μεταβολές της δυσκολίας της εργασίας και στις αλλαγές του περιβάλλοντος

Σχέση με Curriculum Learning και Self-Paced Learning

- **Curriculum Learning:** Όπως ακριβώς ένα πρόγραμμα σπουδών παρουσιάζει σε έναν μαθητή όλο και πιο απαιτητική ύλη, τα δυναμικά όρια καθοδηγούν τον πράκτορα ενός Q-Learning σεναρίου μέσα από μια ακολουθία εργασιών με σταδιακά αυξανόμενη δυσκολία. Αυτή η προσέγγιση διευκολύνει τη μάθηση με το χτίσιμο απλούστερων δεξιοτήτων πριν από την αντιμετώπιση πιο σύνθετων
- **Self-Paced Learning:** Στο Self-Paced Learning, ο πράκτορας καθορίζει το ρυθμό της μάθησής του με βάση την απόδοσή του. Ομοίως, τα δυναμικά όρια ανταμοιβής προσαρμόζονται με βάση την πρόοδο του πράκτορα, επιτρέποντάς του να μαθαίνει με τον δικό του ρυθμό

Σχετικές Έρευνες

- Οι Kumar κ.α.[13], εισάγουν μια νέα προσέγγιση για τη βελτίωση της αποτελεσματικότητας της εκμάθησης μοντέλων λανθάνουσας μεταβλητής. Οι παραδοσιακοί αλγόριθμοι μάθησης συχνά κολλάνε σε μη βέλτιστες λύσεις επειδή εξετάζουν όλα τα δείγματα ταυτόχρονα. Ο προτεινόμενος αλγόριθμος μάθησης SPL, μιμείται την ανθρώπινη διαδικασία μάθησης ξεκινώντας με ευκολότερα δείγματα και ενσωματώνοντας σταδιακά όλο και πιο σύνθετα. Αυτή η επαναληπτική προσέγγιση επιλέγει εύκολα δείγματα σε κάθε επανάληψη και ενημερώνει τις παραμέτρους του μοντέλου, συμπεριλαμβάνοντας σταδιακά περισσότερα δείγματα καθώς η εκμάθηση προχωρά. Τα εμπειρικά αποτελέσματα δείχνουν ότι το SPL υπερτερεί των παραδοσιακών μεθόδων σε διάφορες εργασίες, όπως ο εντοπισμός αντικειμένων και η αναγνώριση χειρόγραφων ψηφίων, επιτυγχάνοντας καλύτερη σύγκλιση και απόδοση
- Οι Zhang κ.α.[25], παρουσιάζουν μια νέα προσέγγιση που βελτιώνει το Q-Learning για το πρόβλημα του πλανόδιου πωλητή Traveling Salesman Problem (TSP). Συνδυάζει το beam search με το Q-Learning, αξιοποιώντας τις υποδιαδρομές για τη διάσπαση

του TSP σε μικρότερα τμήματα. Ο αλγόριθμος δημιουργεί δυναμικά πίνακες ανταμοιβής κατά τη διάρκεια της εκμάθησης, παρέχοντας πιο σχετική ανατροφοδότηση και βελτιώνοντας την αποτελεσματικότητα της μάθησης. Επίσης ρυθμίζει προσαρμοστικά την επιλογή ενεργειών με βάση την τρέχουσα κατάσταση και τις εμπειρίες που μαθαίνει, εξισορροπώντας αποτελεσματικότερα την εξερεύνηση και την εκμετάλλευση. Με την ενσωμάτωση του beam search ο αλγόριθμος διερευνά συστηματικά πολλαπλά υποσχόμενα μονοπάτια ταυτόχρονα, αποτρέποντας την πρόωρη σύγκλιση σε μη βέλτιστες λύσεις. Τα εμπειρικά αποτελέσματα δείχνουν σημαντικές βελτιώσεις στην εύρεση βέλτιστων διαδρομών γρήγορα και αξιόπιστα σε σύγχριση με τις παραδοσιακές μεθόδους Q-Learning. Η προσέγγιση αυτή παρουσιάζει μια πολλά υποσχόμενη κατεύθυνση για μελλοντική έρευνα στην προσαρμοστική μάθηση και βελτιστοποίηση.

Σε αυτό το κεφάλαιο θέσαμε τις βάσεις για την κατανόηση του Q-Learning, ενός ισχυρού εργαλείου στον τομέα της ενισχυτικής μάθησης. Εξετάσαμε τις θεμελιώδεις αρχές του, εμβαθύναμε στις διαφοροποιημένες στρατηγικές της εξερεύνησης Boltzmann και εξετάσαμε τα πλεονεκτήματα των δυναμικών κατωφλίων ανταμοιβής και της προσαρμοστικής μάθησης σε αυτό το πλαίσιο. Αυτές οι έννοιες έχουν συμβάλει καθοριστικά στη διαμόρφωση δύο Q-Learning σεναρίων που χρησιμοπιήθηκαν για την εκπαίδευση ενός ρομποτικού συστήματος και θα αναλυθούν λεπτομερώς στο Κεφάλαιο 4.

Κεφάλαιο 3

Επισκόπηση Τεχνολογιών

Σε αυτό το κεφάλαιο, θα εμβαθύνουμε στις τεχνολογίες που χρησιμοποιήθηκαν στο κομμάτι της υλοποίησης αυτής της ΔΕ, την AutomationML και το ThingsBoard. Η AutomationML είναι μια ουδέτερη μορφή δεδομένων βασισμένη στην XML (Extensible Markup Language) για την αποθήκευση και την ανταλλαγή δεδομένων εργοστασιακών μηχανικών εγκαταστάσεων. Το ThingsBoard, από την άλλη πλευρά, είναι μια IoT πλατφόρμα ανοιχτού κώδικα που επιτρέπει τη σύνδεση και τη συλλογή-επεξεργασία-οπτικοποίηση δεδομένων για ένα ευρύ φάσμα συσκευών και IoT εφαρμογών. Καθώς περιηγούμαστε στο πολύπλοκο πεδίο του βιομηχανικού αυτοματισμού, η διασφάλιση της απρόσκοπτης επικοινωνίας και της ανταλλαγής δεδομένων μεταξύ των διαφόρων κλάδων ενός εργοστασίου αποτελεί σημαντική πρόκληση. Αυτό το κεφάλαιο παρουσιάζει τον τρόπο με τον οποίο αυτές οι καινοτόμες τεχνολογίες αντιμετωπίζουν αυτό το ζήτημα, ανοίγοντας το δρόμο για μια ευέλικτη και αποτελεσματική εφαρμογή.

3.1 AutomationML

Όπως αναφέρθηκε και παραπάνω, η AutomationML (AML) είναι μια περιεκτική αντικειμενοστραφής γλώσσα μοντελοποίησης δεδομένων βασισμένη στην XML. Είναι ανοιχτού κώδικα και παρέχει ένα ολοκληρωμένο σύνολο ευέλικτων μηχανισμών και καινοτομιών για τη μοντελοποίηση των σημερινών και των μελλοντικών πτυχών της μηχανικής [2].

3.1.1 Σκοπός

Ο σκοπός δημιουργίας της AML μπορεί να συμπυχθεί σε μία φράση, στη γεφύρωση του χάσματος στον βιομηχανικό αυτοματισμό. Φανταστείτε μια ορχήστρα όπου κάθε όργανο λειτουργεί απομονωμένο, χωρίς να μπορεί να εναρμονιστεί. Αυτό, δυστυχώς, αντικατοπτρίζει συχνά την πραγματικότητα του βιομηχανικού αυτοματισμού, όπου ξεχωριστοί κλάδοι της μηχανικής, όπως ο είναι ο μηχανολογικός, ο ηλεκτρολογικός και ο τομέας του λογισμικού, χρησιμοποιούν ασύμβατα μεταξύ τους εργαλεία και μορφές δεδομένων. Αυτό είναι το πρόβλημα που έρχεται να αντιμετωπίσει η AML, η οποία έχει σχεδιαστεί με σκοπό την «ενορχήστρωση» μιας συμφωνίας ροής πληροφοριών, βελτιώνοντας τη διαλειτουργικότητα και την αποδοτικότητα.

Ας το σκεφτούμε ως εξής:

1. Διαφορετικές γλώσσες: Κάθε κλάδος της μηχανικής μιλάει κατά κανόνα τη δική του «γλώσσα», μέσω ιδιόκτητου λογισμικού και μορφές δεδομένων
2. Προβλήματα μετάφρασης: Η μη αυτόματη μετάφραση πληροφοριών μεταξύ αυτών των γλωσσών είναι χρονοβόρα και επιρρεπής σε σφάλματα
3. AutomationML ως λύση: Παρέχοντας μια ουδέτερη και τυποποιημένη μορφή, η AML λειτουργεί ως μια κοινή γλώσσα που γίνεται κατανοητή από διάφορα εργαλεία και είδη λογισμικών, ανεξαρτήτου κλάδου, εξαλείφοντας την ανάγκη μετάφρασης

Τα πλεονεκτήματα αυτής της ενιαίας γλώσσας είναι σημαντικά:

- **Ταχύτερη ολοκλήρωση των πρότζεκτ:** Η απρόσκοπη ανταλλαγή δεδομένων εξαλείφει το χρόνο που δαπανάται για μετατροπές και επανεπεξεργασία, με αποτέλεσμα την ταχύτερη διεκπεραίωση των εργασιών
- **Μείωση σφαλμάτων:** Μία γλώσσα σημαίνει μία μορφή δεδομένων οπότε ελαχιστοποιούνται πιθανά σφάλματα μετατροπών και πιθανές ασυμβατότητες. Ως εκ τούτου, βελτιώνεται η συνολική αξιοπιστία του συστήματος
- **Βελτίωση συνεργασίας:** Ομάδες από διαφορετικούς κλάδους μπορούν εύκολα να μοιράζονται πληροφορίες και να συνεργάζονται αποτελεσματικότερα
- **Απλοποίηση διαχείρησης του κύκλου ζωής του εργοστασίου:** Η διαχείριση της πληροφορίας καθ' όλη τη διάρκεια του κύκλου ζωής ενός εργοστασίου, από το σχεδιασμό έως τη λειτουργία και τη συντήρηση, γίνεται πιο αποτελεσματική και εξορθολογισμένη.

Άρα, η AML, συνεχίζοντας την προηγούμενη παρομοίωση, δρα σαν το μαέστρο που επιτυγχάνει την ενορχήστρωση μιας αρμονικής ροής δεδομένων στο βιομηχανικό αυτοματισμό. Εξαλείφει τα εμπόδια στην επικοινωνία, επιτρέποντας την αποτελεσματική συνεργασία και ξεκλειδώνει μια νέα εποχή αυτοματισμού στο χώρο του Industry 4.0.

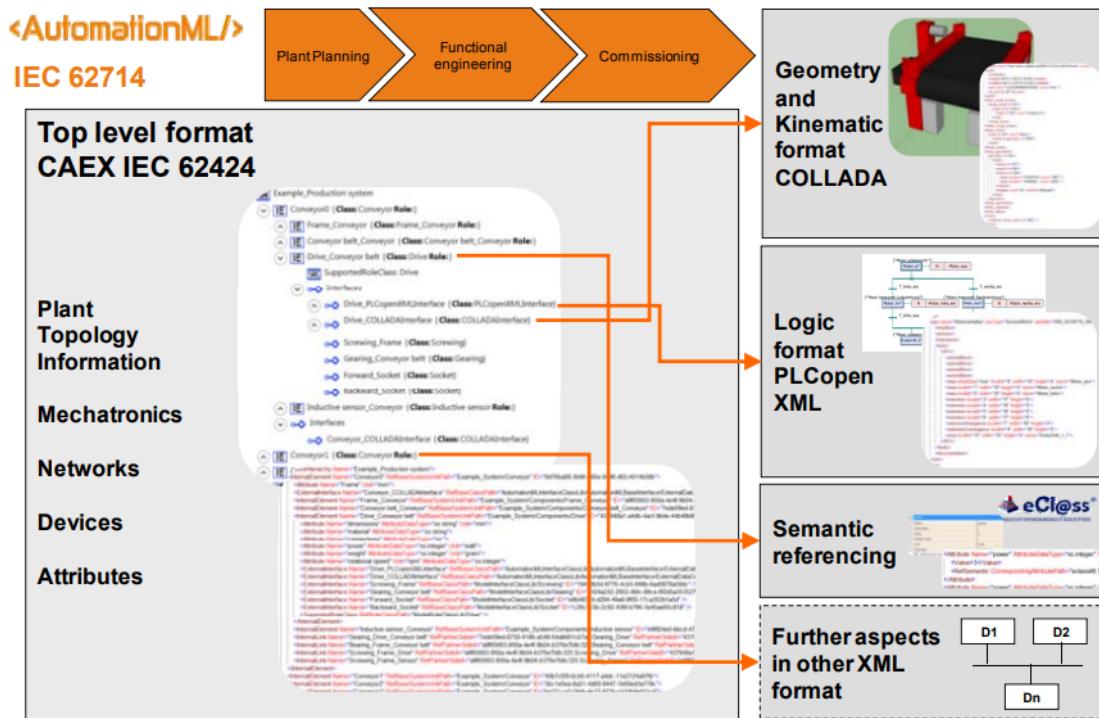
3.1.2 Γενική αρχιτεκτονική

Ακολουθώντας μια αντικειμενοστραφή προσέγγιση, η AML μοντελοποιεί τα φυσικά και λογικά στοιχεία των εγκαταστάσεων ως αντικείμενα δεδομένων με διάφορα χαρακτηριστικά. Αυτά τα αντικείμενα μπορούν να σχηματίζουν ιεραρχίες, όπου μικρότερα αντικείμενα συνθέτουν μεγαλύτερα, μεγαλύτερα αντικείμενα συνθέτουν ακόμη μεγαλύτερα κ.ο.κ. Κάθε αντικείμενο κατέχει διάφορες ιδιότητες, συμπεριλαμβανομένων της γεωμετρίας του, της κινηματικής του, της λογικής του και άλλων σχετικών πληροφοριών [16].

Με την ενσωμάτωση και την προσαρμογή υφιστάμενων XML μορφών, η AML δημιουργεί μια αρθρωτή (modular) δομή με μια κεντρική «μορφή ανώτατου επιπέδου» (top level format).

Λογικά η AML χωρίζεται σε [16]:

- Περιγραφή της τοπολογίας του κατασκευαστικού στοιχείου και των πληροφοριών δικτύωσης, συμπεριλαμβανομένων των ιδιοτήτων των αντικειμένων που εκφράζονται ως ιεραρχία AML αντικειμένων AML και περιγράφονται μέσω CAEX σύμφωνα με το IEC 62424
- Περιγραφή της γεωμετρίας και της κινηματικής των διαφόρων AML αντικειμένων που αντιπροσωπεύονται από το COLLADA 1.4.1 και 1.5.0
- Περιγραφή των δεδομένων λογικής που σχετίζονται με τον έλεγχο των διαφόρων AML αντικειμένων που αντιπροσωπεύονται από το PLCopen XML 2.0 και 2.0.1
- Περιγραφή των σχέσεων μεταξύ των AML αντικειμένων και των αναφορών σε πληροφορίες που είναι αποθηκευμένες σε έγγραφα εκτός της «μορφής ανώτατου επιπέδου» με τη χρήση CAEX



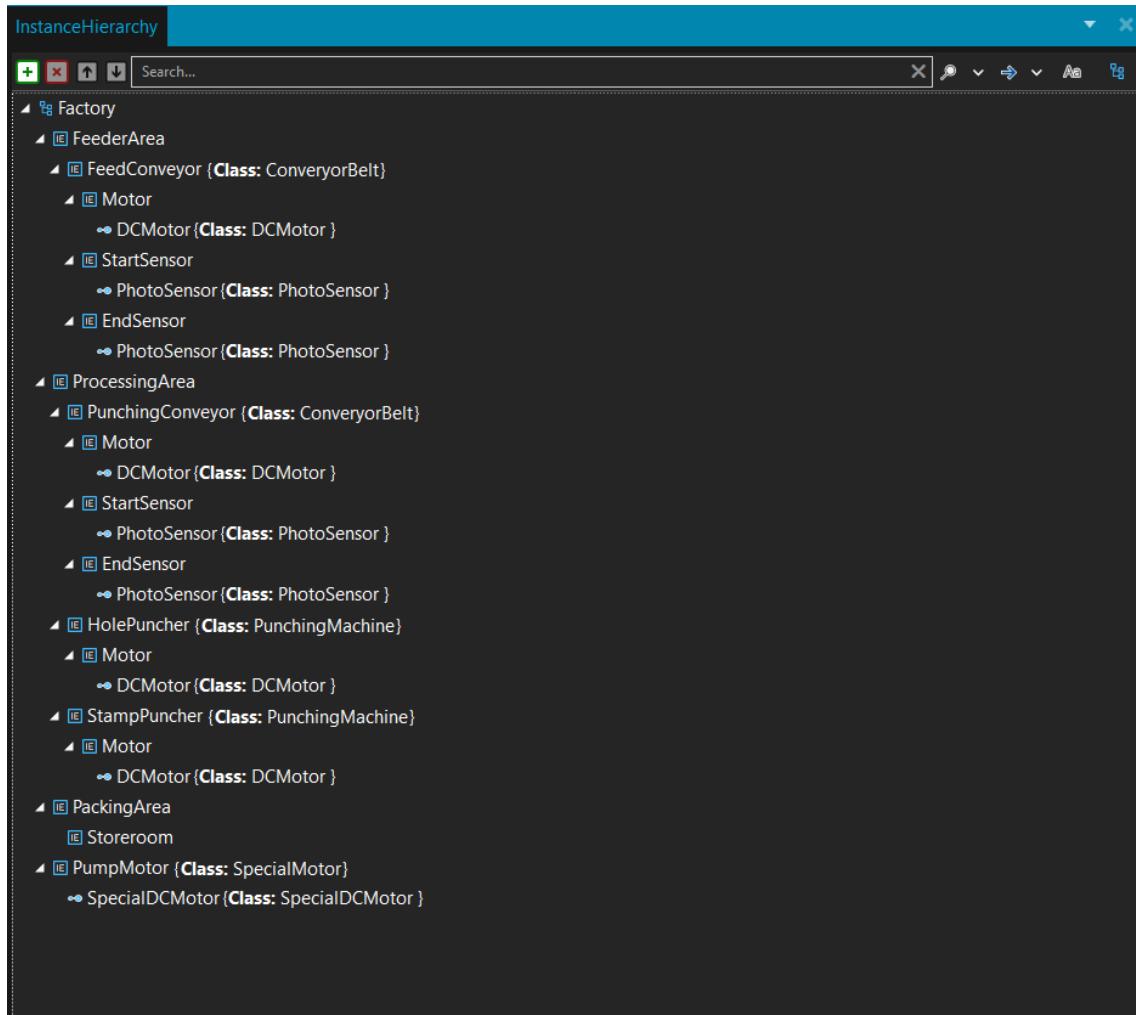
Σχήμα 3.1: Δομή AutomationML project [16]

Όπως αναφέρθηκε και προηγουμένως, η AML αξιοποιεί το CAEX για τη μοντελοποίηση της τοπολογίας και το στοιχείων του συστήματος. Ως εκ τούτου, παρέχει τέσσερα κύρια μέσα μοντελοποίησης:

1. Instance Hierarchy

Είναι το σημαντικότερο μέσο μοντελοποίησης. Ορίζει την τοπολογία του χώρου παραγωγής μέσω μιας ενσωματωμένης ιεραρχίας που αποτελείται από InternalElements και αντιπροσωπεύει τα ουσιαστικά μηχανολογικά χαρακτηριστικά που πρέπει να μοντελοποιηθούν

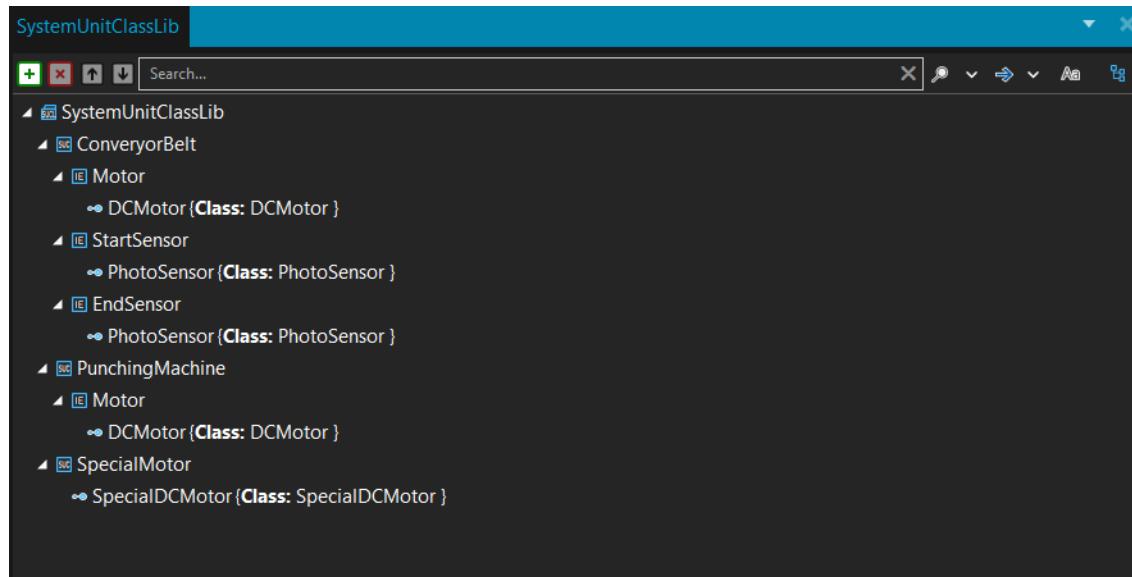
από το CAEX. Ανάλογα το βαθμό αφαιρετικότητας (abstraction), μπορεί να αναπαριστά από ολόκληρα φυσικά στοιχεία (π.χ. χώρος εμφιάλωσης), μέχρι λειτουργικά στοιχεία (π.χ. μηχανές) ακόμα και απλά εξαρτήματα (π.χ. καλώδιο, βίδα).



Σχήμα 3.2: Παράδειγμα Instance Hierarchy

2. System Unit Class

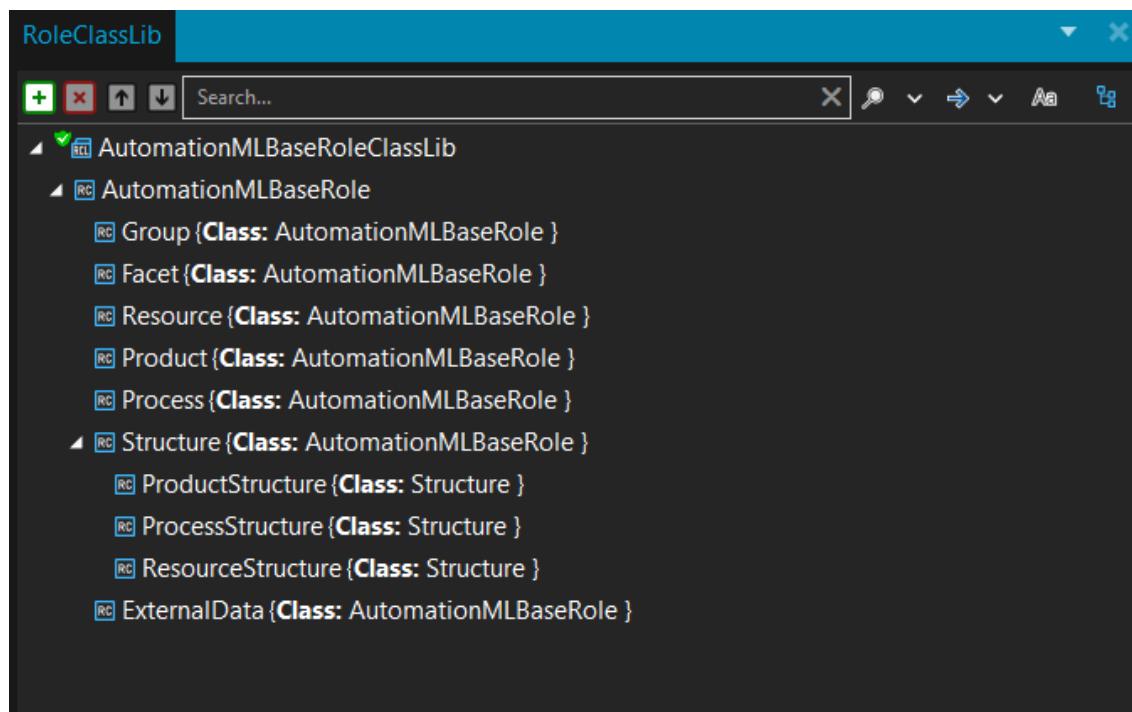
Παρέχουν έναν τρόπο για τον ορισμό της δομής και της σημασιολογίας των αντικειμένων του συστήματος με συνεπή και επαναχρησιμοποιήσιμο τρόπο. Για παράδειγμα, αν έχουμε ένα σύνολο αντικειμένων που αντιπροσωπεύουν διαφορετικούς τύπους αισθητήρων σε ένα εργοστάσιο παραγωγής, θα μπορούσατε να ορίσουμε ένα System Unit Class (SUC) για τους αισθητήρες που περιλαμβάνει χαρακτηριστικά όπως εύρος μέτρησης, ανάλυση και χρόνος απόκρισης. Στη συνέχεια, κάθε μεμονωμένο αντικείμενο αισθητήρα στο Instance Hierarchy θα αναφέρεται σε αυτό το SUC, κληρονομώντας έτσι αυτά τα κοινά χαρακτηριστικά και εξασφαλίζοντας ότι όλοι οι αισθητήρες περιγράφονται με ομοιόμορφο τρόπο.



Σχήμα 3.3: Παράδειγμα System Unit Class

3. Role Class

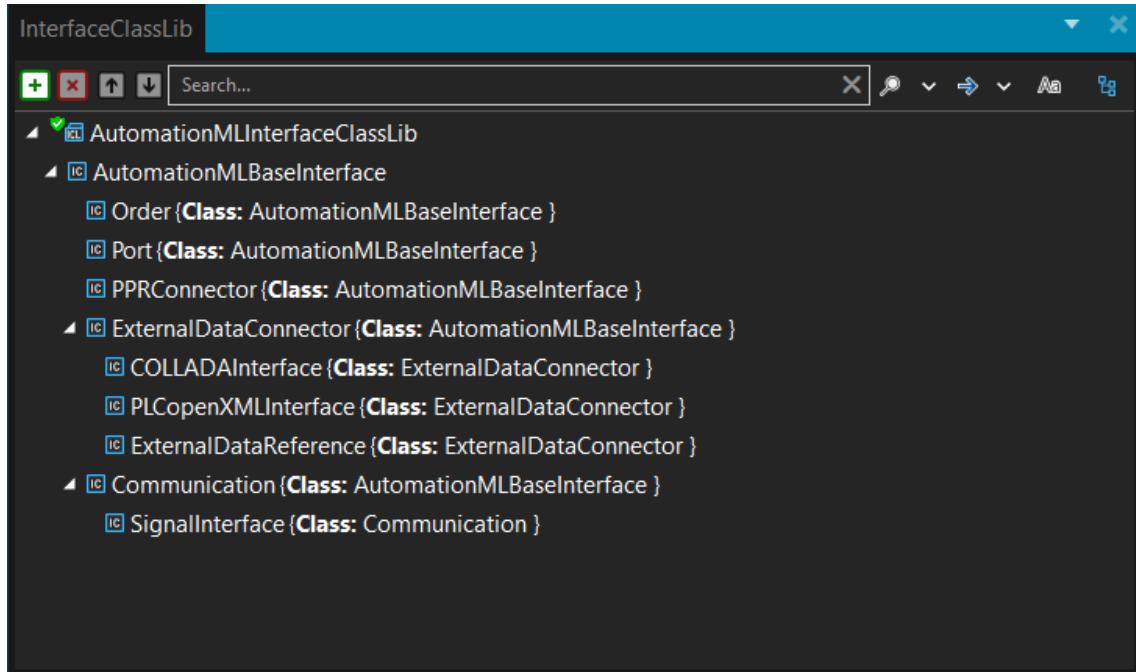
Περιγράφει μια αφηρημένη λειτουργικότητα χωρίς να ορίζει την υποχείμενη τεχνική υλοποίηση, επομένως, πρέπει να χρησιμοποιείται ως δείκτης για τη σημασιολογία ενός αντικειμένου. Για παράδειγμα θα μπορούσαμε να έχουμε μια κλάση με όνομα Εξάρτημα ή με το όνομα Συσκευή οι οποίες υποδεικνύουν τη σημασιολογία της δομής του συστήματος.



Σχήμα 3.4: AutomationML BaseRoleClass Library

4. Interface Class

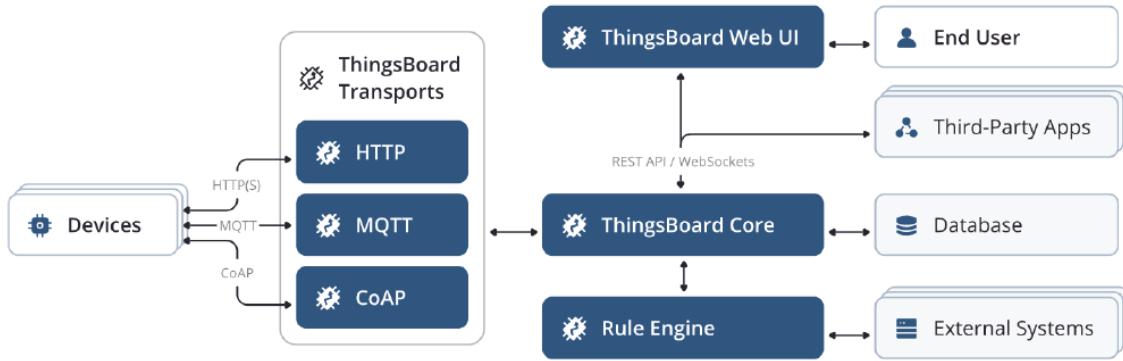
Περιγράφει μια αφηρημένη σχέση που μπορεί να έχει ένα στοιχείο με άλλα στοιχεία ή με πληροφορίες που δεν καλύπτονται από το μοντέλο CAEX. Αυτό θα μπορούσε να είναι, για παράδειγμα, ένα ExternalDataConnector που αντιπροσωπεύει τη συσχέτιση με εξωτερικά αποθηκευμένες πληροφορίες.



Σχήμα 3.5: AutomationML InterfaceClass Library

3.2 ThingsBoard

Όπως αναφέρθηκε και στην εισγωγή του κεφαλαίου, το ThingsBoard είναι μια IoT πλατφόρμα ανοικτού κώδικα που επιτρέπει την ταχεία ανάπτυξη, διαχείριση και κλιμάκωση IoT project [21]. Σε αυτή την υποενότητα θα παρουσιάσουμε τις βασικές δυνατότητες που προσφέρει η πλατφόρμα αυτή και θα αναλύσουμε με ποιον τρόπο μπορεί αξιοποιηθεί για την αποτελεσματική διαχείριση και ανάλυση IoT δεδομένων.



Σχήμα 3.6: Αρχιτεκτονική του ThingsBoard (Πηγή: <https://2021.desosa.nl/print/projects/thingsboard/posts/2.-architecture/>)

3.2.1 ThingsBoard Transports

Το ThingsBoard λειτουργεί ως κεντρικός κόμβος για διάφορες συσκευές και ροές δεδομένων, γεγονός που απαιτεί την υποστήριξη διαφόρων πρωτοκόλλων επικοινωνίας. Σε αυτή την εργασία όμως παρουσιάζουμε μόνο τα τρία κυριότερα, αλλά αξίζει να αναφερθεί ότι παρέχει υποστήριξη και άλλων.

MQTT

Το MQTT, στο οποίο έγινε αναφορά και στο προηγούμενο κεφάλαιο, είναι η κυρίαρχη επιλογή για την επικοινωνία συσκευών. Είναι ένα ελαφρύ πρωτόκολλο δημοσίευσης/συνδρομής, ιδανικό για συσκευές με περιορισμένους πόρους. Η ασύγχρονη φύση του και η αποτελεσματική παράδοση δεδομένων, το καθιστούν ιδανικό για την αποστολή τηλεμετρικών δεδομένων από αισθητήρες και ενεργοποιητές. Το ThingsBoard υποστηρίζει δύο επίπεδα QoS (0 και 1) του MQTT, εξασφαλίζοντας την αξιοπιστία των μηνυμάτων ανάλογα με τις ανάγκες της κάθε εφαρμογής.

CoAP (Constrained Application Protocol)

Το CoAP παρέχει μια αποτελεσματική εναλλακτική λύση αντί του MQTT, για συσκευές με ακόμη αυστηρότερους περιορισμούς. Η προσέγγιση αίτησης-απόκρισης που εφαρμόζει είναι κατάλληλη για περιορισμένα περιβάλλοντα, ενώ παράλληλα επιτρέπει την ανταλλαγή δεδομένων και την απομακρυσμένη διαχείριση. Το ThingsBoard χρησιμοποιεί το CoAP για συγκεκριμένα σενάρια που περιλαμβάνουν συσκευές με χαμηλή κατανάλωση ενέργειας.

HTTP (Hypertext Transfer Protocol)

Επιτρέπει σε συσκευές να δημοσιεύουν τηλεμετρικά δεδομένα, client-side attributes καθώς και να ζητούν attribute values από τον server χρησιμοποιώντας HTTP POST και GET requests. Το HTTP API βασίζεται σε ένα μοντέλο αίτησης-απόκρισης και μπορεί να χρησιμοποιηθεί τόσο σε πρωτόκολλα HTTP όσο και σε HTTPS. Τα API endpoints περιλαμβάνουν

διαδρομές για μεταφόρτωση τηλεμετρικών δεδομένων, ενημερώσεις attribute, RPC εντολές και device provisioning.

3.2.2 ThingsBoard Core

Το ThingsBoard Core αποτελεί το κεντρικό νευρικό σύστημα της πλατφόρμας, υπεύθυνο για τη διαχείριση των συσκευών, την ασφαλή επεξεργασία των δεδομένων και τη διευκόλυνση της επικοινωνίας. Στη συνέχεια θα παρουσιάσουμε τις σημαντικότερες λειτουργίες του.

Διαχείριση Συσκευών

- Καταχωρεί και διαχειρίζεται τις ποικίλες συσκευές που είναι συνδεδεμένες στο ThingsBoard
- Παρέχει λειτουργίες όπως προσθήκη/αφαίρεση συσκευών, ρύθμιση/ανάκτηση attribute και παρακολούθηση της υγείας των συσκευών
- Λειτουργεί ως κεντρικός κόμβος επικοινωνίας των συσκευών, διαχείρισης των ροών δεδομένων και εκτέλεσης των εντολών

Επεξεργασία Δεδομένων

- Επεξεργάζεται δεδομένα πραγματικού χρόνου και ιστορικά δεδομένα που λαμβάνονται από συσκευές
- Χρησιμοποιεί βάσεις δεδομένων από χρονοσειρές για αποτελεσματική αποθήκευση και ανάκτηση δεδομένων
- Επιτρέπει την αποτελεσματική διαχείριση δεδομένων για επιπλέον ανάλυση, οπτικοποίηση και ενέργειες από το rule engine

Ασφάλεια

- Επιβάλλει ισχυρά μέτρα ασφαλείας μέσω ελέγχου ταυτότητας χρήστη, εξουσιοδότησης και κρυπτογράφησης των δεδομένων
- Διαχειρίζεται τον έλεγχο πρόσβασης για διαφορετικούς ρόλους χρηστών και διασφαλίζει την ασφαλή επικοινωνία μεταξύ των δομοστοιχείων (modules)
- Παρέχει εργαλεία για τη διαχείριση API κλειδιών και πιστοποιητικών, διασφαλίζοντας την ακεραιότητα και την ιδιωτικότητα των δεδομένων

Επικοινωνία

- Διαχειρίζεται την επικοινωνία των συνδεδεμένων συσκευών χάνοντας χρήση των πρωτοκόλλων που αναφέρουν νωρίτερα

- Επιτρέπει ευέλικτες γέφυρες πρωτοκόλλων για ενσωμάτωση με ιδιόκτητα πρωτόκολλα ή custom εφαρμογές

Επεκτασιμότητα και Ανοχή Σφαλμάτων

- Είναι σχεδιασμένο με αρχιτεκτονική μικρουπηρεσιών για οριζόντια κλιμάκωση, η οποία επιτρέπει την προσθήκη περισσότερων κόμβων με σκοπό τη διαχείρηση του αυξανόμενου όγκου δεδομένων
- Παρέχει ανοχή σε σφάλματα μέσω κατανευμημένων στοιχείων, εξασφαλίζοντας υψηλή διαθεσιμότητα

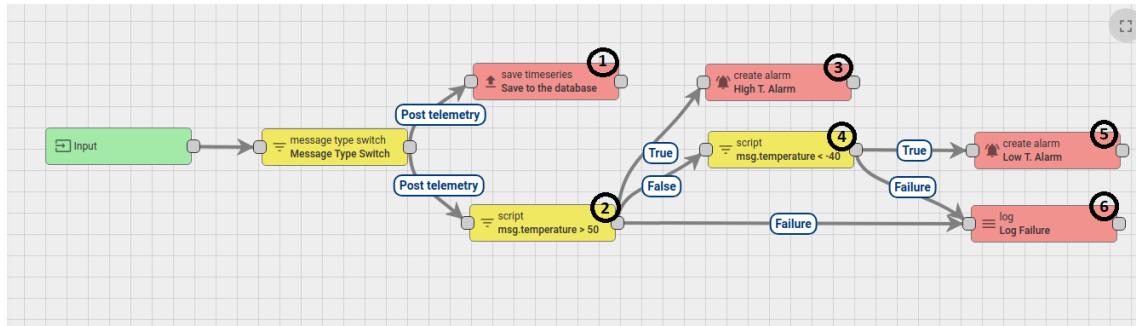
3.2.3 ThingsBoard Rule Engine

To Rule Engine είναι ένα παραμετροποιήσιμο σύστημα το οποίο είναι υπεύθυνο για τη διαχείρηση και επεξεργασία διαφόρων συμβάντων. Αποτελείται από τρία κύρια στοιχεία:

1. **Rule Engine Message:** Αντιπροσωπεύει οποιοδήποτε εισερχόμενο συμβάν, όπως δεδομένα από συσκευές, συμβάντα κύκλου ζωής οντοτήτων (entity life-cycle events), REST API events ή RPC requests
2. **Rule Node:** Είναι βασικό στοιχείο του Rule Engine που επεξεργάζεται ένα μόνο εισερχόμενο μήνυμα κάθε φορά και παράγει ένα ή περισσότερα εξερχόμενα μηνύματα. Μπορεί να φιλτράρει, να εμπλουτίσει, να μετασχηματίσει τα εισερχόμενα μηνύματα, να εκτελεί ενέργειες ή να επικοινωνεί με εξωτερικά συστήματα. Τα διάφορα Nodes μπορούν να συνδέονται μεταξύ τους με σχέσεις
3. **Rule Chain:** Είναι το αποτέλεσμα πολλών συνδεδεμένων Nodes μεταξύ τους. Περνάει τα εξερχόμενα μηνύματα από το ένα Node στο άλλο

Στο Σχήμα 3.7 βλέπουμε παράδειγμα ενός Rule Chain. Η λειτουργία του είναι η εξής:

- Θα αποθηκεύσει όλα τα τηλεμετρικά δεδομένα στη βάση δεδομένων (Node νούμερο 1)
- Θα ενεργοποιήσει το alarm υψηλής θερμοκρασίας αν το πεδίο θερμοκρασίας του μηνύματος είναι μεγαλύτερο από το 50 (Nodes νούμερο 2 και 3)
- Θα ενεργοποιήσει το alarm χαμηλής θερμοκρασίας αν το πεδίο θερμοκρασίας του μηνύματος είναι μικρότερο από το -40 (Nodes νούμερο 4 και 5)
- Θα καταγράψει αποτυχία εκτέλεσης των script ελέγχου θερμοκρασίας στην κονσόλα σε περίπτωση λογικού ή συντακτικού σφάλματος στο script (Node νούμερο 6)



Σχήμα 3.7: Παράδειγμα ενός Rule Chain (Πηγή: <https://thingsboard.io/docs/user-guide/rule-engine-2-0/overview/>)

3.2.4 ThingsBoard WEB UI

To Web UI του ThingsBoard είναι το γραφικό περιβάλλον με το οποίο αλληλεπιδρούν οι χρήστες για να διαχειρίζονται και να παρακολουθούν τις συνδεδεμένες συσκευές, τα δεδομένα και τις εφαρμογές τους. Όπως φαίνεται και από το side-bar στο Σχήμα 3.8, το ThingsBoard παρέχει πολλές δυνατότητες και πληροφορίες για τη διαχείρηση IoT εφαρμογών αλλά στη παρούσα ΔΕ θα μας απασχολήσουν χυρίως τα παρακάτω:

The screenshot shows the ThingsBoard Tenant Administrator WEB UI. The left sidebar contains navigation links for Home, Alarms, Dashboards, Entities, Devices, Assets, Entity Views, Profiles, Device profiles, Asset profiles, Customers, Rule chains, Edge management, Advanced features, Resources, and Notification center. The main dashboard area includes sections for Devices (Inactive: 10, Active: 0, Total: 10), Alarms (Critical: 0), Dashboards (Last viewed: Ball Lifter, Firmware, Rule Engine Statistics, Thermostats, Gateways, Software), Activity (History last 30 days), Quick links (Alarms, Dashboards, Devices), Documentation (Getting started, Rule engine, API, Device profiles), and Usage (Entities: Devices 10, Assets 9, Users 5, Dashboards 6, Customers 3, API calls: 10, 9, 5, 6, 3). A 'Get started' section provides a numbered guide for device creation, connection, dashboard creation, rule configuration, alarm creation, and customer assignment.

Σχήμα 3.8: ThingsBoard Tenant Administrator WEB UI

Devices

Τα Devices είναι φυσικές ή εικονικές οντότητες που αλληλεπιδρούν με την πλατφόρμα για την αποστολή και τη λήψη IoT δεδομένων. Κάθε Device έχει ένα μοναδικό Device ID και πιστοποιείται με τη χρήση ενός Access Token, τα οποία παράγονται κατά τη δημιουργία του. Επίσης κάθε Device έχει μια κατηγορία με το όνομα Attributes. Υπάρχουν τριών ειδών Attributes:

- **Client Attributes:** Αποθηκεύουν δεδομένα που κοινοποιούνται απευθείας από μια συσκευή. Πρόκειται ουσιαστικά για τον τρόπο με τον οποίο η συσκευή επικοινωνεί την τρέχουσα κατάστασή της ή τις ενδείξεις των αισθητήρων της στον server του ThingsBoard. Δεν επιτρέπεται η τροποποίηση τους από το WEB UI
- **Server Attributes:** Ξρησιμοποιούνται για την αποθήκευση δεδομένων σχετικά με μια συσκευή την οποία διαχειρίζεται ο server ThingsBoard. Αυτές οι πληροφορίες συνήθως δεν παράγονται απευθείας από τη συσκευή. Επιτρέπεται η τροποποίηση τους από το WEB UI
- **Shared Attributes:** Παρέχουν ένα αμφίδρομο κανάλι επικοινωνίας μεταξύ της συσκευής και του ThingsBoard server. Προσφέρουν έναν τρόπο και στις δύο πλευρές να βλέπουν και ενδεχομένως να τροποποιούν τα ίδια χαρακτηριστικά. Επιτρέπεται η τροποποίηση τους από το WEB UI

Assets

Τα Assets αντιπροσωπεύουν φυσικές ή λογικές οντότητες σε ένα IoT οικοσύστημα, όπως μηχανήματα, οχήματα, κτίρια κ.α. Κάθε Asset έχει ένα μοναδικό Asset ID το οποίο παράγεται κατά τη δημιουργία του και μπορεί να σχετίζεται με άλλα Assets ή Devices.

Device & Asset Profiles

Τα Device Profiles χρησιμοποιούνται για τη δημιουργία κοινών ρυθμίσεων για πολλαπλές συσκευές. Κάθε συσκευή διαθέτει ένα και μόνο ένα προφίλ. Επίσης έχουν τη δυνατότητα να καθορίσουν σε custom Rule Chains, την επεξεργασία εισερχόμενων μηνυμάτων που αφορούν ένα ή πολλά Device Profiles.

Κατά τον ίδιο τρόπο, τα Asset Profiles χρησιμοποιούνται για τη διαχείριση κοινών ρυθμίσεων πολλαπλών Asset. Κάθε Asset έχει επίσης ένα και μόνο ένα προφίλ. Επιπλέον, τα Asset Profiles επιτρέπουν την επιλογή Rule Chain και Queue για την επεξεργασία διαφόρων δεδομένων ενός Asset. Αυτή η λειτουργία είναι χρήσιμη για το χειρισμό διαφορετικών τύπων Asset και για το διαχωρισμό της επεξεργασίας δεδομένων διαφορετικών Asset, κάτι που μπορεί να είναι σημαντικό για την καλύτερη απόδοση του συστήματος και της διαχείρισης των δεδομένων.

Dashboards

Τα Dashboards είναι παραμετροποιήσιμα interfaces που παρέχουν πληροφορίες σε πραγματικό χρόνο για τις συσκευές και τα δεδομένα IoT. Αποτελούνται από widgets που απεικονίζουν δεδομένα από πολλές πηγές και μπορούν να περιλαμβάνουν διαδραστικά εργαλεία όπως κουμπιά και διακόπτες. Η πρόσβαση στα Dashboard εξαρτάται από το ρόλο του εκάστοτε χρήστη, επιτρέποντας διαφορετικά επίπεδα προνομίων, διασφαλίζοντας την ασφάλεια των δεδομένων. Κάθε Dashboard μπορεί να διαμορφωθεί μέσω διαφόρων ρυθμίσεων, οι οποίες ελέγχουν τη διάταξη των widgets και τη συνολική εμφάνιση του. Επίσης διαθέτουν λειτουργίες όπως είναι η εξαγωγή, η επεξεργασία ή η διαγραφή ενώ παράλληλα παρέχουν τη δυνατότητα ανάθεσής τους σε συγκεκριμένο πελάτη για τον καλύτερο έλεγχο πρόσβασης.

Device	Current SW title	Current SW version	Target SW title	Target SW version	Target SW set time	Progress	Status
Ball Lifter							
EndSensor_1							
EndSensor_2							
Motor_1							
Motor_2							
Motor_3							
Motor_4							
PumpMotor_1							
StartSensor_1							
StartSensor_2							

Σχήμα 3.9: Παράδειγμα Dashboard με διάφορα widgets που παρέχουν πληροφορίες σχετικά με τα υπάρχοντα Devices

Κεφάλαιο 4

Περιγραφή Γλοποίησης

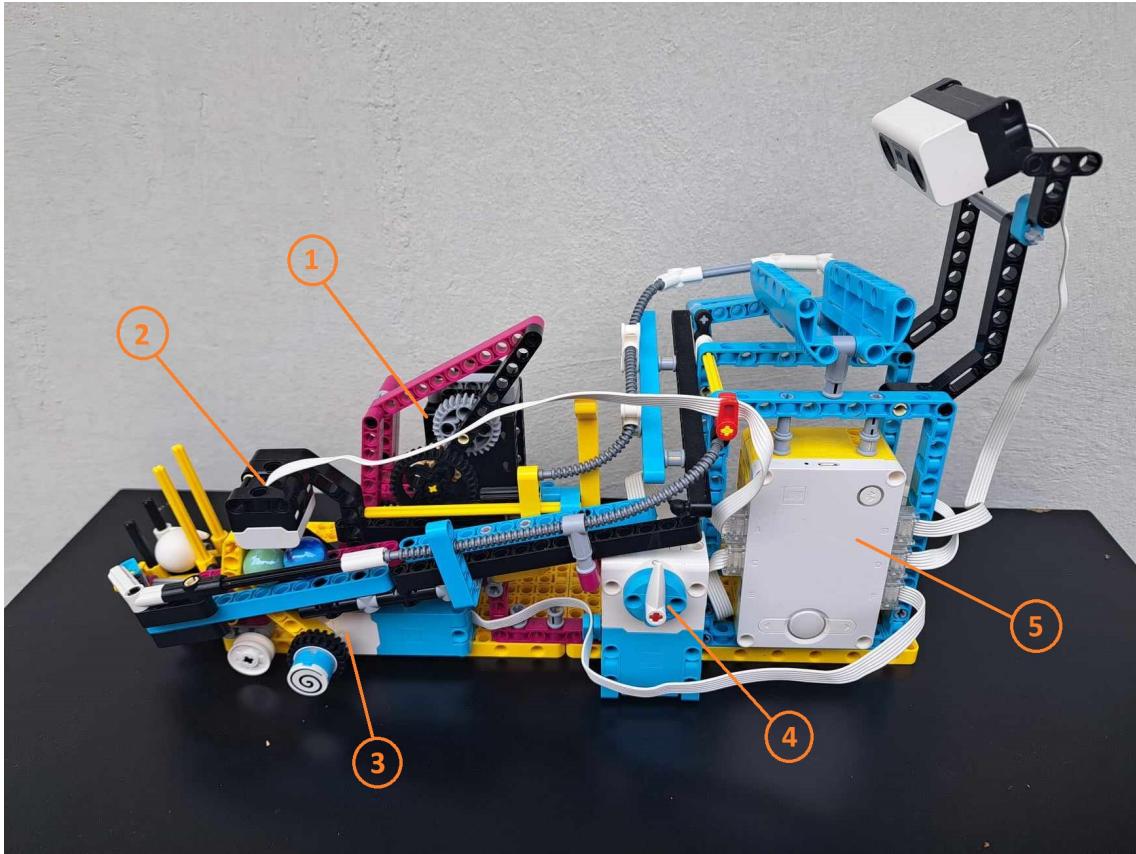
Στην ενότητα αυτή θα περιγράψουμε το σύστημα που υλοποιήθηκε στα πλαίσια αυτής της ΔΕ, παρουσιάζοντας τεχνικές λεπτομέρειες και αναλύοντας την πρακτική εφαρμογή του. Περιλαμβάνει μια ρομποτική εφαρμογή η οποία μοντελοποιήθηκε κατάλληλα μέσω AutomationML και ανταλλάσσει δεδομένα επιτυχώς, επιτυγχάνοντας μια αυτόνομη επικοινωνία, με την πλατφόρμα ThingsBoard. Προχωράμε πέρα από τη θεωρητική βάση που παρουσιάσαμε νωρίτερα και δείχνουμε με ποιον τρόπο το προτεινόμενο σύστημα έχει εφαρμογή σε ένα πραγματικό περιβάλλον.

4.1 LEGO SPIKE Prime

4.1.1 Περιγραφή

To LEGO SPIKE Prime είναι ένα κιτ σχεδιασμένο για μαθητές ηλικίας 10+ που ακολουθεί την εκπαιδευτική προσέγγιση STEAM (Science, Technology, Engineering, Arts and Mathematics) το οποίο προωθεί τη δημιουργικότητα καθώς επίσης εισάγει τους μαθητές στον κόσμο των θετικών επιστημών. Εμείς προφανώς, στα πλαίσια της ΔΕ, αξιοποιήσαμε τις προγραμματιστικές του δυνατότητες για να αναπτύξουμε ένα σύστημα που υπερβαίνει τις τυπικές εκπαιδευτικές εφαρμογές του και προσομοιώνει μια IoT εφαρμογή.

Το μοντέλο που δημιουργήθηκε είναι αυτό που φαίνεται στην παραχώτω εικόνα.



Σχήμα 4.1: LEGO SPIKE Prime BallLifter Robot

Ο σχεδιασμός είναι του Dimitri Dekyvere και οδηγίες συναρμολόγησης μπορούν να βρεθούν στο [3].

Στο Σχήμα 4.1 έχουν αριθμηθεί 5 σημεία. Στο υπόλοιπο της ΔΕ θα αναφερόμαστε σε αυτά τα σημεία ως εξής:

1. **ballLifterMotor:** Πρώτο εκ των δύο μικρών motor που υπάρχουν στο κιτ. Είναι υπεύθυνο για την κίνηση του βραχίονα (δε φαίνεται από αυτή τη γωνία γιατί είναι τοποθετημένο όπως το motor νούμερο 3 αλλά από την πίσω πλευρά)
2. **colorSensor:** Αισθητήρας χρώματος που αναγνωρίζει το χρώμα της μπάλας που περνάει από κάτω του
3. **ballStopperMotor:** Το δεύτερο μικρό motor. Σκοπός του είναι να σταματάει τις μπάλες όταν σταματάει το πρόγραμμα
4. **largeMotor:** Το μεγάλο motor που υπάρχει στο κιτ. Λειτουργεί σαν counter υπολειπόμενων μπαλών
5. **SPIKE:** Ο εγκέφαλος του ρομπότ. Τροφοδοτεί τις υπόλοιπες συσκευές και εκτελεί

τα προγράμματα. Έχει ένα light-matrix που λειτουργεί σαν οθόνη και 4 κουμπιά : Bluetooth button, Hub button, Left-Right button.

4.1.2 Περιβάλλον προγραμματισμού

Ο προγραμματισμός του SPIKE έγινε χρησιμοποιώντας το Pybricks, μια πλατφόρμα προγραμματισμού Python για έξυπνα LEGO Hubs. Το Pybricks δίνει τη δυνατότητα εκτέλεσης MicroPython κώδικα απευθείας στο Hub, δίνοντας πρόσβαση σε πολλές βιβλιοθήκες, επεκτείνοντας έτσι τις δυνατότητες του LEGO.

4.1.3 Επεξήγηση βασικής λειτουργίας

Ο πλήρης κώδικας του BallLifter Robot βρίσκεται στο Παράρτημα A'. Η βασική λειτουργία του BallLifter, όπως φαίνεται και από το όνομα του, είναι:

1. Ανύψωση μπάλας από βραχίονα
2. Ρίψη μπάλας στο υψηλότερο σημείο της κατηφορικής ράμπας
3. Επιστροφή βραχίονα στην αρχική θέση
4. Επανάληψη

Για την κίνηση των motors έχουν υλοποιηθεί δύο συναρτήσεις:

```
def setMotorAngleShortestPath(motor, target_angle, speed)
```

και

```
def setMotorAngleLongestPath(motor, target_angle, speed)
```

οι οποίες περιστρέφουν το motor στην επιθυμητή γωνία επιλέγοντας τη μικρότερη ή τη μεγαλύτερη περιστροφική διαδρομή αντίστοιχα. Η επιλογή ανάμεσα στην μικρότερη ή την μεγαλύτερη διαδρομή που πρέπει να ακολουθήσει το motor είναι αναγκαία για την σωστή κίνηση του βραχίονα.

Επίσης έχουμε τη συνάρτηση

```
def waitForColor(desired_colors)
```

η οποία δέχεται έναν πίνακα από επιθυμητά χρώματα σαν όρισμα. Αν το χρώμα που θα αναγνωρίσει το colorSensor είναι ένα από τα επιθυμητά χρώματα, τότε το επιστρέφει, αλλίως περιμένουμε.

Τέλος, έχουμε τις συναρτήσεις

```
def setLargeMotor(numOfBalls)
```

και

```
def moveLargeMotorForInterval(remainingBalls)
```

Η setLargeMotor δέχεται σαν όρισμα έναν αριθμό υπολοιπόμενων μπαλών και σετάρει το largeMotor. Λειτουργεί σαν ωρολογιακός δείκτης αναλογικού ρολογιού δηλαδή άμα ο αριθμός υπολοιπόμενων μπαλών είναι 9, τότε ο largeMotor θα περιστραφεί 270 μοίρες (αρχίζει πάντα από τις 0 μοίρες όπως φαίνεται και στο Σχήμα 4.1).

Η moveLargeMotorForInterval επίσης δέχεται σαν όρισμα τον αριθμό υπολοιπόμενων μπαλών και μετακινεί το largeMotor για 30 μοίρες κάθε φορά. Ακολουθώντας τη λογική του αναλογικού ρολογιού δηλαδή, άμα για παράδειγμα ο αριθμός υπολοιπόμενων μπαλών μειωθεί από 9 στις 8, τότε το largeMotor θα μετακινηθεί στις 240 μοίρες.

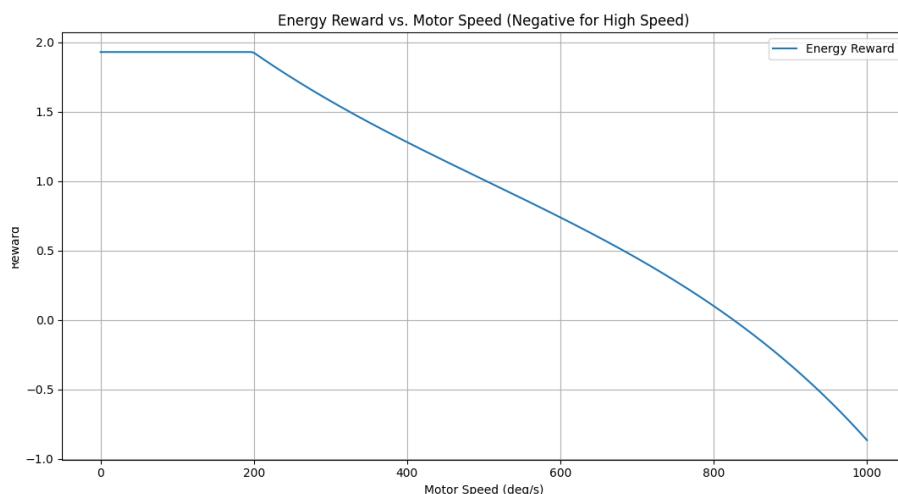
Οι υπόλοιπες συναρτήσεις θα αναλυθούν στη συνέχεια γιατί είναι μέρος των Q-Learning σεναρίων.

4.1.4 Ανάλυση Q-Learning σεναρίων

Τα δύο Q-Learning σενάρια που θα παρουσιάσουμε σε αυτή τη ΔΕ, εξυπηρετούν έναν κοινό σκοπό: τον προσδιορισμό της βέλτιστης ταχύτητας ανύψωσης του ballLifterMotor για μπάλες διαφορετικού βάρους.

Ο προσδιορισμός της βέλτιστης ταχύτητας ανύψωσης του ρομποτικού μας βραχίονα απαιτεί μια λεπτή ισορροπία μεταξύ ανταγωνιστικών παραγόντων:

- Ενεργειακή απόδοση:** Στόχος μας είναι να ελαχιστοποιήσουμε την ενέργεια που καταναλώνει το ballLifterMotor κατά τη διαδικασία ανύψωσης. Οι υψηλότερες ταχύτητες οδηγούν γενικά σε υψηλότερες ενεργειακές δαπάνες. Για να ποσοτικοποιήσουμε αυτή τη σχέση, αναπτύξαμε μια συνάρτηση, την calculateEnergyReward, η οποία προσομοιώνει την ενεργειακή δαπάνη με βάση την επιλεγμένη ταχύτητα και παρέχει μια αντίστοιχη ανταμοιβή. Η ανταμοιβή παραμένει υψηλή για χαμηλότερες ταχύτητες, μειώνεται σταδιακά καθώς αυξάνεται η ταχύτητα και γίνεται αρνητική γύρω στην ταχύτητα 800, υποδεικνύοντας αναποτελεσματική χρήση ενέργειας. Στο Σχήμα 4.2, φαίνεται η γραφική παράσταση αυτής της συνάρτησης



Σχήμα 4.2: Energy Reward vs Motor Speed

2. Αποδοτικότητα χρόνου: Προσπαθούμε να ελαχιστοποιήσουμε το χρόνο που απαιτείται για την ανύψωση της μπάλας. Οι μεγαλύτερες ταχύτητες έχουν φυσικά ως αποτέλεσμα μικρότερους χρόνους ανύψωσης. Για να ποσοτικοποιήσουμε την ανταμοιβή που σχετίζεται με το χρόνο, χρησιμοποιούμε τη συνάρτηση calculateTimeReward, η οποία λαμβάνει ως είσοδο το συνολικό χρόνο ανύψωσης και επιστρέφει μια ανταμοιβή. Η ανταμοιβή είναι υψηλή για μικρότερους χρόνους εκτέλεσης. Καθώς ο χρόνος εκτέλεσης αυξάνεται, η ανταμοιβή μειώνεται μη γραμμικά, και γίνεται αρνητική μετά από 2200 χιλιοστά του δευτερολέπτου. Στο Σχήμα 4.3, φαίνεται η γραφική παράσταση αυτής της συνάρτησης



Σχήμα 4.3: Time Reward vs Execution Time

3. Ποσοστό επιτυχίας: Ο απώτερος στόχος είναι η επιτυχής ανύψωση και μεταφορά της μπάλας. Επομένως, η επιλεγμένη ταχύτητα πρέπει να εξασφαλίζει υψηλή πιθανότητα επιτυχίας

Η σχέση που ορίσαμε και συνδυάζει τους παραπάνω παράγοντες είναι η εξής:

```
combined_reward = success_reward * (2 * energy_reward + (3 * time_reward))
```

όπου το success_reward ορίζεται ως:

```
success_reward = 1 if success else -10
```

Για να προσδιορίσουμε τη βέλτιστη ταχύτητα ανύψωσης, αναζητούμε τη «χρυσή τομή» - το σημείο όπου μεγιστοποιείται η συνδυασμένη ανταμοιβή από την ενεργειακή αποδοτικότητα και τη χρονική αποδοτικότητα, εξασφαλίζοντας προφανώς την επιτυχή ανύψωση της μπάλας.

Χρησιμοποιώντας το Q-Learning, στοχεύουμε να βρούμε τη βέλτιστη ταχύτητα ανύψωσης για μπάλες διαφορετικής μάζας. Για το ζύγισμα των μπαλών χρησιμοποιήθηκε το HX711 Weight Sensor Kit και ένα Arduino UNO. Κάθε Q-learning σενάριο, ακολουθεί μια ξεχω-

ριστή προσέγγιση σε αυτό το πρόβλημα βελτιστοποίησης, χρησιμοποιώντας την εξερεύνηση Boltzmann, δυναμικά κατώφλια ανταμοιβής και την επιλογή σταθμισμένων δράσεων.

Και για τα δύο σενάρια, ως καταστάσεις (states), ψευδούνται οι διαφορετικές μάζες των μπαλών, ως δράσεις (actions), ψευδούνται οι ταχύτητες 50-1000 με βήμα 50 (50, 100, 150, ..., 1000) και για τον υπολογισμό των Q-value γίνεται χρήση της εξίσωσης του Bellman. Στη συνέχεια θα αναλυθούν αυτοί οι αλγόριθμοι και η αποτελεσματικότητά τους στην εύρεση της ιδανικής ταχύτητας ανύψωσης για κάθε διαφορετική μπάλα.

Σενάριο 1: Boltzmann Exploration with Epsilon-Greedy Q-Learning

Βασική ιδέα: Αυτό το σενάριο συνδυάζει την εξερεύνηση Boltzmann με μια στρατηγική Epsilon-Greedy για την επιλογή της ταχύτητας ballLifterMotor. Η Epsilon-Greedy είναι μια κοινή τεχνική στην ενισχυτική μάθηση όπου ο πράκτορας επιλέγει μια τυχαία δράση με πιθανότητα epsilon και την άπληστη δράση (αυτή με την υψηλότερη εκτιμώμενη τιμή) με πιθανότητα (1 - epsilon).

Πώς λειτουργεί:

- Απόφαση Epsilon-Greedy:** Η συνάρτηση αποφασίζει πρώτα αν θα εξερευνήσει ή θα εκμεταλλευτεί με βάση την τιμή του epsilon_in. Στην προκειμένη περίπτωση, κατά τη διαδικασία της εκπαίδευσης, η τιμή του epsilon_in ισούται με 0.9. Εάν ένας τυχαίος αριθμός μεταξύ 0 και 1 είναι μικρότερος από το epsilon_in, ο αλγόριθμος θα διερευνήσει επιλέγοντας μια τυχαία δράση. Διαφορετικά, θα εκμεταλλευτεί επιλέγοντας τη δράση με το μεγαλύτερο Q-value. Κατά τη διάρκεια του Q-Learning, η τιμή του epsilon μειώνεται σταδιακά μετά από κάθε επανάληψη. Αρχικά, όταν ο πράκτορας έχει περιορισμένη γνώση του περιβάλλοντος, ένα υψηλότερο epsilon δίνει προτεραιότητα στην εξερεύνηση, ενθαρρύνοντας τον πράκτορα να δοκιμάσει διάφορες ενέργειες και να συλλέξει πληροφορίες. Καθώς η εκπαίδευση προχωράει και ο πράκτορας αποκτά μεγαλύτερη εμπειρία, ένα χαμηλότερο epsilon δίνει προτεραιότητα στην εκμετάλλευση, επιτρέποντας στον πράκτορα να βασίζεται όλο και περισσότερο στις γνώσεις που έχει αποκτήσει για να λαμβάνει τις βέλτιστες αποφάσεις
- Διαθέσιμες δράσεις (Εξερεύνηση):** Εάν επιλεγεί η εξερεύνηση, η συνάρτηση προσδιορίζει όλες τις διαθέσιμες ταχύτητες (available_speeds). Αυτό γίνεται φιλτράροντας το Q-table, για δράσεις με Q-value μεγαλύτερο ή ίσο του -0.5. Αυτό γίνεται με το σκεπτικό, ότι δράσεις με Q-value μικρότερο του -0.5, είναι σίγουρα μη επιθυμητές οπότε δεν υπάρχει λόγος να ξαναδοκιμαστούν. Προφανώς το Q-table αρχικά είναι αρχικοποιημένο με την τιμή 0, για όλες τις δράσεις
- Βάρη Boltzmann (Εξερεύνηση):** Η συνάρτηση υπολογίζει τα βάρη Boltzmann για κάθε μία από τις διαθέσιμες ταχύτητες. Αυτά τα βάρη είναι ανάλογα με το εκθετικό των Q-value τους διαιρεμένο με την παράμετρο της θερμοκρασίας. Μια υψηλότερη θερμοκρασία οδηγεί σε πιο ομοιόμορφα βάρη (περισσότερη εξερεύνηση), ενώ μια χαμηλότερη θερμοκρασία ευνοεί τη δράση με την υψηλότερη τιμή X

4. **Σταθμισμένη τυχαία επιλογή (Εξερεύνηση):** Ένας τυχαίος αριθμός παράγεται εντός του εύρους του συνολικού αθροίσματος των βαρών Boltzmann. Η συνάρτηση εξτάζει τις διαθέσιμες ταχύτητες και τα αντίστοιχα βάρη τους. Εάν ο τυχαίος αριθμός εμπίπτει στο συνολικό άθροισμα των βαρών μέχρι μια συγκεκριμένη ταχύτητα, επιλέγεται αυτή η ταχύτητα ως δράση.
5. **Υπολογισμός ανταμοιβής:** Με βάση τις συναρτήσεις ανταμοιβής που περιγράψαμε νωρίτερα και της σχέσης που τις συνδέει, υπολογίζεται η συνολική ανταμοιβή για την επιλεγμένη ταχύτητα και ενημερώνεται το Q-value της

Η συνάρτηση παρουσιάζεται στη συνέχεια:

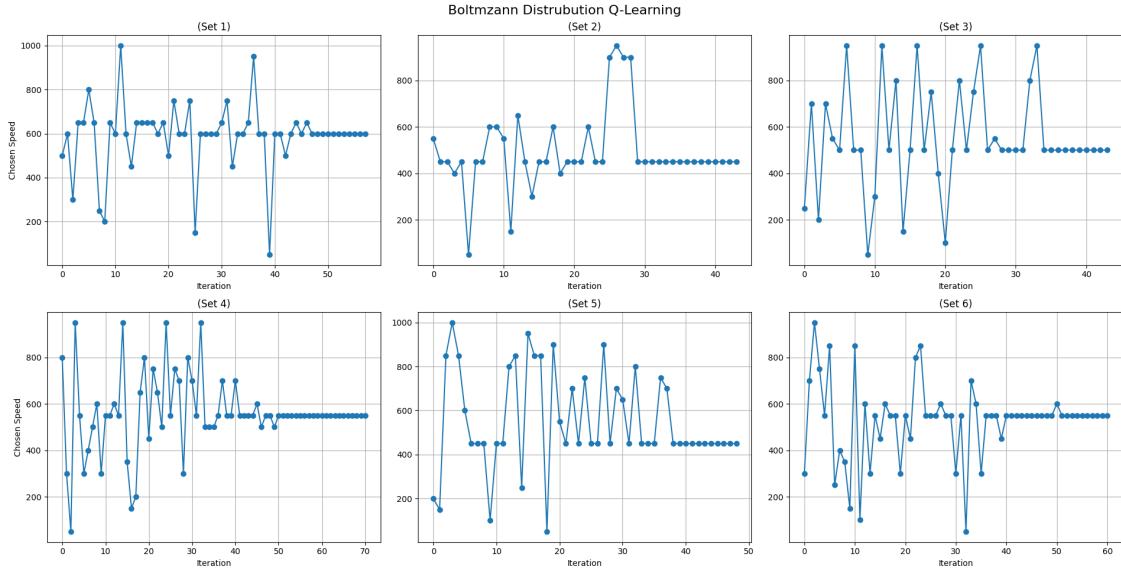
```
def choose_action(state, epsilon_in=epsilon, temperature=1.0):
    if epsilon_in > uniform(0, 1):
        available_speeds = [speed for speed in range(50, 1050, 50) if
qtable[state].get(speed, -1) >= -0.5]

        # Calculate Boltzmann weights
        q_values = [qtable[state][speed] for speed in available_speeds]
        exp_values = [exp(q / temperature) for q in q_values]
        total_weight = sum(exp_values)

        # Generate a random number within the total weight
        rand_num = uniform(0, total_weight)

        # Select the speed based on the random number and weights
        weight_sum = 0
        for i, speed in enumerate(available_speeds):
            weight_sum += exp_values[i]
            if rand_num <= weight_sum:
                return speed
        else:
            return max(qtable[state], key=qtable[state].get)
```

Στο Σχήμα 4.4, παρουσιάζονται τα αποτελέσματα από 6 διαφορετικές «διαδικασίες εκπαίδευσης» για μια μπάλα μάζας 4.4g. Αρχικά η επιλογή ταχυτήτων ποικίλει σε όλο το εύρος του πεδίου τιμών, υποδεικνύοντας ότι ο αλγόριθμος διερευνά διάφορες επιλογές. Καθώς προχωρούν οι επαναλήψεις, η επιλογή ταχύτητας τείνει να συγκλίνει σε μια συγκεκριμένη τιμή, υποδηλώνοντας ότι ο αλγόριθμος εκμεταλλεύεται τη γνώση που έχει μάθει για να επιλέξει την πιο αποδοτική ταχύτητα. Παρατηρούμε ότι κατά μέσο όρο ο αλγόριθμος συγκλίνει μετά από 40 επαναλήψεις σε ένα εύρος ταχυτήτων από 450 έως 600.



Σχήμα 4.4: Boltzmann Distribution Q-Learning

Σενάριο 2: Threshold-Guided Weighted Q-Learning

Βασική ιδέα: Αυτό το σενάριο επεκτείνει το προγούμενο, εισάγοντας δύο βασικές καινοτομίες που ενισχύουν την αποδοτικότητα του αλγορίθμου:

- Δυναμικό κατώφλι ανταμοιβής:** Αντί για μια σταθερή ανταμοιβή, χρησιμοποιείται ένα δυναμικό κατώφλι που προσαρμόζεται με βάση την απόδοση του πράκτορα. Αυτό δημιουργεί έναν «κινούμενο στόχο» που προτρέπει συνεχώς τον πράκτορα να βελτιώνεται, επιταχύνοντας τη μάθηση και ενθαρρύνοντας την αποτελεσματικότερη εξερεύνηση
- Σταθμισμένη επιλογή δράσεων:** Βασιζόμενη στην πιθανολογική προσέγγιση του Boltzmann, αυτό το σενάριο αναθέτει βάρη στις δράσεις με βάση το συνολικό άθροισμα όλων των ανταμοιβών τους. Αυτό επιτρέπει στον αλγόριθμο να εξετάζει όχι μόνο το άμεσο Q-value μιας δράσης, αλλά και τη μακροπρόθεσμη επιτυχία της. Οι δράσεις που οδηγούν σταθερά σε υψηλότερες ανταμοιβές λαμβάνουν μεγαλύτερη βαρύτητα, με αποτέλεσμα να είναι πιο πιθανό να επιλεγούν

Πως λειτουργεί:

1. Αρχικοποίηση:

- Αρχικοποιείται η μεταβλητή `available_speeds`, που ορίζει το εύρος των πιθανών τιμών που μπορεί να λάβει το `ballLifterMotor`
- Αρχικοποιείται ένα λεξικό με όνομα `overall_rewards_per_speed` για να αποθηκεύται η συνολική ανταμοιβή για κάθε ταχύτητα
- Αρχικοποιείται ένα λεξικό με όνομα `action_probabilities` όπου αρχικά δίνει ίση πιθανότητα σε κάθε ταχύτητα, προωθώντας την αρχική εξερεύνηση

- Ορίζεται ένα αρχικό κατώφλι ανταμοιβής (current_threshold)

2. Επιλογή δράσης:

- Η συνάρτηση choose_action εξάγει τις πιθανότητες από το action_probabilities που αντιστοιχούν στις τρέχουσες διαθέσιμες ταχύτητες
- Στη συνέχεια χρησιμοποιείται η συνάρτηση weighted_choice για να επιλαχεί μια ταχύτητα με βάση αυτές τις πιθανότητες. Αυτή η συνάρτηση υλοποιεί μια σταθμισμένη τυχαία επιλογή, όπου η πιθανότητα επιλογής μιας ταχύτητας είναι ανάλογη του βάρους της

3. Υπολογισμός ανταμοιβής:

- Με το ίδιο σκεπτικό όπως προηγουμένως, υπολογίζεται η συνδυασμένη ανταμοιβή combined_reward για τη δράση που επιλέχθηκε. Αυτή τη φορά όμως αυτή η ανταμοιβή συγχρίνεται με το υπάρχων κατώφλι (current_threshold) και υπολογίζεται μια συνολική ανταμοιβή (overall_reward)
- Αν το combined_reward είναι μεγαλύτερο από το current_threshold, τότε το overall_reward είναι θετικό, αλλιώς όμως είναι μικρότερο, το overall_reward είναι αρνητικό. Προφανώς όσο μεγαλύτερο είναι το combined_reward από το current_threshold, τόσο μεγαλύτερο θα είναι και το overall_reward ενώ όσο μικρότερο είναι το combined_reward από το current_threshold, τόσο πιο αρνητικό θα είναι το overall_reward
- Στη συνέχεια ενημερώνεται το overall_rewards_per_speed με το overall_reward της επιλεγμένης ταχύτητας
- Άμα το overall_rewards_per_speed της επιλεγμένης ταχύτητας είναι θετικό, τότε η πιθανότητα επιλογής της αυξάνεται (action_probabilities[chosen_action] *= 1.5), αλλιώς όμως είναι αρνητικό μειώνεται (action_probabilities[chosen_action] *= 0.5)
- Τέλος οι πιθανότητες κανονικοποιούνται για να εξασφαλιστεί ότι το άθροισμά τους ισούται με 1

4. Ενημέρωση κατωφλιού:

Κάθε 5 επαναλήψεις ορίζεται ένα καινούργιο κατώφλι σύμφωνα με την παρακάτω εξίσωση:

```

threshold = max(combined_rewards_array) - (max(
    combined_rewards_array) / decay_rate)

# Keep the higher threshold
new_threshold = max(current_threshold, threshold)

```

Το combined_rewards_array αποθηκεύει όλα τα combined_rewards που έχουν δωθεί μέχρι στιγμής για όλες τις ταχύτητες. Από αυτά επιλέγεται το μεγαλύτερο και έπειτα

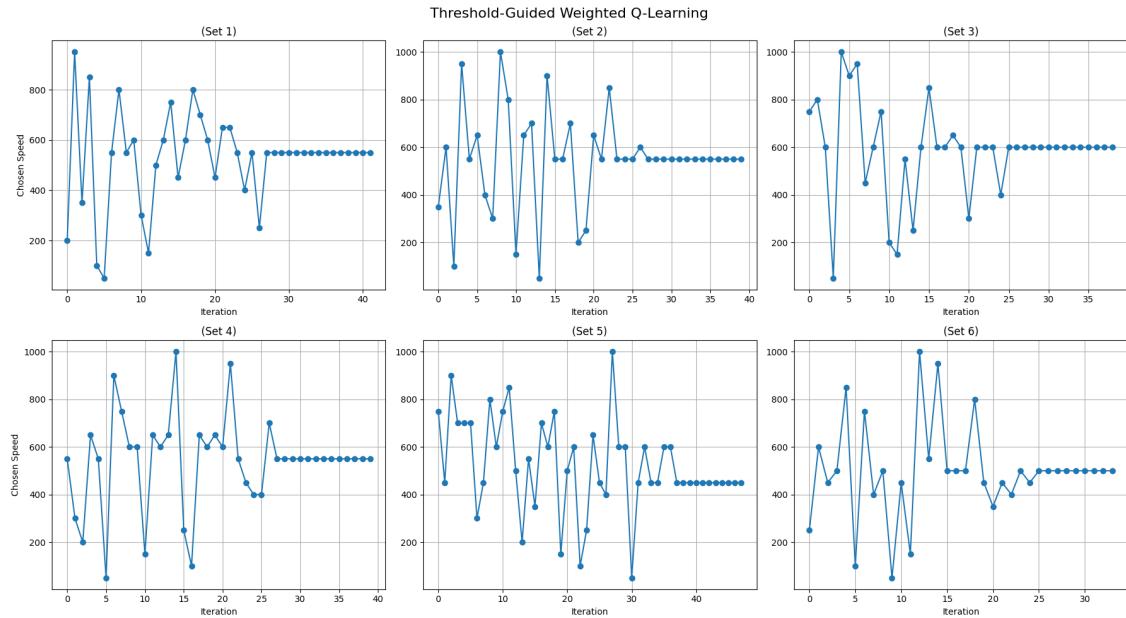
μειώνεται κατά έναν παράγοντα ($\max(\text{combined_rewards_array}) / \text{decay_rate}$), όπου το `decay_rate` αυξάνεται και αυτό γραμμικά κάθε 5 επανλήψεις. Οπότε ουσιαστικά, κάθε 5 επανλήψεις το κατώφλι γίνεται μεγαλύτερο

5. Ενημέρωση πιθανών ταχυτήτων:

Οι δράσεις με `overall_reward` κάτω από -0.2 αφαιρούνται από τη λίστα με τις διαθέσιμες ταχύτητες (`available_speeds`)

```
# Remove speeds with overall_reward lower than -0.2
for speed in list(available_speeds):
    if overall_rewards_per_speed[speed] < -0.2:
        available_speeds.remove(speed)
        del action_probabilities[speed]
```

Στο Σχήμα 4.5, παρουσιάζονται τα αποτελέσματα από 6 διαφορετικές «διαδικασίες εκπαίδευσης» για την ίδια μπάλα μάζας 4.4g. Όπως και προηγούμενως, αρχικά η επιλογή ταχυτήτων ποικίλει σε όλο το εύρος του πεδίου τιμών, υποδεικνύοντας ότι ο αλγόριθμος διερευνά διάφορες επιλογές. Καθώς προχωρούν οι επαναλήψεις, η επιλογή ταχύτητας τείνει να συγκλίνει σε μια συγκεκριμένη τιμή, υποδηλώνοντας ότι ο αλγόριθμος εκμεταλλεύεται τη γνώση που έχει μάθει για να επιλέξει την πιο αποδοτική ταχύτητα. Ωστόσο, αυτή τη φορά παρατηρούμε ότι κατά μέσο όρο ο αλγόριθμος συγκλίνει μετά από 27-28 επαναλήψεις σε ένα εύρος ταχυτήτων από 450 έως 600.



Σχήμα 4.5: Threshold-Guided Weighted Q-Learning

Οπότε συνοψίζοντας, στον Πίνακα 4.1 παρουσιάζονται οι βασικές διαφορές των δύο σεναρίων:

Χαρακτηριστικό	Boltzmann Distribution Q-Learning	Threshold-Guided Weighted Q-Learning
Επιλογή Δράσης	Πιθανολογική με βάση τα Q-values και τη θερμοκρασία	Πιθανολογική με βάση τα Q-values και τα δυναμικά προσαρμοσμένα βάρη
Κατώφλι Ανταμοιβής	Σταθερό	Δυναμικό
Εξερεύνηση & Εκμετάλλευση	Ισορροπείται από την παράμετρο θερμοκρασίας και το epsilon	Ισορροπείται με δυναμικά βάρη και κατώφλι
Ταχύτητα Σύγκλισης	M.O. 40 Επαναλήψεις	M.O. 27-28 Επαναλήψεις

Πίνακας 4.1: Βασικές διαφορές ανάμεσα στα δύο σενάρια

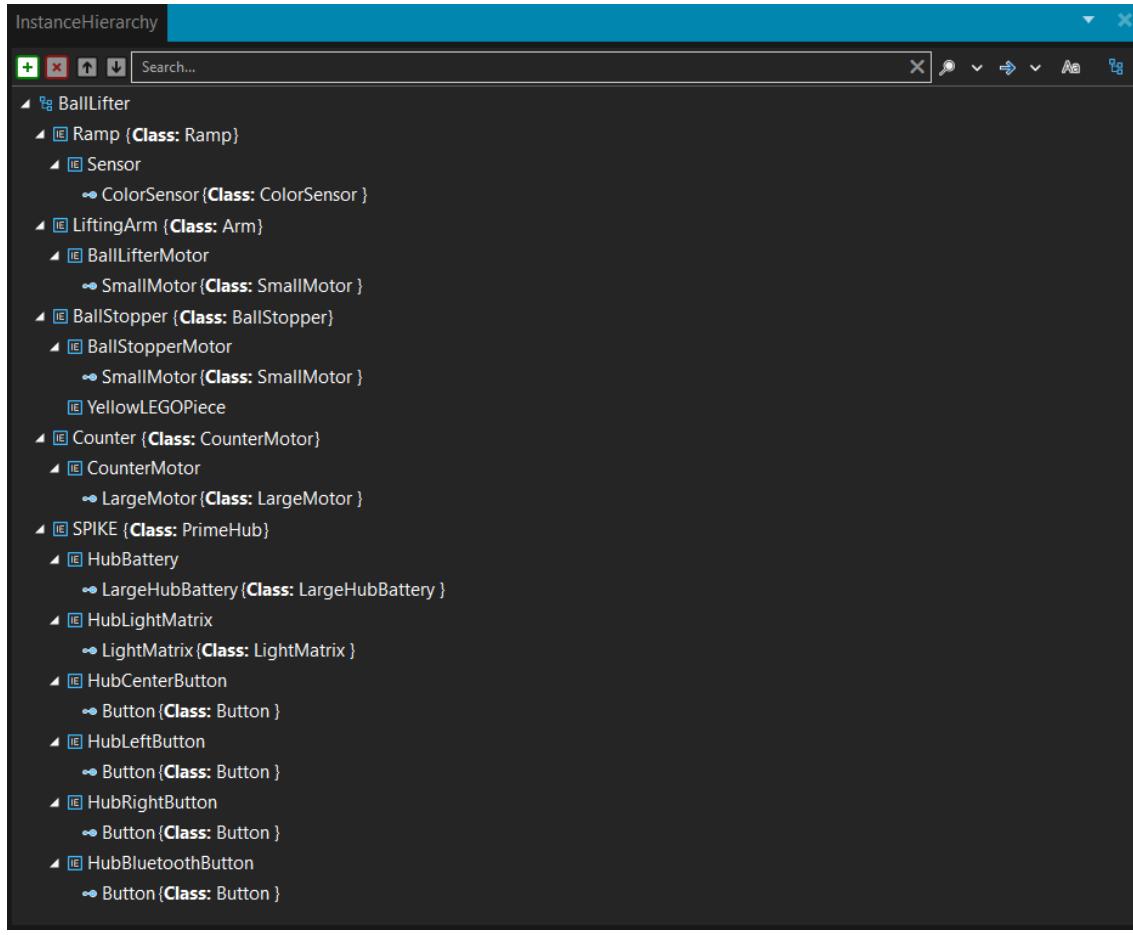
4.2 AutomationML σε ThingsBoard οντότητες

Η ευελιξία του AutomationML δε συμβαδίζει πάντα με τον τρόπο με τον οποίο είναι δομημένα τα άλλα στοιχεία ενός συστήματος. Το Thingsboard, για παράδειγμα, περιορίζει την οργάνωση δεδομένων σε "Assets" και "Devices", γεγονός που μπορεί να καταστήσει δύσκολη την άμεση μετάφραση μοντέλων που έχουν δημιουργηθεί σε AutomationML. Για το λόγο αυτό, είναι σημαντικό να υπάρχει ένα καλά καθορισμένο μοντέλο μετασχηματισμού που να γεφυρώνει το χάσμα μεταξύ της ευελιξίας της AutomationML και των δομικών περιορισμών άλλων πλατφορμών. Στην παρούσα ΔΕ χρησιμοποιήσαμε το μοντέλο που φαίνεται στον Πίνακα 4.2 και είχε προταθεί στο [1].

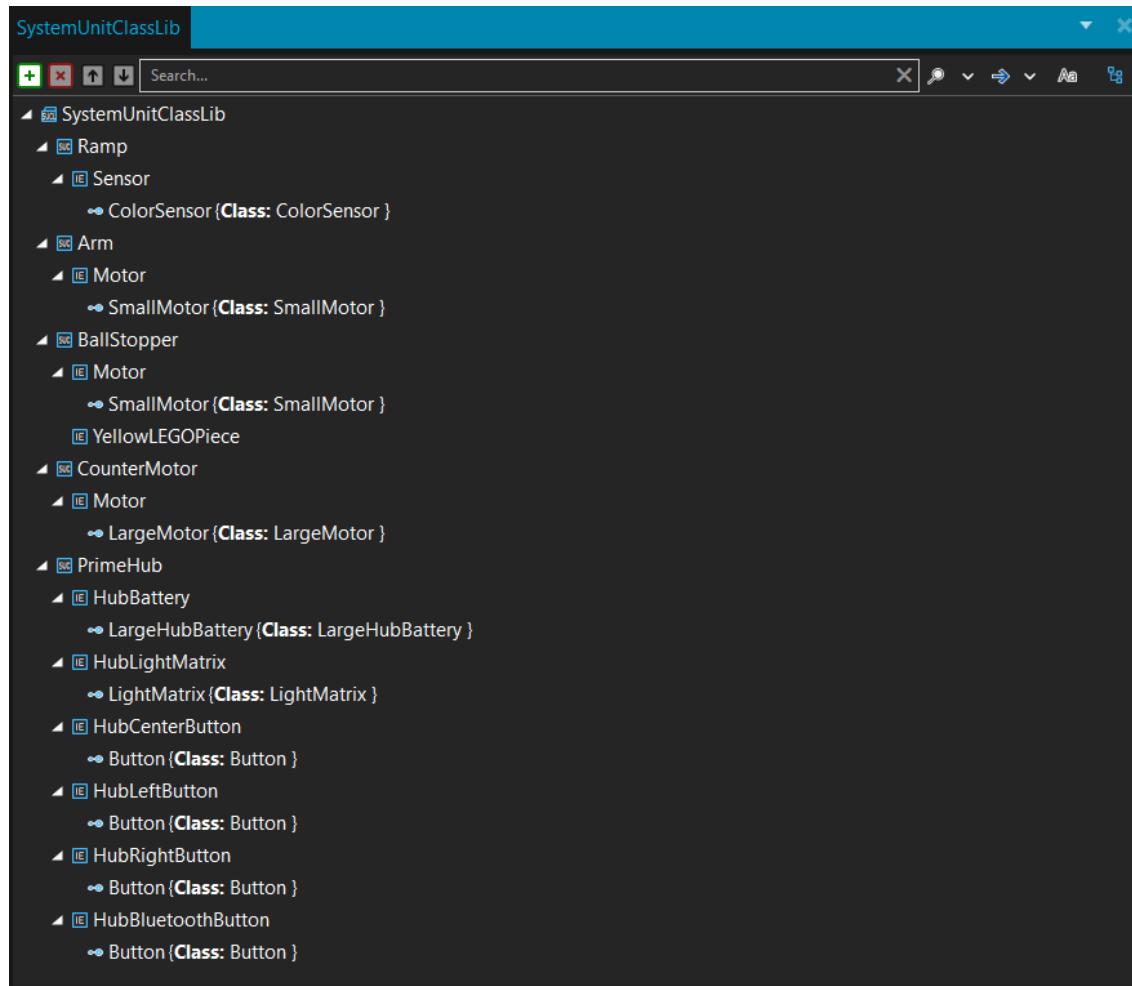
Σχεδιαστικό Αντικείμενο	Αντιστοιχία σε AML	Αντιστοιχία σε Thingsboard
Αντικείμενο φυσικού εξοπλισμού	SUC που υλοποιεί Interface	Device
Λογική ομαδοποίηση εξοπλισμού	SUC χωρίς Interface	Asset
Ιδιότητα εξοπλισμού	Interface attributes	Device additionalInfo property

Πίνακας 4.2: Κανόνες μετατροπής ανάμεσα σε AutomationML και ThingsBoard

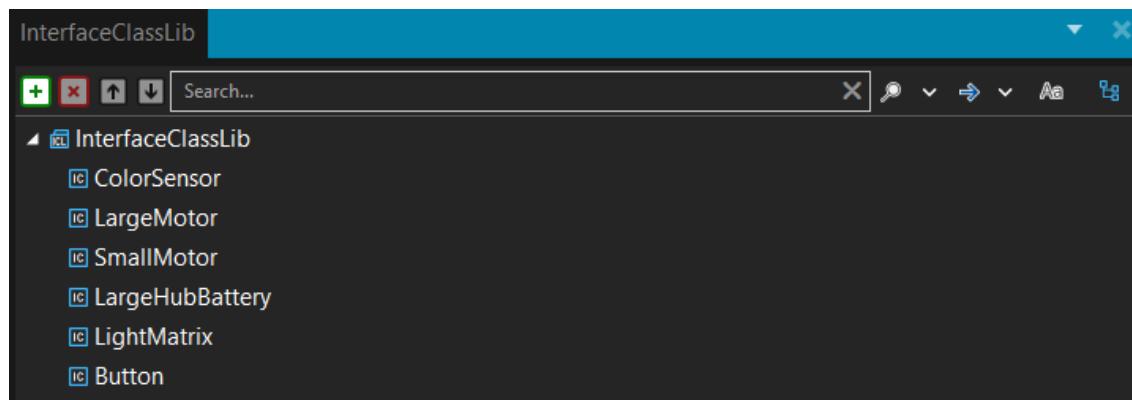
Στη συνέχεια παρουσιάζουμε τη μοντελοποίηση του BallLifter Robot που έγινε στον AutomationML Editor σύμφωνα με τους κανόνες του Πίνακα 4.2.



Σχήμα 4.6: Instance Hierarchy of BallLifter Robot



Σχήμα 4.7: System Unit Class BallLifter Robot



Σχήμα 4.8: Interface Class BallLifter Robot

Στο Σχήμα 4.6 βλέπουμε το Instance Hierarchy του BallLifter Robot το οποίο αποτελείται από πολλά Internal Elements. Χρησιμοποιώντας τους σχεδιαστικούς περιορισμούς του Πίνακα 4.2, όποιο Internal Element δεν υλοποιεί κάποιο Interface, θα μετατραπεί στο ThingsBoard ως Asset ενώ αν υλοποιεί κάποιο Interface, θα μετατραπεί ως Device. Για παράδειγμα, το Internal Element με όνομα LiftingArm δεν υλοποιεί κάποιο Interface οπότε θα οριστεί ως Asset. To BallLifterMotor όμως που είναι ένα InternalElement που υλοποιεί Interface (το SmallMotor), θα οριστεί σαν Device.

Επίσης, κατά την μετατροπή της AML δομής σε οντότητες του ThingsBoard, είναι σημαντικό να διατηρηθούν οι ιεραρχικές σχέσεις που έχουν δημιουργηθεί. Αυτό το πετυχαίνουμε κάνοντας χρήση των "Relations" δομών που προσφέρει το ThingsBoard. Τέλος, όπως ανάφερθηκε και στο προηγούμενο κεφάλαιο, το ThingsBoard προσφέρει την δυνατότητα σχεδίασης προφίλ για Devices και Assets τα οποία αποθηκεύουν κοινές πληροφορίες που μπορεί να έχουν διαφορετικές οντότητες. Εμεία, κατά την δημιουργία ενός Asset, θα του αναθέτουμε στατικά το προφίλ BallLifter Component ενώ κατά τη δημιουργία ενός Device θα του αναθέτουμε προφίλ με το όνομα του Interface που υλοποιεί.

To script που υλοποιήσαμε για την μετατροπή, είναι γραμμένο σε Python. Το .aml αρχείο που θέλουμε να μετατρέψουμε σε ThingsBoard οντότητες, είναι XML μορφής, αλλά για την ευκολότερη διαχείρηση των δεδομένων το μετατρέψαμε σε μορφή JSON. Το Python script διαβάζει το JSON αρχείο και κάνοντας χρήση αναδρομικών συναρτήσεων, αναγνωρίζει τις οντότητες που είναι να δημιουργηθούν καθώς και τις ιεραρχικές σχέσεις που έχουν μεταξύ τους. Η επικοινωνία με το ThingsBoard έγινε μέσω χρήσης του Python REST Client¹.

Στη συνέχεια παρουσιάζονται screenshots από τη λειτουργία του script και τα αποτελέσματα στο ThingsBoard.

```
leo@Megatron:~/Documents/Thesis/AML2TB$ python3 json2tb.py
Asset BallLifter_1 created and assigned to asset profile BallLifter Component
Asset Ramp_1 created and assigned to asset profile BallLifter Component
Relation from BallLifter_1 to Ramp_1 created
Device ColorSensor_1 created and assigned to device profile ColorSensor
Relation from Ramp_1 to ColorSensor_1 created
Asset LiftingArm_1 created and assigned to asset profile BallLifter Component
Relation from BallLifter_1 to LiftingArm_1 created
Device SmallMotor_1 created and assigned to device profile SmallMotor
Relation from LiftingArm_1 to SmallMotor_1 created
Asset BallStopper_1 created and assigned to asset profile BallLifter Component
Relation from BallLifter_1 to BallStopper_1 created
Device SmallMotor_2 created and assigned to device profile SmallMotor
Relation from BallStopper_1 to SmallMotor_2 created
Asset YellowLEGOPiece_1 created and assigned to asset profile BallLifter Component
Relation from BallStopper_1 to YellowLEGOPiece_1 created
Asset Counter_1 created and assigned to asset profile BallLifter Component
Relation from BallLifter_1 to Counter_1 created
Device LargeMotor_1 created and assigned to device profile LargeMotor
Relation from Counter_1 to LargeMotor_1 created
Asset SPIKE_1 created and assigned to asset profile BallLifter Component
Relation from BallLifter_1 to SPIKE_1 created
Device LargeHubBattery_1 created and assigned to device profile LargeHubBattery
Relation from SPIKE_1 to LargeHubBattery_1 created
Device Button_1 created and assigned to device profile Button
Relation from SPIKE_1 to Button_1 created
Device LightMatrix_1 created and assigned to device profile LightMatrix
Relation from SPIKE_1 to LightMatrix_1 created
```

Σχήμα 4.9: Αποτέλεσμα Python Script

¹<https://github.com/thingsboard/thingsboard-python-rest-client>

<input type="checkbox"/>	Created time ↓	Name	Device profile	Label	State	Customer	Public	Is gateway			
<input type="checkbox"/>	2024-02-22 23:50:08	LightMatrix_1	LightMatrix		Inactive		<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:08	Button_1	Button		Inactive		<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:08	LargeHubBattery_1	LargeHubBattery		Inactive		<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:07	LargeMotor_1	LargeMotor		Inactive		<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:07	SmallMotor_2	SmallMotor		Inactive		<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:07	SmallMotor_1	SmallMotor		Inactive		<input type="checkbox"/>	<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:07	ColorSensor_1	ColorSensor		Inactive		<input type="checkbox"/>	<input type="checkbox"/>			

Σχήμα 4.10: Devices

<input type="checkbox"/>	Created time ↓	Name	Asset profile	Label	Customer	Public			
<input type="checkbox"/>	2024-02-22 23:50:08	SPIKE_1	BallLifter Component			<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:07	Counter_1	BallLifter Component			<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:07	YellowLEGOPiece_1	BallLifter Component			<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:07	BallStopper_1	BallLifter Component			<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:07	LiftingArm_1	BallLifter Component			<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:07	Ramp_1	BallLifter Component			<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:07	BallLifter_1	BallLifter Component			<input type="checkbox"/>			

Σχήμα 4.11: Assets

<input type="checkbox"/>	Created time ↓	Name	Profile type	Transport type	Description	Default			
<input type="checkbox"/>	2024-02-22 23:50:08	LightMatrix	Default	Default	Device Profile for LightMatrix	<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:08	Button	Default	Default	Device Profile for Button	<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:08	LargeHubBattery	Default	Default	Device Profile for LargeHubBattery	<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:07	LargeMotor	Default	Default	Device Profile for LargeMotor	<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:07	SmallMotor	Default	Default	Device Profile for SmallMotor	<input type="checkbox"/>			
<input type="checkbox"/>	2024-02-22 23:50:07	ColorSensor	Default	Default	Device Profile for ColorSensor	<input type="checkbox"/>			

Σχήμα 4.12: Device Profiles

Asset profiles			
	Created time ↓	Name	Description
			Default
<input type="checkbox"/>	2024-02-22 23:50:07	BallLifter Component	General BallLifter Component

Σχήμα 4.13: Asset Profiles

4.3 Αμφίδρομη επικοινωνία μεταξύ ThingsBoard και LEGO SPIKE Prime

Το τελευταίο και πιο σύνθετο κομμάτι της υλοποίησης είναι η επίτευξη της αμφίδρομης επικοινωνίας ανάμεσα στο ThingsBoard και το LEGO SPIKE Prime η οποία πραγματοποιήθηκε μέσω ενός Python Script, το οποίο δρα ως μεσάζοντας. Όπως φαίνεται και στο Σχήμα 4.14, το BallLifter Robot επικονωνεί με το PC μέσω Bluetooth, ενώ το ThingsBoard επικονωνεί με το PC κάνοντας χρήση του πρωτοκόλλου MQTT και του ThingsBoard REST API.



Σχήμα 4.14: Αμφίδρομη επικοινωνία μεταξύ LEGO SPIKE Prime και ThingsBoard

4.3.1 Επικοινωνία LEGO SPIKE Prime με PC μέσω Bluetooth

Η επικοινωνία ανάμεσα στο BallLifter Robot και το PC επιτεύχθηκε μέσω Bluetooth, κάνοντας χρήση της open-source Python BLE (Bluetooth Low Energy) βιβλιοθήκης Bleak². Για να μπορέσει να εντοπίσει το SPIKE Hub, είναι απαραίτητο να μην το έχουμε συνδέσει με κάποιο άλλο πρόγραμμα (π.χ. το Pybricks Code).

Η αναζήτηση της συσκευής γίνεται με την εντολή

```
# Search for the hub
device = await BleakScanner.find_device_by_name(HUB_NAME)
```

Αφότου βρεθεί η συσκευή, η σύνδεση γίνεται με την εντολή

```
# Connect to the hub
async with BleakClient(device, handle_disconnect) as client:
```

Μόλις εντοπιστεί το SPIKE και δημιουργήθει η σύνδεση, μπορούμε να ξεκινήσουμε το πρόγραμμα που έχουμε ήδη φορτώσει στο SPIKE, πατώντας το Hub Button.

²<https://github.com/hbldh/bleak>

Για την αποστολή δεδομένων από το SPIKE στο PC χρησιμοποιείται η εντολή stdout ενώ για τη λήψη δεδομένων από το PC στο SPIKE χρησιμοποιείται η εντολή stdin. Στη συνέχεια παρουσιάζουμε ένα παράδειγμα για κάθε περίπτωση.

SPIKE σε PC

Στο script που τρέχει το BallLifter Robot έχουμε την εντολή:

```
stdout.buffer.write('Balls Counter: ' + str(counter) + '\n')
```

Στο script που τρέχει στο PC έχουμε δημιουργήσει τη συνάρτηση

```
async def handle_rx(_, data: bytearray)
```

που είναι υπεύθυνη για τη λήψη και σωστή αποκωδικοποίηση των μηνυμάτων που στέλνει το SPIKE.

Αρχικά αποθηκεύουμε τα εισερχόμενα δεδομένα σε έναν buffer

```
buffer += data.replace(b"\x00", b"").decode()
```

διότι ένα πακέτο δεδομένων BLE μπορεί να μεταφέρει μέγιστο ωφέλιμο φορτίο (payload) 20 bytes. Οπότε αν ένα μήνυμα που σταλθεί από το BallLifter Robot είναι μεγαλύτερο από 20 bytes, πρέπει πρώτα να περιμένουμε να σταλθεί ολόκληρο και μετά να το επεξεργαστούμε. Γι' αυτό το λόγο, στο τέλος κάθε μηνύματος που στέλνει το BallLifter Robot, έχουμε προσθέσει το χαρακτήρα ”\n” και στη συνάρτηση handle_rx έχουμε προσθέσει την εντολή

```
if buffer.endswith("\n"):
```

ώστε να καταλαβαίνουμε πότε τελειώνει το ένα μήνυμα και πότε αρχίζει το επόμενο. Στη συνέχεια, τα μηνύματα φιλτράρονται ανάλογα με το περιοχόμενο τους χρησιμοποιώντας if-else blocks και στέλνονται οι κατάλληλες πληροφορίες στο ThingsBoard. Για το παράδειγμα που δώσαμε παραπάνω, έχουμε ορίσει την εντολή

```
if "Balls Counter:" in buffer:
    ballsCounter = int(buffer.split(":")[-1].strip())
    dataToSend = {"numberOfBalls": ballsCounter}
    requests.post(ballLifterURL, data=json.dumps(dataToSend),
    headers=headers)
```

η οποία στην προκειμένη περίπτωση θα φιλτράρει το μήνυμα και θα κρατήσει μόνο το νούμερο ballsCounter που στάλθηκε από το BallLifter Robot.

PC σε SPIKE

Για να στείλουμε δεδομένα από το PC στο SPIKE χρησιμοποιούμε τη συνάρτηση

```
async def send(data):
    await client.write_gatt_char(
        PYBRICKS_COMMAND_EVENT_CHAR_UUID,
        b"\x06" + data, # prepend "write stdin" command (0x06)
        response=True,
    )
```

και τα στέλνουμε σε μορφή bytes. Για παράδειγμα έχουμε τις παρακάτω εντολές

```
data_to_send = number_of_balls.to_bytes(2, "big")
await send(data_to_send)
```

Το number_of_balls είναι ένας ακέραιος αριθμός μεταξύ 0 και 99 ο οποίος μετατρέπεται σε αναπαράσταση των 2 bytes σε big-endian σειρά (το πιο σημαντικό byte είναι το πρώτο).

To SPIKE διαβάζει το μηνύμα που στέλνουμε με τις παρακάτω εντολές

```
keyboard2 = poll()
keyboard2.register(stdin)
input_from_TB = stdin.buffer.read(2)
if int.from_bytes(input_from_TB, 'big') >= 0 and int.from_bytes(input_from_TB, 'big') <= 99:
    # rest of code
```

δηλαδή διαβάζει 2 bytes και στη συνέχεια τα αποκωδικοποιεί. Αν ο αποκωδικοποιημένος αριθμός είναι μεταξύ του 0 και του 99, εκτελείται ο υπόλοιπος κώδικας.

4.3.2 Επικοινωνία ThingsBoard με PC μέσω REST API και MQTT

Η επικοινωνία ανάμεσα στο ThingsBoard και το PC επιτεύχθηκε μέσω του ThingsBoard REST API και του MQTT πρωτοκόλλου.

MQTT

Στο προηγούμενο κεφάλαιο, έγινε αναφορά στην κατηγορία Attributes που έχουν όλα τα Devices στο ThingsBoard. Στην δικιά μας περίπτωση θα μας χρειαστούν μόνο τα Client και Shared Attributes. Κάνοντας χρήση του MQTT πρωτοκόλλου, μπορούμε να επικοινωνούμε με τα Devices μας στο ThingsBoard, κάνοντας subscribe και publish στα κατάλληλα topics (βλ. κεφάλαιο 2.2.3). Πιο συγκεκριμένα, για να κάνουμε subscribe σε αλλαγές των Shared Attributes των Devices, κάνουμε subscribe στο topic

```
v1/devices/me/attributes
```

Για να ζητήσουμε Client ή Shared Attributes από το server, πρέπει να στείλουμε ένα Publish μήνυμα στο topic

```
v1/devices/me/attributes/request/$request_id
```

Πρωτού στείλουμε όμως στείλουμε Publish μήνυμα με το Request, πρέπει πρώτα να έχουμε κάνει subscribe στο topic

```
v1/devices/me/attributes/response/+
```

Επίσης θα χρειαστεί να κάνουμε subscribe και στο topic

```
v1/devices/me/rpc/request/+
```

για να ενημερωνόμαστε για RPC (Remote Procedure Call) commands από το server.

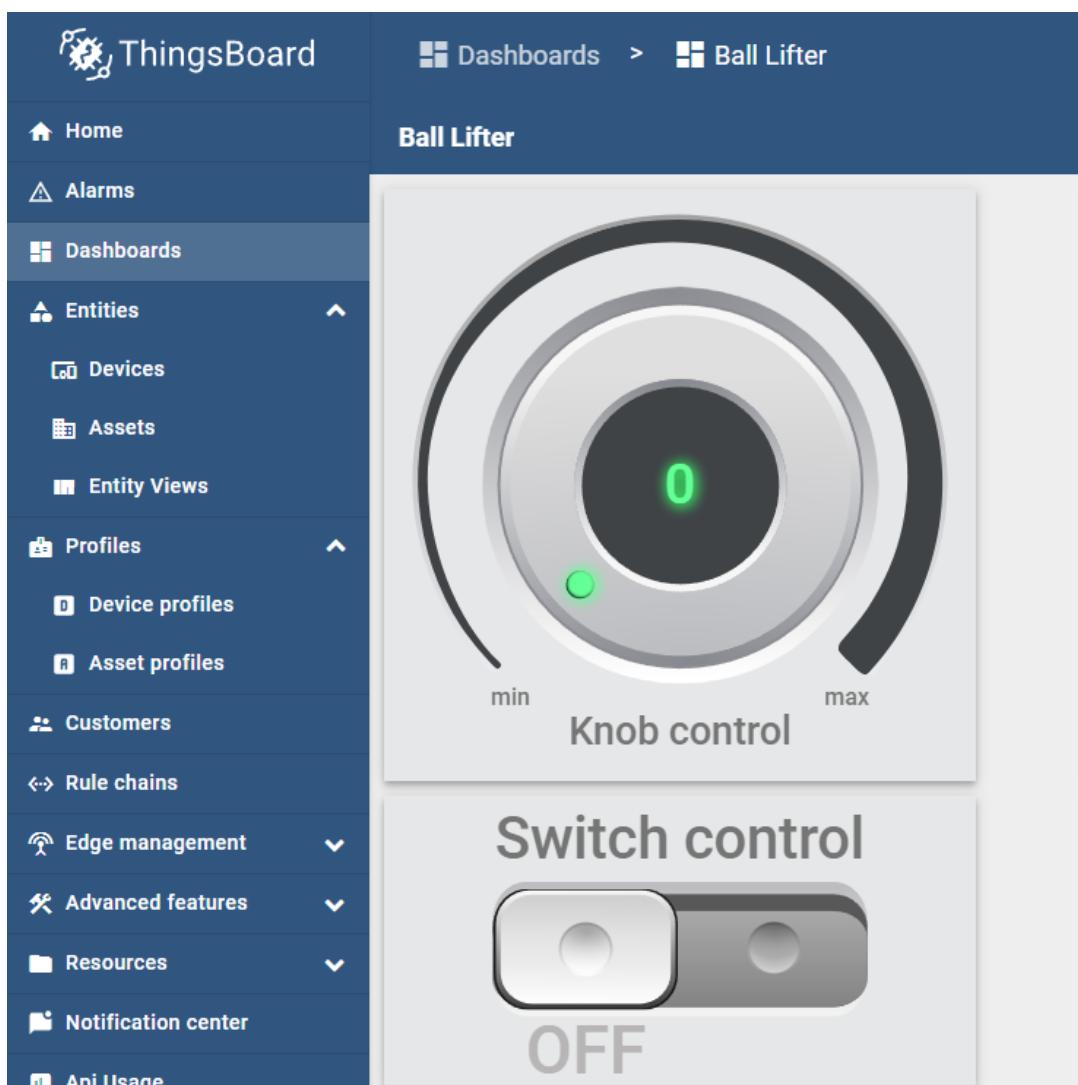
Όλα τα παραπάνω τα υλοποιούμε με την ακόλουθη συνάρτηση

```

def on_connect(client, userdata, flags, rc):
    client.subscribe("v1/devices/me/rpc/request/+")
    client.subscribe("v1/devices/me/attributes")
    client.subscribe("v1/devices/me/attributes/response/+")
    client.publish(
        "v1/devices/me/attributes/request/1",
        json.dumps({"clientKeys": "switch"}),
        1,
    )
)

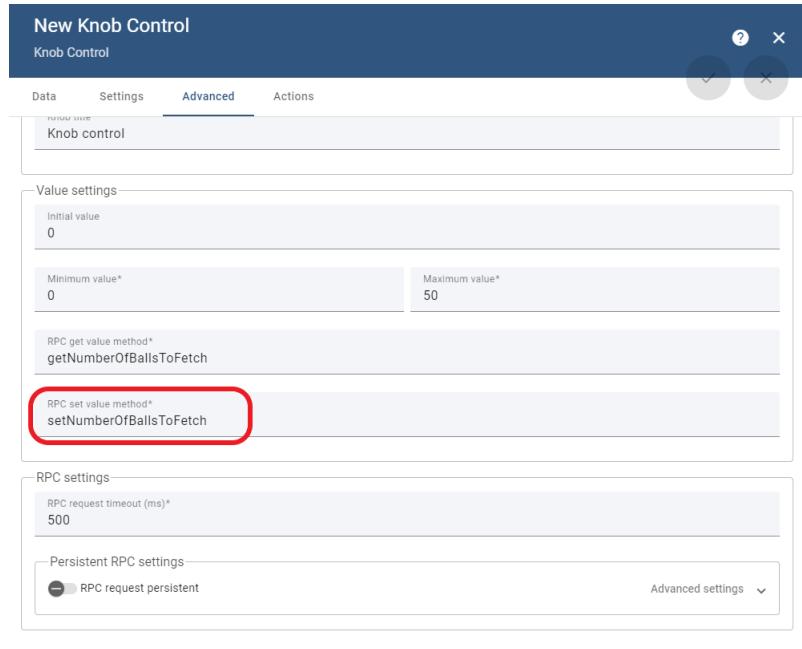
```

Στο Σχήμα 4.15 βλέπουμε 2 Widgets που δημιουργήσαμε σε ένα Dashboard του ThingsBoard, τα οποία μας επιτρέπουν τον απομακρισμένο έλεγχο του BallLifter Robot. Το πρώτο (Knob Control) είναι για να επιλέγουμε τον αριθμό των επιθυμητών μπαλών, δηλαδή των αριθμό των επαναλήψεων που θέλουμε να εκτελέσει το BallLifter και το δεύτερο (Switch Control) είναι ένας διακόπτης ο οποίος ξεκινάει ή σταματάει τη λειτουργία του προγράμματος του BallLifter.



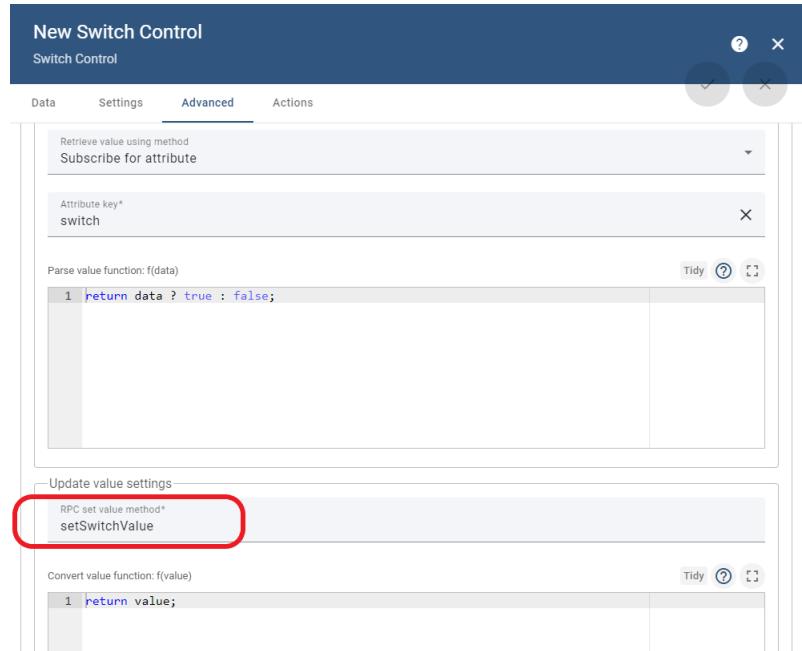
Σχήμα 4.15: Control Widgets

Κάθε φορά που αλλάζουμε την τιμή του Knob Control, στέλνετε ένα RPC command στο server με όνομα setNumberOfBallsToFetch, με τη νεά τιμή που θέτουμε.



Σχήμα 4.16: RPC set value method για Knob Control

Αντίστοιχα, κάθε φορά που αλλάζουμε την τιμή του Switch Control (on/off), στέλνετε ένα RPC command στο server με όνομα setSwitchValue.



Σχήμα 4.17: RPC set value method για Switch Control

Για να ενημερωνόμαστε για τις αλλαγές αυτές έχει υλοποιηθεί η συνάρτηση

```

1 def on_message(client, userdata, msg):
2     data = json.loads(msg.payload)
3     global startProgramFlag, stopProgramFlag, keepRunningFlag,
4         interruptedFlag, numberofBalls
5
6     if msg.topic.startswith("v1/devices/me/rpc/request/"):
7         if data["method"] == "setSwitchValue":
8             client.publish(
9                 "v1/devices/me/attributes",
10                json.dumps({"switch": data["params"]}),
11                1,
12            )
13            if data["params"] == True:
14                startProgramFlag = True
15                keepRunningFlag = True
16            else:
17                keepRunningFlag = False
18                interruptedFlag = True
19        elif data["method"] == "setNumberOfBallsToFetch":
20            client.publish(
21                "v1/devices/me/attributes",
22                json.dumps({"numberOfBallsToFetch": data["params"]}),
23                1,
24            )
25            numberofBalls = data["params"]
26        elif msg.topic == "v1/devices/me/attributes":
27            if data["sh_switch"] == False:
28                client.publish(
29                    "v1/devices/me/attributes",
30                    json.dumps({"switch": False}),
31                    1,
32                )
33                keepRunningFlag = False
34                interruptedFlag = True

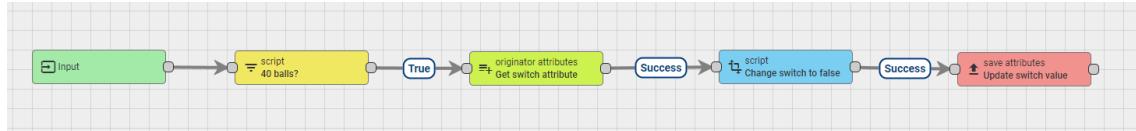
```

Ανάλογα με το RPC command που θα σταλθεί από το server, ενημερώνουμε το αντίστοιχο Client Attribute. Δηλαδή αν το RPC command έχει όνομα setSwitchValue, τότε ενημερώνεται το switch Client Attribute, όλιώς άμα έχει όνομα setNumberOfBallsToFetch ενημερώνεται το numberOfBallsToFetch Client Attribute. Επίσης ενημερώνονται κάποιες global μεταβλητές οι οποίες χρησιμοποιούνται στη συνέχεια για τη σωστή λειτουργία του προγράμματος.

Ακόμα, έχουμε άλλο ένα else-if block το οποίο αφορά το topic

```
v1/devices/me/attributes
```

το οποίο όπως είπαμε αναφέρεται στις αλλαγές των Shared Attributes. Αυτό το κομμάτι κώδικα δεν έχει σχέση με τα Widgets, αλλά έχει να κάνει με μία επιπλέον λειτουργία που υλοποιήσαμε στο ThingsBoard για να αναδείξουμε τις δυνατότητες των Rule Chains. Ήτο συγκεκριμένα υλοποιήθηκε το παρακάτω Rule Chain:



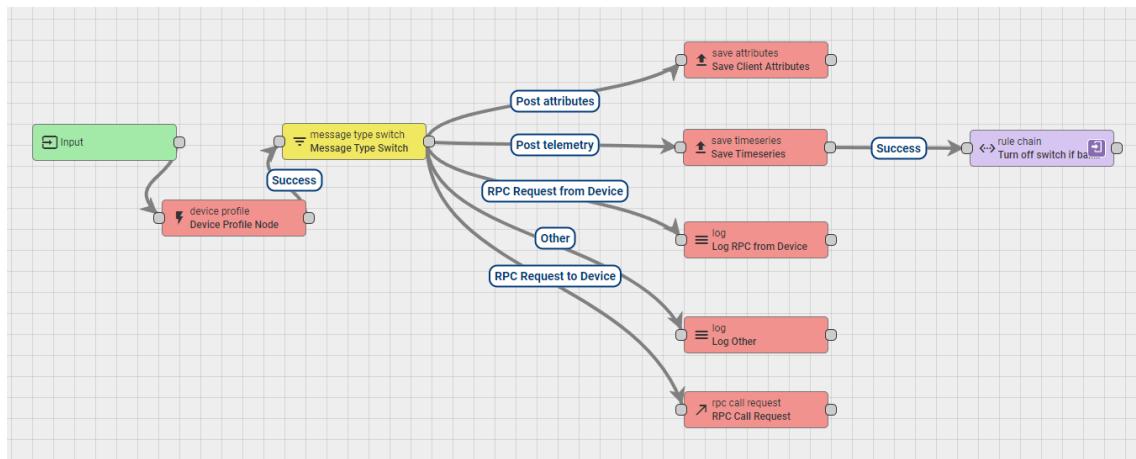
Σχήμα 4.18: Rule Chain που κλείνει το Switch Control Widget αυτόματα μετά από 40 επαναλήψεις του BallLifter

Η λειτουργία του παραπάνω Rule Chain είναι η εξής:

1. Αρχικά έχουμε ένα filter κόμβο ο οποίος εξετάζει αν ο αριθμός των μπαλών είναι ίσος με 40. Αυτό γίνεται χρησιμοποιώντας την τελεμετρική τιμή με όνομα `numberOfBalls`. Αν η συνθήκη του κόμβου είναι αληθής, προχωράμε στον επόμενο κόμβο
2. Στη συνέχεια συνδέεται με έναν enrichment κόμβο ο οποίος ανακτά το Shared Attribute με όνομα `sh_switch`. Αν η ανάκτηση είναι επιτυχής, προχωράμε στον επόμενο κόμβο
3. Στη συνέχεια συνδέεται με έναν transformation κόμβο ο οποίος αλλάζει την τιμή του `sh_switch` σε `false`. Αν η αλλαγή είναι επιτυχής, προχωράμε στον επόμενο κόμβο
4. Τέλος, έχουμε έναν action κόμβο, ο οποίος αποθηκεύει τη νέα τιμή του `sh_switch`

Είναι εύλογο να αναρωτηθεί κανείς γιατί δημιουργήσαμε ένα καινούργιο Shared Attribute (`sh_switch`) και δεν κάναμε χρήση του ήδη υπάρχοντος Client Attribute (`switch`). Η απάντηση σε αυτό το ερώτημα είναι πολύ απλή. Διότι το ThingsBoard δεν επιτρέπει την αλλαγή τιμών των Client Attributes μέσω Rule Chain. Τα Client Attributes έχουν σχεδιαστεί για να αντιπροσωπεύουν τη βασική αλήθεια που αναφέρεται από την ίδια τη συσκευή.

Για να τεθεί σε λειτουργία το παραπάνω Rule Chain πρέπει να το συνδέσουμε με το Root Rule Chain που έχουμε ορίσει.



Σχήμα 4.19: Σύνδεση του Root Rule Chain με το custom Rule Chain

Όπως φαίνεται στο Σχήμα 4.19, συνδέουμε το Rule Chain μας με τον κόμβο του Root Rule Chain που είναι υπεύθυνος για την αποθήκευση των τηλεμετρικών δεδομένων. Με αυτό

τον τρόπο, ο πρώτος filter κόμβος του Σχήματος 4.18, έχει τη δυνατότητα να διαφέρει την τιμή της τηλεμετρικής μεταβλητής `numberOfBalls`.

REST API

Τα δεδομένα του BallLifter Robot, αφού τα επεργαστούμε, τα στέλνουμε στο ThingsBoard με ένα HTTP POST request, χρησιμοποιώντας το ThingsBoard REST API και την βιβλιοθήκη requests της Python. Χρησιμοποιώντας το παράδειγμα που αναφέραμε νωρίτερα,

```
if "Balls Counter:" in buffer:
    ballsCounter = int(buffer.split(":")[-1].strip())
    dataToSend = {"numberOfBalls": ballsCounter}
    requests.post(ballLifterURL, data=json.dumps(dataToSend),
    headers=headers)
```

βλέπουμε τον τρόπο με τον οποίο στέλνονται τα τηλεμετρικά δεδομένα στο ThingsBoard. Στο Σχήμα 4.20, βλέπουμε μια λίστα με τα ονόματα όλων των τηλεμετρικών δεδομένων που στέλνει το BallLifter Robot

- `energyExpenditure`: Δείχνει την ενέργεια που καταναλώνει το BallLifter Robot κατά τη λειτουργία του
- `motorSpeed`: Δείχνει την ταχύτητα του ballLifterMotor σε κάθε επανάληψη
- `numberOfBalls`: Δείχνει το ΣΥΝΟΛΙΚΟ αριθμό μπαλών που έχουν περάσει από το colorSensor του BallLifter Robot κατά τη λειτουργία του προγράμματος

Latest telemetry		
	Key ↑	Value
<input type="checkbox"/>	Last update time	
<input type="checkbox"/>	2024-06-04 15:16:58	energyExpenditure
<input type="checkbox"/>	2024-06-04 15:16:50	motorSpeed
<input type="checkbox"/>	2024-06-04 15:16:50	numberOfBalls

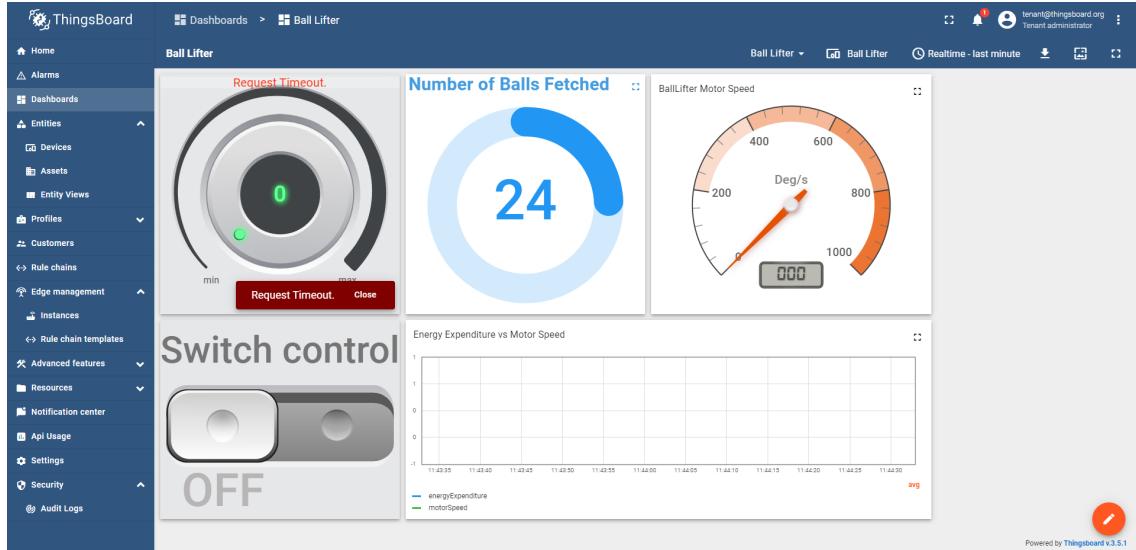
Σχήμα 4.20: Telemetry Keys & Values

4.3.3 Οπτικοποίηση δεδομένων στο ThingsBoard και παράδειγμα λειτουργίας

Στο ThingsBoard, εκτός από τα δύο control Widgets που παρουσιάσαμε νωρίτερα, έχουν προσθέθει και κάποια επιπλέον για να οπτικοποιήσουμε τα δεδομένα που στέλνει το BallLifter Robot. Πιο συγκεκριμένα έχουμε:

- Number of Balls Fetched: Gauge Widget. Οπτικοποιεί τα δεδομένα του τηλεμετρικού κλειδιού `numberOfBalls`

- BallLifter Motor Speed: Gauge Widget. Οπτικοποιεί τα δεδομένα του τηλεμετρικού κλειδιού motorSpeed
- Energy Expenditure vs Motor Speed: Timeseries Line Chart Widget. Οπτικοποιεί τα δεδομένα των τηλεμετρικών κλειδιών energyExpenditure και motorSpeed



Σχήμα 4.21: Widgets

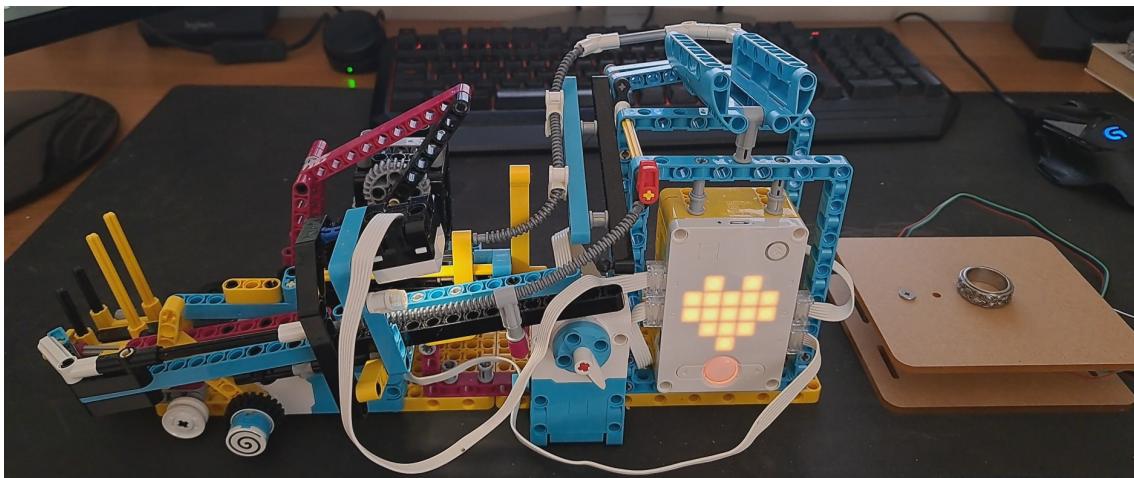
Για την καλύτερη κατανόηση της λειτουργίας του BallLifter Robot και της οπτικοποίησης των δεδομένων στο ThingsBoard, θα παρουσιάσουμε στιγμιότυπα οιθόνης και φωτογραφίες από ένα demo run κάνοντας χρήση του σεναρίου Threshold-Guided Weighted Q-Learning που αναλύσαμε προηγουμένως.

Βήμα 1

Τρέχουμε το Python Script ενώ έχουμε ανοίξει το SPIKE και γίνεται η σύνδεση μέσω Bluetooth. Έπειτα, πατάμε το Hub Button για να εκκινήσει το πρόγραμμα του BallLifter Robot.

```
Windows PowerShell
PS D:\CEID\Thesis\Files\Scripts\Python> python3 .\bidirectional.py
Start the program on the hub now with the button.
```

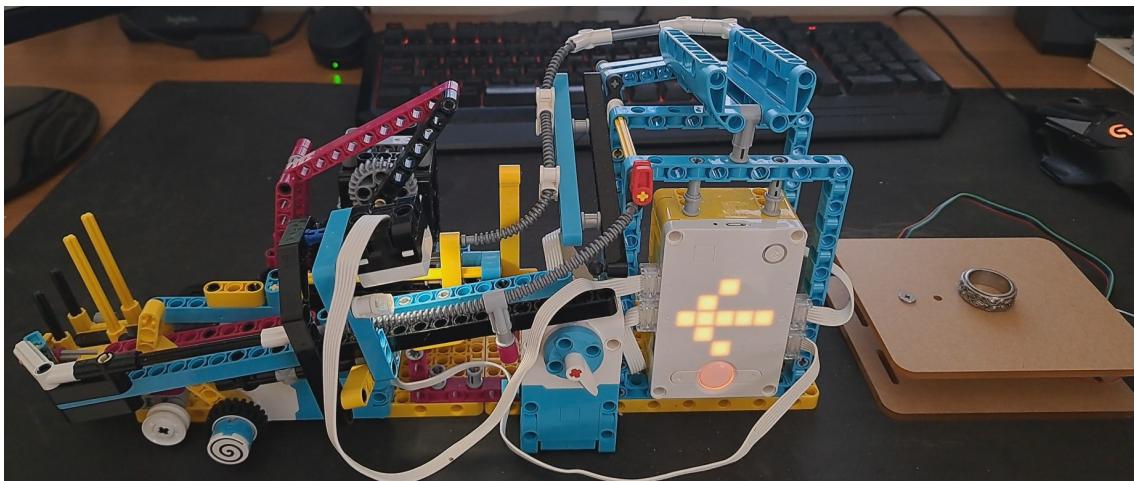
Σχήμα 4.22: Εκτέλεση Python Script



Σχήμα 4.23: BallLifter αφού πατήσουμε το Hub Button

Βήμα 2

Επιλέγουμε τη λειτουργία "Training" του BallLifter Robot, πατώντας το Left Button. Υπάρχει και η λειτουργία "Testing", η οποία χρησιμοποιείται όταν το BallLifter Robot έχει συλλέξει αρκετά δεδομένα για το περιβάλλον του, οπότε δεν υπάρχει πλέον ανάγκη εξερεύνησης. Για την επιλογή της λειτουργίας "Testing" πρέπει να πατήσουμε το Right Button.



Σχήμα 4.24: BallLifter αφού πατήσουμε το Left Button

Βήμα 3

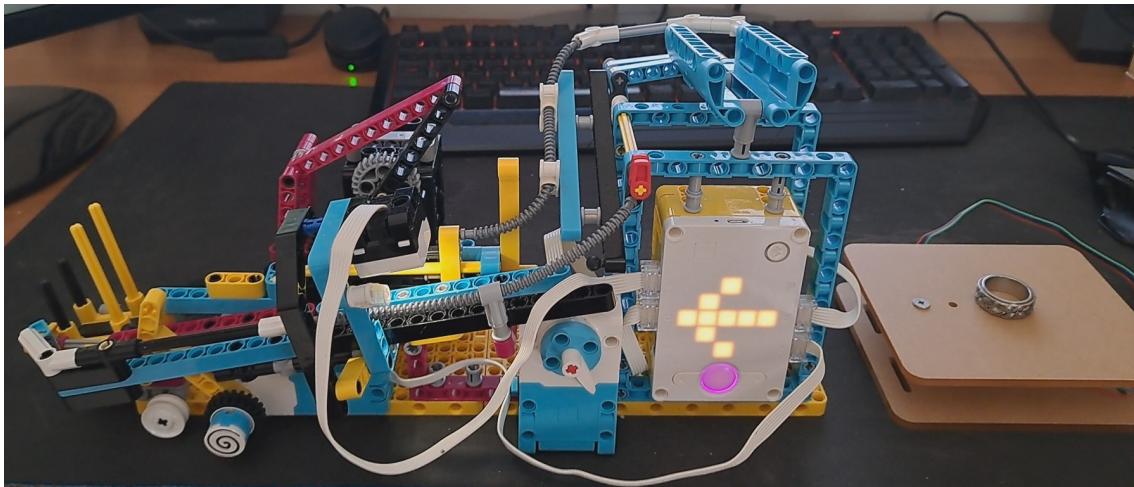
Επιλέγουμε τυχαία τον αριθμό 75 με το Knob Widget και ξεκινάμε το πρόγραμμα με το Switch Widget.



Σχήμα 4.25: Ενωτίνηση BallLifter από το Dashboard

Βήμα 4

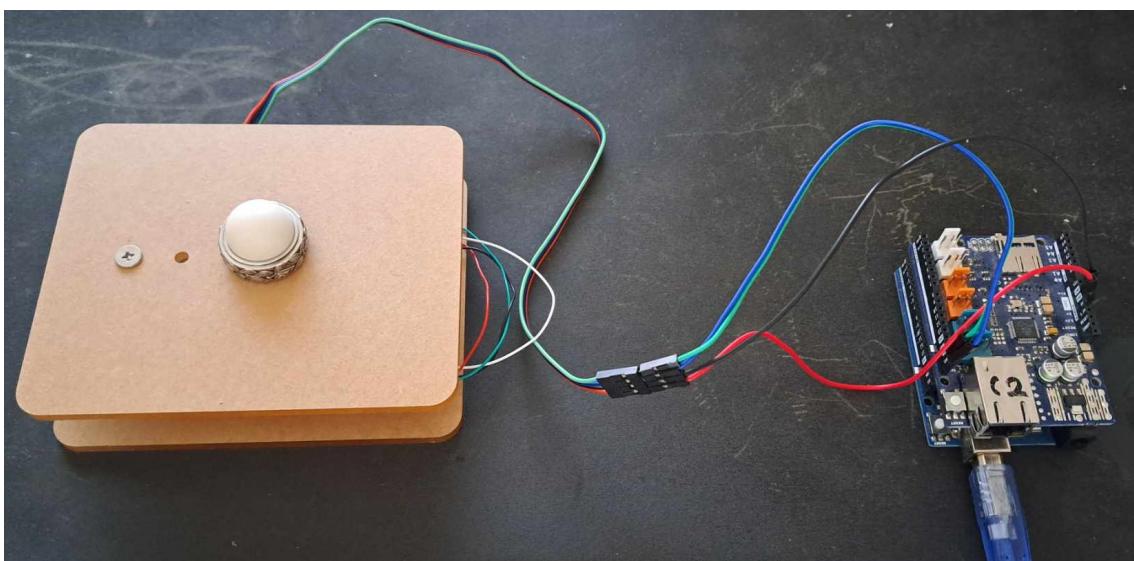
Το μωβ χρώμα του Hub Button υποδεικνύει ότι το BallLifter Robot είναι έτοιμο να λάβει την τιμή βάρους από τη ζυγαριά που είναι συνδεδεμένη στο Arduino.



Σχήμα 4.26: Αναμονή για ζύγισμα μπάλας

Βήμα 5

Τοποθετούμε τη μπάλα πάνω στη ζυγαριά και στέλνουμε τη μέτρηση στο BallLifter Robot.

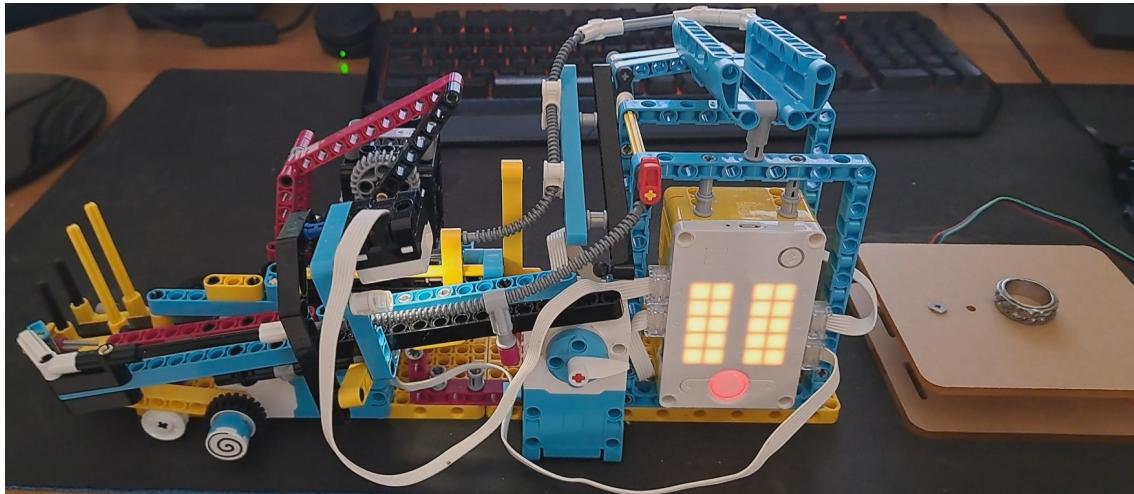


Σχήμα 4.27: Ζύγισμα μπάλας

Βήμα 6

Το κόκκινο χρώμα του Hub Button υποδεικνύει ότι το BallLifter Robot έλαβε την τιμή βάρους από το Arduino. To BallLifter Robot σετάρει το largeMotor να δείχνει στο νοητό

75. Αυτό σημαίνει ότι θα πραγματοποιήσει μια στροφή 2250 μοιρών (6 πλήρεις περιστροφές των 360 μοιρών + 1 στροφή 90 μοιρών). Στο Light Matrix του SPIKE εμφανίζεται ο αριθμός 0. Επίσης το ballStopperMotor κατεβάζει το κίτρινο LEGO piece ώστε να επιτραπεί η ροή των μπαλών.

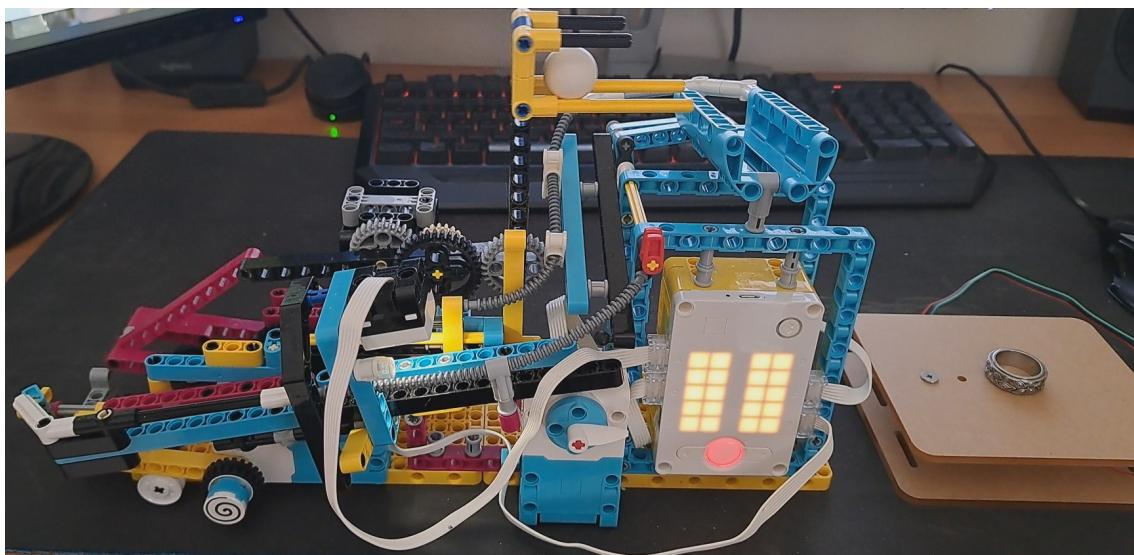


Σχήμα 4.28: Σετάρισμα BallLifter Robot

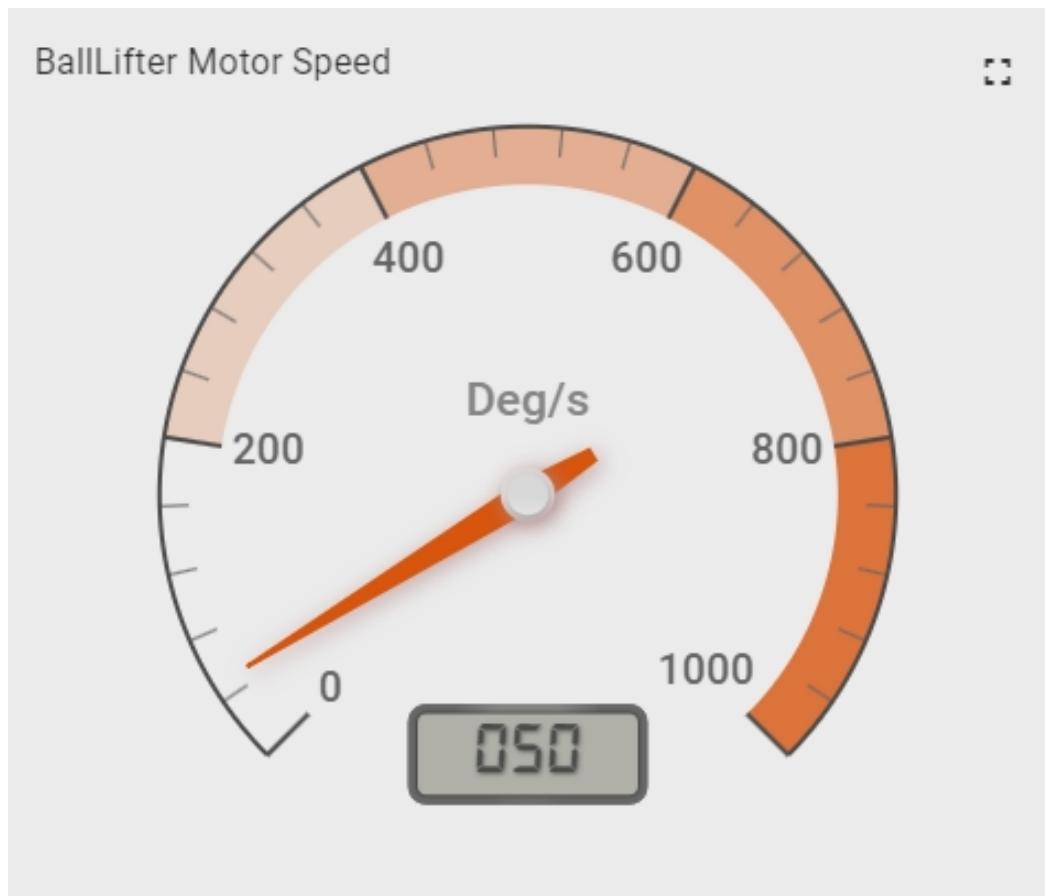
Βήμα 7

Ρίχνουμε την μπάλα και ξεκινάει η λειτουργία του Threshold-Guided Weighted Q-Learning σεναρίου. Στη συνέχεια παραθέτονται κάποια στιγμιότυπα από τη διαδικασία εκπαίδευσης.

1η Επανάληψη: Επιλέχθηκε η ταχύτητα 50



Σχήμα 4.29: 1η Επανάληψη



Σχήμα 4.30: 1η Επανάληψη

9η Επανάληψη: Επιλέχθηκε η ταχύτητα 450

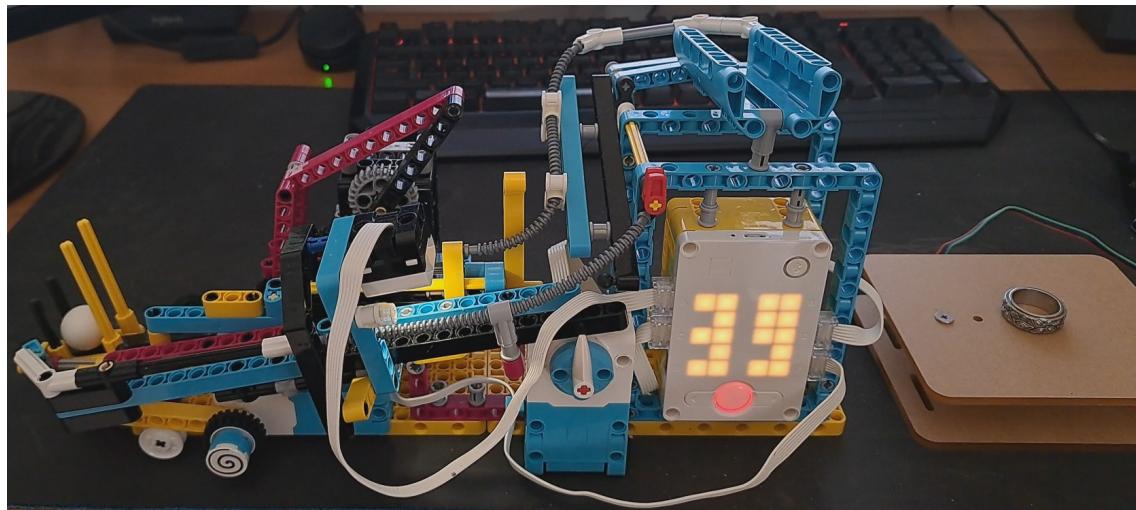


Σχήμα 4.31: Μετά από 8 επαναλήψεις

40η Επανάληψη: Επιλέχθηκε η ταχύτητα 500. Παρατηρούμε από το Energy Expenditure vs Motor Speed Widget, ότι ο αλγόριθμος έχει συγκλίνει εδώ και αρκετές επαναλήψεις στην ταχύτητα 500.



Σχήμα 4.32: Μετά από 39 επαναλήψεις



Σχήμα 4.33: Μετά από 39 επαναλήψεις

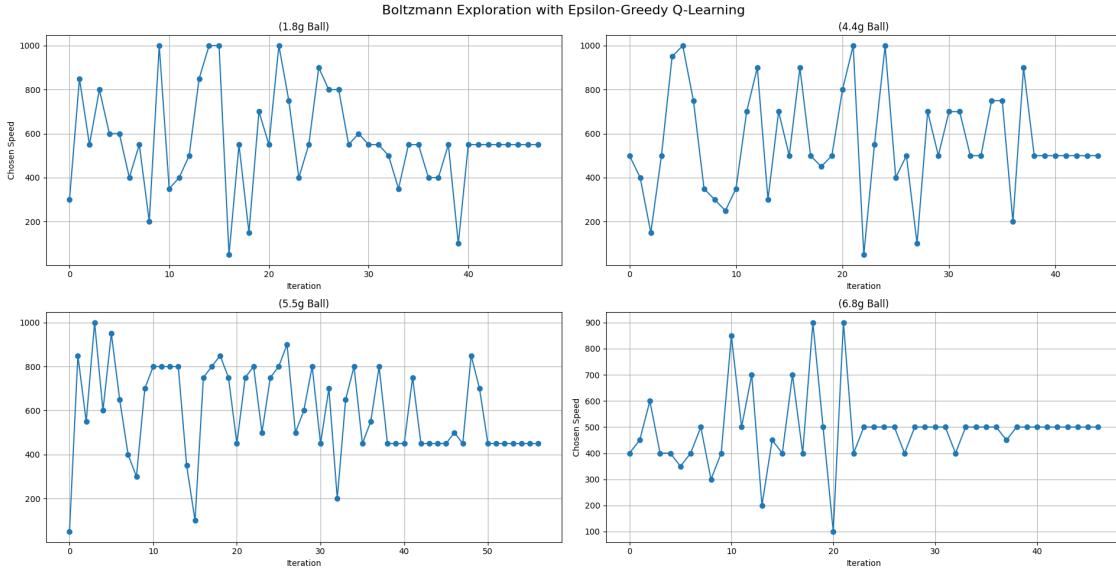
Αξίζει να σημειωθεί ότι η λειτουργία του προγράμματος μπορεί να διακοπεί ανά πάσα στιγμή, κλείνωντας απλά το Switch Widget.

4.3.4 Αποτελέσματα

Σε αυτό το υποκεφάλαιο παρουσιάζονται τα αποτελέσματα των δύο Q-Learning σεναρίων που ανακτήθηκαν έπειτα από δοκιμές με τέσσερα διαφορετικά βάρη μπάλας: 1.8g, 4.4g, 5.4g και 6.8g. Κύριο μέτρο αξιολόγησης των σεναρίων είναι η ταχύτητα σύγκλισης, η οποία αποτελεί

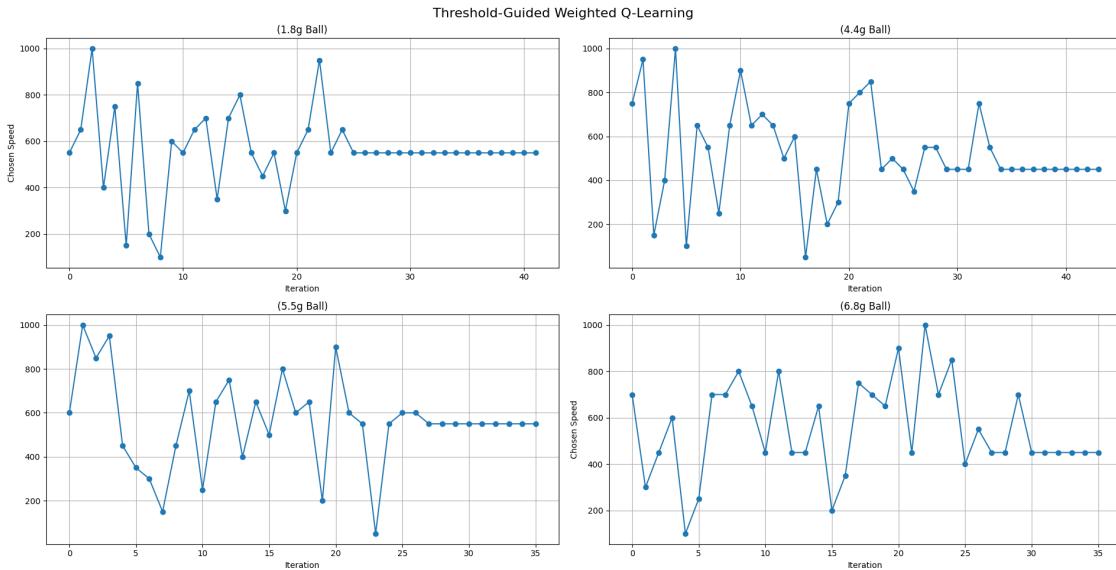
χρίσμα παράγοντα για την προσαρμογή σε πραγματικό χρόνο και την αποτελεσματικότητα των ρομποτικών συστημάτων.

Boltzmann Exploration with Epsilon-Greedy Q-Learning



Σχήμα 4.34: Αποτελέσματα για Boltzmann Exploration with Epsilon-Greedy Q-Learning

Threshold-Guided Weighted Q-Learning



Σχήμα 4.35: Αποτελέσματα για Threshold-Guided Weighted Q-Learning

Από τα γραφήματα είναι εμφανές ότι το Threshold-Guided Weighted Q-Learning (TG-WQL) επιτυγχάνει ταχύτερη σύγκλιση σε σύγκριση με το Boltzmann Exploration with Epsilon-Greedy Q-Learning, ανεξάρτητα από το βάρος της μπάλας. Αυτή η ταχύτερη σύγκλι-

ση αποτελεί σημαντικό πλεονέκτημα για πραγματικές ρομποτικές εφαρμογές όπου η γρήγορη προσαρμογή και η αποδοτικότητα είναι ζωτικής σημασίας.

Γιατί το TGWQL συγκλίνει ταχύτερα:

- **Δυναμικό κατώφλι ανταμοιβής:** Το δυναμικό κατώφλι ανταμοιβής παιζει καθοριστικό ρόλο στην επιτάχυνση της σύγκλισης. Αυξάνοντας σταδιακά το κατώφλι, δημιουργείται ένας «κινούμενος στόχος» για το ρομπότ, ωθώντας το να εξερευνά και να βελτιώνεται συνεχώς. Αυτός ο μηχανισμός εστιάζει τη διαδικασία μάθησης σε ενέργειες που είναι πιθανό να αποφέρουν καλύτερα αποτελέσματα, οδηγώντας σε ταχύτερο καθορισμό της βέλτιστης ταχύτητας.
- **Επιλογή σταθμισμένης δράσης:** Ο σταθμισμένος μηχανισμός επιλογής δράσεων συμβάλλει περαιτέρω στην ταχύτερη σύγκλιση. Ευνοώντας τις ενέργειες με ιστορικό υψηλότερων ανταμοιβών, επιτρέπει στον αλγόριθμο να εκμεταλλευτεί αποτελεσματικότερα τη συσσωρευμένη γνώση του. Αυτό μειώνει τη διερεύνηση μη βέλτιστων ενεργειών και βοηθά το ρομπότ να επικεντρωθεί στις πιο υποσχόμενες ταχύτητες.
- **Συνδυαστικό αποτέλεσμα:** Ο συνδυασμός μεταξύ του δυναμικού κατωφλίου ανταμοιβής και της σταθμισμένης επιλογής δράσης δημιουργεί έναν θετικό βρόχο ανατροφοδότησης. Καθώς το ρομπότ μαθαίνει και επιτυγχάνει υψηλότερες ανταμοιβές, το κατώφλι αυξάνεται, δίνοντάς του περαιτέρω κίνητρο να εξερευνήσει ενέργειες με υψηλότερες επιδόσεις. Αυτή η δυναμική αλληλεπίδραση μεταξύ εξερεύνησης και εκμετάλλευσης επιταχύνει τη διαδικασία σύγκλισης.

Εφαρμογές στον πραγματικό κόσμο

Σε πρακτικές ρομποτικές εφαρμογές, η ταχύτερη σύγκλιση σημαίνει ότι το ρομπότ μπορεί να προσαρμοστεί ταχύτερα σε νέες εργασίες ή μεταβαλλόμενες συνθήκες. Αυτό είναι ιδιαίτερα σημαντικό σε περιβάλλοντα όπου το ρομπότ πρέπει να χειρίστει αντικείμενα με διαφορετικές ιδιότητες, όπως τα διαφορετικά βάρη της μπάλας σε αυτό το σενάριο. Προσδιορίζοντας γρήγορα τη βέλτιστη ταχύτητα ανύψωσης για κάθε μπάλα, το Threshold-Guided Weighted Q-Learning σενάριο επιτρέπει στο ρομπότ να λειτουργεί πιο αποδοτικά και αποτελεσματικά.

Κεφάλαιο 5

Επίλογος

5.1 Συμπεράσματα

Εν κατακλείδι, σε αυτή τη ΔΕ, διερεύνησαμε αναλυτικά τα προβλήματα ενσωμάτωσης εφαρμογών IoT σε βιομηχανικά συστήματα παραγωγής, εστιάζοντας στις προκλήσεις και τις ευκαιρίες που προκύπτουν από την 4η Βιομηχανική Επανάσταση. Μέσω μιας λεπτομερούς βιβλιογραφικής ανασκόπησης και επισκόπησης των τεχνολογιών, θέσαμε στέρεα θεμέλια στον τομέα του Industry 4.0, με ιδιαίτερη έμφαση στις τεχνολογίες AutomationML και ThingsBoard, των οποίων έγινε και χρήση.

Η ενότητα περιγραφής της υλοποίησης της εφαρμογής μας, παρείχε μια πρακτική επίδειξη αυτών των τεχνολογιών μέσω της δημιουργίας του ρομπότ BallLifter, που κατασκευάστηκε χρησιμοποιώντας το LEGO SPIKE Prime κιτ. Υλοποιήσαμε και συγκρίναμε δύο Q-Learning αλγόριθμους για προσαρμοστική συμπεριφορά: Boltzmann Exploration with Epsilon-Greedy Q-Learning και Threshold-Guided Weighted Q-Learning. Αυτοί οι αλγόριθμοι σχεδιάστηκαν με σκοπό την εύρεση της βέλτιστης ταχύτητας ανύψωσης του ρομποτικού βραχίονα για ποικίλα βάρη σφαιρών. Διερευνήσαμε τις επιδόσεις και των δύο αλγορίθμων, δίνοντας έμφαση στην ταχύτητα σύγκλισης και στην ικανότητά τους να εντοπίζουν τις βέλτιστες ενέργειες, αποδεικνύοντας έτσι τις δυνατότητες της ενισχυτικής μάθησης για αυτόνομη προσαρμογή σε βιομηχανικές διαδικασίες.

Η δημιουργία ενός Instance Hierarchy στο πλαίσιο της AutomationML και η επακόλουθη μετάφραση σε οντότητες του ThingsBoard μέσω ενός Python Script, αναδεικνύουν τις δυνατότητες εξομοίωσης και εικονικοποίησης βιομηχανικών περιβαλλόντων παραγωγής. Η προσέγγιση αυτή όχι μόνο διευκολύνει την εξοικονόμηση πόρων και ενέργειας αλλά και προωθεί την ανάπτυξη διαλειτουργικών και ελεγχόμενων συστημάτων εντός του περιβάλλοντος παραγωγής. Επιπλέον, η επίτευξη της αμφίδρομης επικοινωνίας μεταξύ του ρομπότ και της πλατφόρμας ThingsBoard, η οποία επιτυγχάνεται μέσω ενός Python Script και πρωτοκόλλων επικοινωνίας όπως είναι τα Bluetooth/MQTT/REST API, αποτελεί παράδειγμα των δυνατότήτων ανταλλαγής δεδομένων και ελέγχου σε πραγματικό χρόνο σε βιομηχανικά συστήματα παραγωγής.

Τα αποτελέσματα της παρούσας εργασίας δείχνουν ότι η προτεινόμενη προσέγγιση είναι

ένας εφικτός και αποτελεσματικός τρόπος για την ενσωμάτωση συσκευών IoT σε περιβάλλοντα παραγωγής. Η χρήση της AutomationML παρέχει έναν τυποποιημένο τρόπο για τη μοντελοποίηση του περιβάλλοντος παραγωγής, επιτρέποντας την απρόσκοπτη ενσωμάτωση νέων IoT εφαρμογών χωρίς να διαταράσσει τα υπάρχοντα συστήματα. Υπογραμμίζει τη σημασία της διαλειτουργικότητας, της ανταλλαγής δεδομένων και του ελέγχου για την επίτευξη αποδοτικών και βιώσιμων συστημάτων παραγωγής.

Συνοψίζοντας, η παρούσα ΔΕ συμβάλλει στην κατανόηση και την προώθηση των IoT εφαρμογών σε βιομηχανικά συστήματα παραγωγής παρέχοντας μια διεξοδική διερεύνηση των σχετικών τεχνολογιών, μια πρακτική επίδειξη μέσω του ρομπότ BallLifter, καθώς και πληροφορίες σχετικά με τις προκλήσεις και τις ευκαιρίες της ενσωμάτωσης του IoT σε παλαιά συστήματα.

5.2 Μελλοντικές κατευθύνσεις και πιθανές επεκτάσεις

Η ενσωμάτωση εφαρμογών IoT σε βιομηχανικά συστήματα παραγωγής, όπως διερευνάται στην παρούσα διατριβή, ανοίγει πολλούς συναρπαστικούς δρόμους για μελλοντική έρευνα και ανάπτυξη. Μια πολλά υποσχόμενη κατεύθυνση είναι η δημιουργία ενός ψηφιακού διδύμου (digital twin) του BallLifter robot. Το ψηφιακό διδύμο, είναι μια ψηφιακή αναπαράσταση μιας φυσικής οντότητας, η οποία μπορεί να χρησιμοποιηθεί για την προσομοίωση της συμπεριφοράς της, την πρόβλεψη της απόδοσής της και τη βελτιστοποίηση του σχεδιασμού της πριν από την πραγματική υλοποίηση. Η ιδέα αυτή ευθυγραμμίζεται με τη μελλοντική τάση ανάπτυξης των ψηφιακών διδύμων, η οποία προβλέπεται να ακολουθήσει δύο κύριες κατευθύνσεις: την ενσωμάτωση συναφών τεχνολογιών και τη συνεχή βελτίωση των βιομηχανικών εφαρμογών.

Μια άλλη πιθανή επέκταση θα μπορούσε να είναι η ενσωμάτωση μοντέλων μηχανικής μάθησης τα οποία θα επεξεργάζονται και θα αναλύουν τα τηλεμετρικά δεδομένα που συλλέγονται, με σκοπό τη βελτιστοποίηση της λειτουργίας του ρομπότ, την πρόβλεψη πιθανών βλαβών και γενικότερα την άντληση πολύτιμων πληροφοριών για τη βελτίωση της συνολικής αποδοτικότητας.

Τέλος, όπως αναφέρθηκε, τα βιομηχανικά περιβάλλοντα απαιτούν ισχυρά μέτρα ασφαλείας, οπότε στόχος μιας μελλοντικής έρευνας θα μπορούσε να είναι η εφαρμογή μηχανισμών ελέγχου ταυτότητας, αρυπτογράφησης και εξουσιοδότησης για την προστασία της ακεραιότητας των δεδομένων και της συσκευής στο προτεινόμενο πλαίσιο.

Παράρτημα A'

Pybricks Threshold-Guided Weighted Q-Learning

Αυτό είναι το script που περάστηκε στο SPIKE και επιτυγχάνει την επικονωνία του BallLifter με τον υπολογιστή και κατ' επέκταση με το ThingsBoard.

```
1 from pybricks.hubs import PrimeHub
2 from pybricks.pupdevices import Motor, ColorSensor, UltrasonicSensor,
   ForceSensor
3 from pybricks.parameters import Button, Color, Direction, Port, Side, Stop,
   Icon
4 from pybricks.robots import DriveBase
5 from pybricks.tools import wait, StopWatch
6 from usys import stdin, stdout
7 from uselect import poll, select
8 from urandom import uniform, choice
9 from umath import exp
10 from ustruct import unpack
11
12 hub = PrimeHub()
13
14 colorSensor = ColorSensor(Port.A)
15 largeMotor = Motor(Port.C)
16 ballLifterMotor = Motor(Port.D)
17 ballStopperMotor = Motor(Port.F)
18
19 stopwatch = StopWatch()
20
21 epsilon = 0.9
22 alpha = 0.1
23 gamma = 0.9
24
25 States = []
26 chosen_speeds = [] # List to store the chosen speeds and their iterations
27 qtable = {}
28
```

```

29 global total_energy_exp
30 global total_execution_time
31 global avg_reward
32
33 def setMotorAngleShortestPath(motor, target_angle, speed):
34     # Get the current angle of the motor
35     current_angle = motor.angle()
36
37     # Calculate the difference between the target angle and the current
38     # angle
39     angle_difference = target_angle - current_angle
40
41     # Adjust the difference to take the shortest path
42     if abs(angle_difference) > 180:
43         angle_difference -= 360
44
45     # Set the target angle by adding the adjusted difference to the current
46     # angle
47     target_angle = current_angle + angle_difference
48
49     # Rotate the motor to the target angle
50     motor.run_target(speed, target_angle, Stop.HOLD, wait=True)
51
52 def setMotorAngleLongestPath(motor, target_angle, speed, waitFlag):
53     # Get the current angle of the motor
54     current_angle = motor.angle()
55
56     # Calculate the difference between the target angle and the current
57     # angle
58     angle_difference = target_angle - current_angle
59
60     # Adjust the difference to take the longest path
61     if angle_difference > 0:
62         angle_difference -= 360
63
64     # Set the target angle by adding the adjusted difference to the current
65     # angle
66     target_angle = current_angle + angle_difference
67
68     # Rotate the motor to the target angle
69     motor.run_target(speed, target_angle, Stop.HOLD, wait=waitFlag)
70
71
72 # This is a function that waits for a desired color.
73 def waitForColor():
74     while True:
75         if colorSensor.color() != Color.NONE:
76             break
77         stdout.buffer.write('Energy Expenditure: ' + str(50) + '\n')
78         wait(20)
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174

```

```

75 def setLargeMotor(numOfBalls):
76     setMotorAngleShortestPath(largeMotor, 0, 1000)
77     largeMotor.reset_angle(0)
78     wait(1000)
79     largeMotor.run_target(1000, numOfBalls*30, Stop.HOLD, wait=True)
80
81 def moveLargeMotorForInterval(remainingBalls):
82     setMotorAngleShortestPath(largeMotor, remainingBalls*30, 100)
83
84 # Handle weight fluctuations and avoid creating unnecessary states
85 def find_closest_state(weight):
86     weight_tolerance = 0.1
87     for existing_state in qtable:
88         if abs(existing_state - weight) <= weight_tolerance:
89             return existing_state
90     return None # If no state is within tolerance
91
92 def weighted_choice(available_speeds, probabilities):
93     # Create a cumulative distribution list
94     cumulative_distribution = []
95     cumulative_sum = 0
96
97     for prob in probabilities:
98         cumulative_sum += prob
99         cumulative_distribution.append(cumulative_sum)
100
101    # Generate a random number between 0 and the sum of probabilities
102    random_number = uniform(0, cumulative_sum)
103
104    # Find the interval that the random number falls into
105    for i, cumulative_value in enumerate(cumulative_distribution):
106        if random_number <= cumulative_value:
107            return available_speeds[i]
108
109 def choose_action(state, available_speeds, action_probabilities=None):
110     if action_probabilities:
111         # Extract the probabilities for the available speeds
112         probabilities = [action_probabilities[speed] for speed in
113                         available_speeds]
114         return weighted_choice(available_speeds, probabilities)
115     else:
116         # If no probabilities are provided, choose randomly from available
117         # speeds
118         return choice(available_speeds)
119
120 # Function to update Q-value based on Bellman equation
121 def update_q(state, action, reward, next_state):
122     qvalue = qtable[state][action]
123     new_q = (1 - alpha) * qvalue + alpha * (
124         reward + gamma * max(qtable[next_state].values()))

```

```

123     )
124     qtable[state][action] = new_q # update Q-Table with new Q-Value
125
126 def calculate_energy_reward(motor_speed, max_energy=1510):
127     if motor_speed < 200: # If motor speed is below 200 return static
128         reward
129         return 1.93
130     energyFactor = 1.5
131     energy_expenditure = motor_speed * energyFactor
132     stdout.buffer.write('Energy Expenditure: ' + str(energy_expenditure) +
133     '\n')
134     normalized_energy = energy_expenditure / max_energy
135     inverted_energy = 1 - normalized_energy
136     reward = exp(inverted_energy * 3) - exp(normalized_energy * 3)
137     return reward / 10 + 1
138
139 # Define the altered time reward function
140 def calculate_time_reward(execution_time, max_time=3000, threshold_time
141 =2200):
142     if execution_time > threshold_time:
143         normalized_time = (execution_time - threshold_time) / (
144             max_time - threshold_time
145         )
146         reward = -2.2 * (normalized_time**0.6)
147     else:
148         normalized_time = execution_time / threshold_time
149         inverted_time = 1 - normalized_time
150         reward = 2.2 * (inverted_time**0.6)
151     return reward
152
153 def scale_reward(normalized_reward):
154     overall_reward = 2 * (1 / (1 + exp(-0.93 * normalized_reward))) - 1
155     return overall_reward
156
157 def normalize_reward(combined_reward, threshold):
158     normalized_reward = (combined_reward - threshold) # Subtract threshold
159     to center around 0
160     return normalized_reward
161
162 def calculate_overall_reward(combined_reward, threshold):
163     normalized_reward = normalize_reward(combined_reward, threshold)
164     overall_reward = scale_reward(normalized_reward)
165     return overall_reward
166
167 def ballLifterExecution(numOfBallsWanted, numOfBallsRemaining, counter,
168 firstIterationFlag, weight):
169
170     global total_energy_exp # Access the global variable
171     global total_execution_time
172     global avg_reward

```

```

168     global chosen_speeds
169
170     ballLifterMotorStopWatch = Stopwatch()
171
172     keyboard1 = poll()
173     keyboard1.register(stdin)
174
175     total_reward = 0
176     total_energy_exp = 0 # Initialize inside the function
177     iteration = 0 # Track the iteration number
178     combined_rewards_array = []
179
180     available_speeds = list(range(50, 1050, 50))
181     overall_rewards_per_speed = {speed: 0 for speed in available_speeds} #
182     # Track overall rewards for each speed
183     action_probabilities = {speed: 1/len(available_speeds) for speed in
184     available_speeds} # Initialize equal probabilities
185
186     while not keyboard1.poll(0):
187         if firstIterationFlag:
188             setLargeMotor(numOfBallsWanted)
189             setMotorAngleShortestPath(ballStopperMotor, 0, 300)
190             current_threshold = 2.3
191             decay_rate = 9.6
192             hub.display.number(counter)
193             firstIterationFlag = False
194             success = True
195             waitForColor()
196             start_total_time = ballLifterMotorStopWatch.time()
197
198             if success == False:
199                 waitForColor()
200
201             # Choose action
202             chosen_action = choose_action(weight, available_speeds,
203             action_probabilities=action_probabilities)
204             wait(20)
205
206             # Store the chosen action and iteration
207             chosen_speeds.append((iteration, chosen_action))
208             iteration += 1
209             moveLargeMotorForInterval(numOfBallsRemaining)
210
211             wait(100)
212             if not keyboard1.poll(0):
213                 start_lift_time = ballLifterMotorStopWatch.time()
214                 stdout.buffer.write('Motor Speed: ' + str(chosen_action) + '\n')
215
216             #Lifts the arm

```

```

214         setMotorAngleShortestPath(ballLifterMotor, 50, chosen_action)
215         setMotorAngleLongestPath(ballLifterMotor, 120, chosen_action,
216             True)
217         wait(400)
218         setMotorAngleLongestPath(ballLifterMotor, 420, 800, False)
219
220         start_time = stopwatch.time() # Track time for success/fail
221
222         success = False
223
224         total_lift_time = 5000
225         while stopwatch.time() - start_time < 3000: # 3-second timeout
226             if colorSensor.color() != Color.NONE:
227                 total_lift_time = (ballLifterMotorStopWatch.time() -
228                     start_lift_time)
229
230                 success = True
231                 counter += 1
232                 numBallsRemaining -= 1
233                 moveLargeMotorForInterval(numBallsRemaining)
234                 hub.display.number(counter)
235                 break
236
237                 wait(20)
238
239                 energy_expenditure = chosen_action * 1.5
240                 total_energy_exp += energy_expenditure
241                 energy_reward = calculate_energy_reward(chosen_action)
242                 time_reward = calculate_time_reward(total_lift_time)
243                 success_reward = 1 if success else -10
244                 combined_reward = success_reward * (2 * energy_reward + (3 *
245                     time_reward))
246
247                 combined_rewards_array.append(combined_reward)
248
249
250             # Update threshold every 5 iterations
251             if counter > 0 and (counter % 5) == 0:
252                 decay_rate += 0.4
253                 new_threshold = max(combined_rewards_array) - (max(
254                     combined_rewards_array) / decay_rate)
255                 current_threshold = max(current_threshold, new_threshold)
256
257             # Keep the higher threshold
258             print("Updated threshold:", current_threshold)
259             wait(20)
260             total_reward += combined_reward
261
262             overall_reward = calculate_overall_reward(combined_reward,
263                 current_threshold)
264             update_q(weight, chosen_action, combined_reward, weight)
265
266             # Update overall reward for the chosen speed
267             overall_rewards_per_speed[chosen_action] += overall_reward

```

```

258     # Update probabilities based on overall_reward
259     if overall_rewards_per_speed[chosen_action] > 0:
260         action_probabilities[chosen_action] *= 1.5 # Boost
261     elif overall_rewards_per_speed[chosen_action] < 0:
262         action_probabilities[chosen_action] *= 0.5 # Decrease
263
264     # Remove speeds with overall_reward lower than -0.2
265     for speed in list(available_speeds):
266         if overall_rewards_per_speed[speed] < -0.2:
267             available_speeds.remove(speed)
268             del action_probabilities[speed]
269
270     # Normalize probabilities
271     total_prob = sum(action_probabilities.values())
272     for speed in available_speeds:
273         action_probabilities[speed] /= total_prob
274
275     stdout.buffer.write('Balls Counter: ' + str(counter) + '\n')
276     wait(20)
277     stdout.buffer.write('\n')
278
279     if counter == numOfBallsWanted:
280         stdout.buffer.write('Done!' + '\n')
281         total_execution_time = ballLifterMotorStopWatch.time() -
282         start_total_time
283         avg_reward = total_reward / numOfBallsWanted
284         break
285
286     setMotorAngleShortestPath(ballStopperMotor, 310, 200)
287
288 while True:
289     keyboard2 = poll()
290     keyboard2.register(stdin)
291
292     pressed = []
293     counter = 0
294     flag = True
295     stdout.buffer.write('Motor Speed: ' + str(0) + '\n')
296     wait(10)
297     stdout.buffer.write('Select Mode: ' + '\n')
298
299     while not any(pressed):
300         pressed = hub.buttons.pressed()
301         hub.display.icon(Icon.HEART)
302         hub.light.on(Color.ORANGE)
303         stdout.buffer.write('Energy Expenditure: ' + str(50) + '\n')
304         wait(100)
305
306     # Wait for all buttons to be released.
307     while any(hub.buttons.pressed()):

```

```

307     wait(10)
308     if Button.LEFT in pressed:
309         mode = "Training"
310         epsilon = 0.9 # Reset epsilon for each training episode
311         hub.display.icon(Icon.ARROW_LEFT)
312         stdout.buffer.write('Selected Mode: Training' + '\n')
313     elif Button.RIGHT in pressed:
314         mode = "Testing"
315         epsilon = 0 # Set epsilon to 0 for testing
316         hub.display.icon(Icon.ARROW_RIGHT)
317         stdout.buffer.write('Selected Mode: Testing' + '\n')
318     else:
319         mode = "Waiting"
320
321     if mode in ["Training", "Testing"]:
322         stdout.buffer.write('Select Number of Balls' + '\n')
323         wait(20)
324         while not keyboard2.poll(0):
325             stdout.buffer.write('Energy Expenditure: ' + str(50) + '\n')
326             wait(100)
327             inputFromTB = stdin.buffer.read(2)
328
329             if int.from_bytes(inputFromTB, 'big') >= 0 and int.from_bytes(
330                 inputFromTB, 'big') <= 99:
331                 number_of_balls = int.from_bytes(inputFromTB, 'big')
332                 keyboard3 = poll()
333                 keyboard3.register(stdin)
334                 stdout.buffer.write('Waiting to get weight of the ball...' + \
335                                     '\n')
336                 while not keyboard3.poll(0):
337                     hub.light.on(Color.MAGENTA)
338                     wait(100)
339                     hub.light.on(Color.RED)
340                     input_from_arduino = stdin.buffer.read(4)
341                     weight = unpack('!f', input_from_arduino)[0]
342                     stdout.buffer.write("Weight received: " + str(weight) + '\n')
343
344                     closest_state = find_closest_state(weight)
345                     wait(20)
346                     stdout.buffer.write("Closest state " + str(closest_state) + '\n')
347
348             if closest_state:
349                 # State within tolerance exists
350                 weight = closest_state
351             else:
352                 # Create a new state
353                 States.append(weight)
354                 qtable[weight] = {action: 0 for action in range(50, 1050,
355                                         50)}

```

```
352         ballLifterExecution(numberOfBalls, numberOfBalls, counter, flag
353             , weight)
354             for state, inner_dict in qtable.items():
355                 # Create a temporary list to store top 2 elements (key,
356                 value pairs)
357                 top_two = []
358                 # Iterate through each key-value pair in the inner
359                 dictionary
360                 for key, value in inner_dict.items():
361                     # Check if the current element is one of the top 2 (
362                     considering both key and value)
363                     if len(top_two) < 2 or value > top_two[-1][1]:
364                         # If it's one of the top 3, either add it directly
365                         or replace the lowest value
366                         if len(top_two) < 2:
367                             top_two.append((key, value))
368                         else:
369                             top_two[-1] = (key, value)
370                         # Since we only care about the top 2, keep the list
371                         sorted by value (descending)
372                             top_two.sort(key=lambda x: x[1], reverse=True)

373
374             # Print the state and the top 3 elements
375             print(f"State {state}:")
376             for i, (key, value) in enumerate(top_two, 1):
377                 print(f"{i}. Speed: {key}, Q-Value: {value}")
378             wait(20)
```


Βιβλιογραφία

- [1] Christos Alexakos, Andreas Komninos, Christos Anagnostopoulos, George Kalogerias και Athanasios Kalogerias. Iot integration in the manufacturing environment towards industry 4.0 applications. *Στο 2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*, τόμος 1, σελίδες 41–46, 2020.
- [2] What is automationml? URL: <https://www.automationml.org/about-automationml/automationml/>, Visited: February 2024.
- [3] Spike prime sbc balllifter an iot-mqtt demo robot. URL: https://legostudiovives.be/spikeprimesbcballlifter_iot_mqtt_demo/, Visited: February 2024.
- [4] Abishi Chowdhury και Shital A Raut. Benefits, challenges, and opportunities in adoption of industrial iot. *International Journal of Computational Intelligence & IoT*, 2(4), 2019.
- [5] Armando Colombo, Thomas Bangemann, Stamatis Karnouskos, Jerker Delsing, Petr Stluka, Robert Harrison, Francois Jammes και Jose Luis Martinez Lastra. *Industrial cloud-based cyber-physical systems: The IMC-AESOP approach*. 2014.
- [6] Baudouin Dafflon, Nejib Moalla και Yacine Ouzrout. The challenges, approaches, and used techniques of cps for manufacturing in industry 4.0: Aliterature review. *The International Journal of Advanced Manufacturing Technology*, 113:2395–2412, 2021.
- [7] The industrial revolution and work in nineteenth-century europe. URL: <https://web.archive.org/web/20200129200235/https://www.questia.com/read/107622079/the-industrial-revolution-and-work-in-nineteenth-century>, Visited: February 2024.
- [8] Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja και Katarzyna Wasielewska. Semantic interoperability in the internet of things: An overview from the inter-iot perspective. *Journal of Network and Computer Applications*, 81:111–124, 2017.
- [9] Joseph Groot Kormelink, Madalina Drugan και Marco Wiering. Exploration methods for connectionist q-learning in bomberman. 2018.

- [10] Dominique Guinard, Vlad Trifa και Erik Wilde. A resource oriented architecture for the web of things. σελίδες 1 – 8, 2011.
- [11] Shixiang Gu, Timothy P. Lillicrap, Ilya Sutskever και Sergey Levine. Continuous deep q-learning with model-based acceleration. *CoRR*, αβς/1603.00748, 2016.
- [12] N. Jazdi. Cyber physical systems in the context of industry 4.0. Στο *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, σελίδες 1–4, 2014.
- [13] M. Kumar, Benjamin Packer και Daphne Koller. Self-paced learning for latent variable models. Στο *Advances in Neural Information Processing Systems*J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel και A. Culotta, επιμελητές, τόμος 23. Curran Associates, Inc., 2010.
- [14] Georgios Lampropoulos, Kerstin Siakas και Theofylaktos Anastasiadis. Internet of things in the context of industry 4.0: An overview. *International Journal of Entrepreneurial Knowledge*, 7:4–19, 2019.
- [15] In Lee και Kyoochun Lee. The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4):431–440, 2015.
- [16] Arndt Lüder και Nicole Schmidt. Automationml in a nutshell. Στο *Handbuch Industrie 4.0*, 2017.
- [17] Yang Lu. Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6, 2017.
- [18] Τι είναι το mqtt και γιατί το χρειαζόμαστε στο iiot; Περιγραφή του πρωτοκόλλου mqtt. URL: <https://cy.ipc2u.com/articles/articles-and-reviews/ti-einai-to-mqtt-kai-gia-to-chreiazomaste-sto-iiot>, Visited: February 2024.
- [19] History of electricity. URL: <https://www.instituteforenergyresearch.org/history-electricity>, Visited: February 2024.
- [20] Shahryar SOROOSHIAN και Shrikant PANIGRAHI. Impacts of the 4th industrial revolution on industries. *Walailak Journal of Science and Technology (WJST)*, 17(8):903–915, 2020.
- [21] What is thingsboard? URL: <https://thingsboard.io/docs/getting-started-guides/what-is-thingsboard/>, Visited: February 2024.
- [22] Stephan Weyer, Mathias Schmitt, Moritz Ohmer και Dominic Gorecky. Towards industry 4.0 - standardization as the crucial challenge for highly modular, multi-vendor production systems. *IFAC-PapersOnLine*, 48(3):579–584, 2015.
- [23] Hansong Xu, Wei Yu, David Griffith και Nada Golmie. A survey on industrial internet of things: A cyber-physical systems perspective. *IEEE Access*, 6:78238–78259, 2018.

- [24] Xingjie Yu και Huaqun Guo. A survey on iiot security. *Στο 2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, σελίδες 1–5, 2019.
- [25] Jin Zhang, Qing Liu και XiaoHang Han. Dynamic sub-route-based self-adaptive beam search q-learning algorithm for traveling salesman problem. *PLOS ONE*, 18(3):e0283207, 2023.
- [26] H. Zhuge. *The knowledge grid: Toward cyber-physical society, second edition*. 2012.

Συντομογραφίες - Αρχτικόλεξα - - Ακρωνύμια

4IR	4th Industrial Revolution
AI	Artificial Intelligence
API	Application Programming Interface
CoAP	Constrained Application Protocol
DRL	Deep Reinforcement Learning
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IIoT	Industrial Internet of Things
LWT	Last Will and Testament
MQTT	Message Queuing Telemetry Transport
QoS	Quality of Service
RL	Reinforcement Learning
RPC	Remote Procedure Call
SPL	Self-Paced Learning
TGWQL	Threshold-Guided Weighted Q-Learning
UI	User Interface
XML	Extensible Markup Language
ΔΕ	Διπλωματική Εργασία
κ.α.	και άλλα
κ.ο.κ.	και ούτω καθεξής
TN	Τεχνητή Νοημοσύνη

Απόδοση ξενόγλωσσων όρων

administrator	διαχειριστής
alarm	συναγερμός
arts	τέχνες
attestation	πιστοποίηση
attribute	χαρακτηριστικό
broker	διαμεσολαβητής
client-side	πλευρά πελάτη
cloud computing	υπολογιστικό νέφος
configuration	ρύθμιση παραμέτρων
core	πυρήνας
cybersecurity	ασφάλεια κυβερνοχώρου
dashboard	ταμπλό ελέγχου
devices	συσκευές
engineering	μηχανική
interoperability	διαλειτουργικότητα
logistics	επιμελητεία
message	μήνυμα
node	κόμβος
printing	εκτύπωση
profile	προφίλ
publisher	εκδότης
queue	σειρά
rule chain	αλυσίδα κανόνων
rule engine	μηχανή κανόνων
science	επιστήμη
server	διαχομιστής
side-bar	πλαισινή μπάρα
subscriber	συνδρομητής
script	σενάριο
technology	τεχνολογία

