ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΤΩΝ ΚΑΤΕΡΓΑΣΙΩΝ

Σχεδιασμός και Υλοποίηση Ελεγκτών σε Ευέλικτο Σύστημα Κατεργασιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΠΕΤΡΟΠΟΥΛΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

Επιβλέπων : Γ.-Χ. Βοσνιάκος
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2018

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή, κ. Γεώργιο Χ. Βοσνιάκο για την ευκαιρία που μου έδωσε, καθώς και για την βοήθεια του κατά τη διάρκεια εκπόνησης της.

Επίσης, ευχαριστώ τον συμφοιτητή μου Θ. Παπιγγιώτη για τις χρήσιμες συμβουλές του.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για την υπομονή και συμπαράσταση της, όλα αυτά τα χρόνια.

Περίληψη

Σκοπός της διπλωματικής εργασίας είναι, αρχικά η μοντελοποίηση (σχεδιασμός και προσομοίωση), μέσω των δικτύων Petri (PN), του κεντρικού ελεγκτή ενός Ευέλικτου Συστήματος Κατεργασιών (ΕΣΚ) του εργαστηρίου, καθώς και των επιμέρους τοπικών ελεγκτών των σταθμών αυτού. Μετέπειτα, πραγματοποιείται η υλοποίηση των παραπάνω ελεγκτών σε επίπεδο σημάτων εισόδου - εξόδου μέσω κατάλληλων ηλεκτρονικών μονάδων, τους μικρο-ελεγκτές Arduino.

Αρχικά, παρουσιάζεται η γενική δομή και ο εξοπλισμός που συνθέτουν τα ΕΣΚ, οι εφαρμογές και τα οφέλη από τη χρησιμοποίηση τους στην παραγωγική διαδικασία μιας Μονάδας Παραγωγής προϊόντων, καθώς και η ανάγκη ελέγχου τους. Έπειτα, περιγράφεται η σύνθεση του ΕΣΚ και ποιες παραδοχές πρέπει να ισχύουν, ώστε να μοντελοποιηθεί αυτό σε ένα δίκτυο Petri και να μπορεί να εφαρμοστεί κεντρικός έλεγχος από έναν μικρο-ελεγκτή Arduino.

Μετέπειτα, αναπτύσσεται το θεωρητικό υπόβαθρο των δικτύων Petri, με περιγραφή του γραφικού και μαθηματικού μοντέλου που χρησιμοποιεί, τους κανόνες εκτέλεσης των ενεργειών, τις χαρακτηριστικές ιδιότητες δομής και συμπεριφοράς που το διέπουν, τις δυνατότητες μοντελοποίησης και των μεθόδων αναλύσεων του. Επιπλέον, γίνεται αναφορά στη χρησιμότητά τους στον προγραμματισμό συστημάτων διακριτών γεγονότων, ενώ παράλληλα, παρουσιάζονται και ορισμένες επεκτάσεις των κλασσικών δικτύων Petri.

Εν συνεχεία περιγράφεται αναλυτικά η μοντελοποίηση του μελετούμενου συστήματος μέσω των κλασσικών δικτύων Petri. Παρουσιάζεται η εκτέλεση της προσομοίωσης του δικτύου, η γραφική του ανάλυση και η εύρεση των ιδιοτήτων του, τα οποία είναι απαραίτητα για την εφαρμογή ελέγχου που θα ακολουθήσει.

Ακολουθεί μια αναφορά στους μικρο-ελεγκτές τύπου Arduino-Uno και Arduino-Mega και τις δυνατότητες τους στην εφαρμογή ελέγχου. Όπως επίσης, περιγράφονται οι τρόποι επικοινωνίας μεταξύ τους μέσω κατάλληλων πρωτοκόλλων επικοινωνίας σύγχρονης ή ασύγχρονης μετάδοσης δεδομένων. Κατόπιν, γίνεται η εφαρμογή ελέγχου στο υπό μελέτη εξεταζόμενο ΕΣΚ, από έναν κεντρικό και τοπικούς ελεγκτές, με σχέση master-slave, μέσω του διαύλου επικοινωνίας I2C. Εκτελούνται προσομοιώσεις των κωδικών Arduino, που αναπτύχθηκαν και παρατίθενται τα αποτελέσματα τους επί της εφαρμογής ελέγχου.

Τέλος, γίνεται μια συνολική αξιολόγηση των αποτελεσμάτων από τις αναλύσεις και τις προσομοιώσεις που πραγματοποιήθηκαν και αναφέρονται τυχόν επεκτάσεις των εφαρμογών που αναπτύχθηκαν στο πλαίσιο της εργασίας.

Λέξεις Κλειδιά: <<Ευέλικτα Συστήματα, Δίκτυα Petri, Arduino, μικρο-ελεγκτής, I2C, Σειριακή Επικοινωνία>>

Abstract

The aim of the dissertation is, initially the design and simulation of the central and local controllers of a Lab's Flexible Manufacturing System (FMS) and its stations by Petri Nets (PN). Then, it goes on with the implementation of the controllers by receiving and transmitting signals using capable electronic devices, like Arduinos.

Firstly, FMS' general structure and equipment is presented with applications and benefits that are used in the production procedure of goods in a Production Unit, as also the need to control all them together. Afterwards, there is a description for the specific FMS about its composition and the modelling admissions that can convert it to a Petri Net and finally control it by an Arduino microcontroller.

Then, the theory of Petri Nets with their graphic and mathematic model, the rules of engagement, the structure and behavior attributes, modelling abilities and also their analysis methods are described. In addition, the Petri Nets' utility in the discrete event systems modelling is mentioned, while, defined extensions of the classic Petri Nets are presented.

In sequence, there is an analytical description for the procedure modelling of the studying FMS by Petri Nets. Its simulation, graphical analysis and properties are also included, which are essential for control implementation.

A reference to micro-controllers Arduino-Uno and Arduino-Mega2560 and their capability in control implementation is mentioned. As also, their ways of transferring data by synchronous and asynchronous communication protocols. Control is performed in the studying FMS, by a central and local controllers, with master-slave relation, using I2C bus communication. Simulations for the developed Arduinos' programs are executed and their results are included.

At last, there is an evaluation of the results from the performed analysis and simulations, and also potential extensions of dissertation's developed implementations are mentioned.

Λέξεις Κλειδιά: <<Flexible Manufacturing Systems, Petri Nets, Arduino, micro-controller, I2C, Serial Communication>>

Πίνακας Περιεχομένων

Ευχαριστίες **1**

Περίληψη **2**

Abstract **3**

Πίνακας Περιεχομένων **4**

Πίνακας Εικόνων **6**

Πίνακας Πινάκων **8**

Πίνακας Σχημάτων **9**

1 Ευέλικτα Συστήματα Κατεργασιών (ΕΣΚ) **10**

1.1 Συστήματα Παραγωγής 10

1.1.1 Γενικά Στοιχεία – Τύποι 10

1.1.2 Εξέλιξη και Μέθοδοι Οργάνωσης των Συστημάτων Παραγωγής με Η/Υ 14

1.1.3 Κύτταρα (Κυψέλες) Παραγωγής 16

1.2 Εισαγωγή στα Ευέλικτα Συστήματα Κατεργασιών (ΕΣΚ) 19

1.2.1 Ορισμός – Δομή – Ευελιξία 19

1.2.2 Εφαρμογές – Εξοπλισμός – Οφέλη 23

1.2.3 Σχεδιασμός και Έλεγχος 25

2 Παρουσίαση ΕΣΚ Παρούσας Εργασίας **29**

2.1 Γενική Περιγραφή 29

2.2 Σύνθεση – Δομή – Αναλυτικός Εξοπλισμός 31

2.3 Παραδοχές – Παραμετροποίηση 36

2.4 Παραγωγική Διαδικασία 40

3 Δίκτυα Petri (Petri Nets – PN) **42**

3.1 Γενικά Στοιχεία 42

3.2 Ορισμός Μοντελοποίησης – Δομή – Σήμανση 44

3.3 Ιδιότητες – Απλές Δομές Μοντελοποίησης 52

3.4 Μέθοδοι Ανάλυσης 57

3.5 Επιβολή Προτεραιοτήτων – Μηδενικός Έλεγχος 62

3.6 Άλλοι Τύποι – Επεκτάσεις Δικτύων Petri 64

4 Μοντελοποίηση του ΕΣΚ μέσω των Δικτύων Petri **68**

4.1 Δομή – Παρουσίαση του PN 68

4.2 Αντιστοίχιση Καταστάσεων του ΕΣΚ σε Θέσεις του PN 72

4.3 Αντιστοίχιση Γεγονότων του ΕΣΚ σε Μεταβάσεις του PN – Κανόνες Ενεργοποίησης 83

4.4 Προσομοίωση Δικτύου Petri 88

4.4.1 Αρχική Σήμανση 89

4.4.2 Προσομοίωση PN 90

4.4.3 Ιδιότητες PN - Παρατηρήσεις ... 95
5 Μικρο-ελεγκτές Arduino και Σειριακή Επικοινωνία ... 97
5.1 Γενικά Στοιχεία για τους Μικρο-ελεγκτές Arduino ... 97
5.1.1 ArduinoUno ... 99
5.1.2 ArduinoMega2560 ... 103
5.2 Προγραμματισμός ArduinoIDE ... 105
5.3 Επιλογή Ελέγχου με Arduino ... 108
5.3.1 Arduino vs PLC ... 108
5.3.2 Arduino vs RaspberryPi3 ... 109
5.4 Σειριακή Επικοινωνία – I2C ... 112
5.4.1 Τρόποι Επικοινωνίας και Μεταφοράς Δεδομένων ... 112
5.4.2 Inter-Integrated Circuit (I2C) ... 113
5.5 Σύνδεση Arduino – Συσκευής μέσω I2C (Βιβλιοθήκη Wire) ... 114
6 Έλεγχος του ΕΣΚ μέσω Μικρο-ελεγκτών Arduino ... 117
6.1 Έλεγχος Master - Slaves μέσω I2C ... 117
6.1.1 Λογική Ελέγχου ... 118
6.1.2 Εφαρμογή ArduinoMega (master) – 2 ArduinoUno (slaves) ... 120
6.1.3 Πρόγραμμα Ελέγχου με ArduinoMega2560 - 2 ArduinoUno μέσω I2C ... 122
6.2 Προσομοίωση Ελέγχου με ArduinoMega2560 – Διακόπτες – Φωτοδιόδους ... 132
6.2.1 Αιτίες Υλοποίησης ... 132
6.2.2 Προσομοίωση Ελέγχου – Επεξήγηση Προγράμματος ... 134
7 Συμπεράσματα – Επεκτάσεις ... 146
7.1 Συμπεράσματα ... 146
7.2 Μελλοντικές Επεκτάσεις ... 148
8 Βιβλιογραφία ... 149
9 Παραρτήματα ... 151
9.1 Προγράμματα για Σχεδίαση και Προσομοιώσεις ... 151
9.2 Προγράμματα Arduino Master – Slaves μέσω I2C ... 151
9.2.1 Πρόγραμμα ArduinoMega2560 για Master ... 151
9.2.2 Πρόγραμμα ArduinoUno για Slave – Robot (8) ... 153
9.2.3 Πρόγραμμα ArduinoUno για Slave – Mill (9) ... 154
9.3 Πρόγραμμα ArduinoMega2560 – Διακοπτές – Φωτοδιόδοι ... 155

Πίνακας Εικόνων

Εικόνα 1-1 Διάταξη Συστήματος Μαζικής Παραγωγής - Γραμμή Παραγωγής 11
Εικόνα 1-2 Διάταξη Συστήματος Ειδικών Εργασιών - Παραγωγή Μοναδιαίου Προϊόντος 12
Εικόνα 1-3 Σύστημα Παραγωγής σε Παρτίδες (Μερίδες) 13
Εικόνα 1-4 Παραγωγή σε Παρτίδες, Διάταξη κατά Λειτουργία (αριστερά)-σε Κύτταρα (δεξιά) 13
Εικόνα 1-5 Ταξινόμηση Συστημάτων Παραγωγής 14
Εικόνα 1-6 Ομαδοποίηση Τεμαχίων σε Οικογένειες/Ομάδες 15
Εικόνα 1-7 Σύνθετο (Ιδεατό) Τεμάχιο και Απλοποιήσεις του 15
Εικόνα 1-8 Αυτοματοποίηση και Εφαρμογή Ελέγχου στα Συστήματα Κατεργασιών 20
Εικόνα 1-9 Επίπεδα Αυτοματοποίησης στα Συστήματα Κατεργασιών 21
Εικόνα 1-10 Στάδια Κύκλου Ζωής ενός Προϊόντος 22
Εικόνα 1-11 Τυπικό Ευέλικτο Κύτταρο Παραγωγής Μπλοκ Κινητήρων 23
Εικόνα 1-12 Επίπεδα Ιεραρχικού Ελέγχου στα ΕΣΚ[7] 26
Εικόνα 1-13 Σχεδιασμός εκάστου Επιπέδου Ελέγχου[7] 27
Εικόνα 2-1 Κύτταρο Κατεργασιών Εργαστηρίου 30
Εικόνα 2-2 ΟΑΠ (AGV) σε Βιομηχανικό Περιβάλλον 34
Εικόνα 2-3 Ροπμότ του ΕΣΚ 34
Εικόνα 3-1 Δομικά Στοιχεία των Δικτύων Petri 44
Εικόνα 3-2 Δομή του πιο απλού Δικτύου Petri 45
Εικόνα 3-3 Δίκτυο Petri Πριν (αριστερά) και Μετά (δεξιά) την ενεργοποίηση μετάβασης (firing) 45
Εικόνα 3-4 Ενεργοποίηση Μετάβασης (firing) μέσω κλάδων με βάρος >1 (weighted arc) 46
Εικόνα 3-5 Γραφική Δομή - Σήμανση - Ροή κουπονιών ενός Δικτύου Petri [8] 46
Εικόνα 3-6 Αρχική Σήμανση ενός Δικτύου Petri[17] 48
Εικόνα 3-7 Primitive Structures των Δικτύων Petri 56
Εικόνα 3-8 Παράδειγμα δημιουργίας Πεπερασμένου Δένδρου Προσβασιμότητας 59
Εικόνα 3-9 Hierarchical Δίκτυα Petri 66
Εικόνα 3-10 Αλγόριθμος λειτουργίας των Signal Interpreted Nets (Frey-2000)[20] 67
Εικόνα 4-1 Δομή του ΕΣΚ 68
Εικόνα 4-2 Ομαδοποίηση των Σταθμών του ΕΣΚ 70
Εικόνα 5-1 Περιβάλλον Προγραμματισμού του Arduino – ArduinoIDE 97
Εικόνα 5-2 Τροφοδοσία Arduino μέσω συσσωρευτών 98
Εικόνα 5-3 Πλακέτα ArduinoUnoRev3 [24] 99
Εικόνα 5-4 Pinmaping του Επεξεργαστή ATmega328 του ArduinoUno[24] 99
Εικόνα 5-5 Περιληπτική Αντιστοίχιση Θυρών του ArduinoUno 101
Εικόνα 5-6 Αναλυτική Περιγραφή των Θυρών του ArduinoUno 101
Εικόνα 5-7 PinMapping με όλες τις λειτουργίες των θυρών του ArduinoUno 103
Εικόνα 5-8 Πλακέτα Arduino Mega2560Rev3 103
Εικόνα 5-9 Αναλυτική Περιγραφή των Θυρών του ArduinoMega2560 105
Εικόνα 5-10 Μοντέλα PLC της Siemens [26] 108
Εικόνα 5-11 Δομή Λειτουργίας ενός PLC [26] 109
Εικόνα 5-12 RaspberryPi3 και τα I/O pins του 110
Εικόνα 5-13 Σύγκριση Χαρακτηριστικών RaspberryPi και Arduino-Uno [29] 111
Εικόνα 5-14 Micro-controller vs Micro-processor [29] 112
Εικόνα 5-15 Σύνδεση μέσω UART [27] 113
Εικόνα 5-16 Σύνδεση Συσκευών μέσω I2C 114
Εικόνα 6-1 Φόρτωση βιβλιοθήκης Wire() 123
Εικόνα 6-2 Master Code Μεταβλητές των slaves για if conditions 123
Εικόνα 6-3 Master Code Έναρξη Wire και Serial 124

Εικόνα 6-4 Master Code request διαθεσιμότητα από slaves 124
Εικόνα 6-5 Master Code Έλεγχος Ικανοποίησης if condition και εντολές αυτής 125
Εικόνα 6-6 Master Code Έναρξη Μετάδοσης προς slaves για δέσμευση τους 126
Εικόνα 6-7 Slave Code Φόρτωση Wire - Ορισμός ταυτότητας - Αρχικοποίηση μεταβλητής 126
Εικόνα 6-8 Slave Code Εκκίνηση I2C σύνδεσης - Καθορισμός συναρτήσεων στο void setup() 127
Εικόνα 6-9 Slave Code void loop() σε Slave device 128
Εικόνα 6-10 Slave Code Υπορουτίνα requestEvent για μετάδοση data από τον slave στον master 128
Εικόνα 6-11 Slave Code Υπορουτίνα receiveEvent για μετάδοση data από τον master στον slave 129
Εικόνα 6-12 Εκτυπώσεις στη Σειριακή των 3 Arduinos με τη σειρά Master-Slave(Mill)-Slave(Robot) 129
Εικόνα 6-13 Hardware Διάταξη της I2C επικοινωνίας master (ArduinoMega) - 2 slaves (ArduinoUno) 130
Εικόνα 6-14 Δέσμευση και των δύο slaves-τοπικών ελεγκτών 131
Εικόνα 6-15 Hardware της υλοποίησης Ελέγχου με ένα Κεντρικό Ελεγκτή - Διακόπτες – Leds 135
Εικόνα 6-16 Αντιστοίχιση Διακοπτών/Εισόδου σε Pins Εισόδου και Leds 136
Εικόνα 6-17 Βοηθητικές Μεταβλητές εξασφάλισης ορθής συνέχειας των Διεργασιών 137
Εικόνα 6-18 Επεξήγηση Προγράμματος void setup() 137
Εικόνα 6-19 Λειτουργία του countePallet 139
Εικόνα 6-20 Λειτουργία του counterUnit 139
Εικόνα 6-21Λειτουργία των endofPallets και enterUnits 139
Εικόνα 6-22 Εισαγωγή στη διεργασία μεταφοράς παλέτας από Pallet_BufferMillIn στο ID_pos 140
Εικόνα 6-23 Έλεγχος λάθος παλέτας (1) 140
Εικόνα 6-24 Έλεγχος λάθος παλέτας (2) 141
Εικόνα 6-25 Έλεγχος λάθος παλέτας (3) 141
Εικόνα 6-26 Εισαγωγή Προγράμματος στο void loop()- Προσομοίωση Προγράμματος 142
Εικόνα 6-27 Ικανοποίηση 1ης Συνθήκης – Προσομοίωση Προγράμματος 142
Εικόνα 6-28 Προσομοίωση Ελεγκτή (1) 143
Εικόνα 6-29 Προσομοίωση Ελεγκτή (2) 144
Εικόνα 6-30 Προσομοίωση Ελεγκτή (3) 144
Εικόνα 6-31 Εκτυπώσεις στην Σειριακή κλείνοντας τον βρόχο (loop) στην Προσομοίωση 145

Πίνακας Πινάκων

Πίνακας 2-1 Σύνθεση του ΕΣΚ 32
Πίνακας 2-2 Χωρητικότητες των Σταθμών του ΕΣΚ 39
Πίνακας 3-1 Πίνακας Αντιστοίχισης Συστήματος - Μοντελοποίησης στα Δίκτυα Petri[15] 51
Πίνακας 3-2 Αντιστοίχιση Θέσεων - Εργασιών/Πόρων στο παράδειγμα αποτρεπτικού κλάδου 63
Πίνακας 4-1 Πίνακας Θέσεων του PN με Χωρητικότητες και Ερμηνείες των κουπονιών σε αυτές 82
Πίνακας 4-2 Πίνακας Μεταβάσεων του PN με κλάδους εισόδου/εξόδου από/προς Θέσεις 88
Πίνακας 4-3 Αρχική Σήμανση (m_0) του PN 90
Πίνακας 5-1 Τεχνικά Χαρακτηριστικά του ArduinoUno [24] 100
Πίνακας 5-2 Τεχνικά Χαρακτηριστικά ArduinoMega2560 [24] 104
Πίνακας 5-3 Χαρακτηριστικά RaspberrPi 3 110
Πίνακας 5-4 Πλεονεκτήματα - Μειονεκτήματα RaspberryPi - Arduino[28] 111
Πίνακας 5-5 Θύρες σύνδεσης I2C στα Arduino Mega και Uno 114
Πίνακας 6-1 Εξαρτήματα I2C σύνδεσης 121
Πίνακας 6-2 Πίνακας Αντιστοιχίας Σημάτων I/O μεταξύ των υλοποιήσεων Ελέγχου 133
Πίνακας 6-3 Αντιστοιχία Διακοπτών/Εισόδων (Σταθμοί) σε Pins του Arduino 136
Πίνακας 7-1 Αντιστοίχιση Εισόδων - Εξόδων Προγραμμάτων Arduino I2C - Swithes/Leds 147

Πίνακας Σχημάτων

Σχήμα 2-1 Δομή του ΕΣΚ 31
Σχήμα 2-2 Ομαδοποίηση των Σταθμών του ΕΣΚ 33
Σχήμα 2-3 Παράδειγμα Παραγωγικής Διαδικασίας του ΕΣΚ 41
Σχήμα 3-1 Δομή και Σήμανση PN (k=0) 50
Σχήμα 3-2 Δομή και Σήμανση PN (k=1) 50
Σχήμα 3-3 Δομή και Σήμανση PN (k=2) 51
Σχήμα 3-4 Παράδειγμα μη-φραγμένου PN[4] 54
Σχήμα 3-5 Παράδειγμα Μηχανής Κατάστασης[4] 56
Σχήμα 3-6 Παράδειγμα PN που καταλήγει σε αδιέξοδο 58
Σχήμα 3-7 Δένδρο Προσβασιμότητας που καταλήγει σε αδιέξοδο 58
Σχήμα 3-8 Εύρεση P- και T- Αναλλοίωτων μέσω του λογισμικού pipe4.3.0 62
Σχήμα 3-9 Παράδειγμα χρήσης Αποτρεπτικού Κλάδου σε PN 64
Σχήμα 3-10 Αντικατάσταση του αποτρεπτικού κλάδου από μια θέση και μια μετάβαση 64
Σχήμα 4-1 Δίκτυο Petri του ΕΣΚ 69
Σχήμα 4-2 Ομαδοποίηση των Θέσεων (Καταστάσεων) του PN 70
Σχήμα 4-3 Τμήμα του PN για τα Εργαλεία (Tools) 71
Σχήμα 4-4 Τμήμα του PN για τις Παλέτες – Υπο-τμήμα 1 (Pallets) 71
Σχήμα 4-5 Τμήμα του PN για τις Παλέτες - Υπο-τμήμα 2 (Pallets) 71
Σχήμα 4-6 Τμήμα του PN για τις Κοινές Θέσεις (Common) 71
Σχήμα 4-7 Τμήμα του PN για τα Τεμάχια - Υποτμήμα 1 (Units) 72
Σχήμα 4-8 Τμήμα του PN για τα Τεμάχια - Υποτμήμα 2 (Units) 72
Σχήμα 4-9 Αρχική ενεργοποίηση του PN (1) 90
Σχήμα 4-10 Αποτρεπτικοί κλάδοι για επιβολή προτεραιότητας στα εργαλεία έναντι των παλετών (2) 91
Σχήμα 4-11 Εκκίνηση διαδικασιών για παλέτες (3) 91
Σχήμα 4-12 Φόρτωση παλέτας μέσω Robot στο ID_position+Στατιον και μετά στο Mill (4) 92
Σχήμα 4-13 Χρήση Αποτρεπτικού κλάδου για μη περαιτέρω διεργασία λανθασμένης παλέτας (5) 92
Σχήμα 4-14 Απογέμιση της Αποθήκης ετοίμων παλετών μόλις γεμίσει, μέσω ψευδο-θέσης (6) 93
Σχήμα 4-15 Αποτρεπτικοί κλάδοι για έναρξη τεμαχίων μετά την έναρξη τελευταίας παλέτας (7) 93
Σχήμα 4-16 Ολοκλήρωση $1^{ου}$ τεμαχίου έναντι ολοκλήρωσης $1^{ης}$ παλέτας (8) 94
Σχήμα 4-17 Χρήση Αποτρεπτικού κλάδου για μη περαιτέρω διεργασία λανθασμένου τεμαχίου (9) 95
Σχήμα 4-18 Απογέμιση της Αποθήκης ετοίμων τεμαχίων μόλις γεμίσει, μέσω ψευδο-θέσης (10) 95
Σχήμα 5-1 Σύνδεση δύο Arduino μέσω I2C 115
Σχήμα 6-1 ΕΣΚ με Κεντρικό Ελεγκτή (Arduino-Mega) και Τοπικούς Ελεγκτές (Arduino-Uno) 117
Σχήμα 6-2 Διάγραμμα Ροής της Λογικής Ελέγχου που εφαρμόστηκε στο ΕΣΚ 119
Σχήμα 6-3 Συνδεσμολογία I2C με 1 ArdulinoMega και 2 ArduinoUno 121
Σχήμα 6-4 Ηλεκτρονικό Σχέδιο I2C Κυκλώματος 122
Σχήμα 6-5 I2C Επικοινωνία μεταξύ των 3 Arduino με παροχή τάσης 5V - χωρίς σύνδεση USB 131
Σχήμα 6-6 Δομή της υλοποίησης Ελέγχου με ένα Κεντρικό Ελεγκτή - Διακόπτες - Leds 134
Σχήμα 7-1 Εξαγωγή Εντολών Προγράμματος από τους Κανόνες Ενεργοποίησης του PN 147

1

Ευέλικτα Συστήματα Κατεργασιών (ΕΣΚ)

1.1 Συστήματα Παραγωγής

1.1.1 Γενικά Στοιχεία – Τύποι

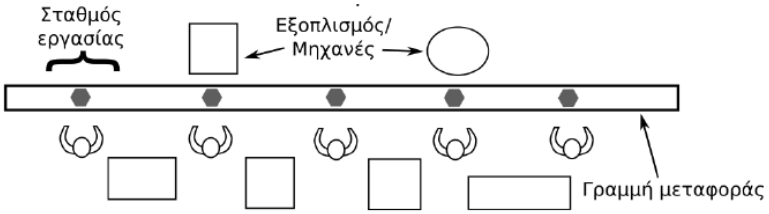
Η οργάνωση της παραγωγικής διαδικασίας είναι συνήθως συνάρτηση του επιδιωκόμενου όγκου παραγωγής και της επιδιωκόμενης ποικιλίας προϊόντων, μεταξύ των οποίων υπάρχει συνήθως μια αντίστροφη σχέση, με αποτέλεσμα η οικονομική παραγωγή μεγάλης ποσότητας προϊόντων να επιβάλει συνήθως μικρή ποικιλία προϊόντων, ενώ αντίστροφα η μεγάλη ποικιλία προϊόντων συνδέεται με μικρούς σχετικά όγκους παραγωγής.

Κατ' επέκταση, προέκυψε η ταξινόμηση των διάφορων κλάδων μεταποίησης σε δύο κύριες κατηγορίες: τους κλάδους που παράγουν *διακριτές μονάδες προϊόντων (discrete product industries)* και τους κλάδους *παραγωγής μέσω διεργασιών (process industries)*. Βασικό χαρακτηριστικό της παραγωγής διακριτών μονάδων (discrete manufacturing) είναι ότι περιλαμβάνει κατά κύριο λόγο εργασίες μορφοποίησης των υλικών και συναρμολόγησης εξαρτημάτων για την κατασκευή πιο σύνθετων προϊόντων. Αντίστοιχα, ο κλάδος των διεργασιών (process industry) αφορά συνήθως την παρασκευή μεγάλης ποσότητας και όγκου προϊόντος, η οποία στη συνέχεια διαμοιράζεται/συσκευάζεται σε διακριτές μονάδες για διανομή και κατανάλωση (φιάλες, δοχεία, κάψουλες, χάπια κλπ.).[1]

Στην προκειμένη περίπτωση θα μας απασχολήσει η παραγωγική μονάδα διακριτών προϊόντων, στην οποία η οργάνωση και ο τεχνολογικός εξοπλισμός εξαρτώνται σε μεγάλο βαθμό από το είδος προϊόντων που παράγει και ειδικότερα από την ποικιλία αυτών και τις αντίστοιχες ποσότητες παραγωγής. Από οικονομική άποψη, παρατηρείται συνήθως μια αντιστρόφως ανάλογη σχέση μεταξύ της ποικιλίας και της ποσότητας, όπου η μεγάλη ποσότητα παραγωγής είναι συνήθως οικονομικά αποτελεσματική για μικρή ποικιλία προϊόντων. Αυτό έχει προφανείς συνέπειες στην οργάνωση και τη λειτουργία μιας παραγωγικής μονάδας, η οποία σχεδιάζεται και εξειδικεύεται σε έναν οικονομικά αποδοτικό συνδυασμό ποικιλίας και ποσότητας παραγωγής. Στο πλαίσιο αυτό διακρίνονται τρεις τύποι συστημάτων παραγωγής διακριτών προϊόντων, όπως παρακάτω:

- Μαζικής παραγωγής (mass production)
- Παραγωγής σε παρτίδες (batch production)
- Ειδικών εργασιών (job-shop production)

Μια *μονάδα μαζικής παραγωγής (mass production)* παράγει σχετικά μικρή ποικιλία προϊόντων (τυποποιημένα προϊόντα) σε μεγάλους αριθμούς (όγκους) παραγωγής, με χαρακτηριστικό παράδειγμα τέτοιου τύπου συστήματος την αυτοκινητοβιομηχανία (Εικόνα 1-1). Σε ένα σύστημα μαζικής παραγωγής χρησιμοποιούνται εκτεταμένα μηχανές ειδικής χρήσης και υπάρχει σχετικά υψηλό επίπεδο αυτοματοποίησης (π.χ. αυτόματοι ταινιόδρομοι, αυτόματες μηχανές λήψης και τοποθέτησης υλικών κλπ.). Η χωροταξία του συστήματος ακολουθεί το πρότυπο της γραμμής παραγωγής (επαναληπτική παραγωγή), σύμφωνα με το οποίο κάθε μονάδα προϊόντος διέρχεται από σειρά στενά συνδεδεμένων σταθμών εργασίας, οι οποίοι τοποθετούνται γραμμικά ο ένας μετά τον άλλον, σύμφωνα με τα στάδια της παραγωγικής διαδικασίας ενός ή λίγων προϊόντων. Ο συγκεκριμένος τύπος χωροταξίας, γνωστός και ως χωροταξία προϊόντος (product layout), ελαχιστοποιεί τον απαιτούμενο, για τη μεταφορά των υλικών, χρόνο, καθώς και τον αριθμό των μονάδων υπό επεξεργασία (ημικατεργασμένα προϊόντα, ενδιάμεσα αποθέματα). Επίσης, η εξειδίκευση στην παραγωγή σχετικά μικρής ποικιλίας προϊόντων απλοποιεί, σημαντικά τη διαχείριση και τον έλεγχο της παραγωγής. Έτσι με τη χρήση εξειδικευμένου εξοπλισμού και τη βέλτιστη εκμετάλλευση χώρου, χρόνου και υλικών γίνεται δυνατή η παραγωγή μεγάλων ποσοτήτων με σχετικά χαμηλό κόστος παραγωγής ανά μονάδα προϊόντος, επιτυγχάνοντας σημαντικές οικονομίες κλίμακας.



Εικόνα 1-1 Διάταξη Συστήματος Μαζικής Παραγωγής - Γραμμή Παραγωγής

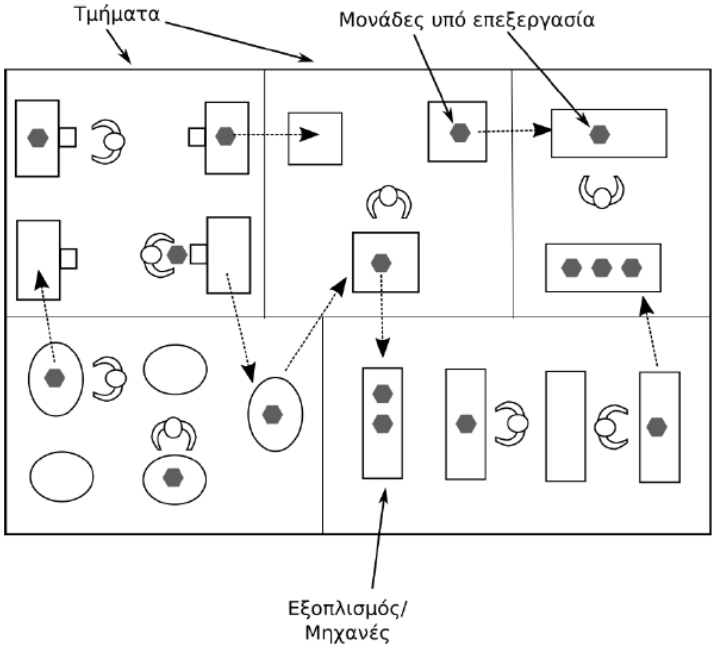
Στο άλλο άκρο του «δίπολου» ποικιλία – ποσότητα παραγωγής, έχουμε το *σύστημα ειδικών εργασιών (job-shop production)* (Εικόνα 1-2), στο οποίο παράγεται υψηλή ποικιλία προϊόντων σε σχετικά μικρές ποσότητες. Το προϊόν που παράγεται είναι διαφορετικό για κάθε πελάτη και ολοκληρωμένο, γι' αυτό και το συγκεκριμένο σύστημα παραγωγής ονομάζεται και *μοναδιαίου προϊόντος (one-off ή make complete)*.

Εδώ, διακρίνονται οι εξής δύο υποπεριπτώσεις, αναλόγως την χρησιμοποιούμενη τεχνολογία:

- Χαμηλή τεχνολογία: τετριμμένη οργάνωση - προσωπικό πολλαπλής ειδίκευσης, πχ μηχανουργείο.
- Υψηλή τεχνολογία: διαχείριση έργων (PM), βέλτιστη εκμετάλλευση ειδικοτήτων, πχ συναρμολόγηση αεροσκαφών.

Στην περίπτωση αυτή, χρησιμοποιείται εξοπλισμός γενικού σκοπού ενώ και το επίπεδο αυτοματοποίησης είναι σχετικά χαμηλό (περιορίζεται κυρίως στο επίπεδο μηχανής). Η χωροταξική οργάνωση του εξοπλισμού ακολουθεί το πρότυπο της *χωροταξίας διαδικασίας ή κατεργασίας (process layout)*, στην οποία μηχανές/εξοπλισμός που επιτελεί τον ίδιο τύπο κατεργασίας τοποθετούνται μαζί. Ο συγκεκριμένος τύπος χωροταξίας, σε συνδυασμό με την χρήση μηχανών γενικού σκοπού, επιτρέπει μεγαλύτερο βαθμό ευελιξίας

στη διαδικασία παραγωγής, άρα και μεγαλύτερη ποικιλία προϊόντων. Συνέπεια, ωστόσο, των παραπάνω είναι ο σχετικά μεγάλος χρόνος παραγωγής, λόγω του μεγάλου απαιτούμενου χρόνου για τον προγραμματισμό και την προετοιμασία των μηχανών, κατά την αλλαγή μεταξύ διαφορετικών προϊόντων, αλλά και η σχετική δυσκολία στον προγραμματισμό και τη διαχείριση της παραγωγής (συνήθως σε μορφή παραγγελιών). Επιπλέον, στα συστήματα ειδικών εργασιών παρατηρείται συχνά, σχετικά μεγάλος αριθμός *μονάδων υπό επεξεργασία (Work In Progress – WIP)* και συνεπώς χαμηλή ταχύτητα διεκπεραίωσης παραγγελιών. Κατόπιν αυτών, το κόστος εργασίας είναι σχετικά υψηλό, καθώς απαιτούνται εξειδικευμένοι εργαζόμενοι για τον προγραμματισμό και έλεγχο των μηχανών, αλλά και τη διαχείριση παραγωγής. Όλα τα παραπάνω έχουν ως συνέπεια, σχετικά υψηλό κόστος και χρόνο παραγωγής ανά τεμάχιο.

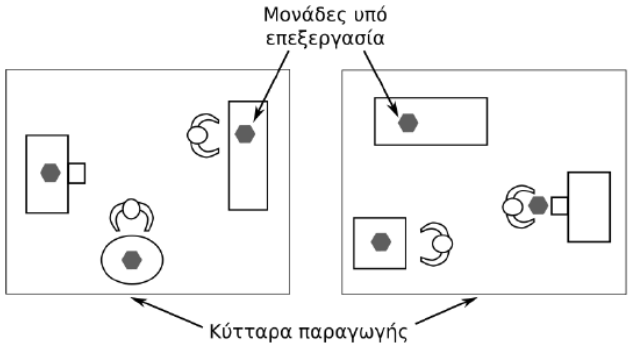


Εικόνα 1-2 Διάταξη Συστήματος Ειδικών Εργασιών - Παραγωγή Μοναδιαίου Προϊόντος

Στο μέσο επίπεδο παραγωγής και ποικιλίας προϊόντων έχουμε το *σύστημα παραγωγής σε παρτίδες (ή μερίδες)* (Εικόνα 1-3), ο οποίος συνδυάζει στοιχεία από τους άλλους δύο τύπους παραγωγικών συστημάτων. Αναλόγως του εξοπλισμού, της οργάνωσης και του βαθμού διαφοροποίησης μεταξύ των προϊόντων μπορεί να έχει περισσότερα κοινά χαρακτηριστικά με ένα σύστημα μαζικής παραγωγής ή με ένα ειδικών εργασιών. Εάν τα προϊόντα παρουσιάζουν σημαντικές διαφορές μεταξύ τους τότε έχουμε έντονη («σκληρή»/“hard”) ποικιλία, τότε το σύστημα προσομοιάζει περισσότερο ένα σύστημα ειδικών εργασιών. Αντίθετα, εάν οι διαφορές των προϊόντων δεν είναι πολύ μεγάλες έχουμε ήπια («μαλακή»/“soft”) ποικιλία και το σύστημα διαθέτει περισσότερα κοινά χαρακτηριστικά με ένα μαζικής παραγωγής. Γενικά, στα συστήματα παραγωγής σε παρτίδες υφίστανται τα εξής χαρακτηριστικά:

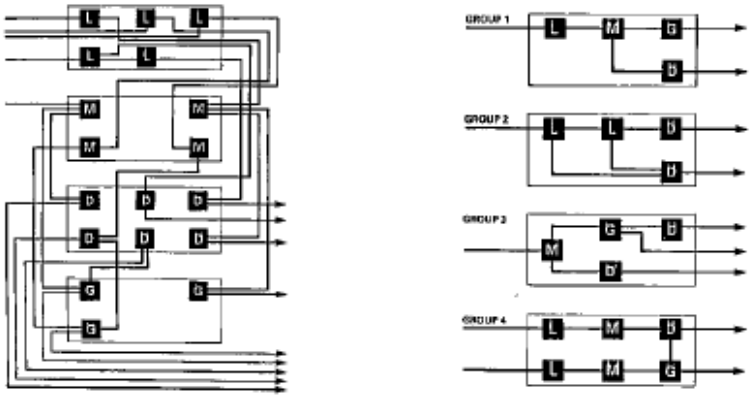
- Η μέση ποικιλία και ο χαμηλός όγκος παραγωγής δεν δικαιολογούν αποκλειστικό εξοπλισμό.
- Υπάρχει ανταγωνισμός μερίδων για τους πόρους παραγωγής άρα και δημιουργία ουρών (WIP).
- Η χρήση πόρων και το μήκος των ουρών εξισορροπούνται μέσω σχεδιασμού και ελέγχου της παραγωγής.

Ανάλογα με το βαθμό διαφοροποίησης στην ποικιλία των προϊόντων, προσαρμόζεται ο βαθμός αυτοματοποίησης και η χρήση εξοπλισμού γενικού σκοπού, που μπορεί να προσαρμοστεί για την παραγωγή διαφορετικών προϊόντων . Η εκμετάλλευση της ψηφιακής τεχνολογίας στα συστήματα παραγωγής σε παρτίδες οδήγησε στην ανάπτυξη των *Ευέλικτων Συστημάτων Παραγωγής – ΕΣΚ (Flexible Manufacturing Systems – FMS)*, τα οποία αποτελούν το κύριο αντικείμενο εξέτασης της υπόψη εργασίας.



Εικόνα 1-3 Σύστημα Παραγωγής σε Παρτίδες (Μερίδες)

Η χωροταξία ενός συστήματος παραγωγής σε παρτίδες ακολουθεί κατά κύριο λόγο δύο διατάξεις (Εικόνα 1-4), την *διάταξη κατά λειτουργία* και την διάταξη *κατά κύτταρα (κυψέλες)*. Στην διάταξη κατά λειτουργία η σχετικά μικρή ποικιλία επιτρέπει τη γρήγορη αλλαγή μεταξύ παρτίδων διαφορετικών προϊόντων, καθότι υπάρχει ευελιξία εκτέλεσης διεργασιών, αλλά δημιουργείται σύνθετη ροή υλικών με επακόλουθο τον μεγάλο χρόνο ολοκλήρωσης και την αύξηση της πολυπλοκότητας προγραμματισμού και ελέγχου της παραγωγής. Στην διάταξη σε *κύτταρα παραγωγής (production cells)*, τα οποία αποτελούνται από διάφορες μηχανές και εξοπλισμό, που συνδέονται με μια ομάδα σχετικά όμοιων προϊόντων (οικογένεια προϊόντων) και διαθέτουν μικρή ευελιξία εκτέλεσης διεργασιών, αλλά έχουν απλή ροή υλικών και μικρό χρόνο ολοκλήρωσης.

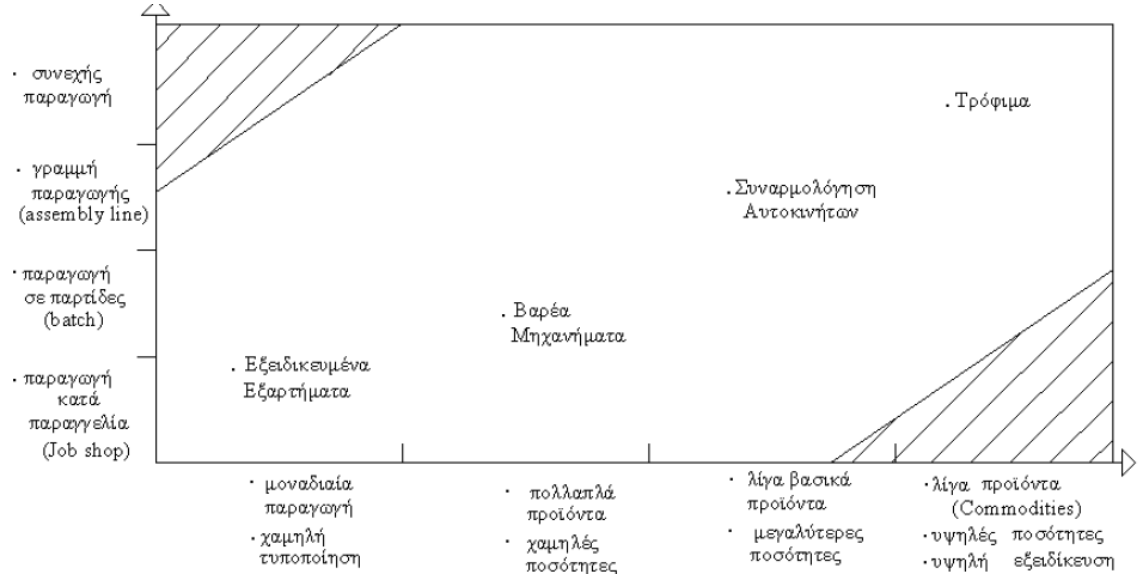


Εικόνα 1-4 Παραγωγή σε Παρτίδες, Διάταξη κατά Λειτουργία (αριστερά)-σε Κύτταρα (δεξιά)

Σε ένα σύστημα παραγωγής σε παρτίδες μπορεί κανείς να δει να παράγονται ταυτόχρονα αρκετά διαφορετικά προϊόντα σε διαφορετικά μεγέθη παρτίδων, γεγονός που δυσκολεύει πολλές φορές τον προγραμματισμό και έλεγχο του συστήματος. Ο αριθμός των μονάδων υπό επεξεργασία και των ενδιάμεσων αποθεμάτων, καθώς και οι χρόνοι διεκπεραίωσης και κύκλου είναι προφανώς μεγαλύτεροι από ενός συστήματος μαζικής παραγωγής, αλλά μικρότεροι από ένα σύστημα ειδικών εργασιών. Όπως είναι προφανές, το κόστος παραγωγής ανά μονάδα προϊόντος σε ένα σύστημα παραγωγής κατά παρτίδες είναι

επίσης κάπου ενδιάμεσα, μεγαλύτερο από το αντίστοιχο σε ένα σύστημα μαζικής παραγωγής και μικρότερο από ότι σε ένα σύστημα ειδικών εργασιών.

Τέλος, αξίζει να σημειωθεί πως δεν υπάρχει εξιδανικευμένο σύστημα παραγωγής που να καλύπτει επαρκώς όλες τις παραγωγικές διαδικασίες, αλλά θα πρέπει να προσαρμόζεται αναλόγως της ζήτησης και του είδους των παραγόμενων προϊόντων, των παραγωγικών δυνατοτήτων και της τεχνικής εξειδίκευσης των εργαζομένων (Εικόνα 1-5).



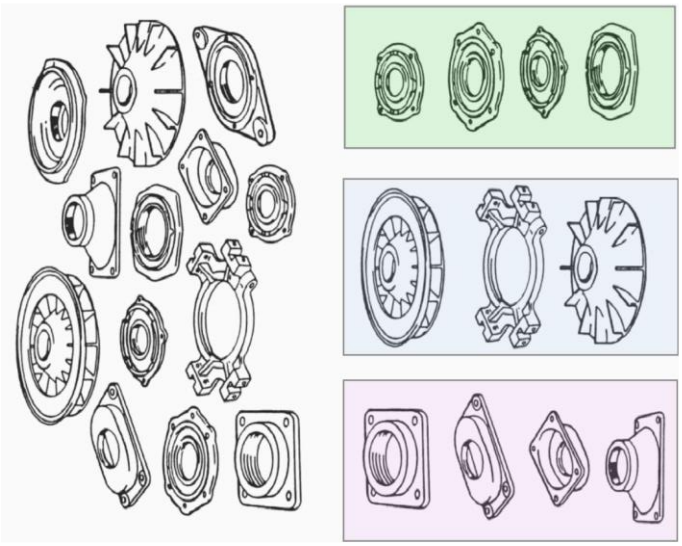
Εικόνα 1-5 Ταξινόμηση Συστημάτων Παραγωγής

1.1.2 Εξέλιξη και Μέθοδοι Οργάνωσης των Συστημάτων Παραγωγής με Η/Υ

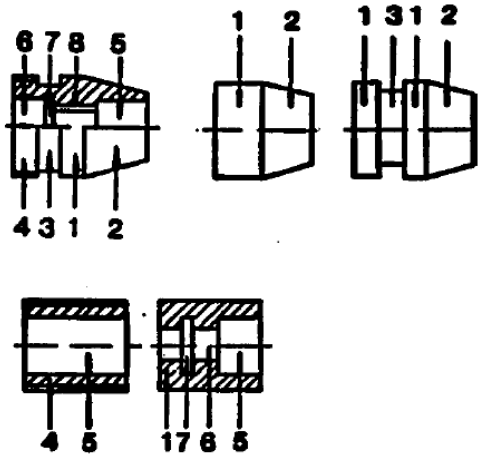
Βασική μονάδα οργάνωσης στην οργανωτική διάσταση των τεχνολογιών κατασκευής με Η/Υ και τα σχετικά θέματα σχεδιασμού και διαχείρισης παραγωγικών συστημάτων, που ενσωματώνουν τις σχετικές τεχνολογίες, αποτελούν, σε ένα πρώτο επίπεδο, το *Κύτταρο Παραγωγής (Manufacturing Cell)* και σε ένα υψηλότερο επίπεδο, το *Ευέλικτο Σύστημα Παραγωγής - ΕΣΚ (Flexible Manufacturing System - FMS).* Βασικό εργαλείο οργάνωσης και σχεδιασμού των συστημάτων αυτών αποτελεί η μεθοδολογία της *Τεχνολογίας Ομάδων (Group Technology - GT).*[1]

Η υπόψη μεθοδολογία αναπτύχθηκε κυρίως τις δεκαετίες μετά το Β' Παγκόσμιο πόλεμο με βάση τις μελέτες του S.P. Mitrofanov στη Σοβιετική Ένωση και άλλων επιστημόνων όπως οι G.M. Ranson, V. A. Petrov, και E. K. Ivanov. Η ανάπτυξη της σχετικής τεχνολογίας οφείλεται κατά κύριο λόγο στην αυξανόμενη πίεση της αγοράς για περισσότερη ευελιξία στην ποικιλία προϊόντων και όγκου παραγωγής αλλά και για μείωση του κόστους και της ταχύτητας παραγωγής. Η Τεχνολογία Ομάδων (Group Technology - GT) βοηθά στην αύξηση της αποτελεσματικότητας στην παραγωγή σε παρτίδες, μέσω της μείωσης του χρόνου, που δαπανάται για την προετοιμασία των μηχανών και του εξοπλισμού (set-up time), αλλά και του χρόνου που σχετίζεται με τη μεταφορά και την αναμονή των υλικών και των ημι-επεξεργασμένων μονάδων (work in progress – WIP) μεταξύ των διάφορων σταδίων της διαδικασίας παραγωγής.

Η *Τεχνολογία Ομάδων (Group Technology - GT)* αποτελεί μία μέθοδο οργάνωσης της παραγωγής διακριτών προϊόντων σε παρτίδες, κατά την οποία τα τεμάχια/προϊόντα ομαδοποιούνται σε *ομάδες/οικογένειες (groups/families)* με βάση ομοιότητες ή κοινά χαρακτηριστικά που παρουσιάζουν τόσο στη μορφή/σχέδιο/γεωμετρία, όσο και στην κατασκευή/φασεολόγιο (Εικόνα 1-6). Η ομαδοποίηση αυτή είναι δυνατή, διότι πολλά προϊόντα αποτελούν ουσιαστικά παραλλαγές ενός σύνθετου - αντιπροσωπευτικού τεμαχίου, το οποίο ονομάζεται ιδεατό τεμάχιο (Εικόνα 1-7) και συγκεντρώνει όλα τα χαρακτηριστικά της οικογένειας, με κάθε επιμέρους τεμάχιο της οικογένειας να προκύπτει μέσω απλοποιήσεων του σύνθετου τεμαχίου.[2]



Εικόνα 1-6 Ομαδοποίηση Τεμαχίων σε Οικογένειες/Ομάδες



Εικόνα 1-7 Σύνθετο (Ιδεατό) Τεμάχιο και Απλοποιήσεις του

Τα κοινά χαρακτηριστικά των τεμαχίων συνεπάγονται και κοινές μεθόδους κατεργασίας, οπότε επαγωγικά προκύπτει ότι είναι πιο αποτελεσματικό να ομαδοποιηθεί οργανωτικά και χωροταξικά και ο σχετικός απαιτούμενος εξοπλισμός, με απώτερο στόχο να μειωθεί ο χρόνος παραγωγής και τα ενδιάμεσα αποθέματα. Οι οργανωτικές ομάδες μηχανών και εξοπλισμού περιγράφονται συχνά με τον όρο *Κύτταρα Παραγωγής (Production Cells)* τα οποία συνδέονται με την παραγωγή μιας συγκεκριμένης οικογένειας προϊόντων. Η σχετική μέθοδος οργάνωσης της παραγωγής αναφέρεται συχνά με τον όρο «*Παραγωγή σε Κύτταρα*» *(Cellular Manufacturing)* και έχει οφέλη, από άποψη διαχείρισης της παραγωγής, καθώς

απλοποιεί τον προγραμματισμό και της διαχείρισης υλικών και μηχανών. Όπως σε κάθε άλλο, έτσι και σε αυτό το πεδίο η ανάπτυξη των ηλεκτρονικών υπολογιστών έπαιξε καθοριστικό ρόλο διευκολύνοντας την ομαδοποίηση καθώς και τον προγραμματισμό/ έλεγχο της παραγωγής.

Συνοπτικά, τα επιδιωκόμενα οφέλη από την εφαρμογή της Τεχνολογίας Ομάδων . είναι τα παρακάτω: [1]

- Τυποποίηση στα εργαλεία παραγωγής και τις διατάξεις συγκράτησης, που συνεπάγεται μείωση του αντίστοιχου κόστους.
- Μείωση των αποστάσεων μεταξύ των μηχανών και κατά συνέπεια η ενέργεια/χρόνος που καταναλώνεται στη μεταφορά υλικών.
- Απλοποίηση της διαδικασίας προγραμματισμού και ελέγχου της παραγωγής με αποτέλεσμα τη μείωση του αντίστοιχου χρόνου.
- Μείωση των χρόνων προετοιμασίας (set-up times) των μηχανών.
- Μείωση των ενδιάμεσων αποθεμάτων και των τεμαχίων υπό επεξεργασία (work in progress) εντός του συστήματος.
- Αύξηση της αποδοτικότητας των εργαζόμενων λόγω του αισθήματος συνεργασίας σε μία ομάδα.
- Υψηλότερη ποιότητα κατασκευής.
- Δυνατότητα γρήγορων αλλαγών και βελτιώσεων στη διαδικασία παραγωγής.

Ένας από τους πρώτους ορισμούς, που έχουν δοθεί για την Τεχνολογία Ομάδων (Group Technology - GT) είναι του E. K. Ivanov:

«Ο κύριος στόχος της GT είναι να παράγεται ένας μικρός αριθμός τεμαχίων, χρησιμοποιώντας πολλές μεθόδους παραγωγής.»[3]

Ενώ, ένας άλλος πιο φιλοσοφικός και γενικευμένος ορισμός είναι ο εξής:

«GT είναι η συνειδητοποίηση, ότι ομαδοποιώντας πολλά όμοια προβλήματα σε ομάδες, μπορεί να βρεθεί μια κοινή λύση για κάθε ομάδα προβλημάτων, εξοικονομώντας χρόνο και προσπάθεια.»[3]

1.1.3 Κύτταρα (Κυψέλες) Παραγωγής

Το βασικό πρόβλημα για τη δημιουργία *κυττάρων παραγωγής* αποτελεί η ταξινόμηση των προϊόντων σε ομάδες/οικογένειες. Το βασικό κριτήριο ταξινόμησης είναι η ομοιότητα στην παραγωγική διαδικασία, η οποία συνήθως αντικατοπτρίζεται και σε ομοιότητα στη μορφή, αλλά δεν είναι απαραίτητο. Για παράδειγμα δύο τεμάχια μπορεί να είναι απολύτως ίδια από πλευράς μορφής, αλλά να διαφέρουν από άποψη υλικού, όγκου παραγωγής και ανοχών ακρίβειας και έτσι να μην ανήκουν στην ίδια οικογένεια. Αντίθετα μια σειρά κομματιών, τα οποία διαφέρουν αρκετά μορφολογικά, αλλά η διαδικασία παραγωγής τους να είναι σε μεγάλο βαθμό παρόμοια π.χ. αξονο-συμμετρικά (κατεργασία σε τόρνο) ενδέχεται να αποτελούν μια οικογένεια/ομάδα τεμαχίων.

Γενικά, η μορφοποίηση των κυττάρων αποβλέπει στην ομαδοποίηση των μηχανών σε κύτταρα και των τεμαχίων σε οικογένειες, που αντιστοιχίζονται μεταξύ τους με δεδομένα τις μηχανές, τα τεμάχια, τα

φασεολόγια και τη ζήτηση παραγωγής. Υπάρχουν τρεις μέθοδοι για την ταξινόμηση των προϊόντων σε ομάδες:

- Η οπτική
- Η κωδικοποίηση
- Η ανάλυση της ροής παραγωγής

Η *οπτική ομαδοποίηση* πραγματοποιείται με βάση εικόνες ή σχέδια των προϊόντων και αποτέλεσε τη βασική μέθοδο ταξινόμησης στις πρώτες επιτυχημένες εφαρμογές της τεχνολογίας ομάδων. Η συγκεκριμένη μέθοδος είναι σχετικά γρήγορη και φθηνή αλλά δεν είναι ιδιαίτερα αξιόπιστη καθώς στηρίζεται πολύ στην εμπειρία του ατόμου που ταξινομεί, ο οποίος πρέπει ιδανικά να έχει πλήρη εικόνα των παραγωγικών διαδικασιών όλων των προϊόντων, καθώς δεν παρέχεται καμιά πληροφορία σχετικά με αυτά.

Η *κωδικοποίηση* αποτελεί την πιο χρονοβόρα μέθοδο ομαδοποίησης, αφού για τη δημιουργία του κωδικού ενός προϊόντος λαμβάνονται υπόψη τα *σχεδιαστικά (design features)* και *κατασκευαστικά (manufacturing features)* του. Τα σχεδιαστικά χαρακτηριστικά σχετίζονται με τη γεωμετρία/μορφή, το μέγεθος και το υλικό του προϊόντος, ενώ τα κατασκευαστικά χαρακτηριστικά αφορούν στα επιμέρους στάδια παραγωγής και τον εξοπλισμό που απαιτείται. Ο κωδικός, που αποδίδεται στο προϊόν, είναι μια ακολουθία αριθμών και χαρακτήρων, η οποία αποτελεί περιγραφή των χαρακτηριστικών του προϊόντος. Η ερμηνεία και η δομή της κωδικοποίησης των τεμαχίων κατηγοριοποιείται στα παρακάτω συστήματα: [4]

- *Ιεραρχικά*: η σημασία κάθε ψηφίου αλλάζει αναλόγως της σημασίας του προηγούμενου ψηφίου.
- *Απόλυτα*: κάθε κωδικοποιημένο χαρακτηριστικό αντιστοιχεί σε συγκεκριμένο ψηφίο.
- *Υβριδικά*: Μίξη των άλλων δύο

Η ιεραρχική δομή πλεονεκτεί στο ότι μπορεί να εμπεριέχει μεγαλύτερο βαθμό πληροφορίας χρησιμοποιώντας λιγότερα ψηφία, ενώ αντίθετα η αλυσιδωτή δομή είναι πιο εύκολη στην ερμηνεία, αλλά απαιτεί μεγαλύτερο αριθμό ψηφίων για τη μετάδοση της ίδιας ποσότητας πληροφορίας. Γι' αυτό πρακτικά, ακολουθούνται υβριδικοί τύποι που συνδυάζουν τους παραπάνω δύο τύπους.

Επίσης, αξίζει να αναφερθεί ότι και η κατάρτιση των κωδικών διαφέρει, η οποία πραγματοποιείται με έναν από τους παρακάτω τρόπους:

- βάσει πινάκων, οι οποίοι περιέχουν τους κανόνες αντιστοίχισης μεταξύ χαρακτηριστικών και κωδικών
- μέσω ενός ψηφιακού συστήματος απόδοσης κωδικών, το οποίο διαθέτει τους κανόνες αντιστοίχισης και λειτουργεί αλληλεπιδραστικά με τον χρήστη, όπως στη χρήση έμπειρων συστημάτων (expert systems), ο οποίος απαντά σε συγκεκριμένες ερωτήσεις
- μέσω της αυτόματης αναγνώρισης των μορφολογικών χαρακτηριστικών εκάστου τεμαχίου (CAD)

Το σύστημα κωδικοποίησης που θα ακολουθηθεί είναι ένας ιδιαιτέρως σημαντικός παράγοντας που επηρεάζει όλα τα τμήματα του συστήματος παραγωγής όπως του σχεδιασμού προϊόντων, σχεδιασμού κατεργασιών, ελέγχου παραγωγής, αγοράς υλικών, καθώς και στην επικοινωνία μεταξύ τμημάτων, γιατί ουσιαστικά η κωδικοποίηση τυποποιεί το προϊόν και συνδέει τον σχεδιασμό με την παραγωγή του. Μερικά

Παραδείγματα συστημάτων κωδικοποίησης προϊόντων για τη δημιουργία ομάδων/οικογενειών τεμαχίων είναι: DCLASS, MICLASS και OPITZ.

Η *Ανάλυση Ροής Παραγωγής* (Production Flow Analysis – PFA) είναι μια άλλη μέθοδος ομαδοποίησης (clustering) των προϊόντων και δημιουργίας κυττάρων παραγωγής, η οποία βασίζεται στην καταγραφή και ανάλυση των επιμέρους σταδίων κατεργασίας ενός προϊόντος, όπως αυτή αποτυπώνεται στο διάγραμμα ροής κατεργασίας εκάστου τεμαχίου. Η σύγκριση των διαγραμμάτων ροής επιτρέπει τη δημιουργία ομάδων από τεμάχια, που παράγονται με παρόμοια διαδικασία. Η PFA περιλαμβάνει συνήθως τα εξής στάδια:

- Κατάρτιση των διαγραμμάτων ροής των τεμαχίων ως μια σειρά εργασιών και σύνδεση τους με τις αντίστοιχες μηχανές κατεργασίας
- Ταξινόμηση των διαγραμμάτων ροής των τεμαχίων σε ομάδες, βάσει του βαθμού ομοιότητας που παρουσιάζουν μεταξύ τους. Οι εργασίες και οι μηχανές κωδικοποιούνται με αριθμούς, το οποίο διευκολύνει στην αναγνώριση παρόμοιων κατεργασιών. Εδώ, τεμάχια με εντελώς παρόμοια διαγράμματα καταρτίζουν ομάδες τεμαχίων.
- Κατάρτιση ενός πίνακα ή χάρτη ροής παραγωγής, όπου απεικονίζεται συνοπτικά η συσχέτιση μηχανών και ομάδων τεμαχίων. Οι τιμές των κελιών του πίνακα παίρνουν την τιμή 1, εάν μια μηχανή υπάρχει στο διάγραμμα ροής ενός τεμαχίου, ή 0 εάν δεν υπάρχει.
- Αναδιάταξη των στηλών και γραμμών του πίνακα ώστε να δημιουργηθούν σχηματικά οι ομάδες τεμαχίων, που έχουν σχετικώς κοινά διαγράμματα ροής.

Για την δημιουργία αυτών των ομάδων/οικογενειών τεμαχίων και κατ' επέκταση τον διαχωρισμό των μηχανών σε κύτταρα, χρησιμοποιούνται διάφοροι αλγόριθμοι. Ενδεικτικά, αναφέρονται οι παρακάτω:

- Βαθμονομικής Ομαδοποίησης (Rank Order Clustering – ROC)
- Ενέργειας Δεσμών (Bond Energy Method)
- Συντελεστών Ομοιότητας Μηχανών
- Δενδροδιάγραμμα Ομοιότητας Μηχανών [4] [5]

Βέβαια, με την ομαδοποίηση των μηχανών και τη δημιουργία ομάδων/οικογενειών τεμαχίων δημιουργούνται και προβλήματα, όπως όταν κάποιες κατεργασίες (π.χ. χύτευση) δεν δύναται να ομαδοποιηθούν και επομένως, πρέπει να παραμείνουν ανεξάρτητες, το οποίο δημιουργεί πληθώρα κυττάρων, ή πρέπει να δημιουργηθούν πολλαπλά αντίγραφα τους σε κάθε κύτταρο, κάτι που μπορεί να έχει απαγορευτικό. Αντίστοιχο πρόβλημα προκύπτει και στην περίπτωση όπου υπάρχουν τεμάχια, που χρειάζονται κατεργασία σε περισσότερα από ένα κύτταρα και είτε δημιουργούνται εικονικά κύτταρα, είτε αλλάζει το φασεολόγιο του τεμαχίου.

Καταλήγοντας, τα πλεονεκτήματα – μειονεκτήματα των Κυττάρων Παραγωγής θα μπορούσαν να συνοψιστούν στα παρακάτω: [4]

- Πλεονεκτήματα
  - Εύκολος προγραμματισμός και έλεγχος παραγωγής.

- Χαμηλότερος αριθμός μονάδων υπό επεξεργασία και νεκρός χρόνος μεταφοράς (WIP).
- Χαμηλός χρόνος προετοιμασίας (set-up time).
- Λιγότερο προσωπικό, με πολλαπλή εξειδίκευση.
- Υψηλή ποιότητα (ενσωμάτωση επιθεώρησης – ποιοτικού ελέγχου παραγωγής).
- Ορθολογικός σχεδιασμός (παραμετροποίηση - τυποποίηση).

- Μειονεκτήματα
  - Πολλές ανθρωπο-ώρες για κωδικοποίηση προϊόντων.
  - Προβληματική ενσωμάτωση 'μη-κυτταρικών' διεργασιών.
  - Έλλειψη ευελιξίας σε σχέση με το μίγμα προϊόντων.

1.2 Εισαγωγή στα Ευέλικτα Συστήματα Κατεργασιών (ΕΣΚ)

1.2.1 Ορισμός – Δομή – Ευελιξία

Τα *Ευέλικτα Συστήματα Κατεργασιών - ΕΣΚ (Flexible Manufacturing system - FMS)* είναι συστήματα Τεχνολογίας Ομάδων (GT) προγραμματισμένου αυτοματισμού και ουσιαστικά αποτελούν ένα ιδιαίτερα αυτοματοποιημένο και τεχνολογικά εξελιγμένο κύτταρο παραγωγής. Αν θα θέλαμε να δώσουμε έναν ορισμό των ΕΣΚ, τότε ο πιο περιεκτικός θα ήταν αυτός του Ranky (1983):

«*Το ΕΣΚ είναι ένα σύστημα που ασχολείται με την επεξεργασία διανομής δεδομένων και αυτοματοποιημένης ροής υλικού, χρησιμοποιώντας μηχανές αριθμητικού ελέγχου, κύτταρα συναρμολόγησης, βιομηχανικά ρομπότ, μηχανήματα επιθεώρησης/ελέγχου, μαζί με ολοκληρωμένα συστήματα διαχείρισης υλικών και αποθεμάτων*»[6]

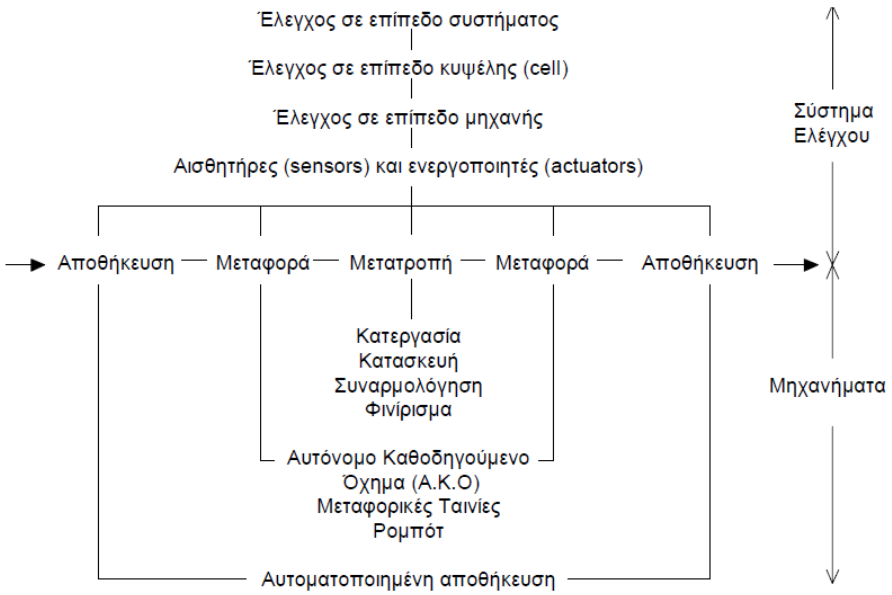
Το βασικό χαρακτηριστικό των Ευέλικτων Συστημάτων Παραγωγής είναι η ***αυτοματοποίηση***. Ένα ευέλικτο σύστημα αποτελείται, συνήθως, από διάφορους σταθμούς επεξεργασίας, μεταξύ των οποίων μπορεί να δρομολογηθεί η ταυτόχρονη παραγωγή προϊόντων διαφόρων οικογενειών/ομάδων. Για να είναι αυτό εφικτό, ένα τυπικό ΕΣΚ ενσωματώνει πολλές τεχνικές αυτοματοποίησης και τεχνολογιών αυτομάτου ελέγχου. Ο αυτοματισμός έγκειται, όχι μόνο στη μηχανοποίηση απλών και σύνθετων μεθόδων παραγωγής, αλλά και στα συστήματα μεταφοράς και ελέγχου που τις συνδέουν. Η ευέλικτη αυτοματοποίηση βρίσκει εφαρμογή στους τομείς της επεξεργασίας, της κατασκευής, της συναρμολόγησης, του φινιρίσματος, της αποθήκευσης, της διαχείρισης και της μεταφοράς υλικών. Ως επακόλουθο αυτών, τα ευέλικτα συστήματα αυτοματοποίησης περιλαμβάνουν εξελιγμένες μηχανές παραγωγής και επιθεώρησης (ποιοτικού ελέγχου), συστήματα ελέγχου με υπολογιστές, συστήματα επικοινωνίας καθώς και συστήματα διαχείρισης υλικών.

Η αυτοματοποίηση μπορεί να υπεισέρχεται, από το χαμηλότερο έως το υψηλότερο επίπεδο ιεράρχησης ενός ΕΣΚ. Έτσι, διακρίνουμε τρία είδη, αναλόγως του επιπέδου ελέγχου που εφαρμόζεται η αυτοματοποίηση (Εικόνες 1-8 και 1-9): [7]

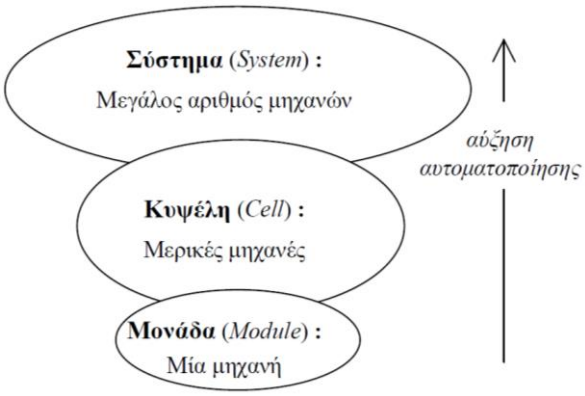
Μονάδα (Module): Μία αριθμητικά ελεγχόμενη εργαλειομηχανή με ενσωματωμένο μικροϋπολογιστή (Computer Numerical Control - CNC). Περιλαμβάνει συσκευή αυτόματης αλλαγής εργαλείων, αποθήκη εργαλείων και ενδεχομένως αυτόματο σύστημα φόρτο-εκφόρτωσης και αποθήκευσης των προς κατεργασία κομματιών (μεταφορική ταινία, ρομποτικό βραχίονα, «ολοκληρωμένο» ρομπότ).

Κυψέλη – Κύτταρο (Cell): Ένα ευέλικτο κύτταρο αποτελείται από μικρό αριθμό συνδεδεμένων CNC μηχανών, με αυτόματη φόρτο-εκφόρτωση (π.χ. ημικυκλική διάταξη γύρω από ένα ρομπότ ή γραμμική διάταξη με συνδεδεμένα κέντρα κατεργασίας). Μία κυψέλη είναι απαραίτητο να περιλαμβάνει σύστημα διακίνησης παλετών - τεμαχίων, αποθήκη εργαλείων και υλικών, καθώς και σύστημα ελέγχου που επιτρέπει στην εγκατάσταση να είναι αυτό-ελεγχόμενη.

Σύστημα (System:) Ένα ευέλικτο σύστημα παραγωγής περιλαμβάνει συνήθως περισσότερους σταθμούς επεξεργασίας (πάνω από τρεις), εξοπλισμό ποιοτικού ελέγχου (π.χ. μετρητικές μηχανές), καθώς και υποστηρικτικό εξοπλισμό, όπως σταθμούς καθαρισμού των κομματιών ή των παλετών. Γι' αυτό και διαθέτει αυτόματα συστήματα ελέγχου, που μπορούν να διαχειριστούν αυξημένη πολυπλοκότητα, τα οποία λειτουργούν συμπληρωματικά των εργασιών των κυψελών ή και των μονάδων. Το σύστημα είναι μεγαλύτερο της κυψέλης και περιέχει επιπροσθέτως αυτόματα συστήματα αποθήκευσης και διακίνησης υλικών/παλετών/τεμαχίων/εργαλείων. Λόγω των μεγαλύτερων αποστάσεων που πιθανόν να υπάρχουν στο επίπεδο του συστήματος, τα υλικά και τα εξαρτήματα μεταφέρονται με οχήματα αυτόματης πλοήγησης (ΟΑΠ – AGV's) ή μεταφορικές ταινίες.



Εικόνα 1-8 Αυτοματοποίηση και Εφαρμογή Ελέγχου στα Συστήματα Κατεργασιών



Εικόνα 1-9 Επίπεδα Αυτοματοποίησης στα Συστήματα Κατεργασιών

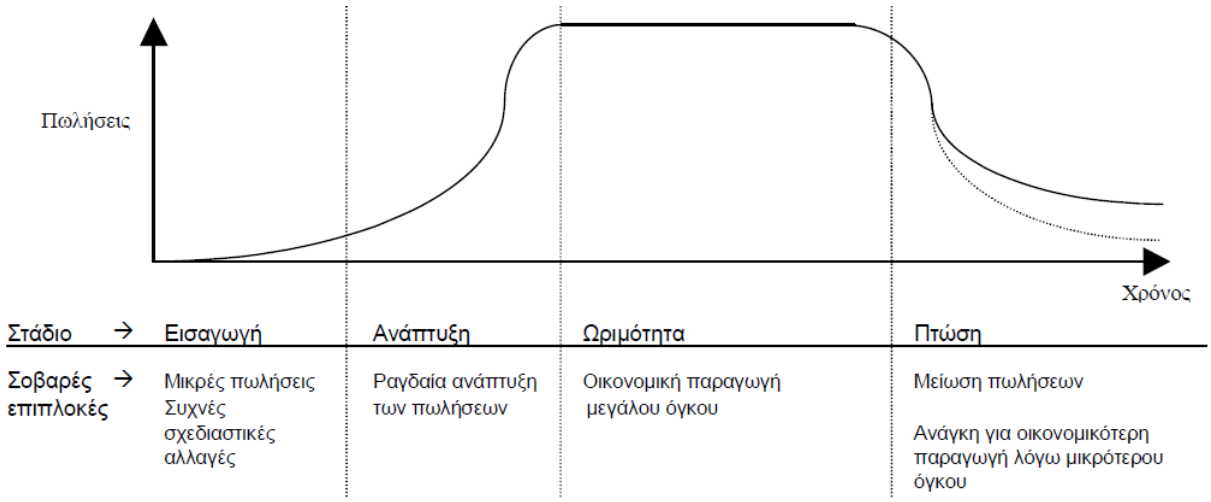
Ένα Ευέλικτο Σύστημα Κατεργασιών (ΕΣΚ) αποτελείται από κάποια βασικά στοιχεία, επομένως η γενική του δομή μπορεί να οριστεί, όπως παρακάτω:[4]

- Σταθμοί Κατεργασίας – Εργαλειομηχανές
  - Προγραμματιζόμενης λειτουργίας
  - Ελεγχόμενες από Η/Υ
  - Διαθέτουν δυνατότητα αποθήκευσης και αλλαγής εργαλείων – τεμαχίων
- Σύστημα Χειρισμού (Διακίνησης) Υλικών
  - Διακίνηση και φορτο-εκφόρτωση τεμαχίων - εργαλείων
  - Παλέτες, Ρομπότ, Οχήματα Αυτόματης Πλοήγησης – ΟΑΠ (Automated Guided Vehicles - AGV's), Μύλοι εργαλείων
- Σύστημα Ενδιάμεσης Αποθήκευσης
  - Automatic Storage and Retrieval Systems (AS/RS)
  - Machine Buffers, Magazines
- Σύστημα Ελέγχου (Λογισμικό + Η/Υ)
  - Ανάθεση Εργασιών, Εντολές, Καθορισμός Διαδρομών, Δήλωση Προτεραιοτήτων

Η *ευελιξία* ενός Συστήματος Κατεργασιών αναφέρεται σε διαφορετικούς τομείς και μπορεί χονδρικά να διακριθεί στην *εσωτερική* ευελιξία, που αφορά στην ικανότητα του συστήματος παραγωγής να ανταποκρίνεται στις αλλαγές που προκαλούνται από την ίδια την επιχείρηση (π.χ. βελτίωση στο σχεδιασμό του προϊόντος), ενώ η *εξωτερική* ευελιξία αφορά στην αντίδραση του συστήματος σε ερεθίσματα προερχόμενα από το ευρύτερο περιβάλλον (π.χ. την αγορά - πελάτες). Πιο αναλυτικά, μπορούμε να διακρίνουμε την εφαρμογή του όρου ευελιξία στα Συστήματα Κατεργασιών, στις εξής έννοιες: [7]

- *Ευελιξία μηχανής*: εκφράζει την προσαρμοστικότητα της μηχανής σε εναλλαγή παραγωγής διαφορετικών προϊόντων, δηλαδή την ευκολία με την οποία επιδέχεται τις απαραίτητες αλλαγές, ώστε να παραχθεί ένα συγκεκριμένο σύνολο τύπων τεμαχίων (Σημ.: το σύνολο τύπων τεμαχίων είναι πιο ευρύ από την οικογένεια τεμαχίων).
- *Ευελιξία κατεργασίας:* εκφράζει τη δυνατότητα παραγωγής ενός συνόλου τύπων τεμαχίων με διαφορετικούς τρόπους, δηλαδή με διαφορετικές κατεργασίες (εναλλαγή στη σειρά ή διαφοροποίηση στο σύνολο κατεργασιών).

- *Ευελιξία ποικιλίας προϊόντων*: εκφράζει την ευκολία με την οποία είναι δυνατή η εναλλαγή στο μείγμα προϊόντων, το οποίο δύναται να κατασκευαστεί από το ΕΣΚ, και συγκεκριμένα:
  - Στον αριθμό των διαφορετικών προϊόντων, που παράγονται από το ΕΣΚ σε οποιαδήποτε χρονική στιγμή.
  - Στη δυνατότητα του ΕΣΚ να εναλλάσσει την παραγωγή μεταξύ δύο προϊόντων γρήγορα και με μικρό κόστος.
  - Στις διάφορες παραλλαγές μεγέθους, σχήματος, πρώτων υλών και απαιτούμενων κατεργασιών για την παραγωγή προϊόντων.
  - Στην ικανότητα εύκολης εισαγωγής νέων προϊόντων στην παραγωγή.
- *Ευελιξία προϊόντος:* εκφράζει την προσαρμογή ενός ΕΣΚ στις απαιτήσεις ενός συγκεκριμένου προϊόντος, αναλόγως και του σταδίου που βρίσκεται στην καμπύλη κύκλου ζωής του προϊόντος (Εικόνα 1-10).



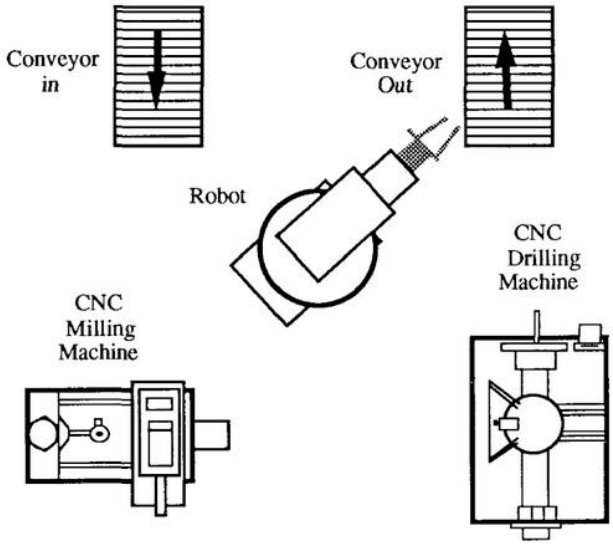
Εικόνα 1-10 Στάδια Κύκλου Ζωής ενός Προϊόντος

- *Ευελιξία διαδρομής*: αναφέρεται στη δυνατότητα εναλλακτικών λύσεων όταν ένα ή περισσότερα σημεία της διαδρομής ενός ή περισσότερων τεμαχίων γίνονται απροσπέλαστα λόγω π.χ. βλάβης μηχανών/εξοπλισμού.
- *Ευελιξία όγκου παραγωγής*: εκφράζει τη δυνατότητα να λειτουργεί το ΕΣΚ με κέρδος σε όσο το δυνατόν μεγαλύτερο εύρος μεγέθους μερίδων.
- *Ευελιξία επέκτασης*: εκφράζει τη δυνατότητα εισαγωγής νέων μηχανών, συστημάτων αποθήκευσης και διακίνησης προϊόντων κτλ, δηλαδή σε περίπτωση αλλαγής μεγέθους του ΕΣΚ πόσο επεκτάσιμο είναι.
- *Ευελιξία διεργασιών:* αναφέρεται στη αλλαγή των λοιπών διαδικασιών που λαμβάνουν χώρα στο ΕΣΚ, πέραν των κατεργασιών, όπως π.χ. στην ικανότητα αναγνώρισης και διόρθωσης σφαλμάτων/βλαβών.
- *Ευελιξία παραγωγής:* περιλαμβάνει ουσιαστικά όλα τα είδη ευελιξίας, που προαναφέρθηκαν, καθώς και αυτά που δεν αναφέρθηκαν σε κάποιο συγκεκριμένο είδος και γενικότερα στην ικανότητα ανάκαμψης του συστήματος από επιδράσεις του ευρύτερου περιβάλλοντος.

1.2.2 Εφαρμογές – Εξοπλισμός – Οφέλη

Τα *ευέλικτα συστήματα παραγωγής* θεωρούνται γενικά κατάλληλα, όταν η παραγωγή έχει χαρακτηριστικά αντίστοιχα με αυτά που υποδεικνύουν την οργάνωση σε κύτταρα παραγωγής, δηλαδή παραγωγή μέσης ποικιλίας προϊόντων, τα οποία μπορούν να ομαδοποιηθούν σε σχετικά λίγες ομάδες/οικογένειες (μερικές δεκάδες) και παράγονται σε μέσο όγκο ετήσιας παραγωγής (5.000-75.000 τεμάχια/έτος). Η βασική διαφορά με ένα τυπικό κύτταρο παραγωγής, που εμπεριέχει κυρίως οργανωτικές και χωροταξικές βελτιώσεις, έγκειται κυρίως στο αρκετά υψηλότερο επίπεδο τεχνολογίας (αυτοματοποίησης) που ενσωματώνει. Όπως γίνεται κατανοητό, το αρχικό κόστος επένδυσης σε ένα ΕΣΚ είναι αρκετά υψηλό, ενώ αντίστοιχα υψηλές είναι και οι απαιτήσεις σε εκπαιδευμένο και εξειδικευμένο εργατικό δυναμικό. Το αυξημένο κόστος επένδυσης και εργασίας αντισταθμίζεται όμως, από την αναμενόμενη αύξηση του βαθμού εκμετάλλευσης του εξοπλισμού μέσω της μείωσης άεργου χρόνου, της σμίκρυνσης του απαιτούμενου χώρου, της καλύτερης ανταπόκρισης σε αλλαγές των παραγωγικών απαιτήσεων, των χαμηλότερων αποθεμάτων, του μειωμένου χρόνου διεκπεραίωσης εντολών παραγωγής, καθώς και της υψηλότερης παραγωγικότητας της εργασίας. Η κύρια εφαρμογή των ΕΣΚ πραγματοποιείται σε προϊόντα που κατασκευάζονται με σχετικά πολύπλοκες και απαιτητικές κατεργασίες.

Ειδικότερα, τα ΕΣΚ βρίσκουν εφαρμογή κυρίως στην παραγωγή προϊόντων πρισματικής γεωμετρίας, δηλαδή κατεργασιών κοπής με φρέζα και δράπανο, όπως π.χ. στην κατασκευή μπλοκ κυλίνδρων μιας μηχανής εσωτερικής καύσης (Εικόνα 1-10). Σε σύγκριση με τα κυλινδρικής μορφής προϊόντα, τα κομμάτια αυτά είναι συνήθως βαρύτερα, το οποίο καθιστά πιο δύσκολη τη διαχείρισή τους από έναν εργαζόμενο ή ρομπότ. Το σχετικό κόστος παραγωγής τείνει να είναι υψηλό, λόγω της πιο πολύπλοκης και χρονοβόρας διαδικασίας παραγωγής. Λόγω των παραπάνω η επένδυση σε ένα ΕΣΚ με δυνατότητα παραγωγής μεγάλης ποικιλίας προϊόντων μπορεί να θεωρηθεί οικονομικά δικαιολογημένη. Τα αξονοσυμμετρικά κομμάτια, αντίθετα, έχουν συνήθως απλούστερη διαδικασία κατασκευής και είναι αρκετά ελαφρύτερα. Στην περίπτωση αυτή είναι σχετικά ευκολότερο να επιτευχθεί η απαιτούμενη ευελιξία με τη χρήση ενός ρομπότ τοποθέτησης και ενός κέντρου κατεργασιών τόρνου.



Εικόνα 1-11 Τυπικό Ευέλικτο Κύτταρο Παραγωγής Μπλοκ Κινητήρων

Τα βασικά στοιχεία, που κατά κύριο λόγο, απαρτίζουν τον *εξοπλισμό* ενός Ευέλικτου Κυττάρου Κατεργασιών αναλύονται στα παρακάτω:[8]

Κύρια Στοιχεία

- *Ανεξάρτητες μηχανές αριθμητικού ελέγχου (CNC)*: είναι συνήθως προγραμματιζόμενοι σταθμοί κατεργασιών στους οποίους περιλαμβάνονται κέντρα τόρνευσης, κέντρα κατεργασιών, κέντρα διαμόρφωσης/κοπής ελάσματος.
- *Αυτόματο σύστημα διακίνησης υλικού:* πραγματοποιεί τη μεταφορά του υλικού μεταξύ των σταθμών αλλά και το χειρισμό του σε κάθε σταθμό (φορτο-εκφόρτωση). Η μεταφορά πραγματοποιείται με σταθερές διατάξεις τύπου ταινιόδρομου ή τύπου γερανογέφυρας καθώς και με οχήματα αυτόματης πλοήγησης πάνω σε ταινιόδρομο (Rail Guided Vehicles – RGV's) ή μη (Automated Guided Vehicles - AGV's), γερανογέφυρες . Χειρισμός σε κάθε σταθμό γίνεται με ρομπότ ή και αυτόματες συσκευές ενσωματωμένες στο σταθμό κατεργασίας.
- *Διατάξεις αποθήκευσης*: τοποθετούνται στην αρχή, στο τέλος αλλά και σε ενδιάμεσα στάδια της παραγωγικής διαδικασίας. Μπορεί να είναι αυτόματα εξυπηρετούμενα σύνολα με ράφια (Automatic Storage and Retrieval Systems – AS/RS), σταθερά ή περιστρεφόμενα τραπέζια (Magazines) με λίγες θέσεις αποθήκευσης ή και απλά ράφια εξυπηρετούμενα από προσωπικό.
- *Ολοκληρωμένη μέθοδος ελέγχου*: συντονίζει τις λειτουργίες των εργαλειομηχανών και των συστημάτων διαχείρισης υλικού, ώστε να επιτύχει βελτιστοποίηση της παραγωγής και να διαθέτει ευελιξία. Παράλληλα, αντικείμενο του σχεδιασμού ελέγχου του συστήματος είναι η κατανομή των εργασιών στο κύτταρο, στοχεύοντας στην ελαχιστοποίηση του συνολικού μη παραγωγικού χρόνου, τη μεγιστοποίηση του βαθμού χρήσης των μηχανών και άλλα κριτήρια κατά περίπτωση. Ο έλεγχος των συστημάτων κατεργασιών πραγματοποιείται με την υλοποίηση κάποιας λογικής, η οποία για τα αυτοματοποιημένα συστήματα ενσωματώνεται σε ένα πρόγραμμα λογισμικού και εκτελείται μέσω μικρο-ελεγκτών ή Η/Υ. Για την επικοινωνία του κεντρικού ελεγκτή με τα υπόλοιπα στοιχεία του κυττάρου χρησιμοποιούνται τεχνολογίες δικτύων ή διαύλων, κυρίως σειριακών (RS-232, RS 422, RS 485).
- *Συστήματα Ταυτοποίησης - Αισθητήρες*: χρησιμοποιούνται για την ταυτοποίηση/προσδιορισμό των τεμαχίων/εργαλείων/παλετών ή αναγνώρισης των καταστάσεων. Υλοποιούνται μέσω διαφόρων διατάξεων όπως μικρο-διακόπτες, φωτοκύτταρα, αναγνώστες bar-code, δέκτες ραδιοκυμάτων, κάμερες κλπ.

Βοηθητικά Στοιχεία

- *Σταθμοί καθαρισμού:* Για τα προϊόντα ή τις μηχανές κατεργασίας.
- *Σταθμοί* μέτρησης: Για τον ποιοτικό έλεγχο των προϊόντων, αλλά και για τον έλεγχο ακρίβειας των μηχανών κατεργασίας
- *Διατάξεις αυτόματης αλλαγής και διαχείρισης εργαλείων (spindle tooling)*: Η σημασία της επιλογής του σωστού εργαλείου, της παρακολούθησης της φθοράς , του πλήθους, της εναλλαξιμότητας και

του κόστους των εργαλείων ενός ΕΣΚ δημιουργεί την ανάγκη για ένα ξεχωριστό σύστημα διαχείρισης εργαλείων. Σε ένα τέτοιο σύστημα είναι δυνατόν η αυτόματη αλλαγή των εργαλείων με κάποιο κριτήριο π.χ. την ελαχιστοποίηση αλλαγών εργαλείων.

- *Προγραμματιζόμενες ιδιοσυσκευές συγκράτησης τεμαχίων (work holding features).*

Το βασικό κίνητρο για την υιοθέτηση των ΕΣΚ, είναι η μείωση του κόστους στην παραγωγή και η προσαρμοστικότητα σε ένα συνεχώς μεταβαλλόμενο περιβάλλον. Επιπλέον λόγοι, αποτελούν η απαιτούμενη μείωση του χρόνου παραγωγής των προϊόντων και η αυξανόμενη πολυπλοκότητά τους.

Τα κυριότερα εκτιμώμενα οφέλη από την υλοποίηση ενός ΕΣΚ μπορούν να συνοψιστούν στα εξής:

- Μεγαλύτερος βαθμός εκμετάλλευσης των μηχανών, λόγω καλύτερου προγραμματισμού, καλύτερης χρήσης των περιοχών ενδιάμεσης αποθήκευσης και *δυναμικού προγραμματισμού* (μέθοδος επίλυσης σύνθετων προβλημάτων, συνήθως βελτιστοποίησης, όπου το αρχικό πρόβλημα αναλύεται σε μια σειρά απλούστερων προβλημάτων αυξανόμενου μεγέθους, τα οποία επιλύονται κατά σειρά[9]).
- Απαιτούνται λιγότερες μηχανές κυρίως λόγω του μεγαλύτερου βαθμού εκμετάλλευσης τους.
- Καλύτερη εκμετάλλευση του χώρου, συγκριτικά με ένα τυπικό κατάστημα εργασιών εκτιμάται ότι το ΕΣΚ απαιτεί περίπου τον μισό χώρο.
- Μεγαλύτερος βαθμός ανταπόκρισης σε μεταβολές. Η εισαγωγή νέων προϊόντων, καθώς και οι αλλαγές στο πρόγραμμα παραγωγής εισάγονται ομαλότερα, σε σχέση με ένα σύστημα γραμμής παραγωγής.
- Μείωση αποθεμάτων/μονάδων υπό επεξεργασία (WIP). Στο ΕΣΚ κατασκευάζεται ένα μίγμα προϊόντων και όχι οι τυπικές ομοιογενείς παρτίδες προϊόντων, οπότε τόσο τα αποθέματα πρώτων υλών και έτοιμων προϊόντων, όσο και τα ενδιάμεσα αποθέματα/μονάδες υπό επεξεργασία μπορούν να περιοριστούν σημαντικά, συνήθως σε ποσοστά 60-80%.[1]
- Ταχύτερη παράδοση των τελικών προϊόντων, καθώς η μείωση των μονάδων υπό επεξεργασία επιφέρει αντίστοιχη μείωση στο χρόνο διεκπεραίωσης μιας εντολής παραγωγής ή παραγγελίας.
- Υψηλότερη παραγωγικότητα εργασίας. Ο υψηλότερος ρυθμός παραγωγής σε συνδυασμό με τη χρήση τεχνολογιών αυτοματοποίησης και ψηφιακής τεχνολογίας αυξάνει την παραγωγικότητα της εργασίας και μπορεί να οδηγήσει σε εξοικονόμηση κόστους εργασίας.
- Δυνατότητα λειτουργίας χωρίς επίβλεψη. Το υψηλό επίπεδο αυτοματοποίησης επιτρέπει θεωρητικά τη λειτουργία του συστήματος συνεχώς (24 ώρες) και με ελάχιστη επίβλεψη.

1.2.3 Σχεδιασμός και Έλεγχος

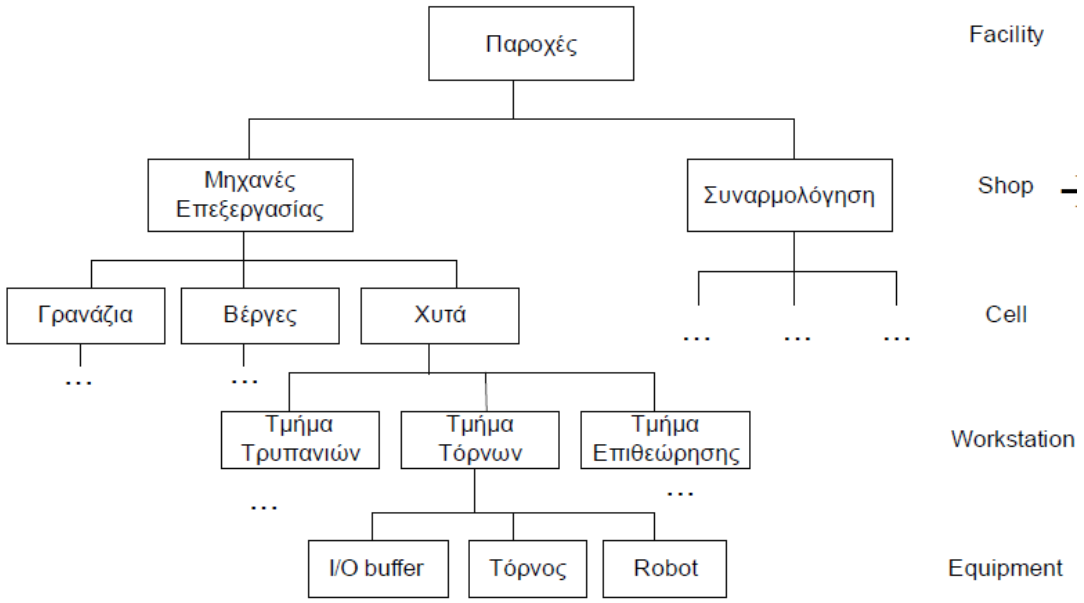
Θεμελιώδες στοιχείο στα ΕΣΚ αποτελεί το ψηφιακό σύστημα ελέγχου, το οποίο περιλαμβάνει έναν κεντρικό ελεγκτή, ο οποίος χρησιμοποιείται για το συντονισμό των επιμέρους στοιχείων του εξοπλισμού, καθώς και τους ελεγκτές των επιμέρους μηχανών και σταθμών εργασιών. Η επικοινωνία μεταξύ του συστήματος ελέγχου και του εξοπλισμού είναι αμφίδρομη, δηλαδή αποστέλλονται δεδομένα από το κεντρικό σύστημα προς τον εξοπλισμό και αντίστροφα. Οι παραπάνω ελεγκτές υλοποιούνται από Η/Υ,

μικρο-ελεγκτές, προγραμματισμένους λογικούς ελεγκτές, οι οποίοι επικοινωνούν μεταξύ τους μέσω διαφόρων προτύπων συνήθως σειριακής μετάδοσης.

Ο έλεγχος αναφέρεται στο «*τι*» πρέπει να επικοινωνεί (από τα στοιχεία του συστήματος) και στο «*πως*» επικοινωνούν τα στοιχεία αυτά μεταξύ τους. Κατόπιν αυτού, τηρείται μια ιεραρχική δομή ελέγχου, με τη ροή πληροφορίας να γίνεται μεταξύ γειτονικών στρωμάτων. Π.χ. ο ελεγκτής του σταθμού κατεργασίας τόρνων μπορεί να επικοινωνήσει μόνο με τον ελεγκτή της κυψέλης στην οποία ανήκει, το ρομπότ, τη μηχανή και το buffer του, ενώ δεν μπορεί να επικοινωνήσει άμεσα με τον ελεγκτή άλλης κυψέλης ή άλλου σταθμού κατεργασίας.

Το Εθνικό Ινστιτούτο Τυποποίησης και Τεχνολογίας της Αμερικής (National Institute of Standards and Technology - NIST) παρουσίασε ένα μοντέλο το οποίο διαιρεί τις παραγωγικές δραστηριότητες σε πέντε επίπεδα ιεραρχικού ελέγχου (Εικόνα 1-12):

- Εξοπλισμού (π.χ. Λειτουργία μύλου αλλαγής εργαλείων)
- Σταθμού κατεργασίας (π.χ. Εκτέλεση κατεργασιών αποπεράτωσης τεμαχίου)
- Κυττάρου (π.χ. Μεταφορά ημι-κατεργασμένου τεμαχίου μεταξύ σταθμών)
- Εργοστασίου (π.χ. Συγχρονισμός κυττάρων για συναρμολόγηση)
- Διοίκησης (π.χ. Προγραμματισμός παραγωγής για ορίζοντα 6 μηνών)



Εικόνα 1-12 Επίπεδα Ιεραρχικού Ελέγχου στα ΕΣΚ[7]

Όσον αφορά τον σχεδιασμό, αυτός υλοποιείται σε κάθε επίπεδο ελέγχου, αλλά διαφέρει ως προς τον ορίζοντα και τις αποφάσεις που λαμβάνονται (Εικόνα 1-13). Πιο συνοπτικά, ο σχεδιασμός. βάσει χρονικών προτεραιοτήτων, μπορεί να διακριθεί σε: *μακροπρόθεσμο σχεδιασμό του συστήματος* (αφορά στην επιλογή των τύπων των κομματιών, τα οποία θα ανατεθούν στο ΕΣΚ και στον εξοπλισμό που απαιτείται για την κατασκευή αυτών των κομματιών), *μεσοπρόθεσμο σχεδιασμό* (περιστρέφεται γύρω από τις εβδομαδιαίες ή ημερήσιες αποφάσεις σχετικά με το τι τύποι κομματιών και εργαλεία θα φορτωθούν στο σύστημα) και

στην *βραχυπρόθεσμη λειτουργία* (περιλαμβάνει το χρονικό προγραμματισμό και τον έλεγχο των μηχανών και του συστήματος διακίνησης υλικών).

Επίπεδο	**Ορίζοντας σχεδιασμού**	**Αποφάσεις**
Παροχές (facility)	Μήνες - χρόνια	Cad (manufacturing eng), σχεδιασμός παραγωγής, λογιστικά, ετήσια απόδοση, δυναμικότητα, συνολική παραγωγή, ποιότητα
Κατάστημα (shop)	Εβδομάδες - μήνες	Ομαδοποίηση παραγγελιών και ταξινόμηση, προληπτική συντήρηση, έλεγχος αποθεμάτων
Κυψέλη (cell)	Ώρες - εβδομάδες	Ακολουθία παρτίδων, δρομολόγηση εργασιών
Σταθμός κατεργασίας (workstation)	Λεπτά - ώρες	Εκκίνηση εργασιών και προσαρμογή, επιθεώρηση
Εξοπλισμός (equipment)	Χιλιοστά του δευτερολέπτου - λεπτά	Έλεγχος σε επίπεδο μηχανής, Συλλογή δεδομένων από αισθητήρες

Εικόνα 1-13 Σχεδιασμός εκάστου Επιπέδου Ελέγχου[7]

Το βασικό δεδομένο για το σχεδιασμό ενός ΕΣΚ, είναι η οικογένεια/ομάδα προϊόντων που πρέπει να παράγει, καθώς και ο αντίστοιχος όγκος παραγωγής. Η οικογένεια προϊόντων καθορίζει σε μεγάλο βαθμό τον τύπο του βασικού εξοπλισμού και τα χαρακτηριστικά του (ακρίβεια), καθώς και την χωροταξική διάταξη του εξοπλισμού στο κύτταρο. Ο όγκος παραγωγής καθορίζει τον αριθμό των μηχανών αλλά και τα χαρακτηριστικά του συστήματος χειρισμού και μεταφοράς των υλικών. Η χωροταξική διάταξη συνδέεται κυρίως με την ποικιλία των διαδρομών ροών που παρατηρείται. Εάν ένα σύστημα έχει μικρό ποσοστό ανάστροφης ροής τότε προτιμάται η γραμμική διάταξη. Αντιθέτως, αν υπάρχει μεγάλη ποικιλία διαδικασιών συνιστάται μια κυκλικού τύπου διάταξη. Οι διατάξεις τύπου σκάλας ή ανοικτού τύπου ενδείκνυνται για σχετικά μεγάλη ποικιλία προϊόντων και μεγάλα ποσοστά ανάστροφης ροής και παρακάμψεων. Σημαντική σχεδιαστική παράμετρος αποτελεί και ο απαιτούμενος βαθμός εκμετάλλευσης (ποσοστό χρησιμοποίησης) του εξοπλισμού, ο οποίος είναι συνήθως αντιστρόφως ανάλογος του αριθμού των μονάδων υπό επεξεργασία (WIP).

Από το σύνολο των διαλαμβανομένων στοιχείων που αναφέρθηκαν, γίνεται εύκολα κατανοητό ότι η εφαρμογή ελέγχου σε ένα ΕΣΚ δεν είναι κάτι απλό, αντιθέτως αποτελεί μια ιδιαιτέρως πολύπλοκη και σύνθετη διαδικασία, η οποία περιλαμβάνει επιμέρους στάδια. Αρχικά, πρέπει να εντοπιστούν τα επιμέρους στοιχεία που συγκροτούν το ΕΣΚ και επηρεάζουν την λειτουργία του, καθώς και πως επιτυγχάνεται η σύνδεση μεταξύ τους. Τα στοιχεία αυτά είναι οι σταθμοί, οι οποίοι πρέπει να ελεγχθούν και στους οποίους τοποθετούνται τοπικοί ελεγκτές (ή ο κάθε σταθμός μπορεί να διαθέτει ελεγκτή). Βάσει των παραπάνω, αρχικός στόχος είναι η μοντελοποίηση του κάθε επιπέδου ελέγχου ξεχωριστά, ξεκινώντας από το κατώτερο προς το υψηλότερο, αναλόγως και του επιπέδου αυτοματοποίησης που θέλουμε να επιτύχουμε. Η μοντελοποίηση στα ΕΣΚ, κατά το πλείστον, πραγματοποιείται μέσω των Δικτύων Petri (Petri Nets). Κατόπιν αυτού, γίνεται η προσομοίωση του συνολικού μοντελοποιημένου συστήματος, συμπεριλαμβανομένων όλων των επιπέδων. Αν όλα βαίνουν καλώς, τελευταίο βήμα αποτελεί η υλοποίηση

των ελεγκτών, η οποία γίνεται μέσω ηλεκτρονικών μονάδων, για τη διασύνδεση των επιμέρους σταθμών (τοπικοί ελεγκτές) με τον κεντρικό ελεγκτή που συντονίζει όλες τις διαδικασίες στο επίπεδο του. Έπειτα, συνεχίζουμε στο παραπάνω επίπεδο, όπου οι κεντρικοί ελεγκτές του κατώτερου επιπέδου αποτελούν πλέον τοπικούς ελεγκτές και υπάγονται σε έναν κεντρικό ελεγκτή του ανώτερου επιπέδου, συνεχίζοντας έως το υψηλότερο επίπεδο αυτοματοποίησης που επιθυμούμε να εφαρμόσουμε στο ΕΣΚ.

2

Παρουσίαση ΕΣΚ Παρούσας Εργασίας

2.1 Γενική Περιγραφή

Αντικείμενο της παρούσας εργασίας αποτελεί ο σχεδιασμός και η υλοποίηση των ελεγκτών ενός ΕΣΚ, του οποίου τα βασικά στοιχεία αναλύονται παρακάτω. Συγκεκριμένα, εξετάζεται η δυνατότητα εφαρμογής τοπικών ελεγκτών στους επιμέρους σταθμούς του ΕΣΚ και ο συντονισμός αυτών, από έναν κεντρικό ελεγκτή. Συναφώς, στο υπόψη ΕΣΚ δεν εξετάζονται αντικείμενα που αφορούν τη χωροταξική διάταξη των σταθμών, τις δυνατότητες παραγωγής, την ποικιλία, ομαδοποίηση/κωδικοποίηση των παραγόμενων προϊόντων ή τη συγκρότηση της δομής του ΕΣΚ (δημιουργία κυττάρων κτλ), καθότι οι σταθμοί που συνθέτουν το ΕΣΚ είναι δεδομένοι.

Κύριος στόχος της εργασίας είναι ο σχεδιασμός (μοντελοποίηση) και η προσομοίωση του δεδομένου ΕΣΚ ως προς την μεταβίβαση τεμαχίων/παλετών/εργαλείων μεταξύ των σταθμών και κατόπιν αυτού, η υλοποίηση ελέγχου, σε ένα επίπεδο αυτοματοποίησης, μέσω της τοποθέτησης των ελεγκτών (τοπικών και κεντρικού) στους σταθμούς του ΕΣΚ. Ο έλεγχος δύναται να υλοποιηθεί, μέσω αποστολής σημάτων εισόδου – εξόδου από κατάλληλες ηλεκτρονικές διατάξεις, οι οποίες θα πρέπει να προγραμματιστούν (software) και να συνδεθούν μεταξύ τους (hardware και software), μέσω κατάλληλων διαύλων επικοινωνίας. Επομένως, τελικό στάδιο της εργασίας αποτελεί η διασύνδεση των ελεγκτών, ώστε να καταστεί δυνατή η επικοινωνία μεταξύ τους.

Ο κάθε σταθμός έχει συγκεκριμένες εισόδους – εξόδους (σήματα) και τοπικό ελεγκτή, ο οποίος είναι ένας μικρο-ελεγκτής, οποίος ενεργοποιεί ιεραρχικά περαιτέρω λογισμικό για την εκτέλεση εξειδικευμένων λειτουργιών, όπως το linuxCNC (λογισμικό των μηχανών κατεργασίας) ή άλλο λογισμικό βάσει του οποίου λειτουργούν οι υπόλοιποι σταθμοί του ΕΣΚ. Ο κεντρικός ελεγκτής, είναι και αυτός ένας μικρο-ελεγκτής με επαυξημένες όμως δυνατότητες, ο οποίος δύναται να ελέγχει τους τοπικούς ελεγκτές και να επικοινωνεί μαζί τους με μέσω κατάλληλων σημάτων που μεταδίδονται μέσω των μικρο-ελεγκτών.

Συγκεκριμένα, ο σχεδιασμός και η προσομοίωση του μοντελοποιημένου συστήματος που αναπτύχθηκε, έγινε βάσει των κλασσικών Δικτύων Petri (Petri Nets – PN), ενώ η υλοποίηση και η ολοκλήρωση μέσω ελεγκτών, έγινε με τη χρήση μικρο-ελεγκτών τύπου ARDUINO-UNO Rev3 για τους τοπικούς ελεγκτές και ARDUINO-MEGA 2560Rev3 για τον κεντρικό ελεγκτή. Η επικοινωνία μεταξύ του κεντρικού ελεγκτή με τους τοπικούς ελεγκτές πραγματοποιήθηκε μέσω του σειριακού διαύλου I2C (Inter-Integrated Circuit), ο οποίος είναι μια διεπαφή δύο καλωδίων και αποτελεί Πρωτόκολλο Σύγχρονης Σειριακής Επικοινωνίας (Sychronous Serial Communication Protocol).

Η δομή του υπό μελέτη εξεταζόμενου συστήματος αποτελείται από ένα Κύτταρο Κατεργασιών, που αποκτήθηκε από το Εργαστήριο Κατεργασιών και περιλαμβάνει τρεις (3) σταθμούς (Εικόνα 2-1), που ελέγχονται από λογισμικό linux CNC:

- Κέντρο Τόρνευσης 2 αξόνων EMCO
- Κέντρο Κατεργασιών 3 αξόνων EMCO
- Ρομπότ 5 αξόνων Mitsubishi RM501

Εικόνα 2-1 Κύτταρο Κατεργασιών Εργαστηρίου

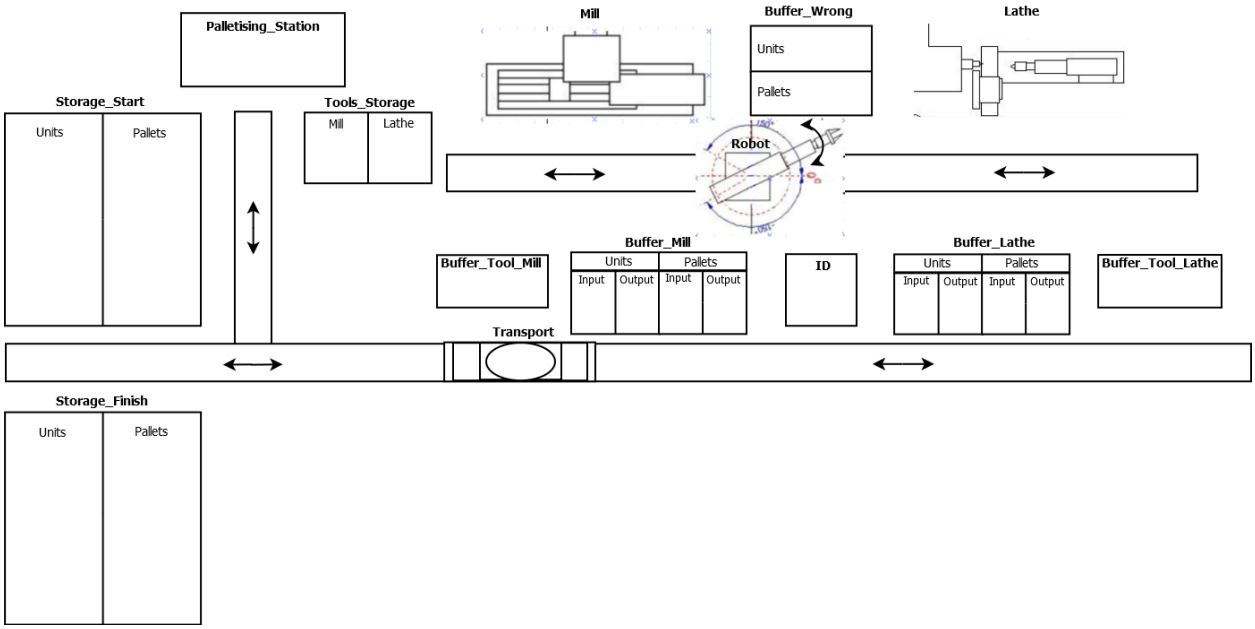
Το υπόψη κύτταρο πρέπει να διασυνδεθεί σε ένα Ευέλικτο Σύστημα Κατεργασιών (ΕΣΚ), το οποίο θα περιλαμβάνει επιπλέον σταθμούς, που είτε είναι υπό κατασκευή, είτε πρόκειται να αγοραστούν σε μελλοντικό χρόνο, και είναι οι εξής:

- Σταθμό Αποθήκευσης Τεμαχίων, Πρώτης Ύλης και Παλετών (Automatic Storage and Retrieval Systems AS/RS), η οποία αποτελεί μια διεπαφή (interface) ανθρώπου – ελεγκτή και σχεδιάζεται σε

άλλη διπλωματική εργασία. Κατόπιν τούτου, η υλοποίηση της είναι εκτός αντικειμένου της παρούσας εργασίας.

- Σταθμό Ταυτοποίησης Τεμαχίων
- Σταθμό Παλετοποίησης εξυπηρετούμενο από άνθρωπο
- Σύστημα Διακίνησης Παλετών/Τεμαχίων/Εργαλείων

Πέραν των αναφερομένων, προβλέφθηκαν ενδιάμεσες αποθήκες (buffers) τεμαχίων, παλετών και εργαλείων, καθώς και αποθήκες για τα λανθασμένα τεμάχια και παλέτες, που θα προκύπτουν ενδεχομένως, κατόπιν αρνητικού ελέγχου ταυτοποίησης από τον αντίστοιχο σταθμό. Κατόπιν των παραπάνω, η δομή του εξεταζόμενου ΕΣΚ παρουσιάζεται στο Σχήμα 2-1. Σημειώνεται ότι, η διάταξη είναι ενδεικτική και δεν αντιπροσωπεύει την πραγματική χωροταξική διάταξη του ΕΣΚ, καθότι αυτή δεν αποτελεί αντικείμενο μελέτης της παρούσας εργασίας.



Σχήμα 2-1 Δομή του ΕΣΚ

2.2 *Σύνθεση – Δομή – Αναλυτικός Εξοπλισμός*

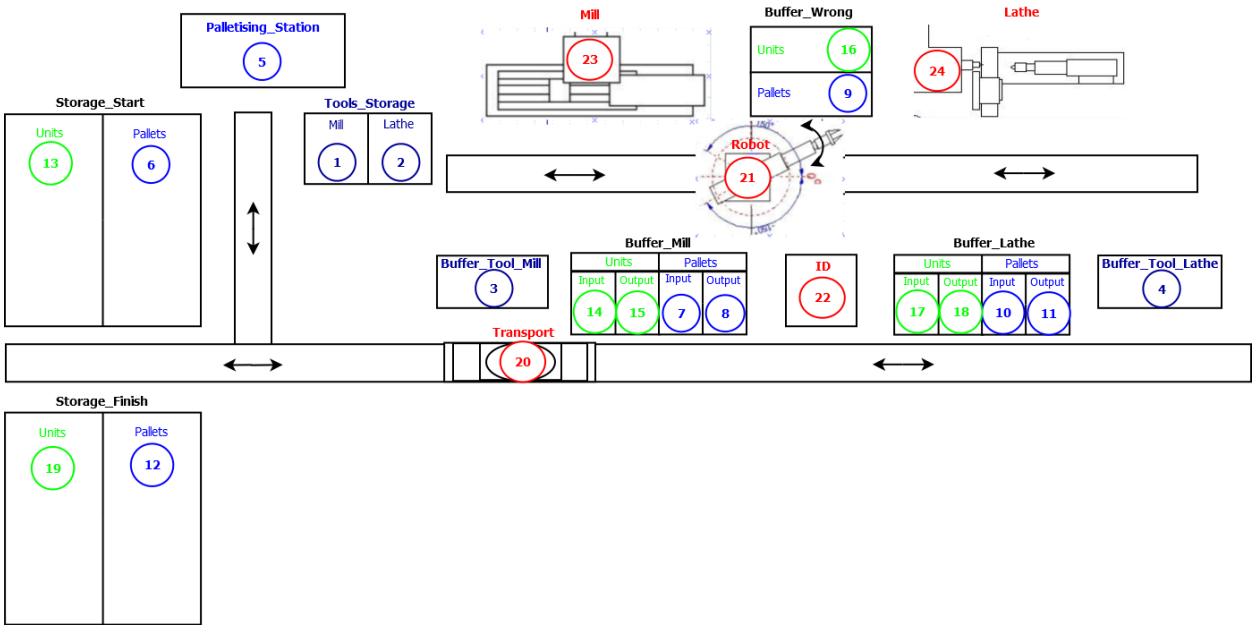
Για ευκολότερη εποπτεία και προκειμένου η διάταξη του ΕΣΚ να είναι ευδιάκριτη, ομαδοποιήθηκαν οι Σταθμοί του ΕΣΚ, αναλόγως του μέσου που εξυπηρετούν (τεμάχια, παλέτες, εργαλεία ή κοινά για όλους). Επομένως, η σύνθεση του ΕΣΚ, με ανάλογο χρωματισμό των ομαδοποιημένων Σταθμών, παρουσιάζεται στο Σχήμα 2-2 και παρατίθεται, όπως παρακάτω στον Πίνακα 2-1:

Εργαλεία (Σκούρο μπλε)-		
1	Αποθήκη Εργαλείων Κέντρου Κατεργασιών	Tool_Storage_Mill
2	Αποθήκη Εργαλείων Κέντρου Τόρνευσης	Tool_Storage_Lathe
3	Ενδιάμεση Αποθήκη Εργαλείων Κέντρου Κατεργασιών	Buffer_Tool_Mill
4	Ενδιάμεση Αποθήκη Εργαλείων Κέντρου Τόρνευσης	Buffer_Tool_Lathe

Παλέτες (Ανοιχτό Μπλε)		
5	Σταθμός Παλετοποίησης	Paletising_Station
6	Αποθήκη α' υλών (ακατέργαστα τεμάχια) σε Παλέτες	Storage_Start_Pallets
7	Ενδιάμεση Αποθήκη Παλετών Εισόδου στο Κέντρο Κατεργασιών	Buffer_Mill_Pallets_Input
8	Ενδιάμεση Αποθήκη Παλετών Εξόδου από το Κέντρο Κατεργασιών	Buffer_Mill_Pallets_Output
9	Αποθήκη Λανθασμένων Παλετών	Buffer_Wrong_Pallets
10	Ενδιάμεση Αποθήκη Παλετών Εισόδου στο Κέντρο Τόρνευσης	Buffer_ Lathe _Pallets_Input
11	Ενδιάμεση Αποθήκη Παλετών Εξόδου από το Κέντρο Τόρνευσης	Buffer_ Lathe _Pallets_ Output
12	Αποθήκη Ετοίμων τεμαχίων σε Παλέτες	Storage_Finish_Pallets
Τεμάχια (Πράσινο)		
13	Αποθήκη ακατέργαστων Τεμαχίων	Storage_Start_Units
14	Ενδιάμεση Αποθήκη Τεμαχίων Εισόδου στο Κέντρο Κατεργασιών	Buffer_Mill_Units_Input
15	Ενδιάμεση Αποθήκη Τεμαχίων Εξόδου από το Κέντρο Κατεργασιών	Buffer_Mill_Units_Output
16	Αποθήκη Λανθασμένων Τεμαχίων	Buffer_Wrong_Units
17	Ενδιάμεση Αποθήκη Τεμαχίων Εισόδου στο Κέντρο Τόρνευσης	Buffer_ Lathe _ Units_Input
18	Ενδιάμεση Αποθήκη Τεμαχίων Εξόδου από το Κέντρο Τόρνευσης	Buffer_ Lathe _Units_ Output
19	Αποθήκη Ετοίμων Τεμαχίων	Storage_Finish_Units
Κοινά (Κόκκινο)		
20	Σταθμός Διακίνησης	Transport
21	Ρομπότ	Robot
22	Σταθμός Ταυτοποίησης	ID
23	Κέντρο Κατεργασιών	Mill
24	Κέντρο Τόρνευσης	Lathe

Πίνακας 2-1 Σύνθεση του ΕΣΚ

Σημείωση: Όποια ονομασία έχει μαύρο χρώμα στο Σχήμα 2-2, αποτελεί κοινή ονομασία για τις παλέτες και τα τεμάχια.



Σχήμα 2-2 Ομαδοποίηση των Σταθμών του ΕΣΚ

Οι *κοινοί σταθμοί* είναι τα *«ενεργητικά»* στοιχεία του συστήματος, πλην του σταθμού ταυτοποίησης, καθώς είναι αυτοί που επιτελούν κάποια διεργασία, είτε διακινώντας τεμάχια/παλέτες/εργαλεία (Transport, Robot), είτε κατεργάζοντας τεμάχια (Mill, Lathe). Ενώ, οι υπόλοιποι σταθμοί αποτελούν *«παθητικά»* στοιχεία και είναι στο σύνολο τους αποθήκες, συν τον σταθμό ταυτοποίησης, από τα οποία διέρχονται τα τεμάχια/παλέτες/εργαλεία. Τα ενεργητικά στοιχεία εκτελούν σύνθετες διεργασίες και οι οποίες αναλύονται παρακάτω:

- *Εργασίες εκτελούμενες από το Σταθμό Διακίνησης (Transport)*

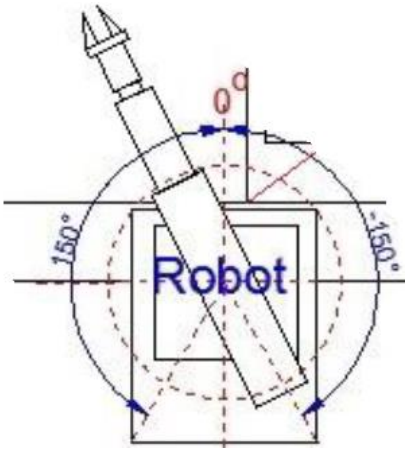
Ο σκοπός του σταθμού διακίνησης είναι η μεταφορά παλετών, τεμαχίων και εργαλείων. Από τον σκοπό της λειτουργίας του, γίνεται αντιληπτό πως πρόκειται για ένα Όχημα Αυτόματης Πλοήγησης (ΟΑΠ - AGV), το οποίο θα κινείται επί προκαθορισμένων διαδρομών, πάνω σε ταινιόδρομους (RGV) ή δρομολόγια με καλώδια ή με χρήση πομπών στον περιβάλλοντα χώρο (Εικόνα 2-2). Επίσης, ενδεχομένως να διαθέτει και ρομποτικό βραχίονα επί του οχήματος, για φορτο-εκφόρτωση των τεμαχίων/εργαλείων/παλετών συγκεκριμένων διαστάσεων και μορφής.[10]

Εικόνα 2-2 ΟΑΠ (AGV) σε Βιομηχανικό Περιβάλλον

- Μεταφορά εργαλείων του κέντρου κατεργασιών από το Tool_Storage_Mill στο Buffer_Tool_Mill
- Μεταφορά εργαλείων του κέντρου τόρνευσης από το Tool_Storage_Lathe στο Buffer_Tool_Lathe
- Μεταφορά παλετών από το Paletising_Station στο Storage_Start_Pallets
- Μεταφορά παλετών από το Storage_Start_Pallets στο Buffer_Mill_Pallets_Input
- Μεταφορά παλετών από το Buffer_ Lathe_Pallets_Output στο Storage_Finish_Pallets
- Μεταφορά τεμαχίων από το Storage_Start_Units στο Buffer_Mill_Units_Input
- Μεταφορά τεμαχίων από το Buffer_Lathe_Units_Output στο Storage_Finish_Units

- *Εργασίες εκτελούμενες από το Robot*

Πρόκειται για έναν ρομποτικό βραχίονα με εκτεταμένο προσπελάσιμο χώρο, καθώς πέραν των κινήσεων του βραχίονα (Εικόνα 2-3), το ρομπότ κινείται και πάνω σε ταινιόδρομο κατά μια διεύθυνση (Σχήμα 2-2). Επίσης, στο Τελικό Σημείο Δράσης (ΤΣΔ) διαθέτει αρπάγη για συγκράτηση τεμαχίων περιορισμένων διαστάσεων.



Εικόνα 2-3 Ροπμότ του ΕΣΚ

Για τις Παλέτες (Pallets):

- Φόρτωση τεμαχίων, που βρίσκονται στις παλέτες, από το Buffer_Mill_Pallets_Input στο ID και έπειτα, χωρίς να αποδεσμεύεται το Robot, φόρτωση στο Mill (αν είναι σωστή η ταυτοποίηση) ή στο Buffer_Wrong_Pallets (αν είναι λανθασμένη η ταυτοποίηση) και τα λανθασμένα παραμένουν εκεί.
- Εκφόρτωση τεμαχίων, που βρίσκονται στις παλέτες, από το Mill στο Buffer_Mill_Pallets_Output
- Μεταφορά της παλέτας, από το Buffer_Mill_Pallets_Output στο Buffer_ Lathe _Pallets_Input
- Φόρτωση τεμαχίων, που βρίσκονται στις παλέτες, από το Buffer_ Lathe_Pallets_Input στο Lathe
- Εκφόρτωση τεμαχίων, που βρίσκονται στις παλέτες, από το Lathe στο Buffer_Lathe _Pallets_Output

Για τα Τεμάχια (Units):

- Φόρτωση τεμαχίου από το Buffer_Mill_Units_Input στο ID και έπειτα, χωρίς να αποδεσμεύεται το Robot, φόρτωση στο Mill (αν είναι σωστή η ταυτοποίηση) ή στο Buffer_Wrong_Units (αν είναι λανθασμένη η ταυτοποίηση) και τα λανθασμένα παραμένουν εκεί.
- Εκφόρτωση τεμαχίου από το Mill στο Buffer_Mill_Units_Output
- Μεταφορά τεμαχίου από το Buffer_Mill_Units_Output στο Buffer_ Lathe _Units_Input
- Φόρτωση τεμαχίου από το Buffer_ Lathe _ Units_Input στο Lathe
- Εκφόρτωση τεμαχίου, από το Lathe στο Buffer_ Lathe _Units_Output

- *Εργασίες εκτελούμενες από το Κέντρο Κατεργασιών (Mill)*
  - Φρεζάρισμα των τεμαχίων, που βρίσκονται στις παλέτες
  - Φρεζάρισμα των ανεξάρτητων τεμαχίων
- *Εργασίες εκτελούμενες από το Κέντρο Τόρνευσης (Lathe)*
  - Τόρνευση των τεμαχίων, που βρίσκονται στις παλέτες
  - Τόρνευση των ανεξάρτητων τεμαχίων
- *Εργασίες εκτελούμενες από τον Σταθμό Ταυτοποίησης (ID)*

Τα τεμάχια μπορούν να ταυτοποιούνται μέσω visual servoing (αναγνώριση με κάμερα), ή αισθητήρων ή bar-code ή μέσω κάποιου άλλου χαρακτηριστικού. Η μέθοδος ταυτοποίησης δεν έχουν αποσαφηνιστεί και ξεφεύγει της μελέτης της παρούσας εργασίας. Οπότε, ο σταθμός πραγματοποιεί:

- Ταυτοποίηση των ανεξάρτητων τεμαχίων.
- Ταυτοποίηση των τεμαχίων, που είναι φορτωμένα στις παλέτες. Εδώ, μπορεί να ταυτοποιείται είτε ένα τεμάχιο από κάθε παλέτα και να εξάγεται το ίδιο συμπέρασμα για τα υπόλοιπα της παλέτας, το οποίο δεν είναι ασφαλές, είτε κάθε τεμάχιο από κάθε παλέτα.

2.3 Παραδοχές – Παραμετροποίηση

Οι επιπλέον Σταθμοί που επιλέχθηκαν για τη σύνθεση του ΕΣΚ, πέραν των αρχικώς καθορισμένων, έγιναν ώστε να καθίστανται διακριτές οι διαδικασίες που επιτελούνται και να επιτυγχάνεται ευελιξία στη συνολική παραγωγή. Επίσης, πρέπει να διευκρινιστεί πως οι επιπλέον σταθμοί δεν αποτελούν απαραίτητα «πραγματικά» μέρη του ΕΣΚ, αλλά μπορεί να είναι και εικονικά, προκειμένου να υλοποιούνται ανεξάρτητες μονάδες για τη μετάδοση σημάτων μεταξύ διακριτών μερών. Γενικά, οι σταθμοί του ΕΣΚ εξετάζονται ως προς τη δυνατότητα ανταλλαγής σημάτων μεταξύ τους και όχι ως προς τη δυνατότητα κατασκευής τους.

Συγκεκριμένα, η διάκριση έγινε μεταξύ:

- Παλετών – Τεμαχίων
- Εργαλείων – Τεμαχίων (ανεξάρτητων ή επί των παλετών)
- Εργαλείων Τόρνου - Φρέζας
- Αποθηκών Εισόδου – Εξόδου από και προς τα Κέντρα Κατεργασιών – Τόρνευσης
- Κατεργασιών στις δύο μηχανές κατεργασίας (Mill – Lathe).

Με την υφιστάμενη διάκριση επιτυγχάνονται διάφορα είδη ευελιξίας, καθώς αποφεύγεται η πολυπλοκότητα, που υπεισέρχεται από τη χρησιμοποίηση κοινών σταθμών. Όθεν, κυρίως μέσω των ενδιάμεσων αποθηκών και της διάκρισης των εργασιών ανεξάρτητων διαδικασιών, έχουμε δημιουργία όγκου προϊόντων υπό επεξεργασία (WIP), τα οποία αναμένουν στις ενδιάμεσες αποθήκες φόρτωσης (Buffers), επιτυγχάνοντας μείωση του άεργου χρόνου (idle time) των μηχανών κατεργασίας. Επιπλέον, το ΕΣΚ καθίσταται πιο εύκολα επεκτάσιμο, καθότι μπορούν να εισέλθουν/εξέλθουν σε/από αυτό νέοι σταθμοί, χωρίς να επηρεαστεί η συνολική του δομή και σύνθεση του ΕΣΚ. Όπως διευκρινίστηκε παραπάνω, *ο διαχωρισμός έγινε για προγραμματιστικούς λόγους κυρίως και όχι για την πραγματική υλοποίηση των συγκεκριμένων σταθμών στο ΕΣΚ*. Συγκεκριμένα:

- Δημιουργήθηκαν ξεχωριστές αποθήκες (αρχικές – ενδιάμεσες – τελικές) για τις παλέτες και τα τεμάχια, ώστε να υπάρχει *ευελιξία στην ποικιλία προϊόντων*. Αυτό συμβαίνει π.χ. όταν μπορεί το φασεολόγιο μιας παλέτας να περιλαμβάνει φρεζάρισμα – τόρνευση και να έχουμε παράλληλα ανεξάρτητα τεμάχια για φρεζάρισμα και τόρνευση. Με την υπάρχουσα δομή μπορεί να φορτωθεί η παλέτα στο Buffer_Mill_Pallets_Input για κατεργασία στο Mill και παράλληλα να φορτωθούν τα τεμάχια που χρήζουν κατεργασία φρεζαρίσματος και τόρνευσης στα αντίστοιχα Buffer_Mill_Units_Input και Buffer_ Lathe _ Units _Input, οπότε μπορούν να κατεργαστούν ανεξάρτητα τεμάχια στην μηχανή τόρνευσης, που δεν είναι δεσμευμένη και μόλις τελειώσει η κατεργασία φρεζαρίσματος της παλέτας, να φορτωθούν άμεσα τα ανεξάρτητα τεμάχια χωρίς την εμπλοκή του Transport. Απαραίτητη προϋπόθεση στα παραπάνω αποτελεί, οι παλέτες και τα τεμάχια να έχουν στο φασεολόγιο τους την ίδια σειρά κατεργασιών. Γενικά, η διακίνηση των τεμαχίων και των παλετών ακολουθά ξεχωριστή πορεία στο ΕΣΚ, διότι τα ανεξάρτητα τεμάχια συνήθως αποτελούν ειδικές παραγγελίες, οι οποίες ξεφεύγουν από την ρουτίνα που λειτουργεί το ΕΣΚ.

- Επίσης, δημιουργήθηκαν ξεχωριστές αποθήκες εργαλείων από αυτές των τεμαχίων και των παλετών, αλλά και των εργαλείων μεταξύ τους (τόρνου – φρέζας), διότι ενδέχεται κάποιο εργαλείο να χρησιμοποιείται σε περισσότερες κατεργασίες από κάποιας συγκεκριμένης παλέτας ή τεμαχίου. Επίσης, για να μην υπάρχει αναμονή του Transport, σε περίπτωση που μεταφέρει εργαλεία για το Mill και το Lathe, μπορεί το Transport να αφήσει το εργαλείο φρέζας στην ενδιάμεση αποθήκη του Mill, στη συνέχεια να τοποθετήσει το εργαλείο τόρνου σε αντίστοιχη αποθήκη του Lathe και μετά να μείνει ελεύθερο, πετυχαίνοντας *ευελιξία διαδρομής*.
- Ακόμη, προβλέφθηκε οι αποθήκες εισόδου – εξόδου για τα τεμάχια και τις παλέτες να είναι διαφορετικές, κυρίως για την περίπτωση που έχουμε παλέτα ή τεμάχιο που χρειάζεται κατεργασία και στις δύο μηχανές. Όταν τελειώσει η κατεργασία της παλέτας στο Mill και πρέπει να συνεχίσει στο Lathe, και υπάρχει ήδη παλέτα που κατεργάζεται στο Lathe δεσμεύοντας το Buffer_Lathe_Pallets_Input, τότε η παλέτα που κατεργάστηκε στο Mill μεταφέρεται μέσω του ρομπότ στην Buffer_Mill_Pallets_Output. Παράλληλα, μπορούμε να έχουμε ήδη φορτωμένη νέα παλέτα στο Buffer_Mill_Pallets_Input, της οποίας η κατεργασία στο Mill μπορεί να ξεκινήσει απευθείας μόλις αυτό αποδεσμευθεί. Έτσι επιτυγχάνουμε μείωση του άεργου χρόνου των μηχανών και πετυχαίνουμε *ευελιξία όγκου παραγωγής*.
- Τέλος, οι δύο μηχανές είναι ανεξάρτητες μεταξύ τους και δύνανται να κατεργαστούν τεμάχια ή παλέτες, που στο φασεολόγιο τους έχουν κατεργασίες και στις δύο μηχανές. Με τη συγκεκριμένη διάκριση των κατεργασιών όμως, δεν εμπλέκονται και οι δύο μηχανές ταυτόχρονα, αλλά κάθε τεμάχιο εμπλέκει τη μηχανή στην οποία κατεργάζεται τη δεδομένη χρονική στιγμή, αφήνοντας την άλλη ελεύθερη να κατεργαστεί κάποιο άλλο κομμάτι.

Για τα τεχνικά και λειτουργικά χαρακτηριστικά των σταθμών που συνθέτουν το υπόψη ΕΣΚ έγιναν κάποιες *παραδοχές*, για λόγους γενίκευσης και δυνατότητας μεταβολής αυτών, όπως:

- Οι διαστάσεις που μπορούν να επεξεργαστούν οι σταθμοί κατεργασιών.
- Ο αριθμός και οι διαστάσεις του κάθε είδους (παλετών/τεμαχίων/εργαλείων), που μπορεί να μεταφέρει ο σταθμός διακίνησης (*Transport*), καθώς και η ικανότητα του να μεταφέρει σε ξεχωριστές θέσεις διαφορετικά είδη (τεμάχια/εργαλεία/παλέτες).
- Η δυνατότητα εκφόρτωσης των εργαλείων στα κέντρα κατεργασιών και τόρνευσης.
- Η χωρητικότητα σε αριθμό και διαστάσεις των τεμαχίων κάθε παλέτας.
- Οι χωρητικότητες σε τεμάχια/παλέτες/εργαλεία των αρχικών/ενδιάμεσων/τελικών αποθηκών.
- Ο τρόπος λήψης και τοποθέτησης των τεμαχίων από το ρομπότ που είναι πιθανόν να διαφοροποιείται σε έκαστο τεμάχιο.
- Η δυνατότητα μεταφοράς παλέτας από το ρομπότ.

Τα περισσότερα από τα παραπάνω, μπορούν να *παραμετροποιηθούν* θέτοντας μια παραδοχή, ένα πρότυπο μέγεθος, τη *μερίδα*, το οποίο θα γίνει καλύτερα κατανοητό και σε επόμενα κεφάλαια κατά την προσομοίωση του ΕΣΚ μέσω των Δικτύων Petri και κατά τον καταρτισμό του προγράμματος Arduino. Περιληπτικά, αναφέρεται ότι χωρίζουμε όλα τα είδη (εργαλείο, τεμάχιο, παλέτα) σε μερίδες. Δηλαδή, ένα

τεμάχιο αντιπροσωπεύει μια μερίδα τεμαχίων, μια παλέτα διαθέτει x αριθμό τεμαχίων που αντιπροσωπεύει μια μερίδα παλετών και ένα εργαλείο αντιπροσωπεύει μια μερίδα εργαλείων. Όσον αφορά τις χωρητικότητες των αρχικών και τελικών αποθηκών όλων των ειδών (εργαλείο, τεμάχιο, παλέτα), ο αριθμός τους μπορεί να ποικίλει, αναλόγως του χρησιμοποιούμενου AS/RS. Ακόμη, οι χωρητικότητες μπορεί να εισάγονται στο πρόγραμμα κάθε φορά από τον χρήστη πριν από την εκκίνηση εκτέλεσης μια συγκεκριμένης παραγωγής. Για τις ενδιάμεσες αποθήκες (Buffers) όλων των ειδών εισόδου - εξόδου, το ρομπότ, τον σταθμό ταυτοποίησης και τα κέντρα κατεργασίας (Mill – Lathe), θεωρούμε πως η χωρητικότητα τους είναι μία μερίδα. Εξαίρεση αποτελεί ο σταθμός διακίνησης (Transport) που θεωρούμε, πως διαθέτει χωρητικότητα για 1 μερίδα τεμαχίων, 1 μερίδα παλετών και 1 μερίδα εργαλείων, αλλά η διακίνηση κάθε είδους από το Transport γίνεται ξεχωριστά. Τέλος, για τις αποθήκες των λανθασμένων τεμαχίων και παλετών (Buffer_Wrong) η χωρητικότητα δεν παίζει κάποιο ρόλο, αλλά την τοποθετούμε ίση με 1 μερίδα τεμαχίων και παλετών αντίστοιχα, για προγραμματιστικούς λόγους. Σύμφωνα με τα παραπάνω, μπορούμε να συγκροτήσουμε τον παρακάτω Πίνακα 2-2, με τις χωρητικότητες των σταθμών σε αριθμό μερίδων:

Α/Α	***Σταθμός***	***Ονομασία Σταθμού***	***Χωρητικότητα - Αριθμός Μερίδων***	***Είδος***
1	Αποθήκη Εργαλείων Κέντρου Κατεργασιών	Tool_Storage_Mill	Σταθερός	Εργαλεία
2	Αποθήκη Εργαλείων Κέντρου Τόρνευσης	Tool_Storage_Lathe	Σταθερός	-//-
3	Ενδιάμεση Αποθήκη Εργαλείων Κέντρου Κατεργασιών	Buffer_Tool_Mill	1	-//-
4	Ενδιάμεση Αποθήκη Εργαλείων Κέντρου Τόρνευσης	Buffer_Tool_Lathe	1	-//-
5	Σταθμός Παλετοποίησης	Paletising_Station	Μεταβλητός	Παλέτες
6	Αποθήκη α’ υλών (κενές θέσεις) σε Παλέτες	Storage_Start_Pallets	Σταθερός	-//-
7	Ενδιάμεση Αποθήκη Παλετών Εισόδου στο Κέντρο Κατεργασιών	Buffer_Mill_Pallets_Input	1	-//-
8	Ενδιάμεση Αποθήκη Παλετών Εξόδου από το Κέντρο Κατεργασιών	Buffer_Mill_Pallets_Output	1	-//-
9	Αποθήκη Λανθασμένων Παλετών	Buffer_Wrong_Pallets	1	-//-
10	Ενδιάμεση Αποθήκη Παλετών Εισόδου στο Κέντρο Τόρνευσης	Buffer_ Lathe _Pallets_Input	1	-//-
11	Ενδιάμεση Αποθήκη Παλετών Εξόδου από το Κέντρο Τόρνευσης	Buffer_ Lathe _Pallets_ Output	1	-//-

Α/Α	*Σταθμός*	*Ονομασία Σταθμού*	*Χωρητικότητα - Αριθμός Μερίδων*	*Είδος*
12	Αποθήκη Ετοίμων τεμαχίων σε Παλέτες (κενές θέσεις)	Storage_Finish_Pallets	Σταθερός	-//-
13	Αποθήκη ακατέργαστων Τεμαχίων	Storage_Start_Units	Σταθερός	Τεμάχια
14	Ενδιάμεση Αποθήκη Τεμαχίων Εισόδου στο Κέντρο Κατεργασιών	Buffer_Mill_Units_Input	1	-//-
15	Ενδιάμεση Αποθήκη Τεμαχίων Εξόδου από το Κέντρο Κατεργασιών	Buffer_Mill_Units_Output	1	-//-
16	Αποθήκη Λανθασμένων Τεμαχίων	Buffer_Wrong_Units	1	-//-
17	Ενδιάμεση Αποθήκη Τεμαχίων Εισόδου στο Κέντρο Τόρνευσης	Buffer_ Lathe _ Units_Input	1	-//-
18	Ενδιάμεση Αποθήκη Τεμαχίων Εξόδου από το Κέντρο Τόρνευσης	Buffer_ Lathe _Units_ Output	1	-//-
19	Αποθήκη Ετοίμων Τεμαχίων (κενές θέσεις)	Storage_Finish_Units	Σταθερός	-//-
20	Σταθμός Διακίνησης - (ΟΑΠ)	Transport	1 μερίδα παλετών 1 μερίδα τεμαχίων 1 μερίδα εργαλείων	
21	Ρομπότ	Robot	1	Παλέτα - Τεμάχιο
22	Σταθμός Ταυτοποίησης	ID	1	-//-
23	Κέντρο Κατεργασιών	Mill	1	-//-
24	Κέντρο Τόρνευσης	Lathe	1	-//-

Πίνακας 2-2 Χωρητικότητες των Σταθμών του ΕΣΚ

Τέλος, μεταξύ των διεργασιών υφίστανται κανόνες προτεραιότητας, κυρίως ως προς τη διαδοχή του είδους (εργαλείο/τεμάχιο/παλέτα) που θα μεταφερθεί πρώτο, μέσω του Transport. Έτσι, θέτουμε την παρακάτω *σειρά προτεραιότητας*:

1. Εργαλεία
2. Παλέτες
3. Τεμάχια

Τα εργαλεία είναι πρώτης προτεραιότητας, διότι πρέπει να τοποθετηθούν πρώτα στις μηχανές κατεργασίας, προκειμένου να κατεργαστούν τα τεμάχια, έπονται οι παλέτες και τελευταία μεταφέρονται τα τεμάχια. Οι παλέτες προηγούνται των τεμαχίων, καθότι πέραν του ότι διαθέτουν περισσότερα τεμάχια και κατ' επέκταση δημιουργούν μεγαλύτερο όγκο προϊόντων υπό-επεξεργασία (wip), επιπλέον οι μηχανές δεν χρειάζονται set-up για κάθε τεμάχιο, αλλά αυτό αλλάζει όταν κατεργαστεί ο αριθμός παλετών μιας παραγγελίας. Σε αντίθεση στα τεμάχια, ενδέχεται μια παραγγελία να περιλαμβάνει μόνο ένα τεμάχιο και

να χρειάζεται νέο set-up οι μηχανές για κάθε κατεργασία ενός τεμαχίου, το οποίο είναι αρκετά χρονοβόρο, ιδιαιτέρως σε πολύπλοκα κομμάτια.

2.4 Παραγωγική Διαδικασία

Η παραγωγική διαδικασία, περιλαμβάνει όλες τις εργασίες που υλοποιούνται στο ΕΣΚ (μεταφορά, φόρτωση/εκφόρτωση, κατεργασίες) κατά σειρά από την στιγμή που θα προγραμματιστεί το ΕΣΚ για μία ανατιθέμενη σε αυτό παραγγελία. Το πρόγραμμα που θα φορτωθεί στο ΕΣΚ για την υλοποίηση μιας παραγγελίας διαφέρει, μόνο σε μία παράμετρο, τη σειρά κατεργασιών. Στο σχεδιασμένο ΕΣΚ της παρούσας εργασίας, δύναται να υπάρχουν οι παρακάτω συνδυασμοί κατεργασιών:

1. Τόρνευση
2. Τόρνευση-Φρεζάρισμα
3. Τόρνευση-Φρεζάρισμα- Τόρνευση
4. Φρεζάρισμα
5. Φρεζάρισμα- Τόρνευση
6. Φρεζάρισμα- Τόρνευση- Φρεζάρισμα

Οπότε, συγκροτούνται έξι (6) διαφορετικά προγράμματα παραγωγικής διαδικασίας, αναλόγως του συνδυασμού κατεργασιών και φορτώνεται κάθε φορά το επιθυμητό. Οι υπόλοιπες παράμετροι παραμένουν ίδιοι και στα 6 προγράμματα. Ο περιορισμός αυτός έγκειται και αυτός, για προγραμματιστικούς λόγους, καθώς το πρόγραμμα ακολουθεί μια συγκεκριμένη ακολουθία εργασιών (ρουτίνα), οπότε ενώ οι αριθμοί παλετών/τεμαχίων/εργαλείων μπορούν να εισάγονται, με τον ίδιο ακριβώς τρόπο και στις 6 περιπτώσεις, από τον χρήστη, η ακολουθία κατεργασιών δεν μπορεί να αναπροσαρμόζεται στο ίδιο πρόγραμμα. Περαιτέρω επεξηγήσεις παρατίθενται στα *Κεφάλαια 4 και 6*.

Τα απαραίτητα δεδομένα που πρέπει να δοθούν στο πρόγραμμα του ΕΣΚ, προκειμένου να πραγματοποιήσει μια παραγωγική διαδικασία είναι:

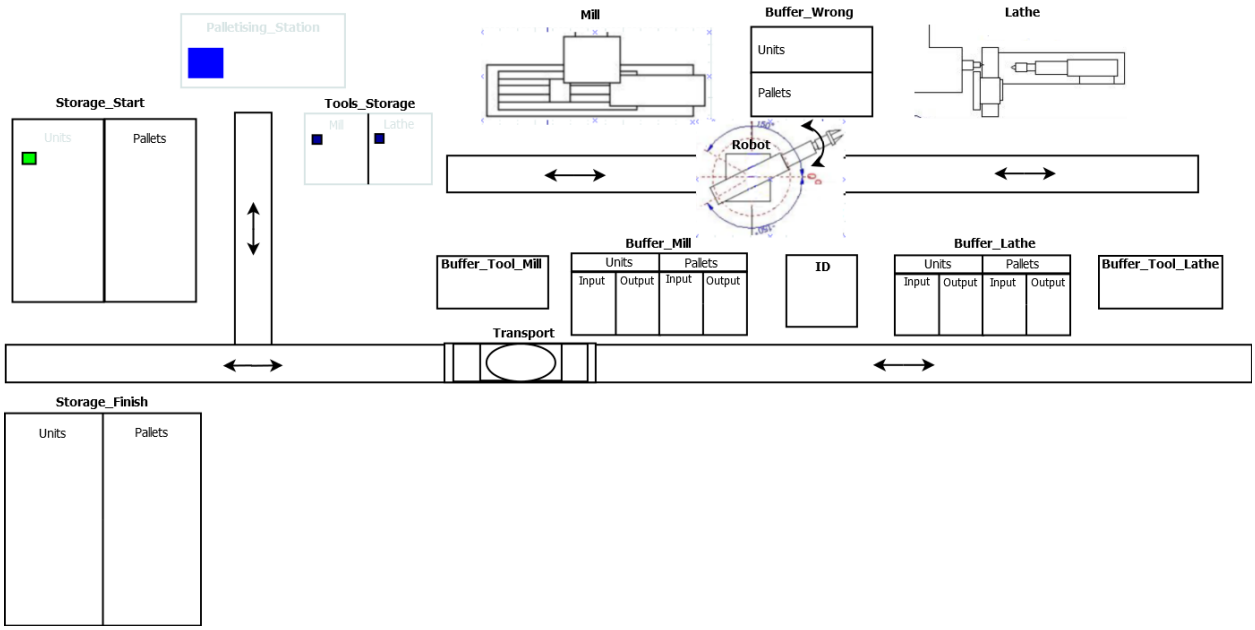
- Η Σειρά Κατεργασιών από τα φασεολόγια παλετών και τεμαχίων (επιλογή ενός από τα έξι προγράμματα παραγωγικής διαδικασίας)
- Ο Αριθμός Παλετών (μερίδες)
- Ο Αριθμός Τεμαχίων (μερίδες)
- Ο Αριθμός Κοπτικών Εργαλείων (ΚΕ) (μερίδα) για το Mill, αν χρειάζονται
- Ο Αριθμός Κοπτικών Εργαλείων (ΚΕ) (μερίδα) για το Lathe, αν χρειάζονται

Το ΕΣΚ κατά την εκτέλεση του προγράμματος, μπορεί να εκτελεί ταυτόχρονα έναν από τους παραπάνω συνδυασμούς κατεργασιών για παλέτες και τεμάχια, όπως προαναφέρθηκε. Καθώς, όταν φορτώνεται το πρόγραμμα στους μικρο-ελεγκτές, έχει μια συγκεκριμένη σειρά εκτέλεσης των εντολών, άρα και τα τεμάχια και οι παλέτες θα πρέπει να ακολουθήσουν την καθορισμένη σειρά εκτέλεσης των εργασιών, βάσει

των εντολών που δίνονται από το πρόγραμμα. Οπότε, μια πιθανή παραγωγική διαδικασία του ΕΣΚ θα μπορούσε να είναι η παρακάτω με τα εξής χαρακτηριστικά:

- Κατεργασίες φρεζαρίσματος – τόρνευσης
- 1 Εργαλείο για Mill
- 1 Εργαλείο για Lathe
- 1 (μερίδα) Παλέτα
- 1 (μερίδα) Τεμάχιο

Οι εμπλεκόμενοι σταθμοί του ΕΣΚ, στην αρχική φάση, είναι όπως στο Σχήμα 2-3. Με κόκκινα γράμματα θα εμφανίζονται οι σταθμοί που είναι δεσμευμένοι, με μαύρο οι ελεύθεροι, ενώ με γκρι οι σταθμοί που θα έχουν δεσμευμένες θέσεις αλλά και κενές (πρόκειται για τους σταθμούς με μεταβλητή χωρητικότητα, βάσει του Πίνακα 2-2). Τα χρωματιστά κουτάκια προσομοιάζουν το είδος (εργαλείο/τεμάχιο/παλέτα) της ομάδας του αντικειμένου και η χρωματική αντιστοίχιση είναι αυτή που δόθηκε παραπάνω (Σχήμα 2-2), σκούρο μπλε για εργαλεία, ανοιχτό μπλε για παλέτες και πράσινο για τεμάχια.



Σχήμα 2-3 Παράδειγμα Παραγωγικής Διαδικασίας του ΕΣΚ

Η λειτουργία του εξεταζόμενου ΕΣΚ παρατίθεται αναλυτικά και στο 4ο κεφάλαιο, στο οποίο γίνεται και παράλληλη προσομοίωση του δικτύου Petri του ΕΣΚ που σχεδιάστηκε και υλοποιείται η παραπάνω παραγωγική διαδικασία.

3

Δίκτυα Petri (Petri Nets – PN)

3.1 Γενικά Στοιχεία

Ο έλεγχος Συστημάτων Κατεργασιών πραγματοποιείται μέσω της υλοποίησης κάποιας λογικής, η οποία για τα αυτοματοποιημένα συστήματα ενσωματώνεται σε ένα πρόγραμμα λογισμικού και εκτελείται συνήθως σε Ελεγκτές Προγραμματισμένης Λογικής (PLC) ή βιομηχανικού τύπου PC ή κάποιου είδους μικρο-ελεγκτές. Στο παρόν Κεφάλαιο παρουσιάζεται ο σχεδιασμός της λογικής του συστήματος ελέγχου με χρήση σαν φορμαλιστικών εργαλείων τα *Δίκτυα Petri*.[11]

Τα *Δίκτυα Petri* επινοήθηκαν κατά τη διδακτορική διατριβή *("Kommunikation mit Automaten" /"Communication with Automata")* του *Carl Adam Petri* στο Technical University of Darmstadt της Δυτικής Γερμανίας, το 1962, ως μία τεχνική μοντελοποίησης βασισμένη στην ιδέα ενός συστήματος ασύγχρονων και ταυτόχρονων λειτουργιών και της υλοποίησης των σχέσεων που συνδέουν αυτές τις λειτουργίες σε ένα δίκτυο. Ο Petri περιέγραψε γραφικά, με μορφή δικτύου, τη μαθηματική σχέση μεταξύ γεγονότων και καταστάσεων σε ένα σύστημα υπολογιστών, δημιουργώντας έτσι ένα μαθηματικό εργαλείο για τη μελέτη της επικοινωνίας μεταξύ αυτόματων μηχανών. Αργότερα, οι Holt and Commoner (1970) εφάρμοσαν τα Δίκτυα Petri για την μοντελοποίηση και ανάλυση συστημάτων που περιλάμβαναν παράλληλες διαδικασίες.[12]

Τη δεκαετία του '70 τα Δίκτυα Petri απετέλεσαν μια ιδιαιτέρως ενεργή περιοχή με τις περισσότερες μελέτες να αφορούν σε συστήματα επεξεργασίας δεδομένων για ηλεκτρονικούς υπολογιστές. Στις αρχές της δεκαετίας του '80, η χρήση δικτύων Petri επεκτάθηκε σε εφαρμογές μηχανικών, και κυρίως στο πεδίο των Αυτόματων Συστημάτων Κατεργασιών, έως ότου ανακαλύφθηκε ότι τα δίκτυα Petri είναι ένα ισχυρό εργαλείο στην περιγραφή συστημάτων που εξαρτώνται από γεγονότα (event driven systems).[13]

Αυτόματα συστήματα κατεργασιών, συστήματα επικοινωνιών, συστήματα υπολογιστών ή συστήματα διαχείρισης πληροφοριών, που μπορεί να είναι ασύγχρονα και να περιέχουν σειριακές ή παράλληλες εργασίες, περιλαμβάνουν παραλληλισμό, συγκρούσεις, αμοιβαίο αποκλεισμό, και μη ντετερμινιστικές μεθόδους χαρακτηρίζονται ως *Συστήματα Διακριτών Γεγονότων (Discrete Event Systems - DES* ή *Δυναμικά Συστήματα Διακριτών Γεγονότων (Discrete Event Dynamic Systems - DEDS)*. Τα DES ή DEDS

περιγράφονται γραφικά από τα Δίκτυα Petri, καθότι περιέχουν χαρακτηριστικά που δεν μπορούν να περιγραφούν από την κλασσική θεωρία ελέγχου, η οποία ασχολείται με συνεχείς ή ασύγχρονες διακριτές μεταβλητές που μοντελοποιούν τα συστήματα σε διαφορικές εξισώσεις.[14]

Αρχικά, τα δίκτυα Petri χρησιμοποιήθηκαν για την αναπαράσταση απλών γραμμών παραγωγής με ουρές, μηχανουργείων, αυτοματοποιημένων συστημάτων παραγωγής, και εν συνεχεία χρησιμοποιήθηκαν για την μοντελοποίηση Ευέλικτων Συστημάτων Παραγωγής (ΕΣΚ), αυτοματοποιημένων γραμμών συναρμολόγησης, συστημάτων με κοινή χρήση/διαχείριση πόρων, και μετέπειτα σε συστήματα παραγωγής με μεθόδους Just-In-Time (JIT) και Kanban (Toyota). Έως τώρα, τα μοντέλα δικτύων Petri έχουν χρησιμοποιηθεί με σκοπό την ανάλυση, αποτίμηση επιδόσεων και έλεγχο συστημάτων σε μια πληθώρα εφαρμογών από διαφορετικά επιστημονικά πεδία, όπως συστήματα λογισμικού, βιομηχανικά συστήματα ελέγχου παραγωγής, συστήματα ελέγχου κυκλοφορίας, χημικές διαδικασίες, συστήματα επικοινωνιών και συστήματα πληροφοριών.[15]

Στα Συστήματα Κατεργασιών, το ενδιαφέρον για τα Δίκτυα Petri ξεκίνησε από την ανάγκη προσδιορισμού και μοντελοποίησης διακριτών συστημάτων παραγωγής. Τα βασικά χαρακτηριστικά σε αυτά τα συστήματα, είναι *η παραλληλία* και *ο μη ντετερμινισμός*, τα οποία συνεπάγονται ότι η εκτέλεση ενεργειών, στα συστήματα, μπορεί να γίνει με πολλούς διαφορετικούς τρόπους. Είναι εύκολο, κατά τη σχεδίαση τέτοιων συστημάτων να αγνοηθούν σημαντικά σημεία αλληλεπίδρασης, οδηγώντας σε εσφαλμένη λειτουργία κατά την εκτέλεσή τους. Οπότε, για να αντιμετωπιστεί η πολυπλοκότητα των σύγχρονων παράλληλων συστημάτων είναι βασικό να παρέχονται μέθοδοι που επιτρέπουν τον έλεγχο και την επιδιόρθωση πριν από την εφαρμογή τους. Ένας τρόπος προσέγγισης του συγκεκριμένου προβλήματος λοιπόν, είναι να δομηθεί ένα εκτελέσιμο μοντέλο του συστήματος και με την προσομοίωση του, να δώσει μια ολοκληρωμένη εικόνα στο σχεδιασμό και στη λειτουργία του συστήματος.[16]

Συνολικά, θα μπορούσαμε να πούμε ότι τα Δίκτυα Petri αποτελούν ένα γραφικό εργαλείο για τον σχεδιασμό *Συστημάτων Διακριτών Γεγονότων* (DES) και παρουσιάζουν τα παρακάτω πλεονεκτήματα:

- Ευκολία στη μοντελοποίηση πολύπλοκων βιομηχανικών συστημάτων με: παράλληλες, σύγχρονες και ασύγχρονες λειτουργίες, συγκρούσεις , αμοιβαίο αποκλεισμό, σχέσεις προτεραιότητας, μη ντετερμινισμό και αδιέξοδα συστημάτων (deadlocks).
- Δυνατότητα δημιουργίας κώδικα κατευθείαν από τη γραφική παράσταση του Δικτύου Petri. Ένας εκτελέσιμος κώδικας δικτύου Petri μπορεί επίσης να δημιουργηθεί σε εφαρμογές πραγματικού χρόνου χρησιμοποιώντας προγραμματιζόμενους λογικούς ελεγκτές (PLC) ή υπολογιστές.
- Δυνατότητα διερεύνησης βασικών ιδιοτήτων του συστήματος, όπως η ύπαρξη αδιεξόδων, μέσω ανάλυσης σε υπολογιστικά προγράμματα βασισμένα στα Δίκτυα Petri, αποφεύγοντας πολυάριθμες προσομοιώσεις για διάφορα σενάρια λειτουργίας του συστήματος.
- Δυνατότητα ανάλυσης λειτουργίας και απόδοσης του συστήματος αξιολογώντας χρόνους παραγωγής, χρόνους αναμονής, βαθμό χρήσης πόρων.
- Η προσομοίωση είναι δυνατόν να οδηγείται από το ίδιο το μοντέλο.

- Η παρακολούθηση, ο έλεγχος και η διόρθωση σφαλμάτων του σχεδιαζόμενου συστήματος, πριν την εφαρμογή του μοντέλου στο πραγματικό σύστημα, μέσω της προσομοίωσης με τα Δίκτυα Petri.

3.2 Ορισμός Μοντελοποίησης – Δομή – Σήμανση

Ένα Δίκτυο Petri, αποτελεί ένα *γραφικό και μαθηματικό εργαλείο* για τη σχεδίαση και την ανάλυση συστημάτων διακριτών γεγονότων (DES), χρήσιμο για τη θεωρητική έρευνα, αλλά και τον πρακτικό σχεδιασμό ελέγχου των συστημάτων αυτών. Παρακάτω, θα αναλυθούν και τα μέρη των Δικτύων Petri, με το γραφικό μέρος των δικτύων να είναι γενικά πιο απλό και κατανοητό, σε σχέση με το μαθηματικό, καθώς η γραφική προσομοίωση ενός Δικτύου αναπαριστά με μεγαλύτερη ομοιότητα την πραγματική λειτουργία ενός ΕΣΚ.

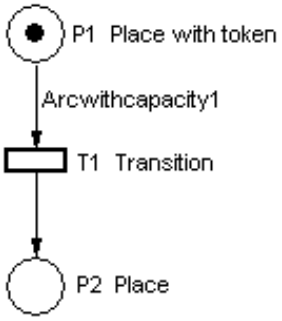
Γραφική Μοντελοποίηση

Τα βασικά στοιχεία που συνθέτουν τη *Δομή* ενός Δικτύου Petri είναι τα παρακάτω (Εικόνα 3-1):

- Οι *θέσεις (Places)*, που εκφράζουν συνθήκες, κατάσταση ενός συστατικού, ή μια εργασία του συστήματος και παριστάνονται με κύκλους.
- Οι *μεταβάσεις (Transitions)*, που εκφράζουν συνήθως γεγονότα και παριστάνονται με τετράγωνα ή με κατακόρυφες γραμμές. Δύο χαρακτηριστικά γεγονότα που μπορεί να περιέχει μία μετάβαση είναι η «έναρξη» και η «λήξη» κάποιας κατάστασης.
- Τα *τόξα ή κλάδοι (Arcs)*, που συνδέουν τις θέσεις με τις μεταβάσεις και παριστάνονται με προσανατολισμένα βέλη. Ένα τόξο μπορεί να έχει κατεύθυνση από μια θέση προς μια μετάβαση (τόξο *εισόδου* στη μετάβαση) ή από μια μετάβαση σε μια θέση (τόξο *εξόδου* από τη μετάβαση) (*one-way*). Επίσης, ένα τόξο μπορεί να έχει ταυτόχρονα και τις δύο κατευθύνσεις (διπλό τόξο). Μια μεταφορά δύο δρόμων (*two-way*) επιτυγχάνεται με ένα τόξο από μια θέση σε μια μετάβαση, και εν συνεχεία με ένα άλλο τόξο από την μετάβαση στην αρχική θέση. Αυτό αποτελεί και έναν *κλειστό βρόγχο (self-loop)*. Τέλος, υπάρχουν και οι αποτρεπτικοί κλάδοι, από μια θέση σε μια μετάβαση και συμβολίζεται με γραμμή και κύκλο στην άκρη αντί βέλους.
- Τα *κουπόνια (Tokens)*, που βρίσκονται στις θέσεις, οι οποίες μπορεί να έχουν κανένα ή θετικό αριθμό κουπονιών. Τα κουπόνια παριστάνονται με μικρούς γεμάτους κύκλους (βούλες).

Εικόνα 3-1 Δομικά Στοιχεία των Δικτύων Petri

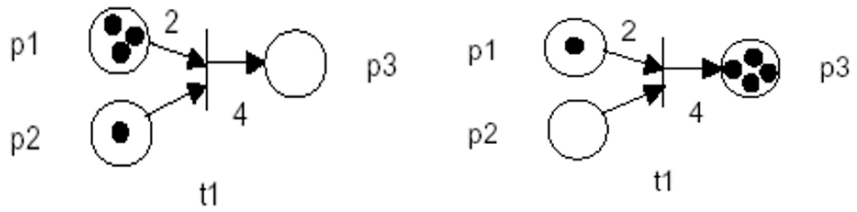
Οι θέσεις και οι μεταβάσεις, συνδεδεμένες μεταξύ τους με τόξα, διαμορφώνουν ένα *προσανατολισμένο γράφημα*, το οποίο αναπαριστά τη *Δομή* του Δικτύου Petri (Εικόνα 3-2). Τα κουπόνια μέσα σε κάποια θέση μπορούν να υποδηλώνουν τον αριθμό των πόρων, αν μια συνθήκη είναι αληθής, αν μια εργασία βρίσκεται σε εξέλιξη ή ακόμα αν κάποιος πόρος είναι δεσμευμένος ή ελεύθερος. Αυτό εξαρτάται από το τι υποδηλώνει η θέση στην οποία βρίσκονται. Όταν όλες οι θέσεις εισόδου μιας μετάβασης διαθέτουν ικανό αριθμό κουπονιών, τότε το γεγονός που προσομοιάζεται από την μετάβαση μπορεί να πραγματοποιηθεί. Αυτό αποκαλείται ενεργοποίηση μετάβασης (*transition firing*). Η ενεργοποίηση της μετάβασης αλλάζει τη διανομή των κουπονιών στις θέσεις, δηλώνοντας την αλλαγή κατάστασης του συστήματος (Εικόνα 3-3). [13]



Εικόνα 3-2 Δομή του πιο απλού Δικτύου Petri

Εικόνα 3-3 Δίκτυο Petri Πριν (αριστερά) και Μετά (δεξιά) την ενεργοποίηση μετάβασης (firing)

Ουσιαστικά, οι θέσεις κατέχουν κουπόνια και συνδέονται αποκλειστικά με μεταβάσεις μέσω κάποιων κλάδων, και έπειτα οι μεταβάσεις συνδέονται και αυτές αποκλειστικά μόνο με άλλες ή και τις ίδιες θέσεις, μέσω κάποιων άλλων κλάδων. Επίσης, οι θέσεις *από και προς* τις μεταβάσεις μπορούν να συνδέονται μεταξύ τους με περισσότερους από έναν κλάδους. Οι κλάδοι έχουν βάρος (weighted) και είναι ο αριθμός των κουπονιών, που πρέπει να περάσουν από μια θέση, μέσω του κλάδου, για να ενεργοποιηθεί μια μετάβαση. Δηλαδή, αν ένας μόνο κλάδος συνδέει μια θέση με μια μετάβαση και αυτός έχει βάρος 2, τότε για να ενεργοποιηθεί η μετάβαση θα πρέπει στη θέση να έχουμε 2 κουπόνια τουλάχιστον. Στην Εικόνα 3-4, έχουμε κλάδους εισόδου με βάρη 2 και 1, ενώ για τον κλάδο εξόδου έχουμε βάρος 4. Αυτό σημαίνει, πως χρειαζόμαστε 2 τουλάχιστόν κουπόνια στη θέση p1 και 1 στη θέση p2 για να ενεργοποιηθεί η μετάβαση t1. Μόλις ενεργοποιηθεί η μετάβαση τότε, λόγω του κλάδου εξόδου με βάρος 4, δημιουργούνται 4 κουπόνια στη θέση p3.



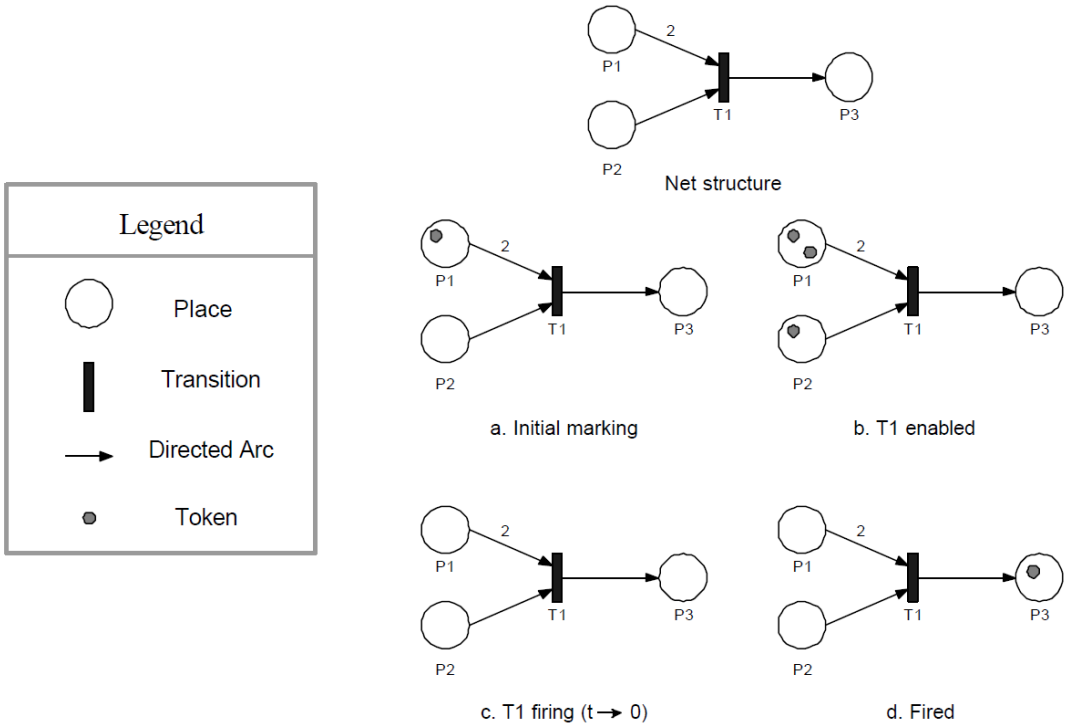
Εικόνα 3-4 Ενεργοποίηση Μετάβασης (firing) μέσω κλάδων με βάρος >1 (weighted arc)

Γενικά, μια μετάβαση ενεργοποιείται (firing), όταν οι θέσεις, που συνδέονται με αυτήν μέσω κλάδων, έχουν τουλάχιστον τόσα κουπόνια όσα τα βάρη των κλάδων που τις συνδέουν. Μετά την ενεργοποίηση της μετάβασης τα κουπόνια της θέσης χάνονται από την συγκεκριμένη θέση. Αν η μετάβαση είναι συνδεδεμένη μετέπειτα με άλλ-η/ες θέσ-η/εις, τότε δημιουργούνται άλλα κουπόνια στις θέσεις αυτές, αναλόγως του βάρους των κλάδων που συνδέουν τη μετάβαση με τις νέες θέσεις. Ακόμη, σημειώνεται ότι, οι θέσεις, *by default*, έχουν θεωρητικά άπειρη χωρητικότητα αποθήκευσης κουπονιών, εκτός αν εμείς θέσουμε κάποιον περιορισμό. Ενώ αντιθέτως, οι μεταβάσεις εξ ορισμού δεν μπορούν να έχουν και να αποθηκεύσουν κουπόνια, καθώς η ενεργοποίηση τους είναι στιγμιαία στα κλασσικά Petri Nets.

Κατόπιν των παραπάνω, θα μπορούσαμε να πούμε πως ένα Δίκτυο Petri αποτελείται από 2 μέρη:

- Τη *Δομή (C) του δικτύου*, που αναπαριστά το *στατικό* μέρος του συστήματος, δηλαδή τη σύνθεση και τον τρόπο σύνδεσης των επιμέρους σταθμών ενός συστήματος μεταξύ τους (τοπολογία).
- Τη *Σήμανση (M)* του δικτύου, που αναπαριστά τη συνολική κατάσταση (state) του συστήματος (συμπεριφορά), η οποία μεταβάλλεται και δείχνει τη *δυναμική* συμπεριφορά του συστήματος. Η σήμανση υλοποιεί τους κανόνες ενεργοποίησης των μεταβάσεων σε ένα Δίκτυο Petri.

Στην Εικόνα 3-5 παρουσιάζεται η δομή, η αρχική σήμανση και η ροή κουπονιών που μεταβάλλει τη σήμανση, σε ένα Δίκτυο Petri.



Εικόνα 3-5 Γραφική Δομή - Σήμανση - Ροή κουπονιών ενός Δικτύου Petri [8]

Στη μοντελοποίηση ενός συστήματος, αρχικά αποφασίζεται η δομή (τοπολογία) του δικτύου και η αρχική σήμανση που μοντελοποιεί την αρχική κατάσταση και έπειτα μέσω της ενεργοποίησης μεταβάσεων (ροή κουπονιών) μεταβάλλεται η σήμανση του δικτύου, η οποία και καθορίζει την συμπεριφορά του συστήματος.

Μαθηματική Μοντελοποίηση

Τόσο οι δομικές ιδιότητες όσο και οι ιδιότητες συμπεριφοράς ενός Δικτύου Petri, πέραν της γραφικής απεικόνισης που αναφέρθηκαν παραπάνω, μπορούν να οριστούν και να μελετηθούν μαθηματικά, μέσω της θεωρίας συνόλων και της γραμμικής άλγεβρας.

Τα Δίκτυα Petri είναι *κατευθυντικά (προσανατολισμένα)* δίκτυα με *δομή (τοπολογία) C* και *σήμανση M*.

Η *δομή* του Δικτύου Petri είναι ένα υπερσύνολο *C={P,T,I,O}*, που αποτελείται από τέσσερα στοιχεία:

- P = {p1, p2, ..., pn}, όπου n > 0, είναι ένα πεπερασμένο σύνολο n θέσεων.
- T = {t1, t2, ..., ts}, όπου s> 0, είναι ένα πεπερασμένο σύνολο s μεταβάσεων. Όπου το σύνολο P∪T≠0 συνθέτει τους κόμβους του γράφου (δικτύου) και ισχύει επίσης P∩T=0.
- I: P X T → N, είναι η συνάρτηση εισόδου που ορίζει το σύνολο των προσανατολισμένων κλάδων (τόξων) εισόδου από τις pi θέσεις του συνόλου θέσεων P, στις tj μεταβάσεις του συνόλου μεταβάσεων T, όπου N = {0, 1, 2, …}είναι το σύνολο τιμών του I και δηλώνει το βάρος των κλάδων. Έτσι, ένας κλάδος εισόδου εκφράζεται ως I(p,t) = N, όπου p η θέση από την οποία ξεκινά και t η μετάβαση στην οποία καταλήγει. Στην ουσία, η συνάρτηση I(p,t) είναι ένας πίνακας διαστάσεων *nxs*, με τις n γραμμές να αναπαριστούν τις pi θέσεις και τις s στήλες να αναπαριστούν τις tj μεταβάσεις και τα στοιχεία του πίνακα αναπαριστούν τα βάρη των κλάδων εισόδου(από θέση προς μετάβαση).
- O: P X T → N, είναι η συνάρτηση εξόδου που ορίζει το σύνολο των προσανατολισμένων κλάδων (τόξων) εξόδου από τις tj μεταβάσεις του συνόλου μεταβάσεων T, στις pi θέσεις του συνόλου θέσεων P, όπου N = {0, 1, 2, …}είναι το σύνολο τιμών του O και δηλώνει το βάρος των κλάδων. Έτσι, ένας κλάδος εξόδου εκφράζεται ως O(p,t) = N, όπου t η μετάβαση από την οποία ξεκινά και p η θέση στην οποία καταλήγει. Στην ουσία, η συνάρτηση O(p,t) είναι και αυτή ένας πίνακας διαστάσεων *nxs*, με τις n γραμμές να αναπαριστούν τις pi θέσεις και τις s στήλες να αναπαριστούν τις tj μεταβάσεις και τα στοιχεία του κλάδου να αναπαριστούν τα βάρη των κλάδων εξόδου (από μετάβαση προς θέση).
- A = O – I, ονομάζεται πίνακας *συμβάντων ή σύμπτωσης* και είναι η αφαίρεση του πίνακα I από τον πίνακα O, ο οποίος είναι και αυτός διαστάσεων nxs.

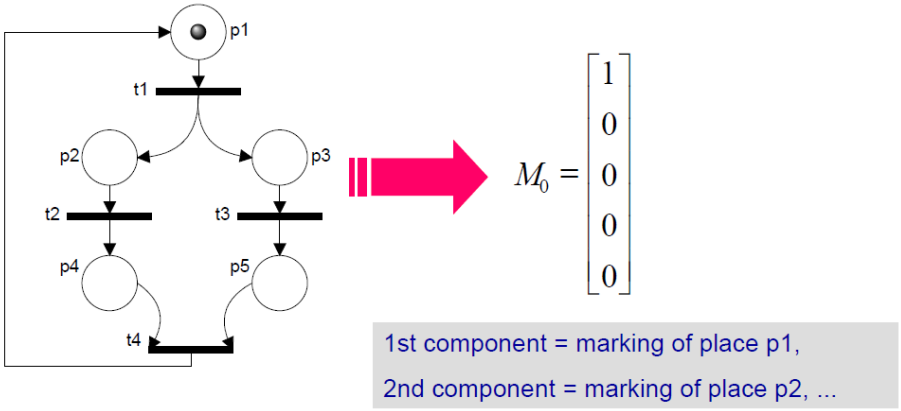
Η δομή ενός Δικτύου Petri μπορεί εναλλακτικά να οριστεί ως *C = {P, T, F, W}*, όπου:

- F ⊆ {PxT}∪{TxP}, είναι ένα υπερσύνολο του συνόλου{PxT}∪{TxP} που αναπαριστά το σύνολο όλων των κλάδων, εισόδου και εξόδου.

- W: F → N, είναι μια γενικευμένη συνάρτηση (εισόδου και εξόδου) και αναπαριστά την πολλαπλότητα ή το βάρος των κλάδων, με N = {0, 1, 2, …}να είναι το σύνολο τιμών του W, που δηλώνει το βάρος των κλάδων.

Η *Σήμανση* ενός Δικτύου Petri είναι ένα διάνυσμα nx1, όπου:

- m_k: P → N, είναι η σήμανση, ένα διάνυσμα στήλη, του οποίου η i-οστή γραμμή αναπαριστά τον αριθμό των κουπονιών N που βρίσκονται στην i-οστή θέση pi ∀ pi ϵ P, σε μια συγκεκριμένη κατάσταση k. Η αρχική σήμανση δηλώνεται ως m_0 ή $m_{k=0}$ ή M_0. Στην Εικόνα 3-6 παρουσιάζεται ένα παράδειγμα του διανύσματος σήμανσης για την αρχική κατάσταση k=0.



Εικόνα 3-6 Αρχική Σήμανση ενός Δικτύου Petri[17]

Η *σήμανση m* είναι ένα *διάνυσμα n γραμμών* (όσες και οι θέσεις του δικτύου), όπου κάθε γραμμή i έχει τα κουπόνια που υπάρχουν στην αντίστοιχη θέση pi ∀ pi ϵ P στη δεδομένη κατάσταση k. Η σήμανση αλλάζει σε κάθε ενεργοποίηση των μεταβάσεων, δηλαδή για κάθε κατάσταση k, δείχνοντας την δυναμική συμπεριφορά του συστήματος.

Η δομή και η σήμανση υλοποιούν ένα Δίκτυο Petri. Έτσι πλέον, μπορούμε πλέον να ορίσουμε το *Δίκτυο Petri* πέντε στοιχείων ως *Z = (P, T, I, O, m_0)*, όπου όταν γνωρίζουμε τη δομή και την αρχική του σήμανση, τότε θεωρούμε πως το Δίκτυο Petri είναι πλήρως ορισμένο.

Κανόνες Εκτέλεσης των Μεταβάσεων (firing)

Γνωρίζουμε πως εκτέλεση των μεταβάσεων t, σε ένα Δίκτυο Petri, σημαίνει διακίνηση των κουπονιών, η οποία μεταβάλει την σήμανση του δικτύου και συνοψίζεται στην παρακάτω μεθοδολογία:

Αν I(p, t) = N (ή O (p, t) = N), όπου p ∀ p ϵ P και t ∀ t ϵ T τότε υπάρχουν N προσανατολισμένα τόξα που συνδέουν τη θέση p με την μετάβαση t (ή την μετάβαση t με την θέση p). Αν ισχύει I(p, t) = 0 (ή O (p, t) = 0), τότε δεν υπάρχουν προσανατολισμένα τόξα που συνδέουν την θέση p με την μετάβαση t (ή την μετάβαση t με την θέση p). Ένα μοναδικό τόξο υπάρχει αν N = 1. Για τις περιπτώσεις όπου N > 1, τότε είτε υπάρχουν N παράλληλα τόξα που συνδέουν μια θέση (ή μια μετάβαση) με μια μετάβαση (ή μια θέση), ή ένα τόξο το οποίο όμως έχει πολλαπλότητα ή βάρος ίσο με N όταν χρησιμοποιείται.

Οι κανόνες εκτέλεσης ενός δικτύου Petri αφορούν κανόνες ενεργοποίησης (ή εκκίνησης), οι οποίες μαζί με τη σήμανση του δικτύου περιγράφονται μαθηματικά, όπως παρακάτω:

- Μια μετάβαση t ϵ T ενεργοποιείται *όταν και μόνο όταν* σε μια κατάσταση k ισχύει:

$$m_k(p) \geq I(p,t), \text{ όπου } p \ \forall p \in P$$

- Η μετάβαση t που ενεργοποιείται σε μια σήμανση m_k, εκκινεί και ως αποτέλεσμα προκύπτει μια νέα σήμανση m_{k+1} της κατάστασης k+1, τότε η σήμανση m_{k+1} λέγεται ότι είναι (άμεσα) προσβάσιμη από την σήμανση m_k, όπου:

$$m_{k+1}(pi) = m_k(pi) - I(pi, t) + O(pi, t), \ \forall pi \in P \ \ ή$$

$$m^{new}(pi) = m^{old}(pi) - I(pi, t) + O(pi, t), \ \forall pi \in P \quad (1)$$

Αξίζει να τονιστεί εδώ, η σχέση μεταξύ της ενεργοποίησης μια μετάβασης t και μιας κατάστασης k, δηλαδή από μια κατάσταση k για να βρεθεί το δίκτυο Petri σε μια κατάσταση k+1, μπορεί να ενεργοποιηθεί μόνο μια μετάβαση t.

Από πλευράς λογικής, οι συνθήκες αυτές δημιουργούν μια σχέση AND. Ο κανόνας εκκίνησης ορίζει ότι μια ενεργοποιημένη μετάβαση t εκκινεί ή ένα γεγονός συμβαίνει. Η εκκίνηση αυτή μπορεί να εξεταστεί σε δύο διαφορετικά στάδια. Στο πρώτο στάδιο, αφαιρείται ο αριθμός των κουπονιών που χρειάζονται από την κάθε μία θέση εισόδου, και ο αριθμός αυτός ισούται με το άθροισμα των τόξων που συνδέουν τη θέση εισόδου με την μετάβαση t. Στην παραπάνω εξίσωση, αυτό δηλώνεται από την αφαίρεση του I(p, t). Σε δεύτερο στάδιο, τοποθετούνται τα κουπόνια στην κάθε μία θέση εξόδου από την μετάβαση t, και ο αριθμός των κουπονιών ισούται με το άθροισμα των τόξων από την t στις συνδεόμενες θέσεις εξόδου. Στην παραπάνω εξίσωση, αυτό δηλώνεται από την πρόσθεση του O(p,t).[13]

Γενικότερα, μπορούμε τροποποιήσουμε την εξίσωση (1) θέτοντας το *Διάνυσμα Ενεργοποίησης Μεταβάσεων ή Συμβάντων* u_k, το οποίο ορίζεται *για κάθε κατάσταση k του συστήματος* ως:

$$u_k{}^T = [u_k(t1)\ u_k(t2)\ \dots\ u_k(ts)] \quad (2)$$

Όπου, για τα στοιχεία $u_k(tj)$ του διανύσματος u_k , ισχύει:

- $u_k(tj) = 1$, αν ενεργοποιείται η μετάβαση tj στην κατάσταση k
- $u_k(tj) = 0$, αν δεν ενεργοποιείται η μετάβαση tj στην κατάσταση k

Πρόκειται για ένα διάνυσμα s γραμμών (όσες οι μεταβάσεις), που τα στοιχεία του παίρνουν τιμές 0 και 1, αναλόγως ποια μετάβαση tj του δικτύου ενεργοποιείται για τη δεδομένη κατάσταση k.

Επομένως, χρησιμοποιώντας και τον Πίνακα Σύμπτωσης A = O – I, για μια δεδομένη κατάσταση k κατά την οποία ενεργοποιείται μια συγκεκριμένη μετάβαση t έχουμε:

$$m_{k+1}(pi) = m_k(pi) + O(pi,tj)*u_k - I(pi,tj)*u_k \quad \text{ή} \quad m_{k+1} = m_k + O*u_k - I*u_k \quad (3)$$

$$m_{k+1} = m_k + A*u_k \quad (4)$$

Μπορούμε από μια αρχική σήμανση m_0 να φτάσουμε σε μια τελική σήμανση m_f , μέσω διαδοχικών ενεργοποιήσεων u_k, u_{k+1} έως u_f, σύμφωνα μα τον τύπο:

$$m_{k+1} = m_k + A*\sum_{k=1}^{k=f} u_k \quad (5)$$

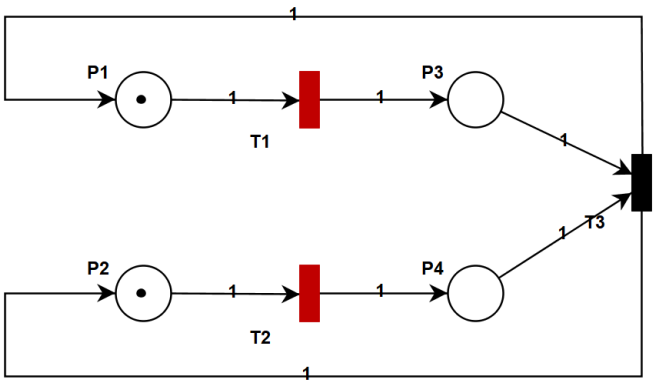
Θέτοντας $\sum_{k=1}^{k=f} u_k = y$, όπου *y το διάνυσμα μέτρησης ενεργοποιήσεων* το οποίο δείχνει πόσες φορές ενεργοποιείται μία μετάβαση, αλλά όχι και με ποια σειρά. Οπότε παίρνουμε την εξίσωση (6), την οποία θα χρησιμοποιήσουμε παρακάτω στην ανάλυση του συστήματος μέσω αναλλοίωτων:

$$m_k = m_{k+1} + A^*y \quad (6)$$

Για την καλύτερη κατανόηση των παραπάνω και του συνδυασμού μεταξύ γραφικού και μαθηματικού μοντέλου παρατίθεται το παρακάτω παράδειγμα προσομοίωσης, με χρήση του προγράμματος PIPEv4.3.0 (με κόκκινο επισημαίνονται οι μεταβάσεις που μπορούν να ενεργοποιηθούν):

Παράδειγμα

Στο Σχήμα 3-1 έχουμε τη Δομή και την Αρχική Σήμανση (k = 0) ενός Δικτύου PN, οπότε μπορούμε να εξάγουμε τους Πίνακες I, O, A και m_0:

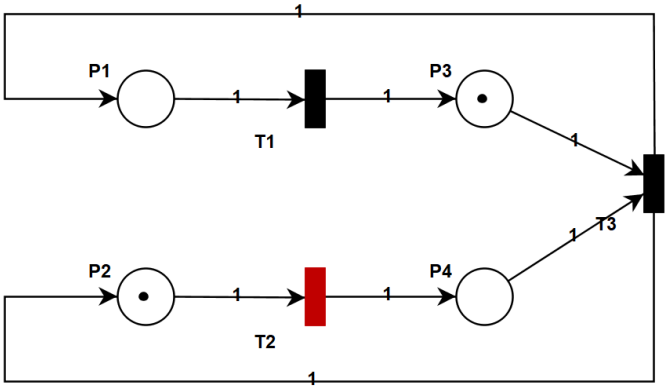


Σχήμα 3-1 Δομή και Σήμανση PN (k=0)

$$I = \begin{matrix} t1 & t2 & t3 & \times \end{matrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} P1 \\ P2 \\ P3 \\ P4 \end{matrix} \quad O = \begin{matrix} t1 & t2 & t3 & \times \end{matrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{matrix} P1 \\ P2 \\ P3 \\ P4 \end{matrix} \quad A = O - I = \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} \quad m_0 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{matrix}$$

Από το Σχήμα 3-1, καταλαβαίνουμε πως μπορούν να ενεργοποιηθούν οι μεταβάσεις T1 και T2. Έστω, επιλέγουμε την T1 πρώτη, οπότε για την κατάσταση k = 1, χρησιμοποιώντας την εξίσωση (1) έχουμε:

$$m_{k=1} = m_0 - I(t1) + O(t1) = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \rightarrow m_1 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$



Σχήμα 3-2 Δομή και Σήμανση PN (k=1)

Ή από την εξίσωση (4) για k=1 ενεργοποιείται η T1 μετάβαση, οπότε το διάνυσμα ενεργοποίησης μεταβάσεων είναι:

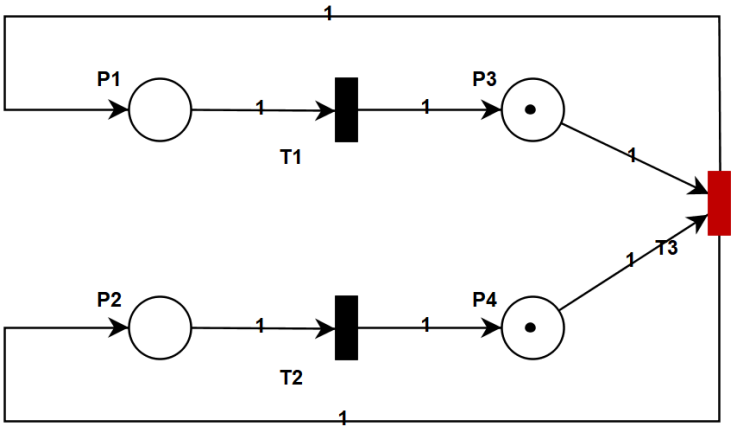
$$u_{k=1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} t1 \\ t2 \\ t3 \end{matrix}$$

Άρα, παίρνουμε το ίδιο αποτέλεσμα με παραπάνω

$$m_1 = m_0 + C * u_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Λειτουργώντας με την ίδια μέθοδο, για την κατάσταση k = 2, όπου ενεργοποιείται η T2 μετάβαση, έχουμε:

$$m_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$



Σχήμα 3-3 Δομή και Σήμανση PN (k=2)

Και για την κατάσταση k = 3, όπου ενεργοποιείται η T3 μετάβαση παίρνουμε:

$$m_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = m_0$$

όπου, η τελική κατάσταση ταυτίζεται με την αρχική $m_3=m_0$ και ουσιαστικά βρεθήκαμε στο ίδιο σημείο με το αρχικό. Το Δίκτυο Petri του παραδείγματος αποτελεί έναν κλειστό βρόχο και λέμε ότι το δίκτυο είναι *αντιστρεπτό*, καθότι $m_{final} = m_{initial}$.

Τέλος, μπορούμε να συνοψίσουμε τη μοντελοποίηση ενός συστήματος, μέσω των Δικτύων Petri, στον παρακάτω Πίνακα 3-1:

Πραγματικό Σύστημα	***Μοντελοποίηση***
Σύστημα	Δίτυο Petri (PN)
Κατάσταση/Εργασία/Συνθήκη	Θέση (p)
Γεγονότα	Μεταβάσεις (t)
Συνθήκες πριν από ένα Γεγονός	Είσοδοι στην ενεργοποιούμενη Μετάβαση (I)
Συνθήκες μετά από ένα Γεγονός	Έξοδοι από την ενεργοποιούμενη Μετάβαση (O)

Πίνακας 3-1 Πίνακας Αντιστοίχισης Συστήματος - Μοντελοποίησης στα Δίκτυα Petri[15]

Συνοψίζοντας, μπορούμε να πούμε πως η εισαγωγή κουπονιών στο δίκτυο και η ροή τους μέσα σε αυτό ρυθμίζεται από τις μεταβάσεις που επιτρέπουν την ξεκάθαρη επίβλεψη της ροής υλικού, ελέγχου, και πληροφοριών. Πιο σημαντικό χαρακτηριστικό του Δικτύου Petri είναι η δυνατότητα που προσφέρει για

έναν τυπικό έλεγχο των ιδιοτήτων του συστήματος και της συμπεριφοράς του, πχ. οι σχέσεις προτεραιότητας των γεγονότων, οι ταυτόχρονες ενέργειες, ο απαιτούμενος συγχρονισμός, η αποφυγή αδιεξόδων, οι επαναληψιμότητα ενεργειών, η αμοιβαία απαγόρευση χρήσης μοιραζόμενων πόρων. Ένας τέτοιος έλεγχος πριν την υλοποίηση του συστήματος στο φυσικό επίπεδο είναι ιδιαίτερα σημαντικός στη δημιουργία και στην ανάπτυξη ευέλικτων αυτοματοποιημένων συστημάτων κατεργασιών.

3.3 Ιδιότητες – Απλές Δομές Μοντελοποίησης

Τα Δίκτυα Petri διαθέτουν κάποιες ιδιότητες, οι οποίες διακρίνονται σε δύο είδη, τις ιδιότητες που αφορούν στη *δομή* του συστήματος, και αυτές που αφορούν στη *συμπεριφορά* του. Οι ιδιότητες συμπεριφοράς είναι οι ιδιότητες, που εξαρτώνται από την αρχική σήμανση του δικτύου, ενώ οι ιδιότητες της δομής του δικτύου δεν εξαρτώνται από την αρχική σήμανση, αλλά εξαρτώνται μόνο από τη δομή ή την τοπολογία του δικτύου. Οι πιο σημαντικές, ιδιότητες των Δικτύων Petri είναι οι κάτωθι[13]:

- Προσβασιμότητα (reachability)
- Φραγή (boundness)
- Ασφάλεια (safeness)
- Συντηρητικότητα (conservativeness)
- Ζωντάνια (liveness)
- Αντιστρεπτότητα (reversibility)
- Αρχική κατάσταση (home state)

Επιπρόσθετες *δομικές* ιδιότητες αποτελούν και οι:

- Επαναληψιμότητα (repetitiveness)
- Συνέπεια (consistency)

Η σημασία των παραπάνω ιδιοτήτων στα συστήματα κατεργασιών είναι ιδιαιτέρως σημαντική, και γι' αυτό θα αναλυθούν παρακάτω. Επίσης, υπάρχουν και άλλες ιδιότητες, όπως η κάλυψη (coverability), η επιμονή (consistency), η απόσταση συγχρονισμού (synchronic distance) και η αμεροληψία (fairness).

Προσβασιμότητα

Με πλήρως ορισμένο το Δίκτυο Petri με Z = (P, T, I, O, m_0) και γνωστό το σύνολο των προσβάσιμων σημάνσεων *R(Z,m_0),* με τον όρο προσβασιμότητα εννοούμε σε πόσες και ποιες σημάνσεις m_k (ή καταστάσεις k) το Z μπορεί να φθάσει. Μία σήμανση m_k είναι προσβάσιμη από την αρχική σήμανση σήμανση m_0, εφόσον υπάρχει τέτοια *ακολουθία ενεργοποιήσεων των μεταβάσεων (0→f),* που μετατρέπουν την m_0 σε m_f, μέσω της εξίσωσης (5):

$$m0 = m_f + A^{*}\sum_{k=0}^{k=f} u_k$$

Μια σήμανση m_{k+1} λέγεται ότι είναι *άμεσα προσβάσιμη* από την m_k, εφόσον η εκκίνηση μιας ενεργοποιημένης μετάβασης tj στη σήμανση m_k οδηγεί στη σήμανση m_{k+1}. Το σύνολο όλων των

προσβάσιμων σημάνσεων από την m_0 συμβολίζεται με R ή $R(m_0)$ ή $R(Z,m_0)$, εφόσον πρόκειται για διαφορετικές αρχικές σημάνσεις και PN. Επίσης, είναι εύκολα αντιληπτό πως η προσβασιμότητα είναι μια *ιδιότητα συμπεριφοράς*, αφού εξαρτάται από την αρχική σήμανση m_0.

Μια πολύ σημαντική παράμετρος κατά τη σχεδίαση ενός συστήματος κατεργασιών είναι κατά πόσο το σύστημα μπορεί να φτάσει σε μια συγκεκριμένη κατάσταση, δηλαδή το PN να φτάσει σε μια συγκεκριμένη σήμανση m_k, ή να επιδείξει συγκεκριμένη λειτουργική συμπεριφορά, δηλαδή στο PN να ενεργοποιηθούν συγκεκριμένες μεταβάσεις tj. Το οποίο ερμηνεύεται αν το σύστημα, που έχει μοντελοποιηθεί με ένα δίκτυο Petri, διαθέτει όλα τα χαρακτηριστικά, όπως ορίστηκαν από τις προδιαγραφές και καμία ανεπιθύμητη ιδιότητα. Άρα, πρέπει να ανακαλύψουμε την ακολουθία των ενεργοποιήσεων μεταβάσεων ($\sum_{k=1}^{k=f} u_k$), που θα οδηγήσουν την αρχική σήμανση m_0 σε m_k.

Τα πραγματικά συστήματα μπορούν να φτάσουν σε μια επιθυμητή κατάσταση ως αποτέλεσμα διαφορετικών λειτουργικών συμπεριφορών. Σε ένα μοντέλο δικτύου Petri αυτό μεταφράζεται στην ύπαρξη συγκεκριμένων ακολουθιών ενεργοποιήσεων μεταβάσεων, που δηλώνουν τις λειτουργικές συμπεριφορές, που θα μετατρέψουν την m_0 σε m_k. Η ύπαρξη επιπλέον ακολουθιών στο δίκτυο Petri που μετατρέπουν την m_0 σε m_k, μπορεί να υποδηλώνει:

- ότι το μοντέλο PN δεν αντικατοπτρίζει ακριβώς τη δομή και τη δυναμική του συστήματος ή
- την ύπαρξη ανεπιθύμητων διαδρομών της λειτουργικής συμπεριφοράς του πραγματικού συστήματος, δεδομένου ότι το μοντέλο του PN αντικατοπτρίζει πλήρως τις προδιαγραφόμενες απαιτήσεις του πραγματικού συστήματος.

Τέλος, η προσβασιμότητα χρησιμεύει στις μεθόδους ανάλυσης ενός δικτύου Petri, μέσω του Γράφου ή Δένδρου Προσβασιμότητας, που θα αναλυθεί παρακάτω και στο οποίο μπορούν να βρεθούν οι προσβάσιμες σημάνσεις εμπειρικά, μέσω ενεργοποίησης των μεταβάσεων και κατασκευής του δένδρου προσβασιμότητας με σκοπό να ελεγχθεί αν το PN οδηγείται σε αδιέξοδο.

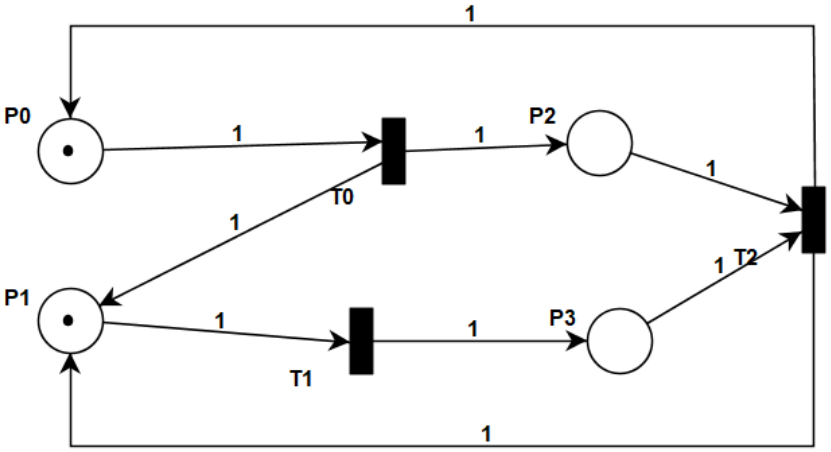
Φραγή και Ασφάλεια

Μια *θέση* $p \in P$ είναι *κ-φραγμένη* αν:

$$m(p) \leq \kappa, \ \forall m \in R, \text{ όπου } \kappa \in N_1$$

Το *Z* ονομάζεται *κ-φραγμένο* αν κάθε θέση $p \in P$, είναι κ-φραγμένη. Το *Z* είναι *δομικά φραγμένο* αν το Z είναι φραγμένο για *οποιοδήποτε* πεπερασμένη αρχική σήμανση m_0. Το *Z* ονομάζεται *ασφαλές* αν είναι *1-φραγμένο*.

Σε ένα *μη-φραγμένο* PN προστίθενται περισσότερα κουπόνια στις εξόδους από όσα αφαιρούνται από τις εισόδους, οπότε έχουμε συνεχή πρόσθεση (δημιουργία) κουπονιών στο PN. Αυτό συμβαίνει σε PN που προσομοιώνουν μια κυκλική διαδικασία (Σχήμα 3-4).



Σχήμα 3-4 Παράδειγμα μη-φραγμένου PN[4]

Σε μερικές περιπτώσεις μπορεί να μην μας ενδιαφέρει η ακριβής τιμή του κ, αλλά αν το Z είναι φραγμένο ή όχι. Οι θέσεις p συνήθως χρησιμοποιούνται για να μοντελοποιήσουμε χώρους αποθήκευσης τεμαχίων/εργαλείων/παλετών ή να αναπαραστήσουν τη διαθεσιμότητα πόρων σε ένα σύστημα κατεργασιών. Είναι, λοιπόν, πολύ σημαντικό να μπορούμε να προσδιορίσουμε αν ο προτεινόμενος έλεγχος, μέσω του PN, εμποδίζει την *υπερχείλιση* των χωρητικοτήτων των αποθηκευτικών χώρων. Η φραγή ενός δικτύου Petri χρησιμοποιείται ακριβώς για να προσδιορίσουμε αν υπάρχουν υπερχειλίσεις στο μοντελοποιημένο σύστημα. Όταν μια θέση ορίζει μια εργασία (π.χ. στην τόρνευση – μία θέση με 1 κουπόνι), η ασφάλεια εγγυάται ότι ο ελεγκτής δεν θα εκκινήσει μια υπό-εκτέλεση εργασία. Η *φραγή* στα PN ερμηνεύεται ως *σταθερότητα* ενός διακριτού συστήματος κατεργασιών, ιδίως αν η μοντελοποίηση του περιλαμβάνει συστήματα ουρών.

Συντηρητικότητα

Το Z είναι *συντηρητικό* σε σχέση με το w, αν υπάρχει ένα διάνυσμα w:

$$w^T = (w_1, w_2, \ldots, w_n) \text{ και } w_i > 0, \text{ όπου } i = 1, 2, \ldots, n$$

τέτοιο ώστε $w^T m = w^T m_0$, $\forall\, m \in R$.

Το Z είναι *αυστηρώς συντηρητικό*, αν το Z είναι συντηρητικό σε σχέση με το $w^T = (1, 1, \ldots, 1)$ ή ισχύει

$$\Sigma m(pi) = \Sigma m_0(pi),\ \forall m \in R.$$

Δεδομένου ότι αυτή η ιδιότητα δεν εξαρτάται από την αρχική σήμανση, η συντηρητικότητα είναι *δομική ιδιότητα*. Στα πραγματικά συστήματα, ο αριθμός των πόρων που χρησιμοποιούνται είναι περιορισμένος. Αν για την αναπαράσταση των πόρων χρησιμοποιούνται κουπόνια, ο αριθμός των οποίων είναι σταθερός, τότε ο αριθμός των κουπονιών σε ένα δίκτυο Petri θα έπρεπε να είναι αμετάβλητος ανεξάρτητα από την σήμανση του δικτύου, καθώς οι πόροι δεν μπορούν ούτε να δημιουργηθούν ούτε να καταστραφούν, εκτός αν υπάρχει μια διάταξη τροφοδότησης ή απομάκρυνσης πόρων, που να επιτρέπει κάτι τέτοιο.

Η αυστηρή συντηρητικότητα σημαίνει ότι ο αριθμός των κουπονιών διατηρείται, ενώ από δομικής πλευράς σημαίνει ότι ο αριθμός των κλάδων εισόδου σε μία μετάβαση πρέπει να ισούται με τον αριθμό των κλάδων εξόδου από τη μετάβαση. Η συντηρητικότητα σημαίνει ότι το άθροισμα των κλάδων παραμένει σταθερό σε κάθε σήμανση.

Ζωντάνια ≠ Αδιέξοδο

Μία μετάβαση t είναι *ζωντανή* αν για μια οποιαδήποτε σήμανση $m_k \in R$, υπάρχει μια ακολουθία μεταβάσεων, των οποίων η εκκίνηση συνεπάγεται μια σήμανση m που ενεργοποιεί την t. Το Z είναι *ζωντανό*, αν κάθε μετάβαση t σε αυτό είναι ζωντανή. Το Z είναι *δομικά ζωντανό*, αν υπάρχει μια πεπερασμένη αρχική σήμανση m_0, η οποία κάνει το δίκτυο ζωντανό.

Μία μετάβαση t είναι *νεκρή*, αν υπάρχει μια σήμανση $m \in R$ από την οποία δεν ξεκινά μια ακολουθία μεταβάσεων, των οποίων η εκκίνηση να συνεπάγεται μια σήμανση η οποία να ενεργοποιεί την t. Το Z περιέχει ένα *αδιέξοδο (deadlock)* αν υπάρχει μια σήμανση $m \in R$, έτσι ώστε να ενεργοποιείται καμία μετάβαση t. Μια τέτοια *σήμανση* ονομάζει νεκρή σήμανση. Καταστάσεις με αδιέξοδα είναι αποτέλεσμα λανθασμένης κατανομής των πόρων ή *εξάντλησης συγκεκριμένων τύπων πόρων*.

Η ζωντάνια ενός δικτύου Petri σημαίνει ότι από μια οποιαδήποτε σήμανση m προσβάσιμη από την αρχική m_0, είναι απολύτως πιθανό να εκκινήσει οποιαδήποτε μετάβαση t μέσα στο δίκτυο προχωρώντας μέσα από ακολουθίες εκκινήσεων. *Γι' αυτό αν ένα δίκτυο Petri είναι ζωντανό, δεν είναι δυνατόν να εμφανισθεί αδιέξοδο.* Η διερεύνηση αδιεξόδου στα μοντελοποιημένα συστήματα είναι ιδιαιτέρως σημαντική, καθώς αν αυτό δημιουργηθεί ανεπιθύμητα, μπορεί να έχει δυσάρεστα αποτελέσματα. Επίσης, η διερεύνηση αδιεξόδου σε ένα PN, μπορεί να γίνει μέσω του δένδρου προσβασιμότητας.

Αντιστρεπτότητα - Αρχική Κατάσταση

Το Z είναι *αντιστρεπτό*, αν $\forall\, m \in R(m_0)$ ισχύει και $m_0 \in R(m)$. Η σήμανση $m' \in R(Z,m_0)$ ονομάζεται *αρχική κατάσταση*, αν $\forall\, m \in R(Z,m_0)$ και η m΄ είναι προσβάσιμη από την m. Ένα δίκτυο Petri είναι *δομικά αντιστρεπτό* αν υπάρχει πεπερασμένη αρχική σήμανση που κάνει το δίκτυο αντιστρεπτό.

Η αντιστρεπτότητα ενός δικτύου Petri σημαίνει ότι για μια οποιαδήποτε σήμανση m προσβάσιμη από την m_0, η m_0 είναι επίσης προσβάσιμη από την m. Ουσιαστικά, η αντιστρεπτότητα αποτελεί ειδική περίπτωση της αρχικής κατάστασης, αφού αν η ισχύει η *αρχική κατάσταση* $m' = m_0$, τότε το δίκτυο είναι αντιστρεπτό.

Πολλά συστήματα απαιτείται να έχουν *ανάδραση (feedback)*, προκειμένου να αυτό-διορθώνονται, γι' αυτό η συγκεκριμένη ιδιότητα είναι πολύ σημαντική για την επαναφορά στη σωστή λειτουργία από λάθος ενός συστήματος κατεργασιών. Επιπροσθέτως, η ιδιότητα εγγυάται την *κυκλική συμπεριφορά* του συστήματος, κάτι το οποίο απαιτείται για όλα τα επαναληπτικά συστήματα κατεργασιών (closed-loop).

Επαναληψιμότητα

Το Z ονομάζεται *επαναληπτικό* αν υπάρχει πεπερασμένη αρχική σήμανση m_0 και μια ακολουθία εκκινήσεων S από την m_0, έτσι ώστε κάθε μετάβαση t να συμβαίνει απείρως συχνά μέσα στην S.

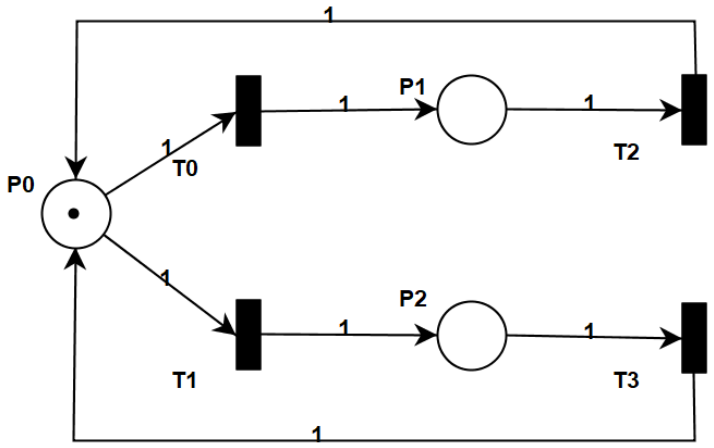
Συνέπεια

Το Z δίκτυο Petri ονομάζεται *συνεπές* αν υπάρχει μια αρχική σήμανση m_0 και μια ακολουθία εκκινήσεων S από την m_0 προς την m_0, έτσι ώστε κάθε μετάβαση t να ενεργοποιείται τουλάχιστον μια φορά μέσα στην S.

Απλές Δομές Μοντελοποίησης

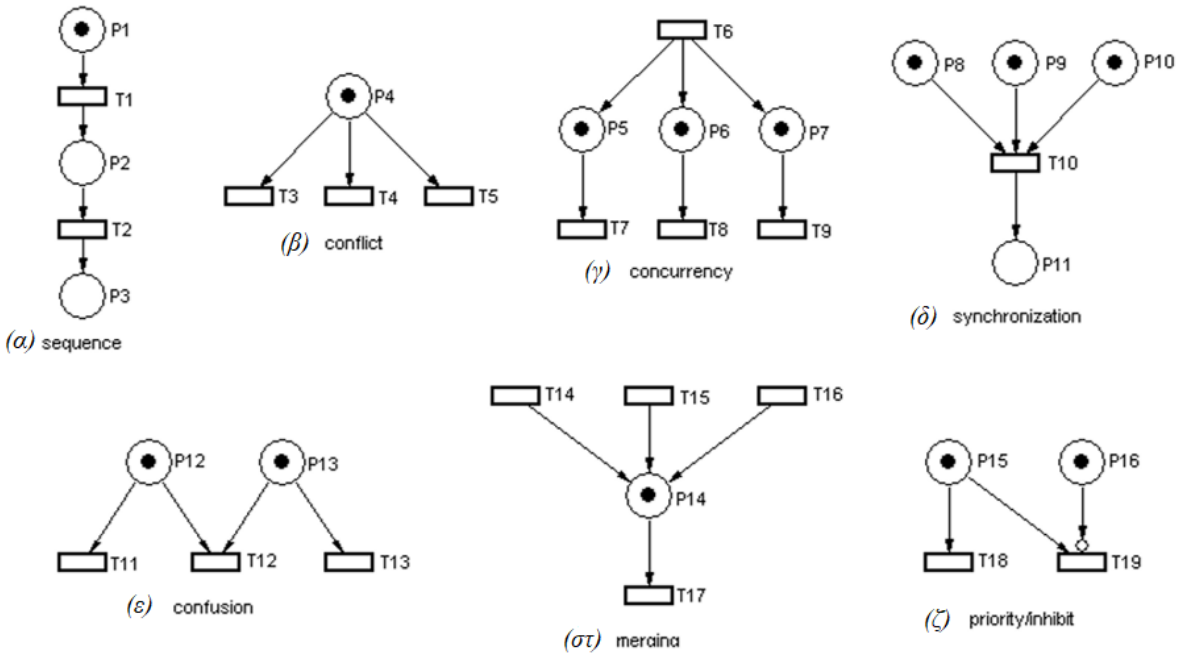
Η μοντελοποίση πραγματικών συστημάτων μέσω των Δίκτυων Petri περιέχει κάποιες συγκεκριμένες δομές, οι οποίες συναντώνται στα περισσότερα δίκτυα και μοντελοποιούν κυρίως συστήματα παράλληλης λειτουργίας και συστήματα στα οποία συμβαίνει διεκδίκηση πόρων (συγκρούσεις). Οπότε, έχουμε:

- Τα παράλληλα γεγονότα που αντιπροσωπεύονται από παράλληλες (ανεξάρτητες) μεταβάσεις. Τα PN που κάθε θέση τους έχει μόνο μια είσοδο και μία έξοδο ονομάζονται *Σημασμένα* και μοντελοποιούν την παραλληλία (Σχήμα 3-1).
- Οι συγκρούσεις, όπου από μια θέση δύναται να ενεργοποιηθούν 2 ή περισσότερες μεταβάσεις και πρέπει να γίνει επιλογή ενεργοποίησης μετάβασης, ονομάζονται *Μηχανές Κατάστασης* (Σχήμα 3-5).



Σχήμα 3-5 Παράδειγμα Μηχανής Κατάστασης[4]

Στα Δίκτυα υπάρχουν απλές δομές, τις οποίες συνθέτουμε για να προβούμε στην μοντελοποίηση πιο πολύπλοκων συστημάτων, οι οποίες αναφέρονται ως *Primitive Structures* (Εικόνα 3-7) και μοντελοποιούν κυρίως προϋποθέσεις, που πρέπει να ισχύουν για κάποιες καταστάσεις, προκειμένου να πραγματοποιηθούν συγκεκριμένα γεγονότα.



Εικόνα 3-7 Primitive Structures των Δικτύων Petri

Οι κυριότερες από αυτές, παρουσιάζονται στην Εικόνα 3-7 και είναι οι εξής:

(α) *Ακολουθία (sequence):* Το κάθε γεγονός συμβαίνει διαδοχικά με καθορισμένη σειρά.

(β) *Σύγκρουση (conflict):*Όπως και στις Μηχανές Κατάστασης. Διαφορετικά γεγονότα (μεταβάσεις) διεκδικούν κάποιον κοινό πόρο (θέση) για να ενεργοποιηθούν. Αν δεν βάλουμε συγκεκριμένη χρονική ενεργοποίηση για κάθε μετάβαση, πρέπει κάθε φορά να επιλέγουμε ποιο γεγονός θα πραγματοποιηθεί, οπότε δεν έχουμε αυτοματοποίηση.

(γ) *Παραλληλία (concurrency):* Όπως και στα Σημασμένα δίκτυα. Από ένα γεγονός (μετάβαση) υλοποιούνται περισσότερες από μια καταστάσεις (θέσεις).

(δ) *Συγχρονισμός (synchronization):* Ένα γεγονός για να υλοποιηθεί εξαρτάται ταυτόχρονα από περισσότερες από μια καταστάσεις/πόρους/εργασίες.

(ε) *Σύγχυση (confusion):* Είναι συνδυασμός σύγκρουσης και παραλληλίας.

(στ) *Συγχώνευση (merging):* Δεν είναι το ίδιο με τον συγχρονισμό. Από την υλοποίηση πολλών διαφορετικών γεγονότων εκτελείται μια συγκεκριμένη κατάσταση.

(ζ) *Προτεραιότητα/Απαγόρευση (priority/inhibit):* Θα εξηγηθεί και παρακάτω στον μηδενικό έλεγχο, όπου για να ενεργοποιηθεί μια μετάβαση δεν θα πρέπει να έχει κουπόνι η θέση με την οποία συνδέεται μέσω αποτρεπτικού κλάδου. Για να ενεργοποιηθεί η T19 δεν πρέπει να έχει κουπόνι η P16 και να έχει κουπόνι η P15. Με αυτόν τον τρόπο μπορούμε να επιβάλουμε και προτεραιότητα, όπως θα δούμε και στο επόμενο Κεφάλαιο 4.

Κατόπιν των παραπάνω ορισμών, καταλαβαίνουμε πως στο PN του Σχήματος 3-1 έχουμε παραλληλία και συγχρονισμό, ενώ στο PN της Εικόνας 3-7 έχουμε σύγκρουση και συγχώνευση.

3.4 Μέθοδοι Ανάλυσης

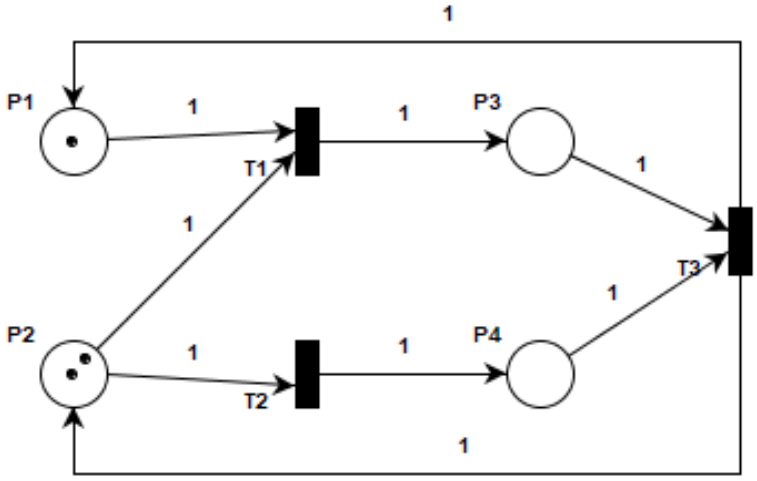
Η μαθηματική ανάλυση των δικτύων Petri πραγματοποιείται κυρίως με τις εξής μεθόδους:

- Ανάλυση μέσω προσβασιμότητας (Reachability Analysis Method)
- Ανάλυση μέσω αναλλοίωτων (Invariant Analysis Method)
- Ανάλυση μέσω απλοποίησης (Reduction Method)

Ανάλυση μέσω Προσβασιμότητας

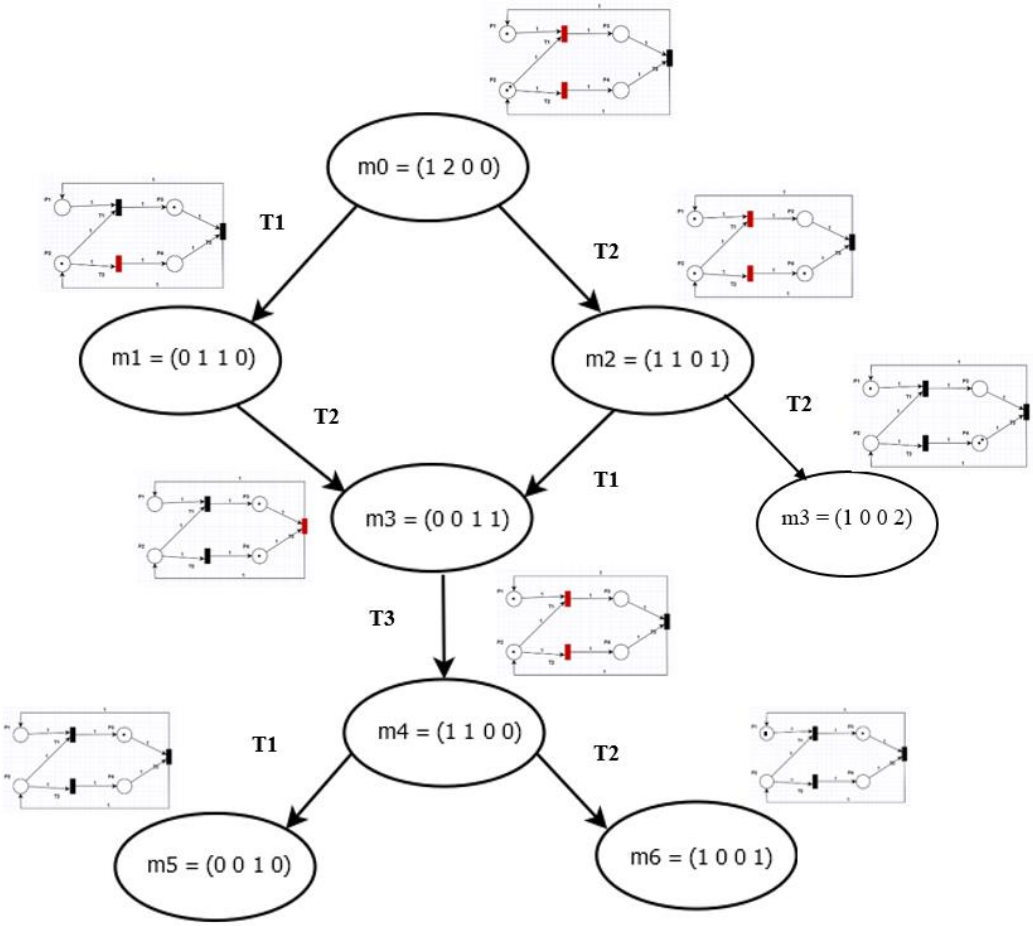
Σε αυτή τη μέθοδο, ξεκινώντας από την αρχική σήμανση m_0 του συστήματος, απαριθμούνται διαδοχικά όλες οι πιθανές καταστάσεις k, στις οποίες μπορεί να βρεθεί το σύστημα. Η βασική μεθοδολογία που ακολουθείται είναι ότι $\forall\ m \in R(m_0)$ αναγνωρίζονται όλες οι ενεργοποιούμενες μεταβάσεις t και η αντίστοιχη σήμανση m_k που προκύπτει για καθεμία ενεργοποιημένη μετάβαση tj. Η γραφική απεικόνιση του αποτελέσματος αποτελείται από κόμβους (ελλείψεις) που είναι οι σημάνσεις m, και κλάδους που είναι οι ενεργοποιημένες μεταβάσεις t. Ένας κλάδος συνδέει έναν κόμβο με έναν άλλο και έχει σημειωμένη την μετάβαση tj πάνω του, με την οποία ενεργοποιήθηκε και δημιούργησε την αλλαγή στην σήμανση. Αυτή η γραφική απεικόνιση ονομάζεται *Δένδρο Προσβασιμότητας ή Κάλυψης* (reachability tree or graph). Η

διαγραφή διπλών σημάνσεων από το δένδρο οδηγεί στην δημιουργία ενός γραφήματος προσβασιμότητα ή κάλυψης. Εν συνεχεία, όλες οι *ιδιότητες* συμπεριφοράς που αναφέρθηκαν παραπάνω μπορούν να προκύψουν, εφόσον ο αριθμός των καταστάσεων είναι πεπερασμένος. Το δένδρο προσβασιμότητας είναι ιδανικό για τον έλεγχο αδιεξόδου, διότι είναι κάτι που γίνεται άμεσα αντιληπτό, όπως στο παρακάτω παράδειγμα:



Σχήμα 3-6 Παράδειγμα PN που καταλήγει σε αδιέξοδο

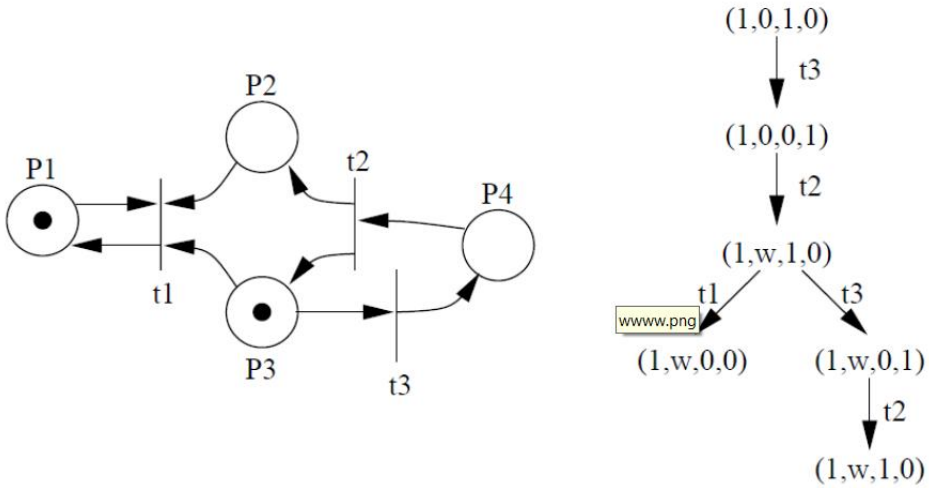
Έχουμε, το PN του Σχήματος 3-6, στο οποίο φαίνεται γραφικά η δομή και η αρχική του σήμανση. Κατόπιν ενεργοποιήσεων των μεταβάσεων και υλοποιώντας το δένδρο προσβασιμότητας (Σχήμα 3-7) φτάνουμε σε αδιέξοδο, καθότι καταλήγουμε σε νεκρές σημάνσεις.



Σχήμα 3-7 Δένδρο Προσβασιμότητας που καταλήγει σε αδιέξοδο

Στο δένδρο προσβασιμότητας έχουμε τη συνολική εικόνα ενεργοποίησης των σημάνσεων σε ένα PN. Η μέθοδος ανάλυσης μέσω προσβασιμότητας είναι εφαρμόσιμη μόνο για την ανάλυση ιδιοτήτων συμπεριφοράς, αλλά αποτελεί μια θεμελιώδης προσέγγιση στην ανάλυση ενός δικτύου Petri. Οπότε, θα μπορούσαμε να πούμε πως, το Δένδρο Προσβασιμότητας είναι ένας χάρτης της δυναμικήςς του PN και κατ' επέκταση του πραγματικού συστήματος.

Όταν ένα δίκτυο Petri είναι *μη-φραγμένο* ή ο αριθμός των καταστάσεων του μοντελοποιημένου συστήματος είναι *μη πεπερασμένος*, τότε η παραπάνω διαδικασία θα συνεχίζεται ες αεί. Για να διατηρηθεί το δένδρο πεπερασμένο χρησιμοποιούνται ορισμένοι κόμβοι που αντιπροσωπεύουν άπειρο αριθμό καταστάσεων. Στη σήμανση των συγκεκριμένων κόμβων μία ή περισσότερες θέσεις που περιέχουν απροσδιόριστο αριθμό κουπονιών βάζουμε ω. Στο παράδειγμα της Εικόνας 3-8 παρουσιάζεται *η δημιουργία ενός πεπερασμένου δένδρου προσβασιμότητας*:



Εικόνα 3-8 Παράδειγμα δημιουργίας Πεπερασμένου Δένδρου Προσβασιμότητας

Η αρχική σήμανση του συγκεκριμένου δικτύου είναι στον κόμβο (1,0,1,0). Από αυτήν την κατάσταση η μόνη μετάβαση που ενεργοποιείται είναι η t3. Όταν η t3 εκκινεί οδηγεί στη σήμανση (1,0,0,1). Εν συνεχεία, ενεργοποιείται η t2 οδηγώντας στη σήμανση (1,1,1,0). Εφόσον η (1,0,1,0) είναι υποσύνολο της (1,1,1,0), η αλληλουχία t3 – t2 θα εμφανίζεται ξανά και ξανά στο δένδρο προσβασιμότητας. Για το λόγο αυτό, σε αυτό το σημείο, τα κουπόνια στη θέση p2 αντικαθίσταται με το σύμβολο ω. Από αυτή τη σήμανση ενεργοποιούνται οι t1 και t3 παράγοντας τις σημάνσεις (1,w,0,0) και (1,w,0,1) αντίστοιχα. Η πρώτη από αυτές δεν ενεργοποιεί καμία μετάβαση (νεκρή σήμανση) ενώ η δεύτερη ενεργοποιεί την t2. Η εκκίνηση της t2 π

αράγει την (1,w,1,0) που υπάρχει ήδη. Άρα το δένδρο προσβασιμότητας έχει ολοκληρωθεί[8].

Από την ανάλυση του δένδρου προσβασιμότητας μπορούμε να εξάγουμε τα παρακάτω γενικά συμπεράσματα:

- Το δίκτυο είναι φραγμένο, αν και μόνο αν το ω δεν εμφανίζεται σε κόμβο. Αν ο αριθμός κ είναι το μέγιστο σε όλες τις σημάνσεις, τότε το δίκτυο είναι κ-φραγμένο. Μια θέση p είναι μη φραγμένη αν υπάρχει κάποια σήμανση m στο δένδρο ώστε m(p) = ω. Το δίκτυο είναι ασφαλές αν και μόνο αν κάθε κόμβος του δένδρου περιέχει μόνο 0 και 1.

- Αν καμία αδιέξοδη σήμανση περιέχει ω, τότε ο αριθμός των διαφορετικών αδιέξοδων σημάνσεων στο δένδρο είναι ο αριθμός των νεκρών σημάνσεων του δικτύου. Αν κάποια αδιέξοδη σήμανση περιέχει ω, τότε το δίκτυο περιέχει άπειρες νεκρές σημάνσεις. Μία μετάβαση θεωρείται νεκρή εάν δεν εμφανίζεται σε κανένα τόξο στο δένδρο.
- Εάν πάρουμε 2 κόμβους του δένδρου (σημάνσεις), που δεν περιέχουν το σύμβολο ω, και στο μονοπάτι που τους συνδέει εμφανίζονται όλες οι μεταβάσεις του δικτύου, τότε το δίκτυο είναι ζωντανό.
- Εάν υπάρχει ένα μονοπάτι από έναν οποιονδήποτε κόμβο προς την αρχική σήμανση χωρίς το σύμβολο ω, τότε το δίκτυο είναι αντιστρεπτό.

Ανάλυση μέσω Αναλλοίωτων

Στο δένδρο προσβασιμότητας έχουμε μια περισσότερο γραφική λύση μέσω αντιστοίχισης των σημάνσεων σε κόμβους, ενώ με τη μέθοδο των αναλλοίωτων εξετάζονται οι βασικές ιδιότητες των δικτύων Petri μελετώντας τις γραμμικές εξισώσεις που προκύπτουν από τους κανόνες εκτέλεσης και με τη χρήση του πίνακα συμβάντων ή σύμπτωσης Α.

P – Αναλλοίωτες

P-αναλλοίωτες είναι οι θετικές ακέραιες λύσεις του διανύσματος x της εξίσωσης:

$$A^T * x = 0 \rightarrow x^T * A = 0 \quad (7)$$

Μας ενδιαφέρει το ελάχιστο σύνολο των P- αναλλοίωτων. Γενικά, υπάρχουν n-r ελάχιστες P-αναλλοίωτες όπου n ο αριθμός των θέσεων και r ο βαθμός (rank) του πίνακα Α (αριθμός των γραμμικά ανεξάρτητων ιδιο-διανυσμάτων) Οι P-αναλλοίωτες δείχνουν την ελάχιστη δομή κλειστών βρόχων (κυκλωμάτων) του PN. Επίσης, αποδεικνύεται ότι αν κάθε θέση στο δίκτυο καλύπτεται από μια P-αναλλοίωτη τότε το δίκτυο είναι φραγμένο.[11] [13]

Για καλύτερη κατανόηση θα χρησιμοποιήσουμε το παράδειγμα του Σχήματος 3-1, όπου έχουμε τον πίνακα συμβάντων Α:

$$A = O - I = \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}$$

Το διάνυσμα x είναι διαστάσεων nx1, όπου n ο αριθμός των θέσεων του PN. Εδώ n = 4, άρα προκύπτει από την εξίσωση (7):

$$x^T * A = 0 \rightarrow [x1 \quad x2 \quad x3 \quad x4] * \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} = 0 \rightarrow \begin{bmatrix} -x1 + x3 \\ -x2 + x4 \\ x1 + x2 - x3 - x4 \end{bmatrix} = 0$$

Οπότε, προκύπτουν 3 γραμμικές εξισώσεις, τις οποίες λύνουμε:

$$-x1 + x3 = 0 \rightarrow x1 = x3 \quad (\alpha)$$

$$-x2 + x4 = 0 \rightarrow x2 = x4 \quad (\beta)$$

$$x1 + x2 - x3 - x4 = 0 \xrightarrow{(\alpha)\ (\beta)} 0 = 0 \text{ ισχύει}$$

Άρα, βάσει των βαρών των κλάδων, που είναι 1 παντού και των περιορισμών (α) και (β) έχουμε 4 δυνατές λύσεις:

- Για x1 = x3 = 1 και x2 = x4 = 0 έχουμε: $s1 = [1 \quad 0 \quad 1 \quad 0]$
- Για x1 = x3 = 0 και x2 = x4 = 1 έχουμε: $s2 = [0 \quad 1 \quad 0 \quad 1]$
- Για x1 = x3 = x2 = x4 =1 έχουμε: $s3 = [1 \quad 1 \quad 1 \quad 1]$
- Για x1 = x3 = x2 = x4 = 0 έχουμε: $s4 = [0 \quad 0 \quad 0 \quad 0]$

Η 4η λύση $s4 = [0 \quad 0 \quad 0 \quad 0]$ απορρίπτεται, διότι είναι αδύνατον να μην έχει τουλάχιστον μια θέση ένα κουπόνι. Από τις s1, s2 και s3 παρατηρώ ότι όλα τα x_i έχουν τουλάχιστον από 1, άρα από όλες τις θέσεις περνούν κουπόνια. Επίσης, οι εφικτές λύσεις s1 και s2 καλύπτουν την s3, οπότε η s3 είναι περιττή.

Καταλήγουμε πως έχουμε 2 P-αναλλοίωτες τις s1 και s2, που καλύπτουν όλες τις θέσεις του δικτύου, οπότε το δίκτυο είναι 1-φραγμένο (ασφαλές) και έχουμε 2 κυκλώματα με θέσεις p1-p3 και p2-p4.

T – Αναλλοίωτες

Τ-αναλλοίωτες είναι οι θετικές ακέραιες λύσεις του διανύσματος y (διάνυσμα μέτρησης ενεργοποιήσεων) της εξίσωσης:

$$A*y = 0 \quad (8)$$

Μια Τ-αναλλοίωτη υποδεικνύει πόσες φορές ενεργοποιείται μια μετάβαση μέσα σε ένα κύκλο ώστε να επανέλθει η αρχική σήμανση στον εαυτό της. Γενικά, υπάρχουν s-r ελάχιστες P-αναλλοίωτες όπου s ο αριθμός των μεταβάσεων και r ο βαθμός του πίνακα Α.

Από το παραπάνω παράδειγμα έχουμε: το διάνυσμα y είναι διαστάσεων sx1, όπου s ο αριθμός των μεταβάσεων του PN. Εδώ s = 3, άρα προκύπτει από την εξίσωση (8):

$$A * y = 0 \rightarrow \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix} * \begin{bmatrix} y1 \\ y2 \\ y3 \end{bmatrix} = 0 \rightarrow \begin{bmatrix} -y1 + y3 \\ -y2 + y3 \\ y1 - y3 \\ y2 - y3 \end{bmatrix} = 0$$

Οπότε, προκύπτουν 4 γραμμικές εξισώσεις, τις οποίες λύνουμε:

$$-y1 + y3 = 0 \rightarrow y1 = y3 \quad (\alpha)$$

$$-y2 + y3 = 0 \rightarrow y2 = y3 \quad (\beta)$$

$$y1 - y3 = 0 \rightarrow y1 = y3 \quad (\gamma)$$

$$y2 - y3 = 0 \rightarrow y2 = y3 \quad (\delta)$$

Άρα, βάσει των βαρών των κλάδων, που είναι 1 παντού και των περιορισμών (α), (β), (γ) και (δ) έχουμε 2 δυνατές λύσεις:

- Για y1 = y2 = y3 = 0 έχουμε: $s1 = [1 \quad 1 \quad 1]$
- Για y1 = y2 = y3 = 1 έχουμε: $s2 = [0 \quad 0 \quad 0]$ Απορρίπτεται

Οπότε, έχουμε μια εφικτή λύση την $s1 = [1 \quad 1 \quad 1]$, που σημαίνει πως οι μεταβάσεις t1, t2 και t4 ενεργοποιούνται μια φορά σε έναν κύκλο του PN.

Στα ίδια ακριβώς συμπεράσματα καταλήγουμε και από την εύρεση P και T αναλλοίωτων μέσω του υπολογιστικού προγράμματος pipev4.3.0 που χρησιμοποιήθηκε για την σχεδίαση, την προσομοίωση και τον υπολογισμό των PN (Σχήμα 3-8).

Petri net invariant analysis results

T-Invariants

T1	T2	T3
1	1	1

The net is covered by positive T-Invariants, therefore it might be bounded and live.

P-Invariants

P1	P2	P3	P4
1	0	1	0
0	1	0	1

The net is covered by positive P-Invariants, therefore it is bounded.

P-Invariant equations

M(P1) + M(P3) = 1
M(P2) + M(P4) = 1

Σχήμα 3-8 Εύρεση P- και T- Αναλλοίωτων μέσω του λογισμικού pipe4.3.0

Ανάλυση μέσω Απλοποιήσεων

Απλοποιώντας ένα υπο-δίκτυο ή ένα κομμάτι, διατηρώντας όμως της ιδιότητες που ψάχνουμε, είναι δυνατόν να βρούμε τις ιδιότητες ενός πολύπλοκου δικτύου Petri. Ο περιορισμός της μεθόδου έγκειται μόνο στη μη ύπαρξη υπο-δικτύων ή στην δυσκολία ανακάλυψής αυτών.

3.5 *Επιβολή Προτεραιοτήτων – Μηδενικός Έλεγχος*

Στα δίκτυα Petri μπορούμε να καθορίσουμε την προτεραιότητα ενεργοποίησης μιας μετάβασης έναντι μιας άλλης, μέσω της χρήσης του δεσμευτικού τόξου ή αποτρεπτικού κλάδου, που αναφέρθηκε παραπάνω. Αυτό μπορεί να συμβεί όταν μια εργασία μοιράζεται τον ίδιο πόρο με μια άλλη εργασία ή έχουμε ανταγωνισμό μιας εργασίας να επιλέξει ανάμεσα σε πολλούς πόρους. Μπορούμε να καθορίσουμε ποια εργασία θα δεσμεύσει πρώτη τον πόρο, μέσω του αποτρεπτικού κλάδου. Το δεσμευτικό τόξο ή αποτρεπτικός κλάδος χρησιμεύει στην επιβολή προτεραιοτήτων ενεργοποιήσεων των μεταβάσεων σε ένα PN.

Εδώ υπεισέρχεται ο όρος του *μηδενικού ελέγχου (zero testing ability)* στα δίκτυα Petri, δηλαδή της δυνατότητας ελέγχου αν μια θέση έχει μηδενικά κουπόνια. Ένα δεσμευτικό τόξο χρησιμοποιείται για να

υλοποιηθεί αυτή η δυνατότητα συνδέοντας μια θέση εισόδου με μια μετάβαση, και σχεδιάζεται με ένα τόξο το οποίο στην αρχή του έχει έναν μικρό κύκλο, και αυτό διότι στα ψηφιακά κυκλώματα ο μικρός κύκλος υποδηλώνει το NOT. Λόγω της παρουσίας του, μια μετάβαση θεωρείται ως ενεργοποιημένη αν κάθε θέση εισόδου στην μετάβαση με κανονικό κλάδο περιέχει αριθμό κουπονιών τουλάχιστον ίσο με το βάρος του κλάδου και κανένα κουπόνι δεν είναι παρόν στις θέσεις εισόδου που συνδέονται με δεσμευτικό τόξο. Οι κανόνες ενεργοποίησης της μετάβασης είναι οι ίδιοι με αυτούς για τις συνδεδεμένες με κανονικά τόξα μεταβάσεις. Η ενεργοποίηση όμως δεν αλλάζει την σήμανση στις θέσεις που είναι συνδεδεμένες με το δεσμευτικό τόξο. Το δεσμευτικό τόξο μπορεί να γενικευθεί με μια πολλαπλότητα. Έτσι, όταν ο αριθμός των κουπονιών στις θέσεις είναι μικρότερος από τον αριθμό πολλαπλότητας του δεσμευτικού τόξου, η μετάβαση επιτρέπεται να ενεργοποιηθεί από τις κανονικές θέσεις εισόδου.

Κατόπιν των παραπάνω, το δεσμευτικό τόξο ορίζεται από τη συνάρτηση δέσμευσης:

$$H: P \times T \rightarrow N, \text{ όπου } N = \{0,1, ..\}$$

Μία μετάβαση t ∈ T θεωρείται ενεργοποιημένη όταν και μόνο όταν:

$$m(p) \geq I(p, t) \text{ και } m(p) < H(p, t), \forall p \not\in P$$

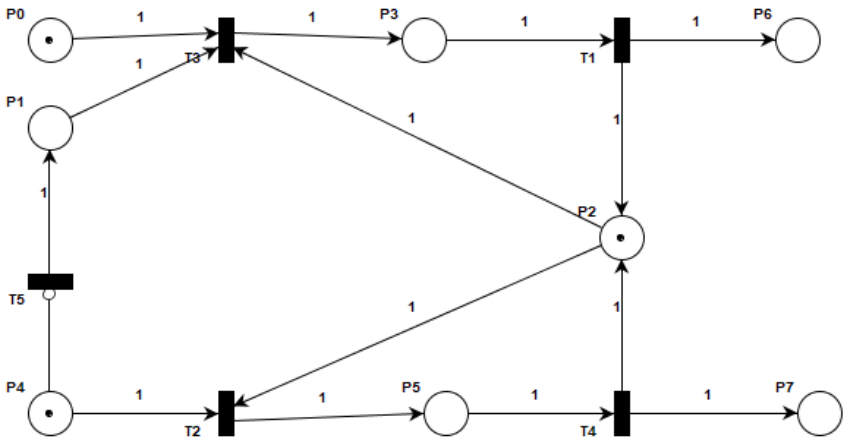
και επίσης, όταν ισχύει: $H(p, t) = 1 \rightarrow I(p, t)=0$

Οι κανόνες ενεργοποίησης της μετάβασης παραμένουν οι ίδιοι. Θεωρητικά, ένα δεσμευτικό τόξο γίνεται απαραίτητο όταν υπάρχουν θέσεις, που ενδεχομένως αποκτούν άπειρο αριθμό κουπονιών και το σύστημα πρέπει να ξέρει τον αριθμό των κουπονιών σε αυτές τις θέσεις. Επίσης, αν τα κουπόνια που ενεργοποιούν τη μετάβαση αποθηκεύονται σε κάποια θέση, τότε θα έχουμε ένα μη-φραγμένο PN.

Σε ένα μοντέλο PN, ο αποτρεπτικός κλάδος μπορεί να υλοποιηθεί και εναλλακτικά με μια πρόσθετη θέση και κατάλληλα τοποθετημένους κλάδους. Όπως φαίνεται στο παρακάτω παράδειγμα των Σχημάτων 3-9 και 3-10. Αν θεωρήσουμε τις παρακάτω θέσεις του Πίνακα 3-2:

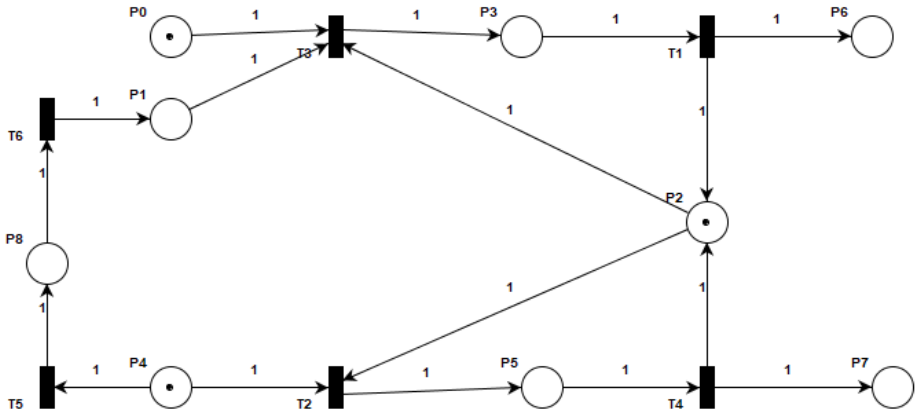
Θέση	***Εργασία – Πόρος***
P0	Τεμάχιο για Τόρνο
P1	Fake Inhibitor
P2	Robot
P3	Τόρνευση
P4	Τεμάχιο για Φρέζα
P5	Φρεζάρισμα
P6	Αποθήκη ετοίμων τόρνου
P7	Αποθήκη ετοίμων φρέζας

Πίνακας 3-2 Αντιστοίχιση Θέσεων - Εργασιών/Πόρων στο παράδειγμα αποτρεπτικού κλάδου



Σχήμα 3-9 Παράδειγμα χρήσης Αποτρεπτικού Κλάδου σε PN

Με χρήση του αποτρεπτικού κλάδου από τη θέση P4 στη μετάβαση T5 δίνουμε προτεραιότητα στη χρήση του ρομπότ να μεταφέρει το τεμάχιο που προορίζεται για φρεζάρισμα έναντι αυτού για τόρνευση. Τον παραπάνω αποτρεπτικό κλάδο, μπορούμε να τον αντικαταστήσουμε από μια επιπλέον θέση P8 και μια μετάβαση T6, που λειτουργούν ως βοηθητικά και αντικαθιστούν ουσιαστικά τον αποτρεπτικό κλάδο (Σχήμα 3-10).



Σχήμα 3-10 Αντικατάσταση του αποτρεπτικού κλάδου από μια θέση και μια μετάβαση

3.6 Άλλοι Τύποι – Επεκτάσεις Δικτύων Petri

Έως τώρα περιεγράφηκαν και αναλύθηκαν όλα τα χαρακτηριστικά και οι λειτουργίες των κλασσικών Δικτύων Petri. Όμως, με το πέρασμα των χρόνων τα δίκτυα Petri επεκτάθηκαν σε πολλές και διαφορετικές κατευθύνσεις, αναλόγως του γνωστικού αντικειμένου και των συστημάτων στα οποία κλήθηκαν να εφαρμοστούν, συμπεριλαμβάνοντας χρόνο, δεδομένα και ιεραρχία. Ως αποτέλεσμα αυτών δημιουργήθηκαν επεκτάσεις των κλασσικών PN με σκοπό την παροχή πιο περιεκτικών και παραστατικών μοντέλων με περισσότερες δυνατότητες μοντελοποίησης. Ενδεικτικά αναφέρονται παρακάτω 4 τύποι δικτύων Petri.

Time Extended or Timed Petri Nets

Αναπτύχθηκαν στα μέσα της δεκαετίας του ΄70, για να συμπεριλάβουν τον χρόνο κατά την προσομοίωση των μοντέλων. Τα βασικά μοντέλα PN είναι ασύγχρονα και δεν περιλαμβάνουν πληροφορίες χρόνου. Η ενσωμάτωση χρονισμού επιτρέπει στα PN να χρησιμοποιηθούν στην προσομοίωση συστημάτων, όπου είναι συχνά σημαντικό να περιγραφεί χρονικά η συμπεριφορά ενός συστήματος. Η υλοποίηση τέτοιων μηχανισμών γίνεται με την εισαγωγή των εννοιών της χρονικής διάρκειας και των χρονικών καθυστερήσεων στα κουπόνια, στις θέσεις και στις μεταβάσεις του μοντέλου.

Η πρώτη εισαγωγή του χρόνου στα PN έγινε από τον C. Ramchandani στο Project MIT, Analysis of Asynchronous Concurrent Systems by Timed Petri Nets, το 1974, όπου τοποθέτησε πεπερασμένες χρονικές καθυστερήσεις, που αντιπροσώπευαν χρονικούς περιορισμούς, αναφερόμενους στις μεταβάσεις του δικτύου. Με αυτόν τον τρόπο, η ενεργοποίηση μιας μετάβασης εμποδιζόταν για χρόνο ίσο με τη χρονική καθυστέρηση. Αν η μετάβαση είχε τις προϋποθέσεις (κουπόνια στις θέσεις εισόδου) και μετά τη λήξη του αντίστοιχου χρονικού διαστήματος, τότε μπορούσε να εκκινήσει. Ωστόσο, οι ντετερμινιστικές χρονικές καθυστερήσεις δεν ήταν κατάλληλες για να περιγράψουν όλες τις περιπτώσεις των συστημάτων, ειδικά στα Ευέλικτα Συστήματα Κατεργασιών που μπορεί να παρουσιάζουν τυχαία συμπεριφορά. Για τη μοντελοποίηση αυτών των συστημάτων δημιουργήθηκαν τα *στοχαστικά δίκτυα Petri*, στα οποία στις χρονικές καθυστερήσεις αντιστοιχίζονται *τυχαίες* τιμές του χρόνου.

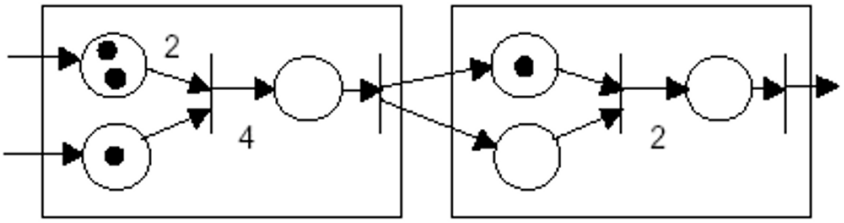
High Level - Colored Petri Nets

Τα κλασσικά PN περιέχουν επαναλαμβανόμενα στοιχεία για τη μοντελοποίηση παρόμοιων συμπεριφορών πολλαπλών αντικειμένων. Τα *δίκτυα Predicate Transition Nets (PrT),* ήταν αυτά που σχεδιάστηκαν πρώτα, με στόχο τη μείωση αυτών των στοιχείων μέσω της συγχώνευσης τους σε ένα. Αυτό επιτυγχάνεται αντικαθιστώντας τα κλασικά με άλλου τύπου κουπόνια (predicates) τα οποία αποδίδουν στην εκάστοτε θέση κάποια συγκεκριμένη σημασία. Επιπλέον, οι *κλάδοι* του δικτύου σημειώνονται με *εκφράσεις* και οι *μεταβάσεις* με *συναρτήσεις/συνθήκες*. Η τοπολογία αυτή καθορίζει τελικά ποια κουπόνια μπορούν να ενεργοποιήσουν μια μετάβαση και ποια είναι αυτά που προκύπτουν από την ενεργοποίηση της. [18]

Στις αρχές του 1980 (Jensen: “Colored Petri Nets and the invariant method” Theoritical Computer Science, 1981), εισήχθησαν τα Coloured Petri nets (CP-nets ή CPN), τα οποία αποτελούν μια γενίκευση των PrT – Nets. Η αιτία για τη δημιουργία των CPN ήταν οι δυσκολίες στην ανάλυση εξαιτίας της αντιστοίχισης των μεταβλητών που εμφανίζονταν στις εκφράσεις των κλάδων. Στα CPN οι δυσκολίες αυτές αποφεύγονται με την αντικατάσταση των εκφράσεων των κλάδων με συναρτήσεις. Σε ένα κλασικό PN τα κουπόνια είναι δυαδικά (binary), δηλαδή δηλώνουν την παρουσία ή μη κουπονιών. Αντίθετα, σε ένα CPN τα κουπόνια είναι τιμές ενός τύπου δεδομένων και ονομάζονται colours (χρώματα). Τα colours μπορεί να είναι τιμές τύπων δεδομένων ή σύνθετων τύπων δεδομένων όπως λίστες και να περιγράφουν αντικείμενα όπως αγαθά, α’ ύλες, πόρους κτλ. Επίσης, εισάγονται δύο τύποι διαδικασιών p και q, ενώ οι μεταβάσεις περιγράφουν την σχέση μεταξύ των κουπονιών εισόδου και εξόδου και επιπλέον είναι δυνατόν να περικλείουν συνθήκες ενεργοποίησης οι οποίες εκφράζονται ως εκφράσεις Boolean.[12]

Hierarchical Petri Nets

Αναπτύχθηκαν στα τέλη της δεκαετίας του ‘80 επειδή τα πραγματικά συστήματα μοντελοποιούνταν από τεράστια δίκτυα και η δομή τους είχε μεγάλη έκταση. Προκειμένου να κάνουν την μορφοποίηση των PN πιο πρακτική και εύκολη δημιουργήθηκαν τα ιεραρχικού τύπου PN. Η ιεραρχία αναφέρεται στη δομή των δικτύων, δημιουργώντας υπο-δίκτυα (subnets) και υπο-συστήματα (sub-systems). Οπότε, έχουμε αντικατάσταση των στοιχείων (θέσεων ή μεταβάσεων) του δικτύου με ένα χαμηλότερου επιπέδου δίκτυο. Ένα τέτοιο υπο-δίκτυο παρουσιάζει μια πιο αναλυτική και λεπτομερή απεικόνιση της εργασίας που περιγράφει το στοιχείο που αντικαθιστά. Κατά αυτόν τον τρόπο, τα μοντέλα PN μπορούν να δομηθούν ως ένα σύνολο επιπέδων. Τα υπο-δίκτυα αλληλεπιδρούν μεταξύ τους μέσα από ένα σύνολο κατάλληλα ορισμένων διεπιφανειών ώστε να μπορούν διαχειριστούν μεγαλύτερες εφαρμογές. Κάθε υπο-δίκτυο απεικονίζεται με ένα τετράγωνο κουτί(Εικόνα 3-9), που εμπεριέχει τμήμα του δικτύου PN.



Εικόνα 3-9 Hierarchical Δίκτυα Petri

Signal Interpreted Petri Nets (SIPN)

Ο προγραμματιστικές γλώσσες μέσω PLC χρησιμοποιούνται ευρέως για τον έλεγχο σύγχρονων συστημάτων κατεργασιών. Όμως, είναι ατελής σε συγκεκριμένες καταστάσεις. Ο Minas [2002] καταλήγει ότι στις γλώσσες των PLC λείπουν τα παρακάτω:[19]

- Δυνατότητα γραφικής περιγραφής διαδοχικών και παράλληλων αλγορίθμων
- Δυνατότητα οπτικής ανάδρασης της ροής ελέγχου στους παραπάνω αλγόριθμους
- Ευκολία εφαρμογής
- Ευκολία υλοποίησης (γρήγορη εξαγωγή κώδικα)

Τα Signal Interpreted Petri Nets (SIPN) αναπτύχθηκαν για να ικανοποιήσουν αυτές τις απαιτήσεις. Ένα SIPN είναι ένα ειδικό είδος PN με δυαδικές σημάνσεις, το οποίο ορίζεται από τα παρακάτω χαρακτηριστικά [Frey 2002]: $SIPN = (P, T, F, m_0, I, 0, \varphi, \omega, \Omega)$

Όπου, P,T, F, m0 είναι γνωστά, τα υπόλοιπα ορίζονται ως εξής:

I: μη κενός, πεπερασμένο σύνολο λογικών σημάτων εισόδου

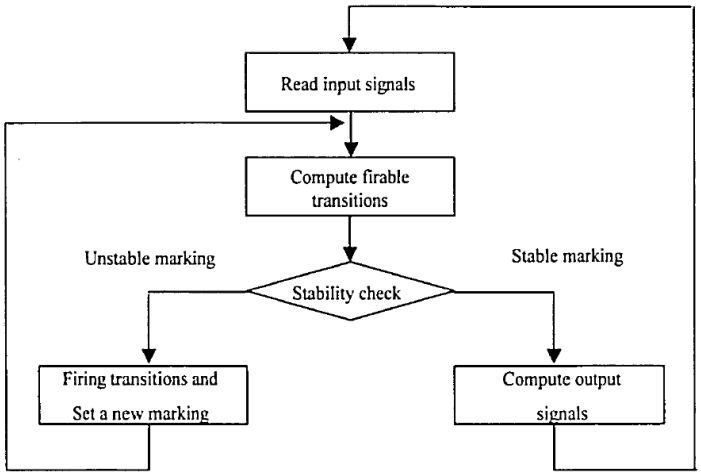
O: μη κενός, πεπερασμένο σύνολο λογικών σημάτων εξόδου, όπου $I \cap O = \emptyset$

φ: είναι ένας χάρτης/διάγραμμα που σχετίζεται με κάθε μετάβαση

ω: είναι ένας χάρτης/διάγραμμα που σχετίζεται με κάθε θέση

Ω: είναι η συνάρτηση εξόδου που συνδυάζει την ω με όλες τις σημασμένες θέσεις.

Η δυναμική συμπεριφορά του SIPN δίνεται από τη ροή των κουπονιών σε αυτό, η λειτουργία και η ροή των σημάτων στο δίκτυο φαίνεται στην Εικόνα 3-10, όπως περιγράφηκε από τον Frey (2000).



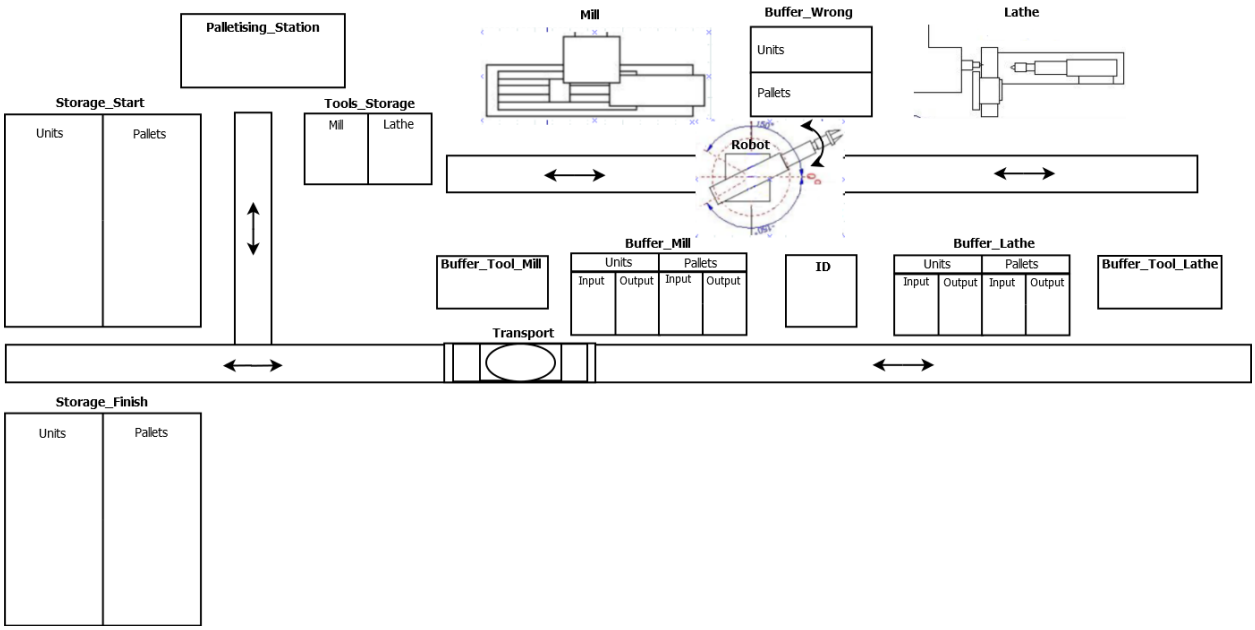
Εικόνα 3-10 Αλγόριθμος λειτουργίας των Signal Interpreted Nets (Frey-2000)[20]

4

Μοντελοποίηση του ΕΣΚ μέσω των Δικτύων Petri

4.1 Δομή – Παρουσίαση του PN

Στο παρόν κεφάλαιο παρουσιάζεται η μοντελοποίηση του ΕΣΚ, το οποίο παρουσιάστηκε στο 2ο κεφάλαιο (Εικόνα 4-1) σε Δίκτυο Petri (Σχήμα 4-1). Το δίκτυο Petri σχεδιάστηκε και προσομοιώθηκε, μέσω του λογισμικού *PIPEv4.3.0*. για μια συγκεκριμένη σειρά κατεργασιών (φρεζάρισμα → τόρνευση) από το σύνολο των έξι διαφορετικών που αναφέρθηκαν στο 2ο Κεφάλαιο.



Εικόνα 4-1 Δομή του ΕΣΚ

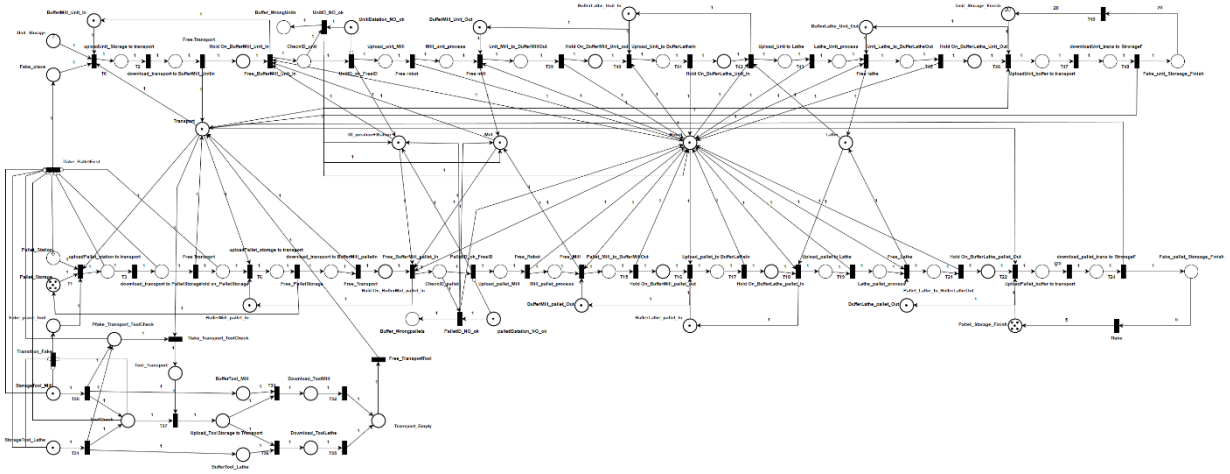
Κατά την μοντελοποίηση δημιουργήθηκαν θέσεις, στις οποίες η ύπαρξη ή μη κουπονιών περιγράφουν τη διαθεσιμότητα των κοινών πόρων (Robot, Transport, Mill, Lathe, ID), αλλά και των αποθηκών (Αρχικές, Ενδιάμεσες, Τελικές) ως προς διαθεσιμότητα τους ή μη σε τεμάχια/παλέτες/εργαλεία. Επιπλέον, μοντελοποιήθηκαν θέσεις που προσομοιάζουν εργασίες, όπως η μεταφορά τεμαχίων/παλετών/εργαλείων που πραγματοποιείται μέσω του Transport και του Robot, η εκτέλεση κατεργασιών (τόρνευση, φρεζάρισμα) μέσω των Mill και Lathe, καθώς και ειδικές καταστάσεις (βοηθητικές λειτουργίες) που

συμβαίνουν στο ΕΣΚ, όπως και η *αναμονή* σε κάποια αποθήκη, προκειμένου να αποδεσμευθεί κάποιος πόρος που συντελεί στην εκτέλεση κάποιας εργασίας. Τέλος, μοντελοποιήθηκαν θέσεις χωρίς να αντικατοπτρίζονται στο πραγματικό σύστημα (fake places), αλλά εξυπηρετούν την ροή κουπονιών εντός του PN ή επιβάλουν προτεραιότητες. Πέραν των παραπάνω, το παρόν δίκτυο Petri περιέχει αποτρεπτικούς κλάδους, προκειμένου να δίνει προτεραιότητα στην εκτέλεση συγκεκριμένων μεταβάσεων (γεγονότων), αναλόγως των αναγκών του συστήματος.

Στις παρακάτω ενότητες παρατίθενται αναλυτικά:

- Η μοντελοποίηση καταστάσεων/εργασιών/πόρων σε θέσεις (P) και γεγονότων (έναρξη και λήξη) σε μεταβάσεις (T).
- Η προσομοίωση του PN, με παράλληλη αντιστοίχιση στην λειτουργία του πραγματικού συστήματος, υπό το πρίσμα της υλοποίησης μιας παραγωγική διαδικασίας του ΕΣΚ.
- Η αιτιολόγηση της δομής και της αρχικής σήμανσης του μοντελοποιημένου PN.

Η μοντελοποίηση του ΕΣΚ σε PN παρουσιάζεται στο Σχήμα 4-1, αφορά μια συγκεκριμένη σειρά κατεργασιών (φρεζάρισμα → τόρνευση), διότι για διαφορετική σειρά κατεργασιών, θα πρέπει να αλλάξει η διαδοχή ενεργοποίησης των μεταβάσεων του δικτύου και κατ' επέκταση ολόκληρη η μοντελοποίηση του δικτύου Petri. Παρόλ' αυτά, οι γενικές αρχές της δομής του PN παραμένουν ίδιες και για τις έξι (6) περιπτώσεις, οπότε αρκεστήκαμε στη μοντελοποίηση μόνο της συγκεκριμένης περίπτωσης, όπως και στη συγγραφή του αντίστοιχου κώδικα για την συγκεκριμένη περίπτωση που θα αναλυθεί στο 6ο κεφάλαιο. Η μοντελοποίηση και ανάλυση των 5 επιπλέον περιπτώσεων είναι περιττή και τέτοιας έκτασης που ξεφεύγει από τα πλαίσια της παρούσας εργασίας.

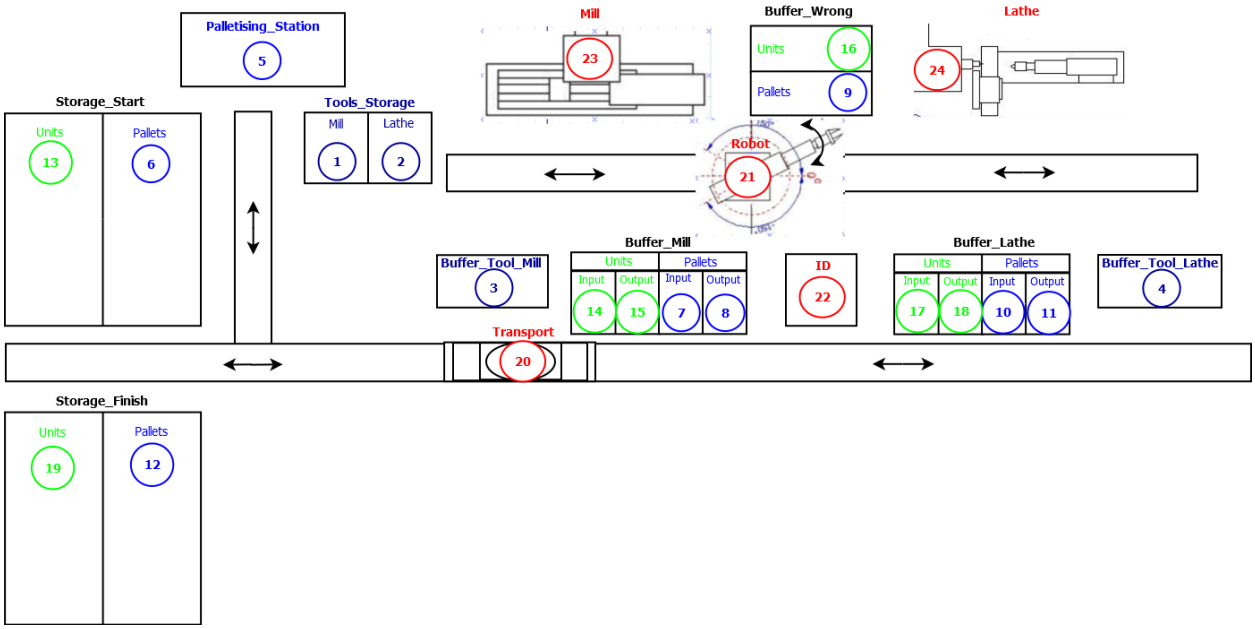


Σχήμα 4-1 Δίκτυο Petri του ΕΣΚ

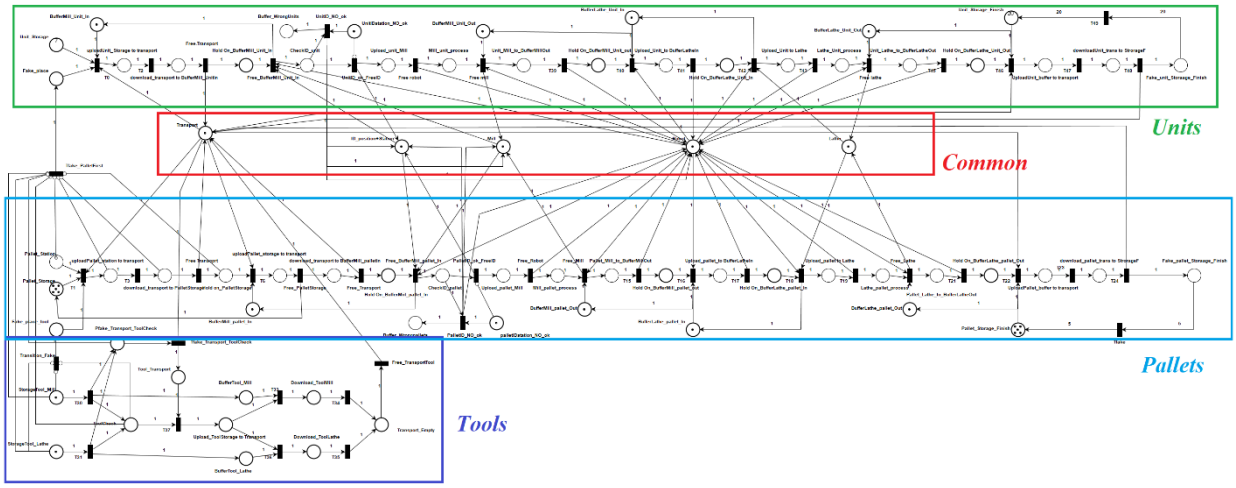
Αρχικά, τηρήθηκε και στο PN η ομαδοποίηση που είχε γίνει και στο ΕΣΚ στο 2ο κεφάλαιο (Εικόνα 4-2), κατά την οποία ομαδοποιήθηκαν οι Σταθμοί του ΕΣΚ, χωρίζοντας όλες τις θέσεις σε ομάδες, αναλόγως της διαδικασίας που εκτελούν (Σχήμα 4-2) όπως παρακάτω:

- Θέσεις για την μετακίνηση των Εργαλείων (Tools - μπλε κλειστό).
- Θέσεις για την μετακίνηση και κατεργασία των Παλετών (Pallets - μπλε ανοιχτό).
- Θέσεις για την μετακίνηση και κατεργασία των Τεμαχίων (Units - πράσινο).

- Θέσεις Κοινές για την μετακίνηση Εργαλείων/Παλετών/Τεμαχίων και για την κατεργασία των Παλετών/Τεμαχίων (Common – κόκκινο).



Εικόνα 4-2 Ομαδοποίηση των Σταθμών του ΕΣΚ

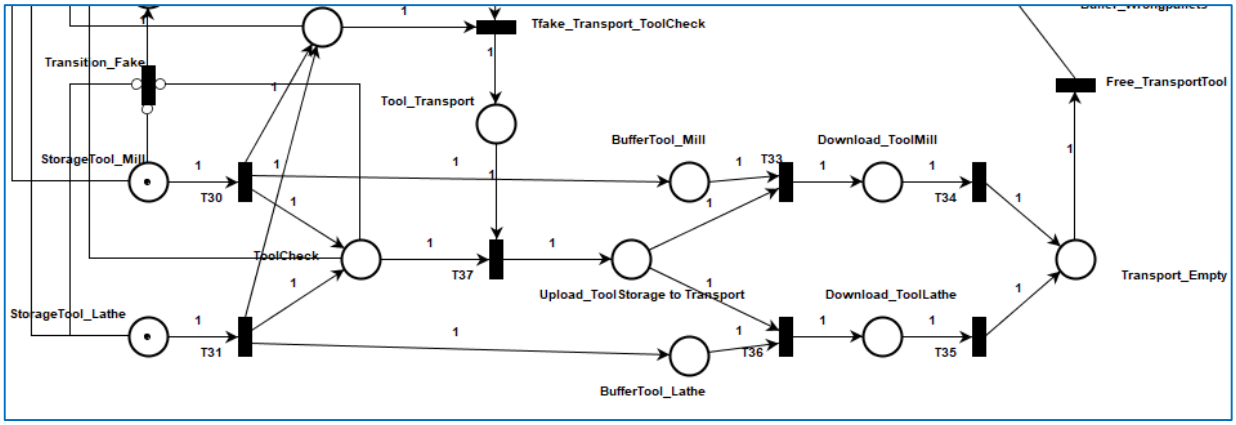


Σχήμα 4-2 Ομαδοποίηση των Θέσεων (Καταστάσεων) του PN

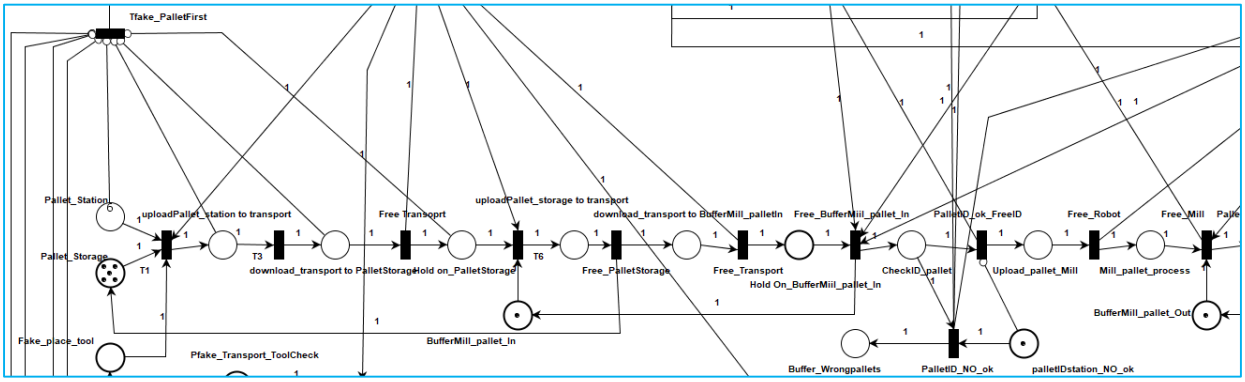
Στο PN μοντελοποιήθηκαν οι εργασίες στο ΕΣΚ με αυστηρή διάκριση αυτών, ώστε να δεσμεύονται και αποδεσμεύονται οι πόροι που χρησιμοποιούνται σε κάθε εργασία, και η σειρά εκτέλεσης παράλληλων (concurrency) εργασιών να έχει στοχαστικότητα. Η διάκριση αναφέρεται σε κάθε ξεχωριστή εργασία που εκτελείται στο PN. Π.χ. η μεταφορά ενός εργαλείου από την αρχική αποθήκη στην ενδιάμεση μέσω του Transport, διακλαδώνεται σε 2 ξεχωριστές εργασίες. Τη φόρτωση του εργαλείου από την αρχική αποθήκη στο Transport και την εκφόρτωση από το Transport στην ενδιάμεση αποθήκη.

Οι *χωρητικότητες* των θέσεων στο PN παίζουν πολύ σημαντικό ρόλο στις ιδιότητες των δικτύων Petri, όπως η *προσβασιμότητα*, η *φραγή* και η *ζωντάνια* του PN. Είναι ένα επιπλέον εργαλείο, πέραν της δομής και της αρχικής σήμανσης, με τις οποίες μπορούμε να καταλήξουμε νεκρές σημάνσεις στο PN με το δένδρο προσβασιμότητας να οδηγείται σε αδιέξοδο, κατά την ολοκλήρωση της προσομοίωσης, λόγω *εξάντλησης συγκεκριμένων τύπων πόρων* και όχι λανθασμένης κατανομής αυτών, το οποίο είναι ανεπιθύμητο. Αυτό

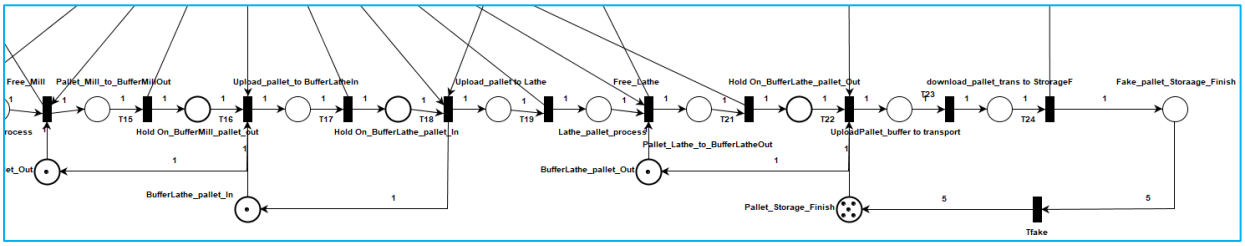
συμβαίνει, διότι το PN λειτουργεί κατά κύριο λόγο με *διαδοχικές ενέργειες (sequence)* και δεν αποτελεί κυκλική διεργασία. Εντός του PN βέβαια, υπάρχουν κυκλικές διεργασίες σε υπο-τμήματα του (τρία), θυμίζοντας την δομή που ακολουθείται στα Hierarchical Petri Nets, αλλά δεν δομήθηκαν τα υπο-τμήματα αυτά σαν sub-nets. Τα υπο-τμήματα, στα οποία εκτελείται κυκλική διαδικασία είναι οι 3 ομάδες (εργαλεία/παλέτες/τεμάχια) από τις 4 ομάδες που χωρίσαμε τις θέσεις παραπάνω στο Σχήμα 4-2. Εκτός κυκλικής διαδικασίας είναι η ομάδα των κοινών (common) θέσεων που χρησιμοποιούνται από τις υπόλοιπες. Οι 4 ομάδες θέσεων και οι αντίστοιχες μεταβάσεις τους παρουσιάζονται στα παρακάτω Σχήματα 4-3 έως 4-8,σε μεγέθυνση για μεγαλύτερη ευκρίνεια των θέσεων και μεταβάσεων.



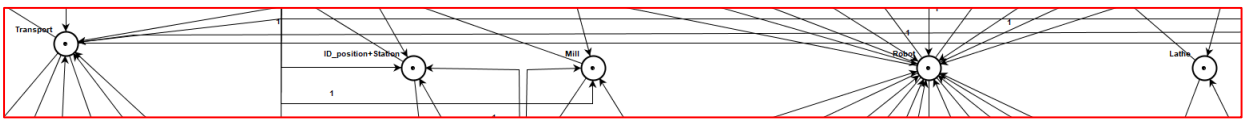
Σχήμα 4-3 Τμήμα του PN για τα Εργαλεία (Tools)



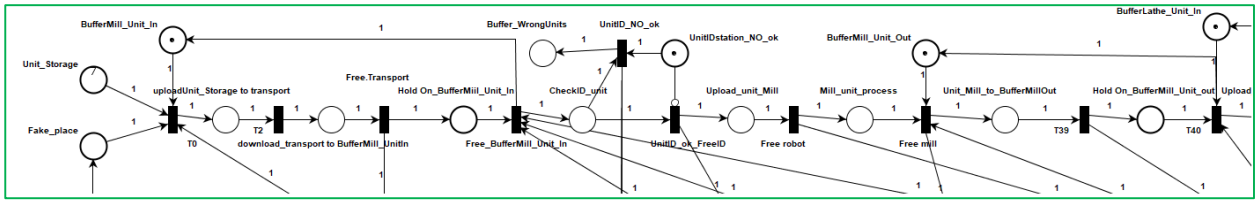
Σχήμα 4-4 Τμήμα του PN για τις Παλέτες – Υπο-τμήμα 1 (Pallets)



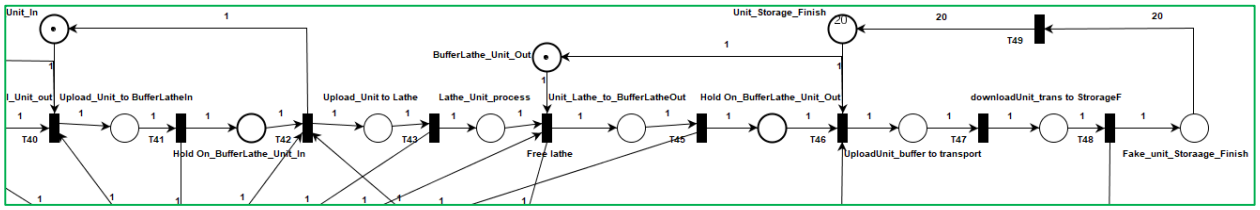
Σχήμα 4-5 Τμήμα του PN για τις Παλέτες - Υπο-τμήμα 2 (Pallets)



Σχήμα 4-6 Τμήμα του PN για τις Κοινές Θέσεις (Common)



Σχήμα 4-7 Τμήμα του PN για τα Τεμάχια - Υποτμήμα 1 (Units)



Σχήμα 4-8 Τμήμα του PN για τα Τεμάχια - Υποτμήμα 2 (Units)

Η κυκλική διαδικασία των 3 ομάδων είναι ουσιαστικά η εφαρμογή κανόνων προτεραιότητας στο PN, οι οποίες υλοποιούνται με τη χρήση αποτρεπτικών κλάδων. Οπότε, εδώ η προτεραιότητα έγκειται όχι μόνο στην έναρξη των διαδικασιών για κάθε ομάδα, αλλά και στην λήξη αυτών. Συγκεκριμένα, η διαδικασία μεταφοράς και κατεργασίας των παλετών δεν θα ξεκινήσει, αν δεν έχει τελειώσει πρώτα η διαδικασία μεταφοράς των εργαλείων και έπειτα η διαδικασία μεταφοράς και κατεργασίας των τεμαχίων δεν θα ξεκινήσει, αν δεν έχει ξεκινήσει πρώτα η τελευταία παλέτα για την αντίστοιχη διαδικασία. Με την ακόλουθη δομή καθορίζουμε τη ροή των διαδικασιών στο PN. Βασική προϋπόθεση όλων των παραπάνω είναι τα τεμάχια και οι παλέτες να έχουν στο φασεολόγιο τους την ίδια σειρά κατεργασιών, όπου εδώ είναι φρεζάρισμα → τόρνευση (milling → turning).

Τέλος υπενθυμίζεται από το 2ο κεφάλαιο ότι, όλες οι διαδικασίες μεταφοράς εργαλείων/παλετών/τεμαχίων γίνονται κατά 1 μερίδα αναλόγως του είδους, διότι για τη μοντελοποίηση του ΕΣΚ σε PN και μετέπειτα την υλοποίηση του σε κώδικα για τον έλεγχο μέσω του Arduino, πρέπει να γνωρίζουμε:

- Τον ακριβή αριθμό διαθεσιμότητας θέσεων για κάθε είδος του σταθμού διακίνησης (Transport) και
- Τον ακριβή αριθμό διαθεσιμότητας θέσεων των ενδιάμεσων αποθηκών (Buffers) όλων των ειδών.

4.2 Αντιστοίχιση Καταστάσεων του ΕΣΚ σε Θέσεις του PN

Στον Πίνακα 4-1 παρατίθεται η αντιστοίχιση των καταστάσεων/εργασιών/πόρων του ΕΣΚ σε θέσεις του PN, η ερμηνεία της αντιστοίχισης και η χωρητικότητα σε κουπόνια κάθε θέσης. Όπως είχε αναφερθεί και στο 2ο κεφάλαιο ισχύει η παραμετροποίηση των εργαλείων/τεμαχίων/παλετών σε μερίδες. Άρα οι μερίδες κάθε είδους μοντελοποιούνται σε κουπόνια. Όπου έχει τοποθετηθεί *μεταβλητή χωρητικότητα* σε κάποια θέση, αυτή θα πρέπει να ελέγχεται, πριν την έναρξη της προσομοίωσης, σε σχέση με τον αριθμό των κουπονιών της αρχικής σήμανσης (m_0). Ενώ, όπου έχουμε τοποθετήσει *σταθερή χωρητικότητα*, σε θέσεις που προσομοιάζουν αποθήκες, αυτή μπορεί να την αλλάξει ο χρήστης πριν την έναρξη της προσομοίωσης, αναλόγως των δυνατοτήτων των αποθηκών που εξαρτώνται από την αρχική σήμανση (m_0). Επειδή, οι

θέσεις στις παλέτες λειτουργούν με τις ίδιες ακριβώς διαδικασίες όπως και στα τεμάχια, παραλείπεται η αναλυτική ερμηνεία σε αυτές που επιτελούν την ίδια ακριβώς κατάσταση/διαδικασία.

Α/Α	***Κατάσταση (ΕΣΚ)***	***Θέση (PN)***	***Χωρητικό-τητα (σε κουπόνια)***	***Ερμηνεία Ύπαρξης Κουπονιού***
1	Αποθήκη Εργαλείων φρέζας	StorageTool_Mill	Σταθερό (>1) = 10	Δέσμευση - Εργαλείο φρέζας - 1 κουπόνι αντιστοιχεί σε 1 μερίδα εργαλείων φρέζας ***Τοποθετείται από τον χρήστη πριν την προσομοίωση. Όμως με προσοχή, καθώς η αποθήκη έχει πεπερασμένο αριθμό θέσεων (χωρητικότητας).*** Εδώ βάλαμε μια λογική τιμή ίση με 10.
2	Αποθήκη Εργαλείων τόρνου	StorageTool_Lathe	Σταθερό (>1) = 10	Δέσμευση - Εργαλείο τόρνου - 1 κουπόνι αντιστοιχεί σε 1 μερίδα τόρνου ***Τοποθετείται από τον χρήστη πριν την προσομοίωση. Όμως με προσοχή, καθώς η αποθήκη έχει πεπερασμένο αριθμό θέσεων(χωρητικότητας).*** Εδώ βάλαμε μια λογική τιμή ίση με 10
3	Έλεγχος Εργαλείων	ToolCheck	2	Δέσμευση - Εργαλεία (Βοηθητική) - Προκειμένου να παίρνει κάθε φορά από 1 μερίδα εργαλείων φρέζας και τόρνου
4	*Σταθμός Διακίνησης*	*Transport*	*1*	*Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι είναι διαθέσιμο*
5	Ψευδοθέση για δέσμευση Transport	Pfake_Transport_ToolCheck	2	Δέσμευση - Εργαλεία (Βοηθητική) - Προκειμένου μαζί με το Transport να ενεργοποιεί τη μετάβαση για διάθεση του, σε μεταφορά εργαλείων
6	Διάθεση Transport για εργαλεία	Tool_Transport	1	Διαθεσιμότητα (Βοηθητική) - 1 κουπόνι σημαίνει ότι είναι διαθέσιμο. Διάθεση του Transport για μεταφορά ενός εργαλείου σε κάθε μετακίνηση προς τα BufferTools Mill και Lathe.
7	Φόρτωση εργαλείου στο Transport	Upload_ToolStorage to Transport	1	Εργασία - Φόρτωση εργαλείου (Mill ή Lathe) από Αποθήκη Εργαλείων φρέζας στο Transport

Α/Α	*Κατάσταση (ΕΣΚ)*	*Θέση (PN)*	*Χωρητικότητα (σε κουπόνια)*	*Ερμηνεία Ύπαρξης Κουπονιού*
8	Ενδιάμεση Αποθήκη εργαλείων φρέζας	BufferTool_Mill	1	Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι έχει ελεύθερη θέση για εργαλείο φρέζας. Αν έχουμε στην Αποθήκη Εργαλείων φρέζας εργαλεία>1, τότε σε κάθε κυκλική διαδικασία για τη φόρτωση των εργαλείων θεωρούμε πως το εργαλείο φορτώνεται στη μηχανή πριν έρθει το επόμενο και ελευθερώνει το BufferTool_Mill.
9	Ενδιάμεση Αποθήκη εργαλείων τόρνου	BufferTool_Lathe	1	Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι έχει ελεύθερη θέση για εργαλείο τόρνου. Όπως και για τα εργαλεία φρέζας.
10	Εκφόρτωση εργαλείου φρέζας στην Ενδιάμεση Αποθήκη	Download_ToolMill	1	Εργασία - Εκφόρτωση εργαλείου φρέζας από Transport στην Ενδιάμεση Αποθήκη εργαλείων φρέζας
11	Εκφόρτωση εργαλείου τόρνου στην Ενδιάμεση Αποθήκη	Download_ToolLathe	1	Εργασία - Εκφόρτωση εργαλείου τόρνου από Transport στην Ενδιάμεση Αποθήκη εργαλείων τόρνου
12	Αποδέσμευση Transport	Transport_Empty	1	Εργασία (Βοηθητική) - Αποδέσμευση του Transport μετά την εκφόρτωση των εργαλείων στις Ενδιάμεσες Αποθήκες
13	Ψευδοθέση για την επιβολή προτεραιότητας στη μεταφορά εργαλείων	Fake_place_tool	1	Διαθεσιμότητα (Βοηθητική) - 1 κουπόνι σημαίνει πως έχουμε ολοκλήρωση μεταφοράς εργαλείων και συνέχιση με παλέτες. Επιβολή προτεραιότητας στην μεταφορά εργαλείων έναντι παλετών και τεμαχίων, μέσω αποτρεπτικών κλάδων. Βάζουμε χωρητικότητα 1, ώστε να *μην* ενεργοποιείτε συνεχώς η μετάβαση προς αυτήν και κάνουμε το PN μη-φραγμένο.
14	Αρχική Αποθήκη παλετών	Pallet_Storage	Σταθερό (>1) = 5	Διαθεσιμότητα - Όσα κουπόνια διαθέτει τόσες ελεύθερες θέσεις για παλέτες υπάρχουν. Μπορεί την χωρητικότητα να την αλλάζει ο

Α/Α	*Κατάσταση (ΕΣΚ)*	*Θέση (PN)*	*Χωρητικό-τητα (σε κουπόνια)*	*Ερμηνεία Ύπαρξης Κουπονιού*
				χρήστης πριν την προσομοίωση. Εδώ, θεωρήσαμε μια λογική τιμή π.χ. ίση με 5 θέσεις. Επίσης, κάθε φορά που φορτώνεται τεμάχιο στο Transport ελευθερώνεται και μια θέση της.
15	Σταθμός παλετοποίησης	Pallet_Station	Μεταβλητό	Δέσμευση - Παλέτα - 1 κουπόνι αντιστοιχεί σε 1 μερίδα παλετών. Όσα κουπόνια διαθέτει, τόσες παλέτες είναι έτοιμες για μετακίνηση στην Αρχική Αποθήκη παλετών ***Τοποθετείται από τον χρήστη πριν την προσομοίωση, χωρίς περιορισμό.***
16	Φόρτωση παλέτας στο Transport	uploadPallet_station to transport	1	Εργασία - Φόρτωση της παλέτας στο Transport από τον Σταθμό παλετοποίησης. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να εξυπηρετηθεί από το Transport
17	Εκφόρτωση της παλέτας στην Αρχική Αποθήκη παλετών	download_transport to PalletStorage	1	Εργασία - Εκφόρτωση της παλέτας στην Αρχική Αποθήκη παλετών από το Transport
18	Αναμονή στην Αρχική Αποθήκη παλετών	Hold on_PalletStorage	Σταθερό (>1) = 5	Δέσμευση (Βοηθητική) - Η παλέτα αναμένει στην Αρχική Αποθήκη παλετών
19	Ενδιάμεση Αποθήκη παλετών εισόδου στο Mill	BufferMill_pallet_In	1	Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι είναι διαθέσιμη. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να είναι στο Buffer.
20	Φόρτωση παλέτας στο Transport	uploadPallet_storage to transport	1	Εργασία - Φόρτωση της παλέτας στο Transport από την Αρχική Αποθήκη παλετών. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να εξυπηρετήθεί από το Transport
21	Εκφόρτωση παλέτας στο στην	download_transport to BufferMill_palletIn	1	Εργασία -

A/A	***Κατάσταση (ΕΣΚ)***	***Θέση (PN)***	***Χωρητικό-τητα (σε κουπόνια)***	***Ερμηνεία Ύπαρξης Κουπονιού***
	Ενδιάμεση Αποθήκη παλετών εισόδου στο Mill			Εκφόρτωση της παλέτας στην Ενδιάμεση Αποθήκη παλετών από το Transport
22	Αναμονή στην Ενδιάμεση Αποθήκη παλετών εισόδου στο Mill	Hold On_BufferMiil_pallet _In	1	Δέσμευση (Βοηθητική) - Η παλέτα αναμένει στην Ενδιάμεση Αποθήκη παλετών εισόδου στο Mill
23	*Ρομπότ Mitsubishi*	*Robot*	*1*	*Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι είναι διαθέσιμο*
24	*Σταθμός Ταυτοποίησης (ID)*	*ID_position+Station*	*1*	*Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι είναι διαθέσιμο*
25	Έλεγχος Ταυτοποίησης Παλέτας	CheckID_pallet	1	Εργασία – Έλεγχος Ταυτοποίησης της παλέτας από τον σταθμό ID Και μεταφοράς της στον ID μέσω του Robot
26	Αποθήκη λάθος παλετών	Buffer_Wrongpallets	Μεταβλητό	Δέσμευση - Λάθος Παλέτα – 1 κουπόνι σημαίνει 1 λάθος παλέτα κατόπιν ταυτοποίησης. Χωρίς περιορισμό, διότι μπορεί οι λάθος παλέτες να απομακρύνονται και απευθείας.
27	Ψευδοθέση για διακοπή της πορείας της παλέτας προς κατεργασία	palletIDstation_ NO_ok	1	Διαθεσιμότητα (Βοηθητική) – 1 κουπόνι σημαίνει πως μετά την ταυτοποίηση έχουμε λάθος παλέτα και διακοπή της ροής του κουπονιού (παλέτα). Μέσω του αποτρεπτικού κλάδου, δεν ενεργοποιείται η συνέχιση της ροής του κουπονιού (παλέτα), αλλά πηγαίνει στην Αποθήκη Buffer_Wrongpallets
28	*Κέντρο Κατεργασιών EMCO*	*Mill*	*1*	*Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι είναι διαθέσιμο*
29	Φόρτωση παλέτας στο Mill	Upload_pallet_Mill	1	Εργασία - Φόρτωση της παλέτας στο Mill από τον Σταθμό ID μέσω του Robot. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να εξυπηρετηθεί από το Robot
30	Φρεζάρισμα παλέτας	Mill_pallet_process	1	Εργασία -

Α/Α	*Κατάσταση (ΕΣΚ)*	*Θέση (PN)*	*Χωρητικό-τητα (σε κουπόνια)*	*Ερμηνεία Ύπαρξης Κουπονιού*
				Φρεζάρισμα παλέτας. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να κατεργαστεί το Mill
31	Ενδιάμεση Αποθήκη παλετών εξόδου από Mill	BufferMill_pallet_ Out	1	Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι είναι διαθέσιμη. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να είναι στο Buffer.
32	Εκφόρτωση παλέτας στην Ενδιάμεση Αποθήκη παλετών εξόδου από Mill	Pallet_Mill_to_Buffer MillOut	1	Εργασία - Εκφόρτωση της παλέτας από το Mill στο BufferMill_pallet_ Out μέσω του Robot. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να εξυπηρετηθεί από το Robot
33	Αναμονή στην Ενδιάμεση Αποθήκη παλετών εξόδου από το Mill	Hold On_BufferMill_ pallet_out	1	Δέσμευση (Βοηθητική) - Η παλέτα αναμένει στην Ενδιάμεση Αποθήκη παλετών εξόδου από το Mill
34	Ενδιάμεση Αποθήκη παλετών εισόδου στο Lathe	BufferLathe_pallet_In	1	Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι είναι διαθέσιμη. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να είναι στο Buffer.
35	Μεταφορά παλέτας στο BufferLathe_ pallet_In	Upload_pallet_to BufferLatheIn	1	Εργασία - Μεταφορά της παλέτας από το BufferMill_pallet_ Out στο BufferLathe_pallet_In, μέσω του ρομπότ. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να εξυπηρετηθεί από το Robot
36	Αναμονή στην Ενδιάμεση Αποθήκη παλετών εισόδου στο Lathe	Hold On_BufferLathe_ pallet_In	1	Δέσμευση (Βοηθητική) - Η παλέτα αναμένει στην Ενδιάμεση Αποθήκη παλετών εισόδου στο Lathe
37	*Κέντρο Τόρνευσης EMCO*	*Lathe*	*1*	*Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι είναι διαθέσιμο*
38	Φόρτωση παλέτας στο Lathe	Upload_pallet to Lathe	1	Εργασία - Φόρτωση της παλέτας από το BufferLathe_pallet_In στο Lathe,

A/A	*Κατάσταση (ΕΣΚ)*	*Θέση (PN)*	*Χωρητικό-τητα (σε κουπόνια)*	*Ερμηνεία Ύπαρξης Κουπονιού*
				μέσω του Robot. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να εξυπηρετηθεί από το Robot
39	Τόρνευση παλέτας	Lathe_pallet_process	1	Εργασία - Τόρνευση παλέτας. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να κατεργαστεί το Lathe
40	Ενδιάμεση Αποθήκη παλετών εξόδου από το Lathe	BufferLathe_ pallet_Out	1	Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι είναι διαθέσιμη. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να είναι στο Buffer.
41	Εκφόρτωση παλέτας στην Ενδιάμεση Αποθήκη παλετών εξόδου από Lathe	Pallet_Lathe_to_BufferLatheOut	1	Εργασία - Εκφόρτωση της παλέτας από το Lathe στο BufferLathe_ pallet_Out μέσω του Robot. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να εξυπηρετηθεί από το Robot
42	Αναμονή στην Ενδιάμεση Αποθήκη παλετών εξόδου από το Lathe	Hold On_BufferLathe _pallet_Out	1	Δέσμευση (Βοηθητική) - Η παλέτα αναμένει στην Ενδιάμεση Αποθήκη παλετών εξόδου από το Lathe
43	Τελική Αποθήκη ετοίμων παλετών	Pallet_Storage_Finish	Σταθερό (>1) = 5	Διαθεσιμότητα - Όσα κουπόνια διαθέτει τόσες ελεύθερες θέσεις για παλέτες υπάρχουν. Μπορεί την χωρητικότητα να την αλλάζει ο χρήστης πριν την προσομοίωση. Εδώ, θεωρήσαμε μια λογική τιμή π.χ. ίση με 5 θέσεις. Επίσης, σε κάθε έτοιμη παλέτα δεσμεύεται μια θέση της, έως ότου γεμίσει. Για να μην υπάρξει υπερχείλιση ελευθερώνονται όλες οι θέσεις μαζί, μέσω ενός τεχνάσματος με την Fake_pallet_Storage_Finish
44	Φόρτωση παλέτας στο Transport από την Αποθήκη	UploadPallet_buffer to transport	1	Εργασία - Φόρτωση της παλέτας από το BufferLatheOut στο Transport. Η χωρητικότητα είναι 1, διότι κάθε

Α/Α	*Κατάσταση (ΕΣΚ)*	*Θέση (PN)*	*Χωρητικό-τητα (σε κουπόνια)*	*Ερμηνεία Ύπαρξης Κουπονιού*
	παλετών εξόδου από το Lathe			φορά 1 παλέτα (μερίδα) δύναται να φορτωθεί στο Transport
45	Εκφόρτωση παλέτας από το Transport στην Τελική Αποθήκη ετοίμων παλετών	download_pallet _trans to StrorageF	1	Εργασία - Εκφόρτωση της παλέτας από το Transport. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 παλέτα (μερίδα) δύναται να δύναται να φορτωθεί στο Transport
46	Ψευδοθέση για την Τελική Αποθήκη ετοίμων παλετών	Fake_pallet_Storage_ Finish	Σταθερό (>1) = 5	Δέσμευση (Βοηθητική) - 1 κουπόνι σημαίνει 1 παλέτα στην ψευδο-αποθήκη ετοίμων. Χρησιμεύει, καθώς εδώ συγκεντρώνονται τα κουπόνια (έτοιμες παλέτες) και όταν γίνουν ίσα με το βάρος του κλάδου εξόδου (=5), στέλνει τα κουπόνια (5) στην Pallet_Storage_Finish, ώστε να ελευθερωθούν οι θέσεις της.
47	Ψευδοθέση για την επιβολή προτεραιότητας των παλετών έναντι των τεμαχίων	Fake_place	1	Διαθεσιμότητα (Βοηθητική) - 1 κουπόνι σημαίνει πως έχουν ξεκινήσει όλες οι παλέτες και μπορούν να ξεκινήσουν οι διαδικασίες για τα τεμάχια. Επιβολή προτεραιότητας στις παλέτες έναντι τεμαχίων, μέσω αποτρεπτικών κλάδων. Βάζουμε χωρητικότητα 1, ώστε να <u>μην</u> ενεργοποιείτε συνεχώς η μετάβαση προς αυτήν και κάνουμε το PN μη-φραγμένο.
48	Αποθήκη Αρχικών Τεμαχίων (Ακατέργαστων)	Unit_Storage	Σταθερό (>1) = 20	Δέσμευση - Τεμάχιο - 1 κουπόνι αντιστοιχεί σε 1 μερίδα τεμαχίων. Όσα κουπόνια διαθέτει, τόσα τεμάχια είναι έτοιμα για μετακίνηση στην BufferMill_Unit_In. ***Τοποθετείται από τον χρήστη πριν την προσομοίωση. Όμως πρέπει να είναι μικρότερος της χωρητικότητας της Αποθήκης***.
49	Ενδιάμεση Αποθήκη τεμαχίου εισόδου στο Mill	BufferMill_Unit_In	1	Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι είναι διαθέσιμη. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 τεμάχιο

A/A	*Κατάσταση (ΕΣΚ)*	*Θέση (PN)*	*Χωρητικό-τητα (σε κουπόνια)*	*Ερμηνεία Ύπαρξης Κουπονιού*
				(μερίδα) δύναται να είναι στο Buffer.
50	Φόρτωση παλέτας στο Transport	uploadUnit_Storage to transport	1	Εργασία - Φόρτωση τεμαχίου στο Transport
51	Εκφόρτωση τεμαχίου στην Ενδιάμεση Αποθήκη τεμαχίων εισόδου στο Mill	download_transport to BufferMill_UnitIn	1	Εργασία - Εκφόρτωση τεμαχίου στην Ενδιάμεση Αποθήκη τεμαχίων
52	Αναμονή στην Ενδιάμεση Αποθήκη παλετών εισόδου στο Mill	Hold On_BufferMiil _Unit_In	1	Δέσμευση (Βοηθητική) - Το τεμάχιο αναμένει στην Ενδιάμεση Αποθήκη τεμαχίων εισόδου στο Mill
53	Έλεγχος Ταυτοποίησης Τεμαχίου	CheckID_unit	1	Εργασία – Έλεγχος Ταυτοποίησης του Τεμαχίου από τον σταθμό ID Και μεταφοράς του στον ID μέσω του Robot
54	Αποθήκη λάθος τεμαχίων	Buffer_WrongUnits	Μεταβλητό	Δέσμευση - Λάθος Τεμάχιο – 1 κουπόνι σημαίνει 1 λάθος τεμάχιο κατόπιν ταυτοποίησης. Ότι ισχύει με την αντίστοιχη των παλετών.
55	Ψευδοθέση για διακοπή της πορείας του τεμαχίου προς κατεργασία	UnitIDstation_NO_ok	1	Διαθεσιμότητα (Βοηθητική) – 1 κουπόνι σημαίνει πως μετά την ταυτοποίηση έχουμε λάθος τεμάχιο και διακοπή της ροής του κουπονιού (τεμάχιο). Μέσω του αποτρεπτικού κλάδου, δεν ενεργοποιείται η συνέχιση της ροής του κουπονιού, αλλά πηγαίνει στην Αποθήκη Buffer_WrongUnits
56	Φόρτωση τεμαχίου στο Mill	Upload_unit_Mill	1	Εργασία - Φόρτωση τεμαχίου στο Mill από τον Σταθμό ID μέσω του Robot.
57	Φρεζάρισμα τεμαχίου	Mill_unit_process	1	Εργασία - Φρεζάρισμα τεμαχίου
58	Ενδιάμεση Αποθήκη τεμαχίων εξόδου από Mill	BufferMill_Unit_Out	1	Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι είναι διαθέσιμη
59	Εκφόρτωση παλέτας στην Ενδιάμεση	Unit_Mill_to_Buffer MillOut	1	Εργασία -

Α/Α	*Κατάσταση (ΕΣΚ)*	*Θέση (PN)*	*Χωρητικό-τητα (σε κουπόνια)*	*Ερμηνεία Ύπαρξης Κουπονιού*
	Αποθήκη τεμαχίων εξόδου από Mill			Εκφόρτωση του τεμαχίου από το Mill στο BufferMill_Unit_Out μέσω του Robot.
60	Αναμονή στην Ενδιάμεση Αποθήκη τεμαχίων εξόδου από το Mill	Hold On_BufferMill _Unit_out	1	Δέσμευση (Βοηθητική) - Το τεμάχιο αναμένει στην Ενδιάμεση Αποθήκη τεμαχίων εξόδου από το Mill
61	Ενδιάμεση Αποθήκη τεμαχίων εισόδου στο Lathe	BufferLathe_Unit_In	1	Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι είναι διαθέσιμη.
62	Μεταφορά τεμαχίου στο BufferLathe_ pallet_In	Upload_Unit_to BufferLatheIn	1	Εργασία - Μεταφορά της τεμαχίου από το BufferMill_unit_ Out στο BufferLathe_Unit_In, μέσω του ρομπότ.
63	Αναμονή στην Ενδιάμεση Αποθήκη τεμαχίων εισόδου στο Lathe	Hold On_BufferLathe _Unit_In	1	Δέσμευση (Βοηθητική) - Το τεμάχιο αναμένει στην Ενδιάμεση Αποθήκη τεμαχίων εισόδου στο Lathe
64	Φόρτωση τεμαχίου στο Lathe	Upload_Unit to Lathe	1	Εργασία - Φόρτωση της τεμαχίου από το BufferLathe_Unit_In στο Lathe, μέσω του Robot
65	Τόρνευση τεμαχίου	Lathe_Unit_process	1	Εργασία - Τόρνευση τεμαχίου.
66	Ενδιάμεση Αποθήκη τεμαχίων εξόδου από το Lathe	BufferLathe_ Unit_Out	1	Διαθεσιμότητα - 1 κουπόνι σημαίνει ότι είναι διαθέσιμη.
67	Εκφόρτωση τεμαχίου στην Ενδιάμεση Αποθήκη τεμαχίων εξόδου από Lathe	Unit_Lathe_to_Buffer LatheOut	1	Εργασία - Εκφόρτωση του τεμαχίου από το Lathe στο BufferLathe_ Unit_Out μέσω του Robot
68	Αναμονή στην Ενδιάμεση Αποθήκη τεμαχίων εξόδου από το Lathe	Hold On_BufferLathe_ Unit_Out	1	Δέσμευση (Βοηθητική) - Το τεμάχιο αναμένει στην Ενδιάμεση Αποθήκη τεμαχίων εξόδου από το Lathe
69	Τελική Αποθήκη ετοίμων τεμαχίων	Unit_Storage_Finish	Σταθερό (>1) = 20	Διαθεσιμότητα - Όσα κουπόνια διαθέτει τόσες ελεύθερες θέσεις για τεμάχια

A/A	*Κατάσταση (ΕΣΚ)*	*Θέση (PN)*	*Χωρητικό-τητα (σε κουπόνια)*	*Ερμηνεία Ύπαρξης Κουπονιού*
				υπάρχουν. Μπορεί την χωρητικότητα να την αλλάζει ο χρήστης πριν την προσομοίωση. Εδώ, θεωρήσαμε μια λογική τιμή π.χ. ίση με 20 θέσεις. Επίσης, σε κάθε έτοιμο τεμάχιο δεσμεύεται μια θέση της, έως ότου γεμίσει. Για να μην υπάρξει υπερχείλιση ελευθερώνονται όλες οι θέσεις μαζί, μέσω ενός τεχνάσματος με την Fake_unit_Storaage_Finish
70	Φόρτωση τεμαχίου στο Transport από την Αποθήκη τεμαχίων εξόδου από το Lathe	UploadUnit_buffer to transport	1	Εργασία - Φόρτωση του τεμαχίου από το BufferLatheOut στο Transport. Η χωρητικότητα είναι 1, διότι κάθε φορά 1 τεμάχιο (μερίδα) δύναται να φορτωθεί στο Transport
71	Εκφόρτωση τεμαχίου από το Transport στην Τελική Αποθήκη ετοίμων τεμαχίων	downloadUnit_trans to StrorageF	1	Εργασία - Εκφόρτωση του τεμαχίου από το Transport
72	Ψευδοθέση για την Τελική Αποθήκη ετοίμων τεμαχίων	Fake_unit_Storage _Finish	Σταθερό (>1) = 20	Δέσμευση (Βοηθητική) - 1 κουπόνι σημαίνει 1 τεμάχιο στην ψευδο-αποθήκη ετοίμων. Χρησιμεύει, καθώς εδώ συγκεντρώνονται τα κουπόνια (έτοιμα τεμάχια) και όταν γίνουν ίσα με το βάρος του κλάδου εξόδου (=20), στέλνει τα κουπόνια (20) στην Unit_Storage_Finish, ώστε να ελευθερωθούν οι θέσεις της.

Πίνακας 4-1 Πίνακας Θέσεων του PN με Χωρητικότητες και Ερμηνείες των κουπονιών σε αυτές

Επομένως από το σύνολο των θέσεων, αυτές που πρέπει να τοποθετούνται από τον χρήστη στην Αρχική Σήμανση m_0 για την εκτέλεση της προσομοίωσης είναι οι κάτωθι:

- Αποθήκη Εργαλείων φρέζας (StorageTool_Mill): Ο χρήστης τοποθετεί τον αριθμό κουπονιών = αριθμός μερίδων εργαλείων φρέζας$\in N_0$ < χωρητικότητα του StorageTool_Mill.
- Αποθήκη Εργαλείων τόρνου (StorageTool_Lathe): Ο χρήστης τοποθετεί τον αριθμό κουπονιών = αριθμός μερίδων εργαλείων τόρνου $\in N_0$ < χωρητικότητα του StorageTool_Lathe.
- Σταθμός Παλετοποίησης (Pallet_Station): Ο χρήστης τοποθετεί τον αριθμό κουπονιών = αριθμός μερίδων παλετών$\in N_0$, χωρίς κάποιο περιορισμό, καθώς μπορούν να φτιάχνονται παλέτες ες αεί.

- Αρχική Αποθήκη Τεμαχίων (Unit_Storage): Ο χρήστης τοποθετεί τον αριθμό κουπονιών = αριθμός μερίδων τεμαχίων $\in N_0$ < χωρητικότητα του Unit_Storage

Αν έχω μηδέν κουπόνια σε κάποια από τις παραπάνω θέσεις, τότε δεν ενεργοποιείται η μετάβαση που εκκινεί την αντίστοιχη διαδικασία για τα εργαλεία/τεμάχια/παλέτες, αλλά βάσει των κανόνων προτεραιότητας που ορίσαμε στον Πίνακα 4-1 και στο 2ο Κεφάλαιο (εργαλεία-παλέτες-τεμάχια), συνεχίζει στην επόμενη διαδικασία. Αν σε όλες τις παραπάνω θέσεις έχω μηδέν κουπόνια, τότε δεν ενεργοποιείται κάποια μετάβαση στο PN, καθώς δεν εκτελείται κάποια διαδικασία στο ΕΣΚ.

Τέλος, αναφέρεται πως δεν έχει διευκρινιστεί ο ακριβής τρόπος με τον οποίο φορτώνονται τα εργαλεία από τις ενδιάμεσες αποθήκες τους στις μηχανές κατεργασίας. Αυτό έγινε, διότι το ρομπότ δε δύναται μέσω του βραχίονα να εκτελέσει αυτήν την ενέργεια, γι' αυτό θεωρούμε πως τα εργαλεία φορτώνονται στις μηχανές, μέσω αντίστοιχων συστημάτων που κατέχουν οι ίδιες, είτε χειρωνακτικά.

4.3 Αντιστοίχιση Γεγονότων του ΕΣΚ σε Μεταβάσεις του PN – Κανόνες Ενεργοποίησης

Στον Πίνακα 4-2 παρατίθεται η αντιστοίχιση των γεγονότων (έναρξη ή λήξη) του ΕΣΚ σε μεταβάσεις του PN, μαζί με τις θέσεις εισόδου/εξόδου προς/από τις μεταβάσεις, συνοδευόμενες με τα αντίστοιχα βάρη των κλάδων που τις συνδέουν (όπου έχουμε βάρος κλάδου = 1 παραλείπεται). Επίσης, στη στήλη παρατηρήσεις δίνεται ερμηνεία για τις σημαντικότερες μεταβολές που πραγματοποιούνται κατά την έναρξη-λήξη κάθε μετάβασης, ιδιαίτερα κατά την εφαρμογή των αποτρεπτικών κλάδων, αλλά και γενικότερα, όπου δεν είναι ευδιάκριτα τα αποτελέσματα από την ενεργοποίηση μιας μετάβασης.

Σε κάποιες μεταβάσεις δεν έχει δοθεί συγκεκριμένη ονομασία, αλλά αναφέρονται ως Tj, όπου j $\in N_1$, κυρίως διότι έχουν πολλούς κλάδους εισόδου και εξόδου και επιτελούν ενεργοποιήσεις, που αλλάζουν σημαντικά τη σήμανση του PN. Η σειρά των μεταβάσεων στον Πίνακα 4-2, είναι και η σειρά που ενεργοποιούνται, όπως θα δούμε και παρακάτω στην προσομοίωση του δικτύου, όπου δίνονται παραπομπές. Τέλος, σημειώνεται ότι, οι διεργασίες στα τεμάχια (units) και αυτές των παλετών (pallets) είναι ακριβώς ίδιες, το μόνο που αλλάζει είναι η χωρητικότητα των αποθηκών αρχικών και τελικών τεμαχίων.

Α/Α	***Θέσεις (P) - Βάρη κλάδων εισόδου>1 ή 0***	***Μετα-βάσεις***	***Θέσεις (P) - Βάρη κλάδων εξόδου>1 ή 0***	***Παρατηρήσεις (Έναρξη εργασίας)***
1	StorageTool_Mill	T30	ToolCheck Pfake_Transport_Tool Check BufferTool_Mill	Κουπόνι στο BufferTool_Mill = Διαθέσιμη Θέση ***(Σχήμα 4-9)***
2	StorageTool_Lathe	T31	ToolCheck Pfake_Transport_Tool Check BufferTool_Lathe	Κουπόνι στο BufferTool_Lathe = Διαθέσιμη Θέση ***(Σχήμα 4-9)***
3	Pfake_Transport_Tool Check		Tool_Transport	

Α/Α	***Θέσεις (Ρ) - Βάρη κλάδων εισόδου>1 ή 0***	***Μετα-βάσεις***	***Θέσεις (Ρ) - Βάρη κλάδων εξόδου>1 ή 0***	***Παρατηρήσεις (Έναρξη εργασίας)***
	Transport	Tfake_ Transport_ ToolCheck		Tool_Transport έχει κουπόνι, άρα μπορεί να ενεργοποιηθεί Τ37
4	ToolCheck Tool_Transport	T37	Upload_ToolStorage toTransport	Φόρτωση εργαλείου (φρέζας ή τόρνου) στο Transport
5	BufferTool_Mill Upload_Tool StoragetoTransport	T33	Download_ToolMill	Εκφόρτωση εργαλείου (φρέζας) στο BufferMill
6	Upload_ToolStorage to Transport BufferTool_Lathe	T36	Download_ToolLathe	Εκφόρτωση εργαλείου (τόρνου) στο BufferLathe
7	Download_ToolMill	T34	Transport_Empty	Κουπόνι στο Transport_Empty = μπορεί να αποδεσμευθεί το Transport
8	Download_ToolLathe	T35	Transport_Empty	-//-
9	Transport_Empty	Free_ Transport Tool	Transport	Κουπόνι στο Transport = Αποδέσμευση
10	***StorageTool_Mill = 0*** ***ToolCheck = 0*** ***StorageTool_ Lathe = 0***	**Transition _ Fake**	**Fake_place_tool**	***Αποτρεπτικοί κλάδοι Εισόδου***. Δεν πρέπει να έχουν κουπόνι οι αποθήκες εργαλείων και το ToolCheck, προκειμένου να δεσμεύσει το Transport πριν αυτό διατεθεί στις παλέτες. ***(Σχήμα 4-10)***
11	Pallet_Station Pallet_Storage Fake_place_tool Transport	T1	uploadPallet_station to transport	Φόρτωση παλέτας στο Transport για μεταφορά από Pallet_Station στην Pallet_Storage ***(Σχήμα 4-11)***
12	uploadPallet_station to transport	T3	download_transport to PalletStorage	Εκφόρτωση παλέτας στο PalletStorage
13	download_transport to PalletStorage	Free Transport	Hold on_PalletStorage Transport	Κουπόνι στο Transport = Αποδέσμευση
14	BufferMill_pallet_In Hold on_PalletStorage Transport	T6	uploadPallet_storage to transport	Φόρτωση παλέτας στο Transport για μεταφορά στο BufferMill_pallet_In
15	uploadPallet_storage to transport	Free_Pallet Storage	download_transport to BufferMill_palletIn Pallet Storage	Εκφόρτωση παλέτας, Κουπόνι στο Pallet Storage = αποδέσμευση θέσης
16	download_transport to BufferMill_palletIn	Free_ Transport	Transport Hold On_BufferMiil_ pallet_In	Κουπόνι στο Transport = Αποδέσμευση

Α/Α	***Θέσεις (P) - Βάρη κλάδων εισόδου>1 ή 0***	***Μετα-βάσεις***	***Θέσεις (P) - Βάρη κλάδων εξόδου>1 ή 0***	***Παρατηρήσεις (Έναρξη εργασίας)***
17	Hold On_BufferMiil _pallet_In ID_position+Station Mill Robot	Free_ BufferMiil _pallet_In	BufferMill_pallet_In CheckID_pallet	Κουπόνι στο BufferMill_pallet_In = Αποδέσμευση Και πάει για ταυτοποίηση. ***(Σχήμα 4-12)***
18	CheckID_pallet palletIDstation_ NO_ok = 1	PalletID_ NO_ok	Buffer_Wrongpallets ID_position+Station Mill Robot	Αν έχει λάθος ταυτότητα η παλέτα, τότε palletIDstation_ NO_ok = 1, και πάει στο Buffer_Wrongpallets ***(Σχήμα 4-13)***
19	**CheckID_pallet** ***palletIDstation_ NO_ok = 0***	**PalletID_ ok_FreeID**	**ID_position+Station** **Upload_pallet_Mill**	***Αποτρεπτικός κλάδος Εισόδου.*** Απαγόρευση συνέχειας της διεργασίας της παλέτας αν υπάρχει κουπόνι στο palletIDstation_ NO_ok = 1 ***(Σχήμα 4-13)***
20	Upload_pallet_Mill	Free_Robot	Mill_pallet_process Robot	Κουπόνι στο Robot = Αποδέσμευση και Φρεζάρισμα
21	BufferMill_pallet_Out Mill_pallet_process Robot	Free_Mill	Pallet_Mill_to_Buffer MillOut Mill	Κουπόνι στο Mill = Αποδέσμευση Εκφόρτωση από Mill στο BufferMill_pallet_Out
22	Pallet_Mill_to_Buffer MillOut	T15	Hold On_BufferMill _pallet_out Robot	Κουπόνι στο Robot = Αποδέσμευση
23	Hold On_BufferMill _pallet_out Robot BufferLathe_pallet_In	T16	BufferMill_pallet_Out Upload_pallet_to BufferLatheIn	Μεταφορά παλέτας μέσω Robot από BufferMill_pallet_Out στο BufferLathe_pallet_In
24	Upload_pallet_to BufferLatheIn	T17	Hold On_Buffer Lathe_ pallet_In Robot	Κουπόνι στο Robot = Αποδέσμευση
25	Hold On_Buffer Lathe_pallet_In Robot Lathe	T18	BufferLathe_pallet_In Upload_pallet to Lathe	Φόρτωση στο Lathe
26	Upload_pallet to Lathe	T19	Lathe_pallet_process Robot	Κουπόνι στο Robot = Αποδέσμευση και Τόρνευση
27	Lathe_pallet_process Robot BufferLathe_ pallet_Out	Free_Lathe	Lathe Pallet_Lathe_to_ BufferLatheOut	Κουπόνι στο Lathe = Αποδέσμευση και εκφόρτωση από Lathe στο BufferLathe_pallet_Out

A/A	***Θέσεις (P) - Βάρη κλάδων εισόδου>1 ή 0***	***Μετα-βάσεις***	***Θέσεις (P) - Βάρη κλάδων εξόδου>1 ή 0***	***Παρατηρήσεις (Έναρξη εργασίας)***
28	Pallet_Lathe_to _BufferLatheOut	T21	Hold On_BufferLathe _pallet_Out Robot	Κουπόνι στο Robot = Αποδέσμευση
29	Pallet_Storage_ Finish > 0 Transport Hold On_BufferLathe _pallet_Out	T22	UploadPallet_buffer to transport BufferLathe_ pallet_Out	Μεταφορά μέσω Transport της παλέτας στο Pallet_Storage_Finish
30	UploadPallet_ buffer to transport	T23	download_pallet _trans to StrorageF	εκφόρτωση παλέτας
31	download_pallet_ trans to StrorageF	T24	Fake_pallet_ Storage_Finish Transport	Κουπόνι στο Transport = Αποδέσμευση και παλέτα στην Ψευδοθέση
32	Fake_pallet_ Storage_Finish = 5	Tfake	Pallet_Storage_ Finish = 5	Μόλις γεμίσει η Pallet_Storage_Finish = 0 κουπόνια, τότε όταν η Fake_pallet_ Storage_Finish = 5 κουπόνια, απογεμίζεται αυτομάτως η Pallet_Storage_Finish = 5 κουπόνια. ***(Σχήμα 4-14)***
33	***Hold on_ PalletStorage = 0*** ***download_transport to PalletStorage = 0*** ***uploadPallet_station to transport = 0*** ***Pallet_Station = 0*** ***ToolCheck = 0*** ***Pfake_Transport _ToolCheck = 0*** ***StorageTool_ Lathe = 0*** ***StorageTool_Mill = 0***	**Tfake_ PalletFirst**	**Fake_place**	***Αποτρεπτικοί κλάδοι εισόδου***, που δεν επιτρέπουν την εκκίνηση διεργασιών των τεμαχίων, αν πρώτα δεν έχουν τελειώσει αυτές των εργαλείων και να έχουν ξεκινήσει οι διεργασίες για όλες τις παλέτες (μπορεί η διεργασία των τεμαχίων να ξεκινήσει πριν την ολοκλήρωση όλων παλετών, όχι όμως πριν την έναρξη της τελευταίας παλέτας). ***(Σχήμα 4-15)***
34	Transport Fake_place Unit_Storage BufferMill_Unit_In	T0	uploadUnit_Storage to transport	Έναρξη Διεργασιών Τεμαχίων, Μεταφορά τεμαχίου από το Unit_Storage στο BufferMill_Unit_In μέσω του Transport
35	uploadUnit_ Storage to transport	T2	download_transport to BufferMill_UnitIn	Εκφόρτωση τεμαχίου
36	download_transport to BufferMill_UnitIn	Free. Transport	Transport Hold On_BufferMiil _Unit_In	Κουπόνι στο Transport = Αποδέσμευση

Α/Α	***Θέσεις (P) - Βάρη κλάδων εισόδου>1 ή 0***	***Μετα-βάσεις***	***Θέσεις (P) - Βάρη κλάδων εξόδου>1 ή 0***	***Παρατηρήσεις (Έναρξη εργασίας)***
37	Hold On_BufferMiil _Unit_In ID_position+Station Mill Robot	Free_ BufferMill _Unit_In	CheckID_unit BufferMill_Unit_In	Κουπόνι στο BufferMill_Unit_In = Αποδέσμευση Και πάει για ταυτοποίηση.
38	CheckID_unit UnitIDstation_NO_ok	UnitID_ NO_ok	Buffer_WrongUnits	Αν έχει λάθος ταυτότητα το τεμάχιο, τότε UnitIDstation_NO_ok = 1, και πάει στο Buffer_WrongUnits ***(Σχήμα 4-17)***
39	**CheckID_unit** ***UnitIDstation_ NO_ok = 0***	**UnitID_ok _FreeID**	**ID_position+Station** **Upload_unit_Mill**	***Αποτρεπτικός κλάδος Εισόδου.*** Απαγόρευση συνέχειας της διεργασίας της παλέτας αν υπάρχει κουπόνι στο UnitIDstation_ NO_ok = 1 ***(Σχήμα 4-17)***
40	Upload_unit_Mill	Free robot	Mill_unit_process Robot	Κουπόνι στο Robot = Αποδέσμευση και Φρεζάρισμα
41	Mill_unit_process BufferMill_Unit_Out Robot	Free mill	Mill Unit_Mill_to_ BufferMillOut	Κουπόνι στο Mill = Αποδέσμευση Εκφόρτωση από Mill στο BufferMill_Unit_Out
42	Unit_Mill_to_ BufferMillOut	T39	Hold On_BufferMill_ Unit_out Robot	Κουπόνι στο Robot = Αποδέσμευση
43	BufferLathe_Unit_In Hold On_BufferMill_ Unit_out Robot	T40	BufferMill_Unit_Out Upload_Unit_to BufferLatheIn	Μεταφορά τεμαχίου μέσω Robot από BufferMill_Unit_Out στο BufferLathe_Unit_In
44	Upload_Unit_to BufferLatheIn	T41	Robot HoldOn_BufferLathe_ Unit_In	Κουπόνι στο Robot = Αποδέσμευση
45	Hold On_BufferLathe _ Unit_In Robot Lathe	T42	BufferLathe_Unit_In Upload_Unit to Lathe	Φόρτωση στο Lathe
46	Upload_Unit to Lathe	T43	Lathe_Unit_process Robot	Κουπόνι στο Robot = Αποδέσμευση και Τόρνευση
47	Robot Lathe_Unit_process BufferLathe_ Unit_Out	Free lathe	Lathe Unit_Lathe_to_Buffer LatheOut	Κουπόνι στο Lathe = Αποδέσμευση και εκφόρτωση από Lathe στο BufferLathe_Unit_Out

Α/Α	***Θέσεις (P) - Βάρη κλάδων εισόδου>1 ή 0***	***Μετα-βάσεις***	***Θέσεις (P) - Βάρη κλάδων εξόδου>1 ή 0***	***Παρατηρήσεις (Έναρξη εργασίας)***
48	Unit_Lathe_to_Buffer LatheOut	T45	Robot HoldOn_BufferLathe_ Unit_Out	Κουπόνι στο Robot = Αποδέσμευση
49	Unit_Storage_Finish HoldOn_BufferLathe_ Unit_Out Transport	T46	BufferLathe_Unit_Out UploadUnit_buffer to transport	Μεταφορά μέσω Transport του τεμαχίου στο Unit_Storage_Finish
50	UploadUnit_buffer to transport	T47	downloadUnit_trans to StrorageF	Εκφόρτωση τεμαχίου
51	downloadUnit_trans to StrorageF	T48	Fake_unit_Storaage_ Finish Transport	Κουπόνι στο Transport = Αποδέσμευση και τεμάχιο στην Ψευδο-θέση
52	Fake_unit_Storage_ Finish = 20	T49	Unit_Storage_Finish = 20	Μόλις γεμίσει η Unit_Storage_Finish = 0 κουπόνια, τότε όταν η Fake_unit_Storage_Finish = 20 κουπόνια, απογεμίζεται αυτομάτως η Unit_Storage_Finish = 20 κουπόνια. ***(Σχήμα 4-18)***

Πίνακας 4-2 Πίνακας Μεταβάσεων του PN με κλάδους εισόδου/εξόδου από/προς Θέσεις

Η αποτύπωση του συνόλου των κανόνων ενεργοποίησης των μεταβάσεων είναι πολύ σημαντική για την εφαρμογή του ελέγχου, που θα εφαρμοστεί στο 6ο Κεφάλαιο, διότι βάσει αυτών των κανόνων θα δομηθεί ο κώδικας του μικρο-ελεγκτή Arduino. Ουσιαστικά, οι είσοδοι από μια θέση σε μια μετάβαση είναι οι Συνθήκες «Αν» («if» conditions), όπου αν ικανοποιούνται, τότε εκτελούνται οι εξισώσεις εντός της συνθήκης, οι οποίες είναι οι έξοδοι από τις μεταβάσεις προς τις θέσεις.

4.4 Προσομοίωση Δικτύου Petri

Στο μοντελοποιημένο PN έχουμε *72 Θέσεις και 52 Μεταβάσεις*. Όπως είναι φυσικό, η ανάλυση του δικτύου μέσω κάποιας μεθόδου καθίσταται αδύνατη, είτε μέσω του δένδρου προσβασιμότητας είτε μέσω αναλλοίωτων. Καθώς, τα υπολογιστικά πακέτα προσομοίωσης PN δεν μπορούν να παρέχουν το δένδρο προσβασιμότητας ή τις P-, T- αναλλοίωτες, εξαιτίας του τεράστιου όγκου των πράξεων που πρέπει να πραγματοποιήσουν σε ένα τόσο μεγάλο PN. Οπότε, ο μόνος τρόπος για την ανάλυση του δικτύου είναι η προσομοίωση του μέσω του προγράμματος PIPEv4.3.0, με την ενεργοποίηση διαδοχικά των μεταβάσεων, που δύναται να ενεργοποιηθούν.

4.4.1 Αρχική Σήμανση

Η Αρχική Σήμανση του PN, μαζί με τη δομή που είναι πλέον καθορισμένη και σταθερή, παίζει σημαντικό ρόλο για την ερμηνεία των ιδιοτήτων του δικτύου, αλλά και για την εκτέλεση των ενεργοποιήσεων των μεταβάσεων. Λόγω του μεγάλου όγκου των θέσεων, θα παρουσιάσουμε την αρχική σήμανση του δικτύου, στον παρακάτω Πίνακα 4-3, μόνο με τις θέσεις που δεν έχουν μηδενικά κουπόνια κατά την αρχική σήμανση m_0. Έχουμε αναφέρει ήδη, πως οι μόνες θέσεις, στις οποίες πρέπει να τοποθετεί κουπόνια ο χρήστης στην αρχή της προσομοίωσης είναι οι παρακάτω, στις οποίες επιλέγουμε κατά το δοκούν να βάλουμε τα εξής κουπόνια:

- StorageTool_Mill = 2
- StorageTool_Lathe = 3
- Pallet-Station = 6
- Unit_Storage = 8

Από τις παραπάνω θέσεις, η μόνη που δεν έχει περιορισμό χωρητικότητας, για την τοποθέτηση κουπονιών, είναι η Pallet_Station, καθώς ο σταθμός παλετοποίησης μπορεί να συσκευάζει συνέχεια παλέτες, χωρίς να περιορίζεται από κάπου. Αντίθετα, οι άλλες θέσεις προσομοιάζουν αποθήκες που έχουν πεπερασμένο αριθμό θέσεων στην πραγματικότητα, γι΄ αυτό και στο PN τοποθετήθηκαν συγκεκριμένες χωρητικότητες.

Στον Πίνακα 4-3, παρουσιάζεται η αρχική σήμανση του δικτύου για τις μη μηδενικές Θέσεις (Θέσεις που στην αρχική σήμανση έχουν 0 κουπόνια, παραλείπονται).

Α/Α	***Θέση***	***Αρχική Σήμανση***	***Χωρητικότητα***
1	*StorageTool_Mill (εργαλεία φρέζας)*	*2*	*10*
2	*StorageTool_Lathe (εργαλεία τόρνου)*	*3*	*10*
3	Pallet_Storage (ελεύθερες θέσεις)	5	5
4	*Pallet_Station (παλέτες)*	*6*	∞
5	*Unit_Storage (τεμάχια)*	*8*	*20*
6	BufferMill_pallet_In (ελεύθερες θέσεις)	1	1
7	**palletIDstation_NO_ok (ένδειξη λάθους παλέτας)**	**1**	**1**
8	BufferMill_pallet_Out (ελεύθερες θέσεις)	1	1
9	BufferLathe_pallet_In (ελεύθερες θέσεις)	1	1
10	BufferLathe_pallet_Out (ελεύθερες θέσεις)	1	1
11	Pallet_Storage_Finish (ελεύθερες θέσεις)	5	5
12	Transport (διαθέσιμο)	1	1
13	ID_position+Station (διαθέσιμο)	1	1
14	Mill (διαθέσιμο)	1	1
15	Robot (διαθέσιμο)	1	1
16	Lathe (διαθέσιμο)	1	1
17	BufferMill_Unit_In (ελεύθερες θέσεις)	1	1
18	**UnitIDstation_NO_ok (ένδειξη λάθους τεμαχίου)**	**1**	**1**
19	BufferMill_Unit_Out (ελεύθερες θέσεις)	1	1
20	BufferLathe_Unit_In (ελεύθερες θέσεις)	1	1
21	BufferLathe_Unit_Out (ελεύθερες θέσεις)	1	1

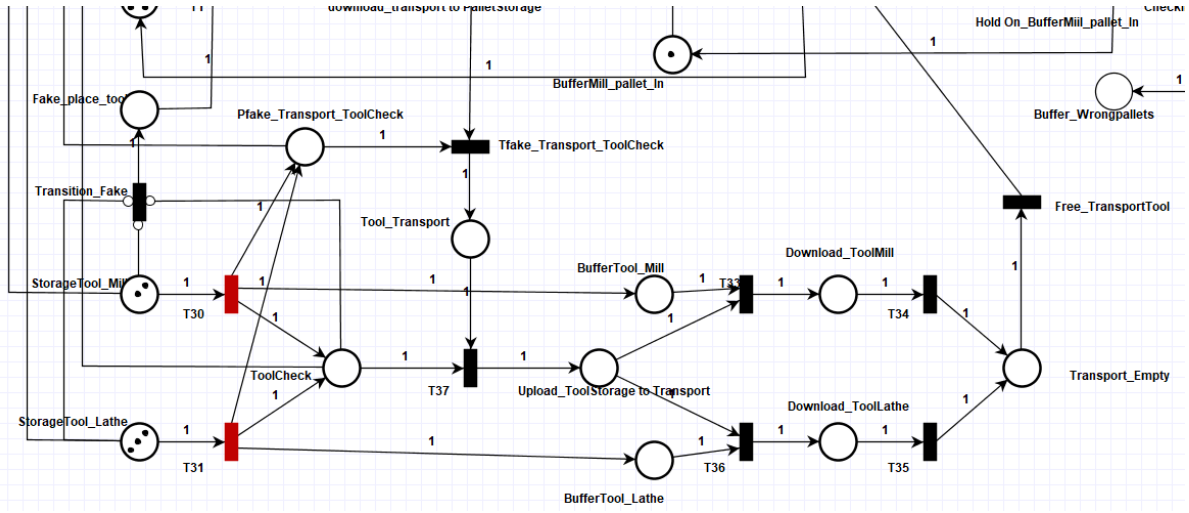
22	Unit_Storage_Finish (ελεύθερες θέσεις)	20	20

Πίνακας 4-3 Αρχική Σήμανση (m_0) του PN

4.4.2 Προσομοίωση PN

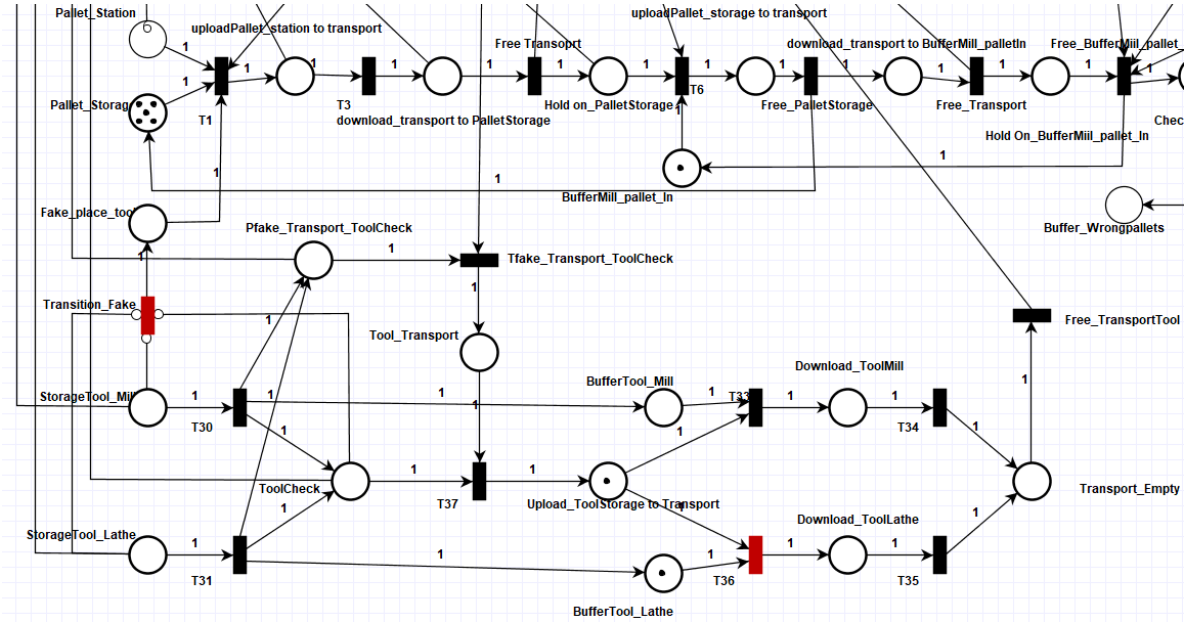
Βάση της αρχικής σήμανσης του Πίνακα 4-3, εκτελούμε την προσομοίωση του PN μέσω του λογισμικού PIPEv4.3.0. Οι εικόνες που παρουσιάζονται παρακάτω είναι σε μεγέθυνση, αναλόγως της περιοχής ενδιαφέροντος και προσομοιάζουν την κίνηση των κουπονιών *σε* επιλεγμένα/ενδεικτικά σημεία ενεργοποίησης των μεταβάσεων. *(Σημείωση: Όπου η μετάβαση είναι κόκκινη, δύναται να ενεργοποιηθεί).*

Στο Σχήμα 4-9 φαίνεται εκκίνηση της προσομοίωσης η δυνατότητα ενεργοποίησης και των 2 μεταβάσεων, για τη μεταφορά των εργαλείων από τα Tool_Storage στα αντίστοιχα Buffers, μέσω του Transport και της βοηθητικής θέσης ToolCheck.



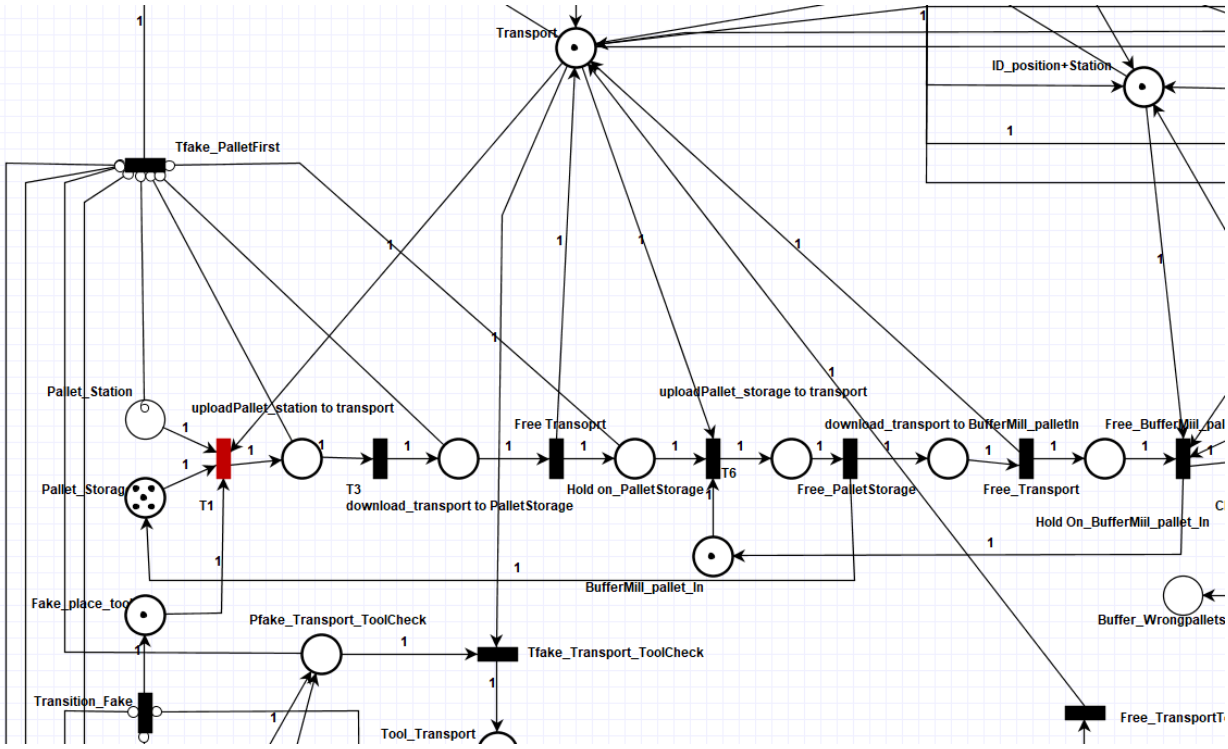
Σχήμα 4-9 Αρχική ενεργοποίηση του PN (1)

Στο Σχήμα 4-10 έχουμε Επιβολή Προτεραιότητας εκκίνησης των εργαλείων έναντι των παλετών, όπου οι αποτρεπτικοί κλάδοι δεν επιτρέπουν την έναρξη διεργασιών για τις παλέτες, αν δεν ολοκληρωθούν αυτές των εργαλείων. Έχουμε αποτρεπτικά τόξα από τις αποθήκες εργαλείων StorageTool_Mill/Lathe για να εξασφαλίσουμε ότι όλα τα εργαλεία έχουν ξεκινήσει την μεταφορά τους προς τα Buffrers, αλλά και από το ToolCheck για να δεσμεύσει πρώτο το Transport, έναντι των παλετών για την φόρτωση του τελευταίου εργαλείου, διότι δεν επαρκούν μόνο τα δεσμευτικά τόξα από τις StorageTool_Mill και StorageTool_Lathe.



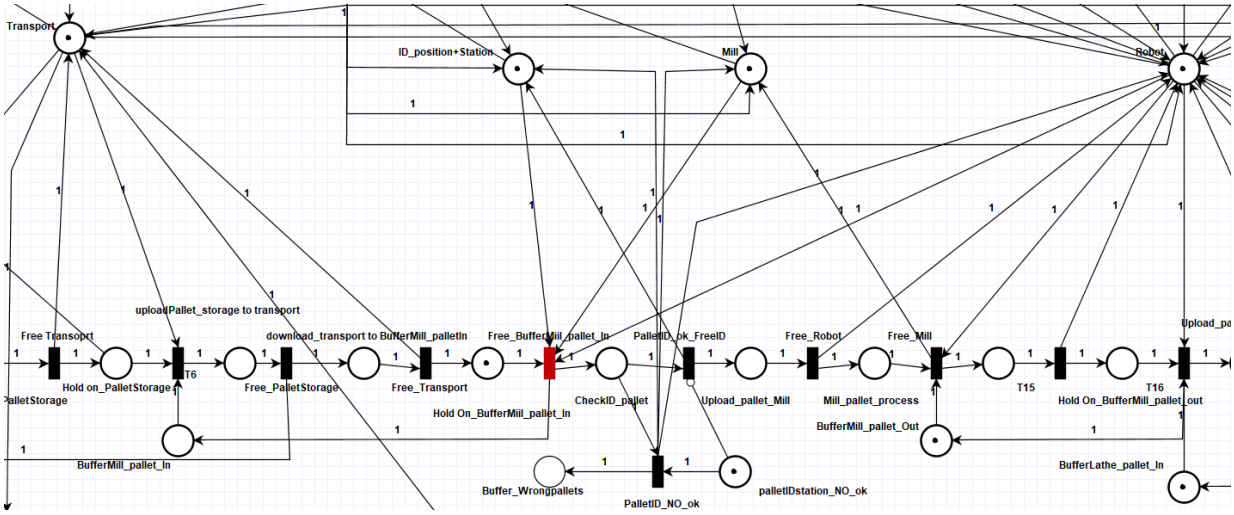
Σχήμα 4-10 Αποτρεπτικοί κλάδοι για επιβολή προτεραιότητας στα εργαλεία έναντι των παλετών (2)

Στο Σχήμα 4-11 παρουσιάζεται η μεταφορά παλέτας από το Pallet_Station στο Pallet_Storage, μέσω του Transport. Θα πρέπει να υπάρχουν κουπόνια στο Pallet_Station (παλέτα), στη Pallet_Storage (διαθέσιμη θέση), το Transport (διαθέσιμο) και στη Fake_place_tool, που πιστοποιεί ότι έχουν τελειώσει οι διεργασίες των εργαλείων.



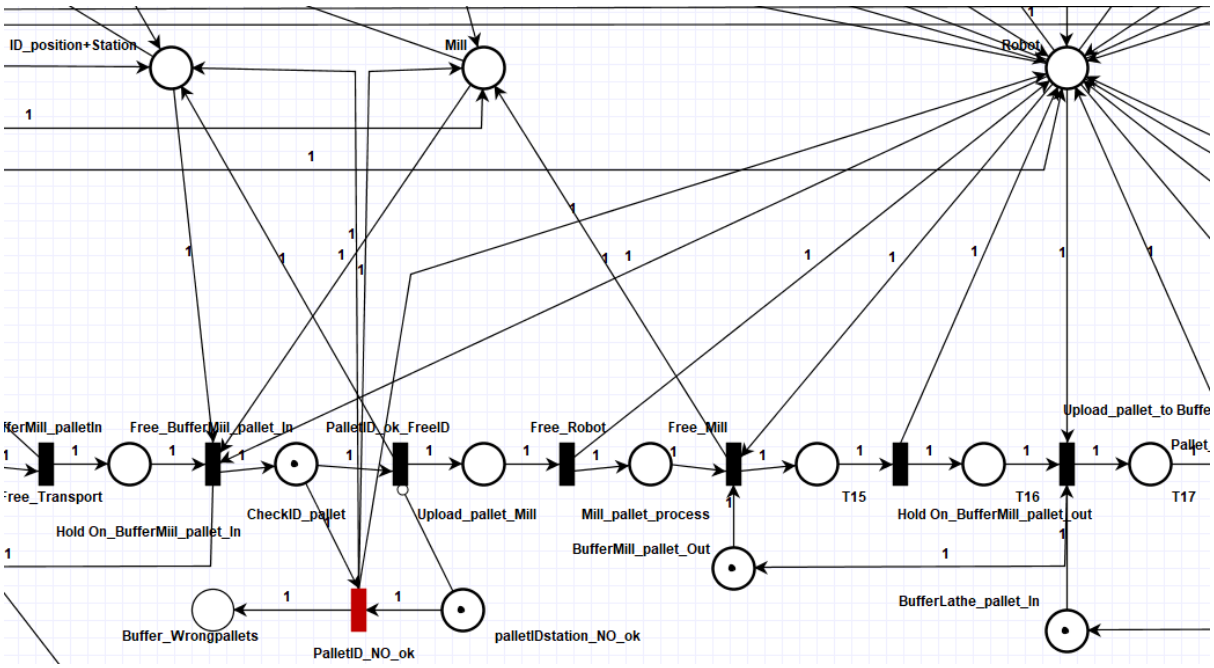
Σχήμα 4-11 Εκκίνηση διαδικασιών για παλέτες (3)

Στο Σχήμα 4-12 φαίνεται η διαδικασία για να φορτωθεί στο Mill η παλέτα, μέσω του Robot, αφού περάσει πρώτα από τη ID_position+station, και εφόσον ταυτοποιηθεί ως σωστό πάει στο Mill, διαφορετικά πηγαίνει στο Buffer_Wrongpallets.



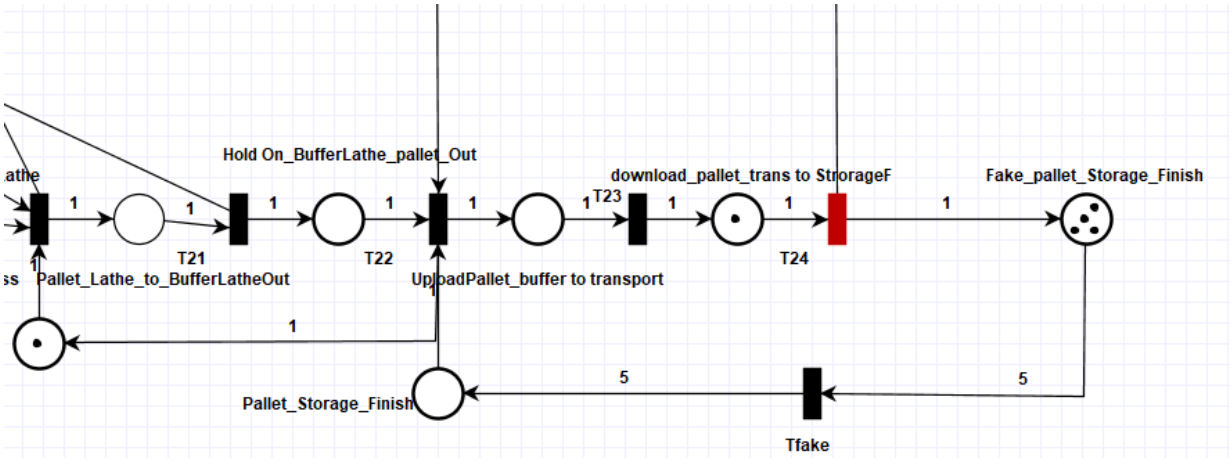
Σχήμα 4-12 Φόρτωση παλέτας μέσω Robot στο ID_position+Στατιον και μετά στο Mill (4)

Στο Σχήμα 4-13, κουπόνι στη palletIDstation_NO_ok σημαίνει πως έχουμε λανθασμένη παλέτα! Άρα, δεν έχω περαιτέρω διεργασία για αυτήν, λόγω του αποτρεπτικού κλάδου δεν ενεργοποιείται η μετάβαση PalletID_ok_FreeID, αλλά η παλέτα πηγαίνει στο Buffer_Wrongpallets.



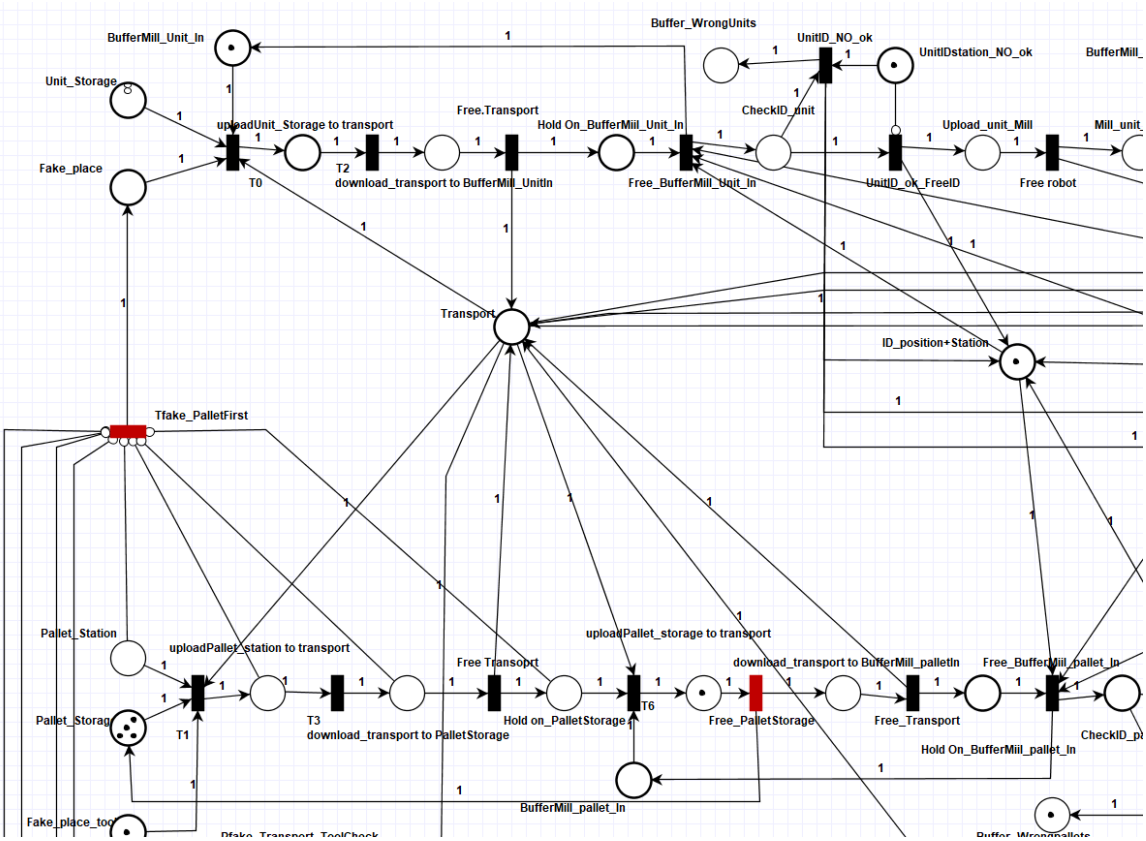
Σχήμα 4-13 Χρήση Αποτρεπτικού κλάδου για μη περαιτέρω διεργασία λανθασμένης παλέτας (5)

Στο Σχήμα 4-14, η θέση Pallet_Storage_Finish = 0 ερμηνεύεται πως είναι γεμάτη από παλέτες (0 κενές θέσεις). Οι παλέτες, μόλις φτάσουν στο Fake_pallet_Storage_Finish, πρέπει να συγκεντρωθούν 5 για να ενεργοποιηθεί η μετάβαση Tfake και να αδειάσει από παλέτες το Pallet_Storage_Finish = 5 (κενές θέσεις). Αυτό υλοποιείται εισάγοντας βάρη στους κλάδους εισόδου και εξόδου στην Tfake, ίσα με 5 κουπόνια.



Σχήμα 4-14 Απογέμιση της Αποθήκης ετοίμων παλετών μόλις γεμίσει, μέσω ψευδο-θέσης (6)

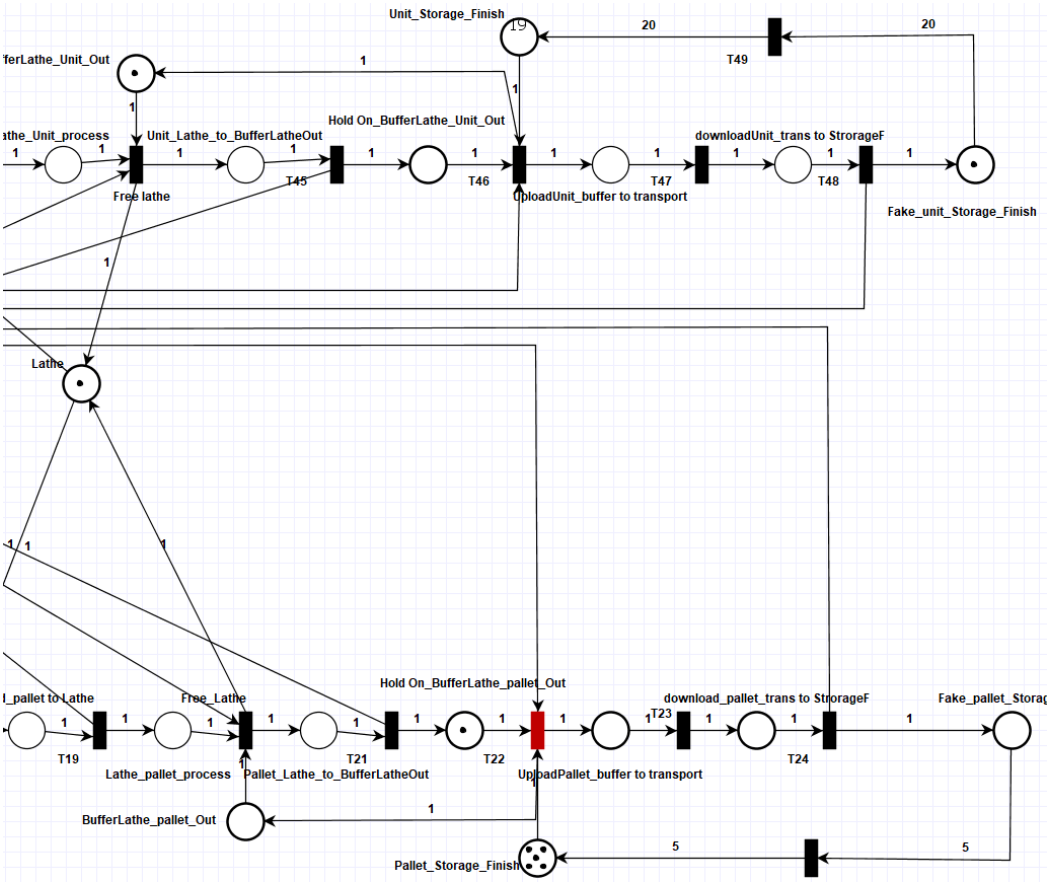
Στο Σχήμα 4-15, φαίνεται η ενεργοποίηση της μετάβασης Tfake_PalletFirst, μόλις η τελευταία παλέτα φύγει από την Αποθήκη Αρχικών Παλετών, δηλαδή όταν μεταφερθεί (κουπόνι) στη θέση *uploadPallet_storage to Transport*. Έτσι, μέσω οκτώ (8) αποτρεπτικών κλάδων καθορίζουμε την προτεραιότητα έναρξης των διεργασιών για κάθε είδος (εργαλείο/παλέτα/τεμάχιο). Οι οκτώ αποτρεπτικοί κλάδοι (Πίνακας 4-2), προέρχονται από θέσεις των εργαλείων και των παλετών, έτσι καθορίζεται η εκκίνηση των διεργασιών των τεμαχίων να γίνει τελευταία, ενώ πρωτύτερα στο Σχήμα 4-10 η επιβολή προτεραιότητας ήταν μεταξύ παλετών – εργαλείων.



Σχήμα 4-15 Αποτρεπτικοί κλάδοι για έναρξη τεμαχίων μετά την έναρξη τελευταίας παλέτας (7)

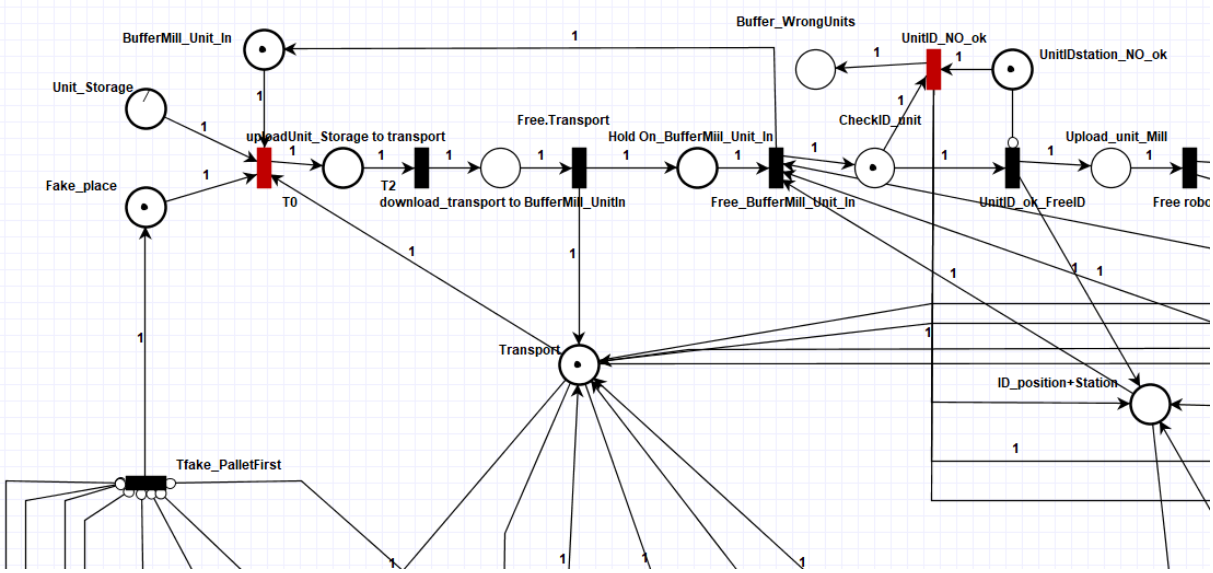
Στην προκειμένη περίπτωση, η έναρξη διεργασιών για τα τεμάχια ξεκινούν όταν η τελευταία παλέτα φθάσει στη θέση *uploadPallet_storage to Transport*. Οπότε, γίνεται αντιληπτό, ότι κάποια τεμάχια μπορεί να περάσουν στην ροή των διεργασιών κάποιες από τις τελευταίες παλέτες (Σχήμα 4-16), ακόμα να

τελειώσουν και πρώτα. Για να το αποτρέψουμε αυτό, δηλαδή να τελειώσουν πρώτα όλες οι παλέτες και μετά να ξεκινήσουν τα τεμάχια, θα έπρεπε να βάλλουμε αποτρεπτικούς κλάδους από όλες τις θέσεις , από τις οποίες διέρχονται οι παλέτες, προς τη μετάβαση *Tfake_PalletFirst* (Σχήμα 4-15).



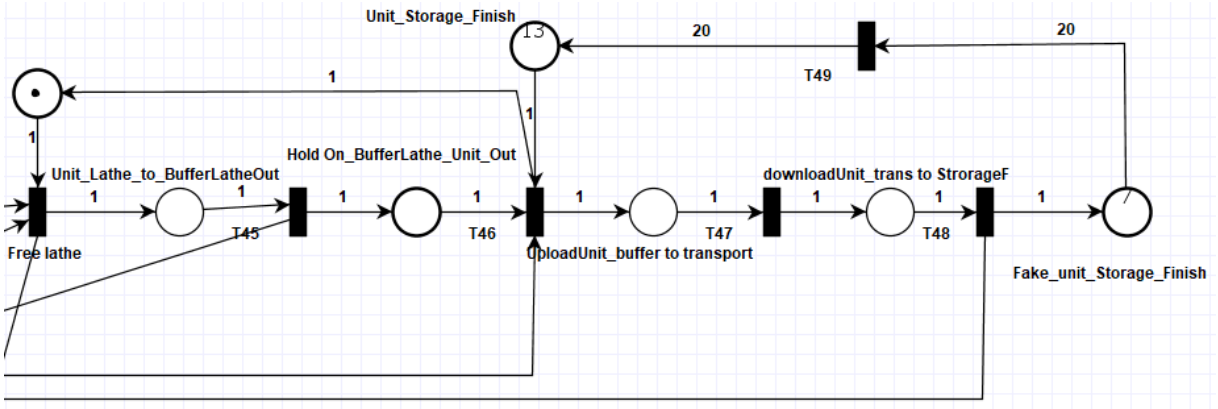
Σχήμα 4-16 Ολοκλήρωση 1ου τεμαχίου έναντι ολοκλήρωσης 1ης παλέτας (8)

Στο Σχήμα 4-17, όπως και για τις παλέτες, κουπόνι στο UnitIDstation_NO_ok = 1, σημαίνει πως έχουμε λανθασμένο τεμάχιο! Άρα, δεν έχω περαιτέρω διεργασία για αυτήν λόγω του αποτρεπτικού κλάδου, οπότε το τεμάχιο πηγαίνει στο Buffer_WrongUnits.



Σχήμα 4-17 Χρήση Αποτρεπτικού κλάδου για μη περαιτέρω διεργασία λανθασμένου τεμαχίου (9)

Στο Σχήμα 4-18 (ολοκλήρωση της προσομοίωσης), η θέση Unit_Storage_Finish = 13 διαθέτει 7 έτοιμα τεμάχια (13 κενές θέσεις). Τα τεμάχια, μόλις φτάσουν στο Fake_unit-Storage_Finish, πρέπει να συγκεντρωθούν 20 κουπόνια για να ενεργοποιηθεί η T49 και να αδειάσει από τεμάχια το Unit_Storage_Finish = 20 (κενές θέσεις). Αυτό υλοποιείται εισάγοντας βάρη στους κλάδους εισόδου και εξόδου στην T49 ίσα με 20 κουπόνια.



Σχήμα 4-18 Απογέμιση της Αποθήκης ετοίμων τεμαχίων μόλις γεμίσει, μέσω ψευδο-θέσης (10)

4.4.3 Ιδιότητες PN - Παρατηρήσεις

Παραλληλία - Συγκρούσεις

Στο κεφάλαιο 4.4.2 των προσομοιώσεων, παρατηρήθηκε για κάθε σήμανση του δικτύου, να μπορούν να ενεργοποιηθούν περισσότερες από μία μεταβάσεις. Αυτό συμβαίνει, είτε επειδή έχουμε παράλληλες εργασίες (*concurrency*), είτε όταν δύο ή περισσότερες μεταβάσεις ανταγωνίζονται τον ίδιο πόρο (κουπόνι από μία θέση) και έχουμε σύγκρουση (*conflict*). Στην περίπτωση της σύγκρουσης, ο ανταγωνισμός πόρου γίνεται για κάποια από τις κοινές θέσεις (*common*), οπότε η σειρά ενεργοποίησης κάθε μετάβασης διαθέτει μια *στοχαστικότητα*, η οποία επαφίεται στις επιλογές του προγράμματος που είναι «τυχαίες». Βέβαια, ο ανταγωνισμός πόρων που πραγματοποιείται στο υπόψη PN, είναι ελεγχόμενος και δεν επηρεάζει την παραγωγική διαδικασία, έτσι όπως ορίστηκε αρχικώς.

Λανθασμέν-η/ο Παλέτα/Τεμάχιο

Όσον αφορά την ένδειξη λάθους τεμαχίου στα PN, αυτή δεν μπορεί να μοντελοποιηθεί με ακρίβεια καθώς δεν υπάρχει τρόπος κατά την προσομοίωση του δικτύου να «διαβάζεται» σε κάθε παλέτα/τεμάχιο η ταυτότητα της/του, γι' αυτό επιλέγουμε η 1η παλέτα και το 1ο τεμάχιο να είναι τα λανθασμένα. Στο 6ο κεφάλαιο κατά την υλοποίηση του δικτύου σε κώδικα, μέσω του μικρο-ελεγκτή Arduino μπορούμε σε κάθε παλέτα ή τεμάχιο να αλλάζουμε την συνθήκη ταυτοποίησης του, μέσω αντίστοιχης εντολής που διαβάζει την ταυτότητα.

Ιδιότητες PN

Ακόμα και πριν την προσομοίωση του PN, είναι δυνατόν να διακρίνουμε τις ιδιότητες του. Συγκεκριμένα, για την δεδομένη αρχική σήμανση είναι *20-φραγμένο*, διότι καμία θέση δεν θα λάβει μεγαλύτερο αριθμό κουπονιών από 20, για οποιαδήποτε σήμανση του PN. Επιπλέον, με τους περιορισμούς που έχουμε θέσει στις χωρητικότητες των θέσεων, το PN θα είναι πάντα φραγμένο, διότι οι μόνες περιπτώσεις να είναι μη – φραγμένο, είναι είτε για άπειρο αριθμό κουπονιών στο Pallet_Station, κάτι που δεν είναι εφικτό καθώς οι παλέτες είναι πεπερασμένες, είτε αν δημιουργούνται νέα κουπόνια σε κάποια σήμανση από αποτρεπτικούς κλάδους, το οποίο επίσης δεν γίνεται λόγω των χωρητικοτήτων/περιορισμών που έχουν τεθεί.

Στο PN τέλος, γνωρίζουμε πως έχει ένα αδιέξοδο (*deadlock*), καθώς θα οδηγηθούμε σε νεκρή σήμανση, κατά την οποία καμία μετάβαση θα ενεργοποιείται, *λόγω εξάντλησης των πόρων*, που πόροι του συστήματος που εξαντλούνται είναι τα εργαλεία, τα τεμάχια και οι παλέτες.

5

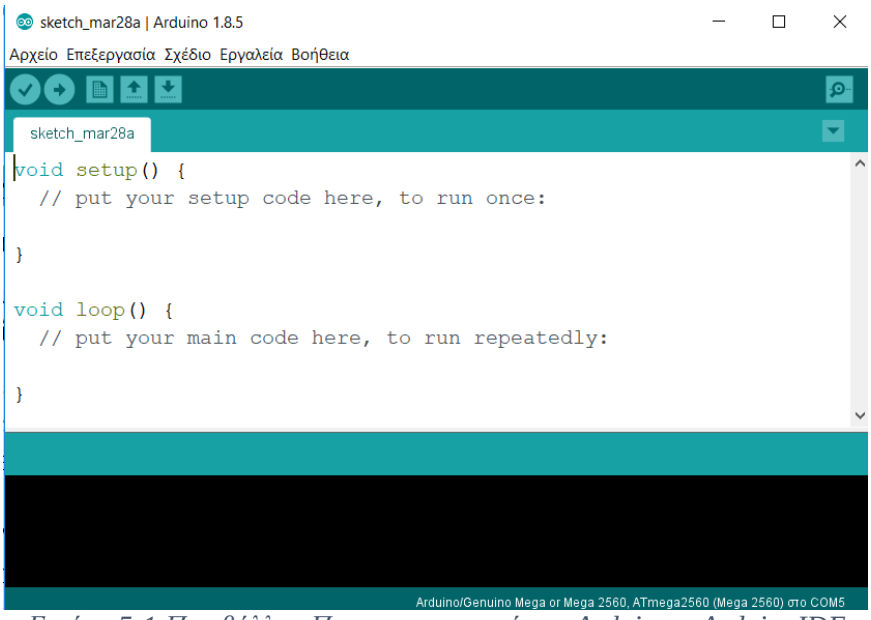
Μικρο-ελεγκτές Arduino και Σειριακή Επικοινωνία

5.1 Γενικά Στοιχεία για τους Μικρο-ελεγκτές Arduino

Για την υλοποίηση ελέγχου στο εξεταζόμενο ΕΣΚ, χρησιμοποιήθηκαν δύο τύποι (πλακέτες) μικρο-ελεγκτών Arduino:

- Το Arduino-UnoRev3 για τους τοπικούς ελεγκτές
- Το Arduino-Mega 2560Rev3 για τον κεντρικό ελεγκτή.

Το Arduino είναι μια «ανοικτού κώδικα» πλατφόρμα «προτυποποίησης» ηλεκτρονικών. Αποτελείται από μια προγραμματιζόμενη πλακέτα, έναν μικρο-ελεγκτή καθώς και από το λογισμικό που χρειάζεται για την λειτουργία του. Αναλυτικότερα, το Arduino είναι ένας *single-board* μικρο-ελεγκτής, δηλαδή μια απλή μητρική πλακέτα ανοικτού κώδικα, με ενσωματωμένο μικρο-ελεγκτή και εισόδους/εξόδους (I/O pins), και η οποία μπορεί να προγραμματιστεί στο περιβάλλον προγραμματισμού *Arduino IDE (Εικόνα 5-1)* που έχει βασιστεί στη *Wiring* (πρόκειται για τη γλώσσα προγραμματισμού C++ και ένα σύνολο από βιβλιοθήκες, υλοποιημένες επίσης στην C++). [21][22]



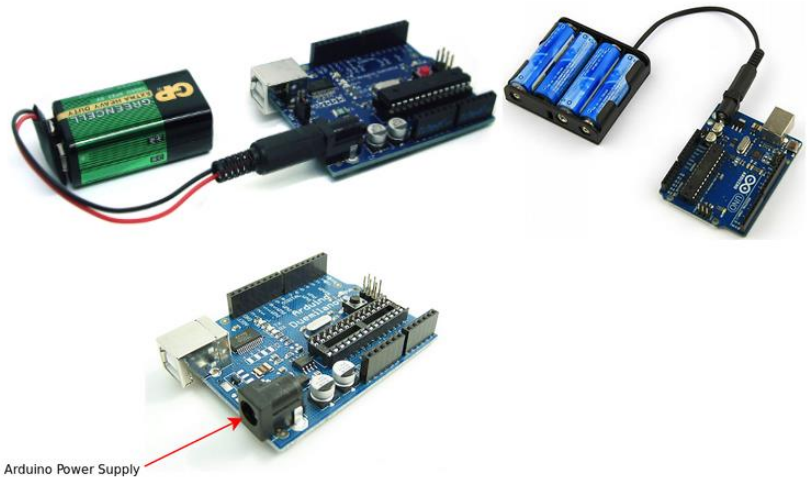
Εικόνα 5-1 Περιβάλλον Προγραμματισμού του Arduino – ArduinoIDE

Εκτός της βασικής έκδοσης του περιβάλλοντος *Arduino IDE*, υπάρχει και μια παραλλαγμένη έκδοση του *Scratch*, η οποία μπορεί να χρησιμοποιηθεί για να γράψουμε προγράμματα για το Arduino, η *S4A - Scratch For Arduino*, η οποία είναι επίσης ανοικτού κώδικα και δωρεάν. Το πλεονέκτημα της έκδοσης αυτής είναι ο οπτικός προγραμματισμός (blocks όπως στο Scratch) σε σχέση με τη γραφή εντολών στο κλασσικό περιβάλλον. Παρόμοιας λογικής είναι και το ArduBlock, το οποίο επίσης χρησιμοποιεί οπτικό προγραμματισμό μέσω έτοιμων blocks για τον προγραμματισμό του. Ακόμα, υπάρχουν οπτικές εκδόσεις στο διαδίκτυο, όπως το BlocklyDuino ή το ArduinoMio. [23]

Το Arduino αποτελείται από έναν *μικρο-επεξεργαστή (micro-processor)*, τον *ATmega* (αναλόγως της πλακέτας διαφορετικός) της *Atmel*. Οι μονάδες εισόδου / εξόδου (I/O pins) χωρίζονται σε Ψηφιακές και Αναλογικές. Επίσης, υπάρχει πληθώρα συσκευών, συμβατές με τις πλακέτες Arduino. Κάποιες από αυτές είναι: αισθητήρες θερμοκρασίας-υγρασίας-δύναμης-πίεσης, μαγνητόμετρα, γυροσκόπια, επιταχυνσιόμετρα, κ.α. Επίσης, με το Arduino ελέγχονται motors DC (βηματικοί (stepper) και servo), leds, φώτα (220v), ρελέ κ.α.

Το Arduino μπορεί να προγραμματιστεί από έναν κοινό Η/Υ, συνδέοντας την σειριακή θύρα που υποστηρίζει ο μικροεπεξεργαστής ATmega (USB B) με τη θύρα USB του υπολογιστή (USB 2.0 ή 3.0). Η σειριακή σύνδεση (Serial over USB) χρησιμοποιείται για τη μεταφορά προγραμμάτων προς την πλακέτα Arduino και αντίστροφα μεταφέρει δεδομένα, που λαμβάνει ο Arduino από τις συνδεόμενες περιφερειακές συσκευές, προς τον Η/Υ.[24]

Οι περισσότερες πλακέτες Arduino μπορούν να τροφοδοτηθούν από μπαταρία ή τροφοδοτικό, συνδέοντας στο power jack (Εικόνα 5-2), η οποία πρέπει να είναι 7V έως 12V, ή από το ίδιο USB που χρησιμοποιείτε για τη σύνδεση με τον Η/Υ. Ενώ, η ίδια παρέχει σταθερή τάση 5V στις εξόδους της (pins). Επίσης, μπορεί να λάβει τάση 5V και από τα pins Vin και GND μέσω καλωδίων τύπου jumper (Εικόνα 5-2).



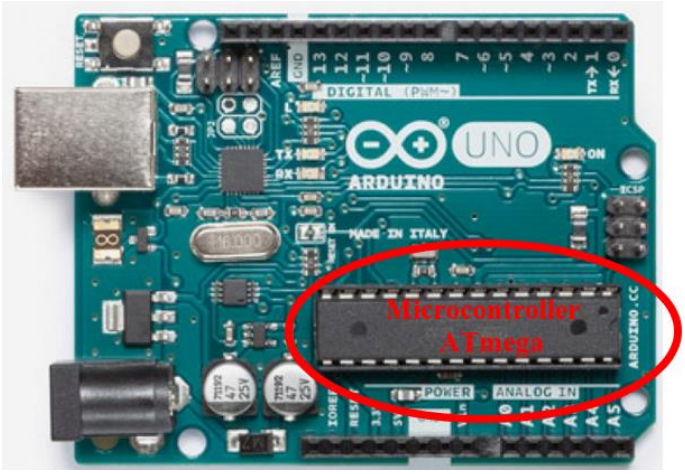
Εικόνα 5-2 Τροφοδοσία Arduino μέσω συσσωρευτών

Οι πλακέτες Arduino διαφέρουν μεταξύ τους, κυρίως στον επεξεργαστή ATmega και στον αριθμό των I/O pins, με τα υπόλοιπα χαρακτηριστικά να παραμένουν σχεδόν ίδια. Τα Arduino διακρίνονται στις παρακάτω κύριες κατηγορίες:[25]

- Τις βασικές πλακέτες: Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro, Arduino ADK, Arduino DUE κ.α.
- Τις πλακέτες με πρόσβαση στο Internet: Arduino Ethernet
- Τα shields για Arduino: Wi-Fi Shield, Motor Shield, Ethernet Shield, SD Shield κ.α., τα οποία επεκτείνουν τις δυνατότητες του.

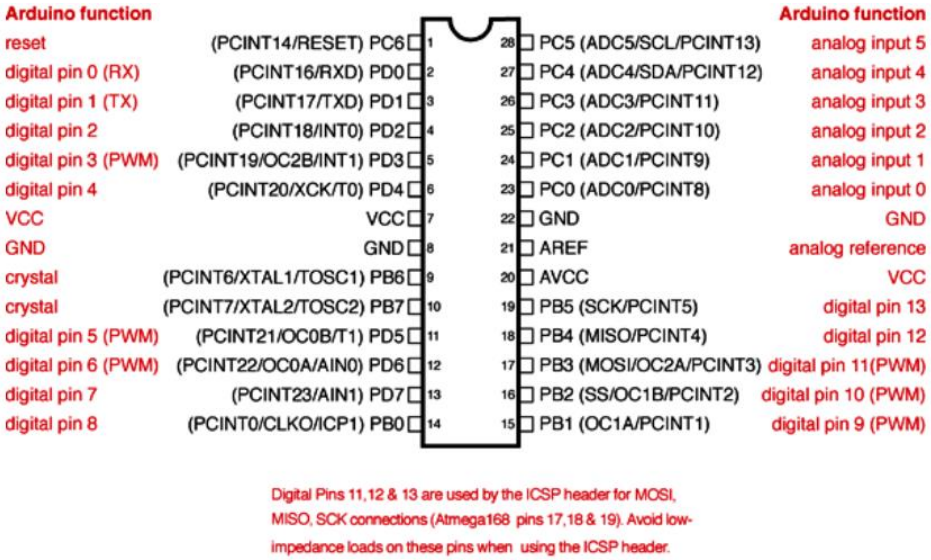
5.1.1 ArduinoUno

Το ArduinoUnoRev3 ή Uno (Εικόνα 5-3) είναι η πιο διαδεδομένη πλακέτα Arduino, που χρησιμοποιείται ευρέως σε πληθώρα εφαρμογών, από τον έλεγχο σε quadcopters έως τον έλεγχο συσκευής ποτίσματος.



Εικόνα 5-3 Πλακέτα ArduinoUnoRev3 [24]

Τα βασικά του χαρακτηριστικά είναι: ο επεξεργαστής που διαθέτει, ο *ATmega328P*, τα 14 ψηφιακά I/O pins, όπου τα 6 μπορούν να γίνουν έξοδοι PWM και τα 6 αναλογικά I/O pins. Στην Εικόνα 5-4 παρουσιάζεται η αντιστοίχιση των ακίδων του επεξεργαστή (pinmapping - ηλεκτρονικό σχέδιο) *ATmega328P*, ο οποίος είναι βασισμένος στην αρχιτεκτονική AVR, που παρέχει 3 είδη μνήμης Flash Memory, ARAM και EEPROM).



Εικόνα 5-4 Pinmaping του Επεξεργαστή ATmega328 του ArduinoUno[24]

Στον Πίνακα 5-1 παρατίθενται τα κύρια χαρακτηριστικά της πλακέτας Arduino-UnoRev3:

Τεχνικά Χαρακτηριστικά του ArduinoUno	
Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

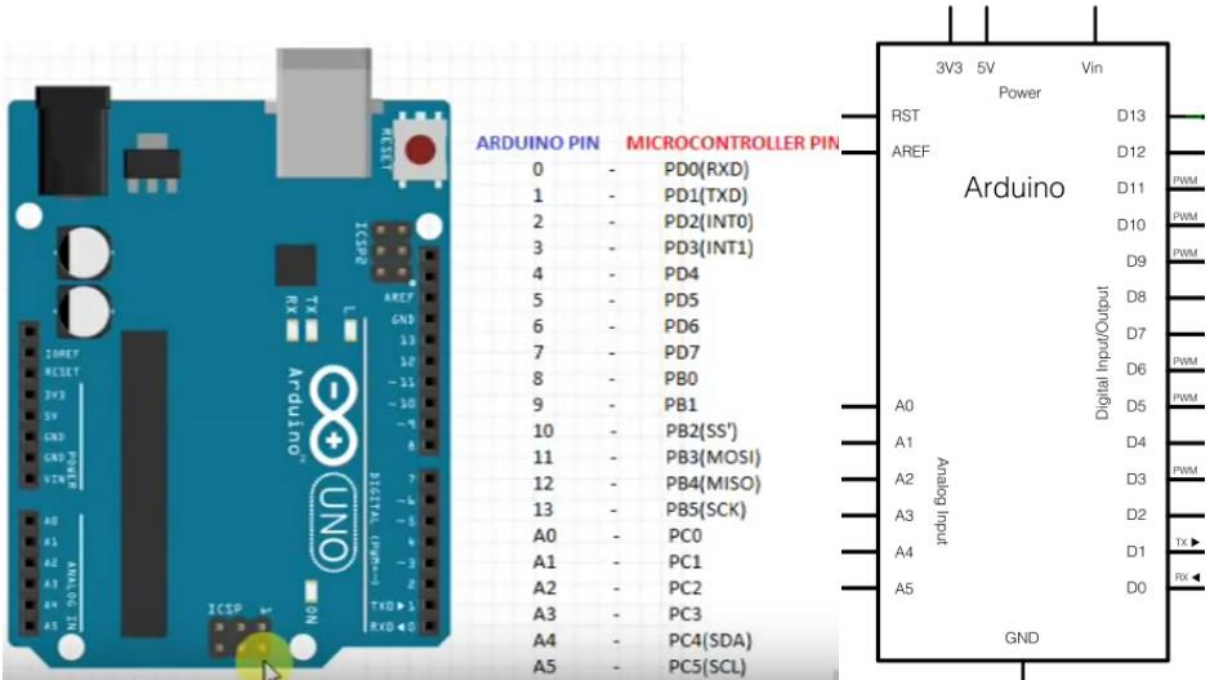
Πίνακας 5-1 Τεχνικά Χαρακτηριστικά του ArduinoUno [24]

Θύρες Εισόδου – Εξόδου (I/O pins)

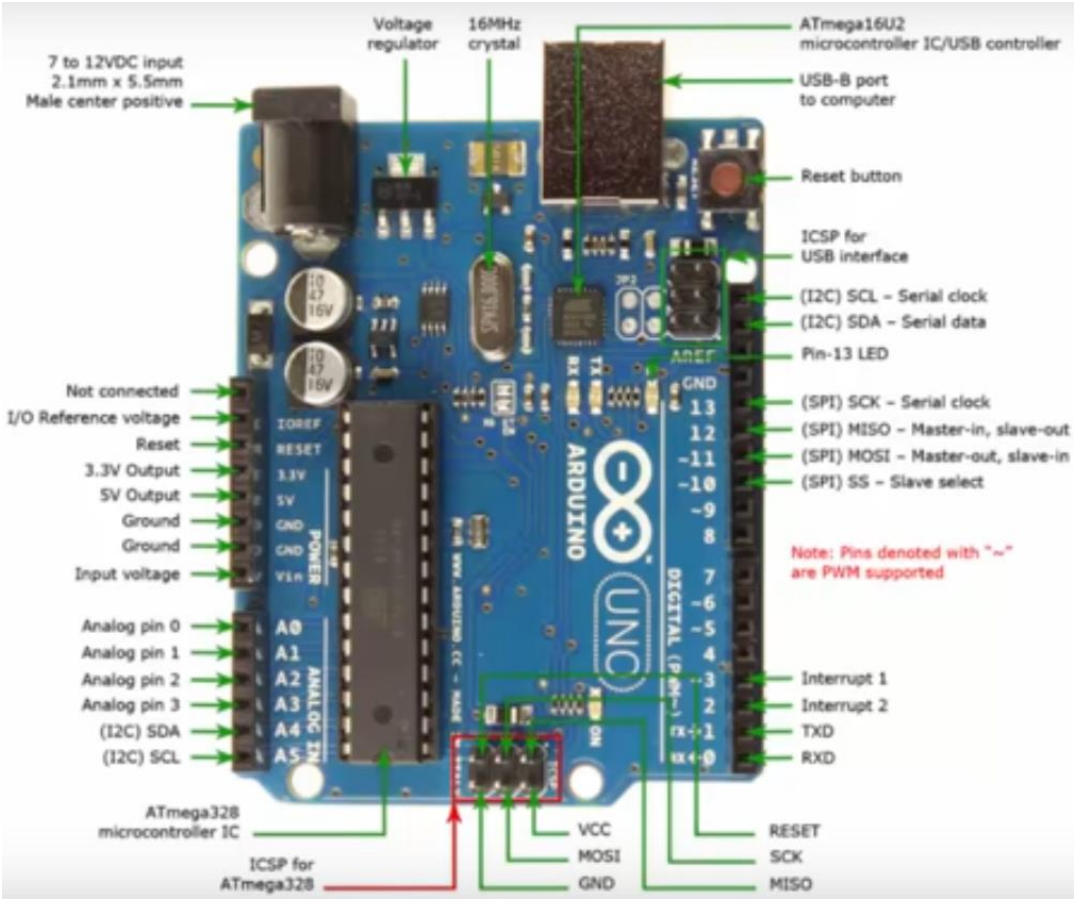
Οι 14 ψηφιακές θύρες ονομάζονται με νούμερα από το 0 έως το 13 (D I/O), ενώ οι 6 αναλογικές με το γράμμα Α ακολουθούμενο από ένα νούμερο από 0 μέχρι το 5 (A I/O). Όλες οι θύρες (pins) είναι θηλυκά. Οι θύρες (pins) μπορούν να πάρουν και να δώσουν τάση από 0V έως 5V και να παρέχουν ή να δεχτούν το πολύ 20mA ένταση ρεύματος. Από τις 14 ψηφιακές θύρες οι 6 (3, 5, 6, 9, 10, 11) είναι και PWM θύρες (Pulse Width Modulation), δηλαδή μπορούν να προσομοιώσουν αναλογικές εξόδους. Έτσι, συνοπτικά έχουμε:

- Για **ψηφιακή είσοδο**, χρησιμοποιούμε τις 14 ψηφιακές 0..13. Όταν δουλεύουν ψηφιακά, η είσοδος μπορεί να είναι 0 ή 5V, με τον χαρακτηρισμό LOW ή HIGH αντίστοιχα.
- Για **ψηφιακή έξοδο**, χρησιμοποιούμε τις 14 ψηφιακές 0..13. Όταν δουλεύουν ψηφιακά, η έξοδος μπορεί να είναι 0 ή 5V, με τον χαρακτηρισμό LOW ή HIGH αντίστοιχα.
- Για **αναλογική είσοδο**, δηλαδή να διαβάσουμε τιμές τάσης ρεύματος στο διάστημα 0 έως 5V, χρησιμοποιούμε τις έξι αναλογικές θύρες A0 έως A5.
- Για **αναλογική έξοδο**, μπορούμε να χρησιμοποιήσουμε τις έξι PWM ψηφιακές θύρες (3, 5, 6, 9, 10, 11), οι οποίες θα μας δώσουν ρεύμα εξόδου όποιας τιμή θέλουμε στο διάστημα από 0 έως 5V.

Γράφοντας κάποιον κώδικα θα πρέπει να αρχικοποιήσουμε τις θύρες που χρησιμοποιούμε με τη συνάρτηση: *pinMode()*, δηλαδή δίνουμε την πληροφορία αν θα είναι για είσοδο ή για έξοδο. Επίσης, στη θύρα 13 υπάρχει συνδεδεμένο ήδη ένα Led πάνω στην πλακέτα Arduino Uno, κι έτσι μπορούμε να το χρησιμοποιούμε για σχετικές λειτουργίες. Στις Εικόνες 5-5 (περιληπτικά) και 5-6 (αναλυτικά) παρουσιάζεται η ερμηνεία κάθε εξωτερικής θύρας του ArduinoUno.



Εικόνα 5-5 Περιληπτική Αντιστοίχιση Θυρών του ArduinoUno



Εικόνα 5-6 Αναλυτική Περιγραφή των Θυρών του ArduinoUno

Επιπλέον, όταν χρησιμοποιείται η σειριακή οθόνη παρακολούθησης της επικοινωνίας (Serial Monitoring) με τον υπολογιστή, τα pins 0 και 1 λειτουργούν ως RX (Receiver) και TX (Transmitter) της σειριακής, διότι το πρόγραμμά ενεργοποιεί την σειριακή θύρα. Έτσι, όταν λόγου χάρη το πρόγραμμά στέλνει δεδομένα στην σειριακή (pin 1), αυτά προωθούνται και στη θύρα USB μέσω του ελεγκτή Serial-Over-

USB, για να τα διαβάσει ενδεχομένως μια άλλη συσκευή, ενώ στις απαντήσεις του Η/Υ προς το Arduino μέσω USB αυτά στέλνονται και στο pin 0. Επομένως, δεν συνίσταται να χρησιμοποιούνται για άλλο σκοπό, αν έχουμε ανοικτή τη σειριακή παρακολούθηση. Ακόμη, υπάρχουν και 2 μικρά leds πάνω στο Arduino με την ίδια ονομασία (RX και TX), τα οποία ανάβουν αναλόγως αν έχουμε σειριακή λήψη ή μετάδοση δεδομένων. Τέλος, το πρωτόκολλο επικοινωνίας που χρησιμοποιείται για την μεταφορά των δεδομένων από μια συσκευή στην άλλη είναι το πρωτόκολλο ασύγχρονης επικοινωνίας UART (Universal Asynchronous Transmitter/Receiver).

Όπως αναφέρθηκε τα pins 3, 5, 6, 9, 10 και 11 μπορούν να λειτουργήσουν και ως ψευδο-αναλογικές έξοδοι με το σύστημα PWM (Pulse Width Modulation - Διαμόρφωση Εύρους Παλμών). Ως PWM ορίζεται η τεχνική η οποία δίνει αναλογικά αποτελέσματα με την χρήση ψηφιακών μέσων. Μέσω ψηφιακού ελέγχου δημιουργείται ένα τετραγωνικό κύμα, δηλαδή ένα σήμα που εναλλάσσεται μεταξύ on και off (ή αλλιώς μεταξύ 5V και 0V). Η διάρκεια του «on time» είναι το pulse width, δηλαδή το εύρος του παλμού (ή αλλιώς duty cycle).[21]

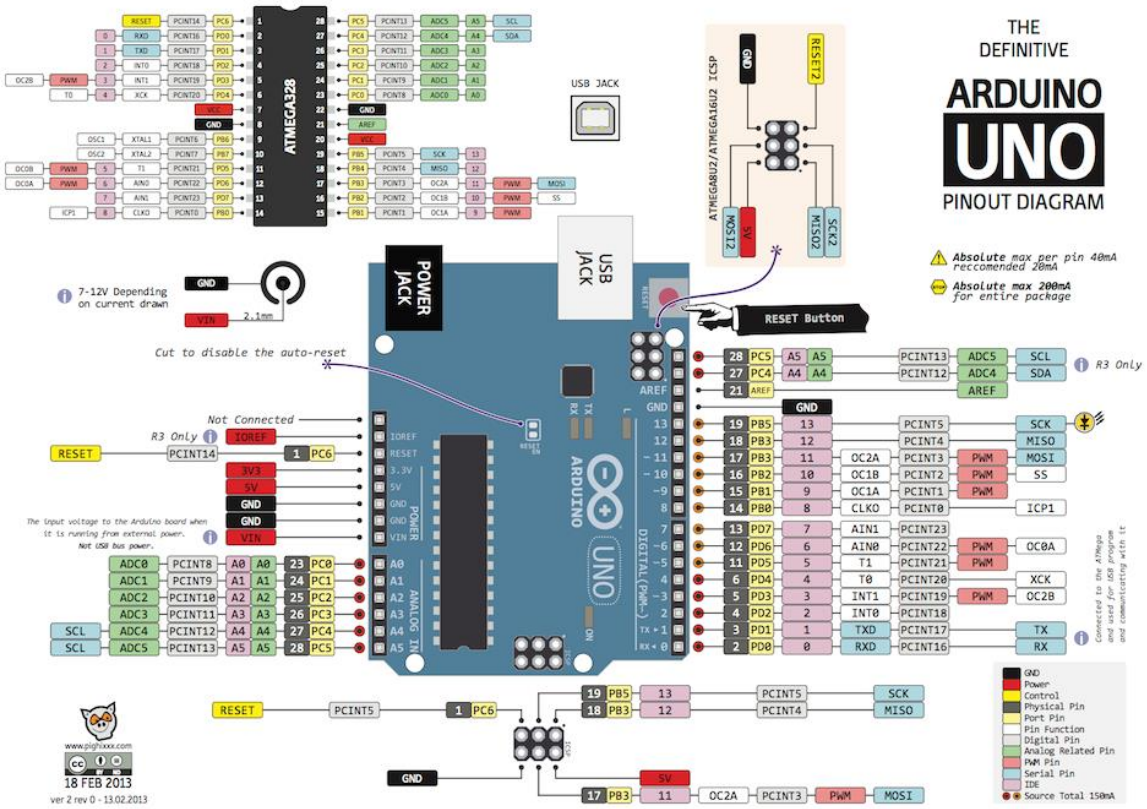
Τα 6 Analog pins λειτουργούν ως αναλογική είσοδος κάνοντας χρήση του ADC (Analog to Digital Converter) που είναι ενσωματωμένο στον μικροελεγκτή. Επομένως, είναι δυνατό να τροφοδοτηθεί ένα από τα pin αυτά με μία τάση, που μπορεί να κυμαίνεται από 0V ως μια τάση αναφοράς Vref, η οποία αν δεν προηγηθεί κάποια αλλαγή είναι προ-ρυθμισμένη στα 5V. Τότε, μέσα από το πρόγραμμά λαμβάνεται η τιμή του pin ως ένα ακέραιο αριθμό ανάλυσης 10-bit, από 0 (όταν η τάση στο pin είναι 0V) μέχρι 1023 (όταν η τάση στο pin είναι 5V). Η τάση αναφοράς μπορεί να ρυθμιστεί με μια εντολή στο 1.1V, ή σε όποια άλλη τάση απαιτείται (μεταξύ 2 και 5V) τροφοδοτώντας εξωτερικά με αυτή την τάση το pin με την σήμανση AREF που βρίσκεται στην απέναντι πλευρά της πλακέτας από τις αναλογικές θύρες (Εικόνα 5-6). Οι αναλογικές θύρες επίσης, μπορούν να μετατραπούν σε ψηφιακές, εφόσον οι ψηφιακές δεν επαρκούν, αν τις ορίσουμε ως digital αντί για analog, μέσω μιας σειράς συναρτήσεων όπως η digitalWrite ,[24] π.χ.

pinMode(A0, OUTPUT);

digitalWrite(A0, HIGH);

Μέσω των αναλογικών θυρών A4 (SDA) και A5 (SCL) ή των θυρών SDA-SCL (πάνω από το AREF), μπορεί να γίνει σύνδεση του Arduino, μέσω του διαύλου επικοινωνίας I2C (Inter Integrated Communication) με περιφερειακές συσκευές ή άλλες πλακέτες όπως Arduino και RaspberryPi σε σχέση master-slave ή πιο γενικά multimaster-slaves, δηλαδή στο δίκτυο να υπάρχουν περισσότεροι από ένας master. Πέραν του I2C, μπορούν να γίνουν και άλλες συνδέσεις του Arduino, όπως η SPI (Sychronous Data Transmission Protocol), όπου γίνεται σύνδεση 4 καλωδίων χρησιμοποιώντας τις ψηφιακές θύρες 10 (SS), 11 (MOSI), 12 (MISO), 13 (SC), ή TWI (ίδια με την I2C).

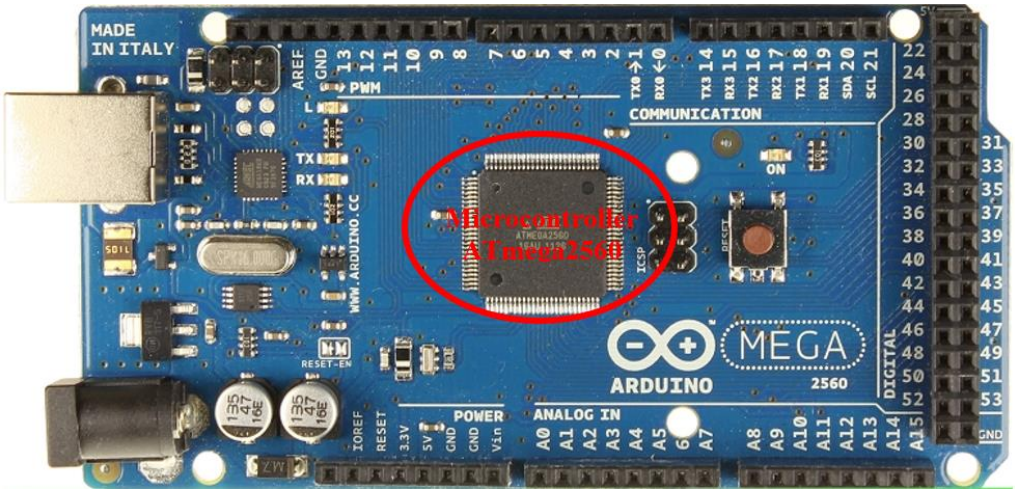
Τέλος, οι ψηφιακές θύρες 2 και 3 μπορούν να λειτουργήσουν ως interrupts (διακοπές). Αυτές οι θύρες χρησιμοποιούν την συνάρτηση *attachInterrupt()* και μπορούν να διαμορφωθούν ώστε να διακόπτουν το πρόγραμμα, όταν διαπιστώσουν μια τιμή που θα έχουμε ορίσει. Στην εικόνα 5-7 περιγράφονται αναλυτικά όλες οι δυνατότητες που έχουν οι θύρες του ArduinoUno.



Εικόνα 5-7 PinMapping με όλες τις λειτουργίες των θυρών του ArduinoUno

5.1.2 ArduinoMega2560

Οι περισσότερες λειτουργίες του ArduinoMega (Εικόνα 5-8) είναι ίδιες με αυτές της πλακέτας ArduinoUno, όπως οι τάσεις τροφοδοσίας και εξόδου, οι εντάσεις ρεύματος, η ταχύτητα του ρολογιού (clockspeed) και το είδος των θυρών που διαθέτει. Οι κύριες διαφορές τους εντοπίζονται στον microcontroller (ATmega2560), στις μνήμες RAM (8 KB), Flash Memory (256 KB) και EEPROM (4 KB) και στον αριθμό των θυρών I/O, αφού διαθέτει 54 ψηφιακές και 16 αναλογικές, καθώς και επιπλέον θέσεις για παροχή τάσης εξόδου 5 V και γειώσεις (GND). Ακόμη, περιέχει 4 σειριακές θύρες UART (0-1,14-15,16-17,18-19) και έναν κρυσταλλικό ταλαντωτή 16 MHz [22]



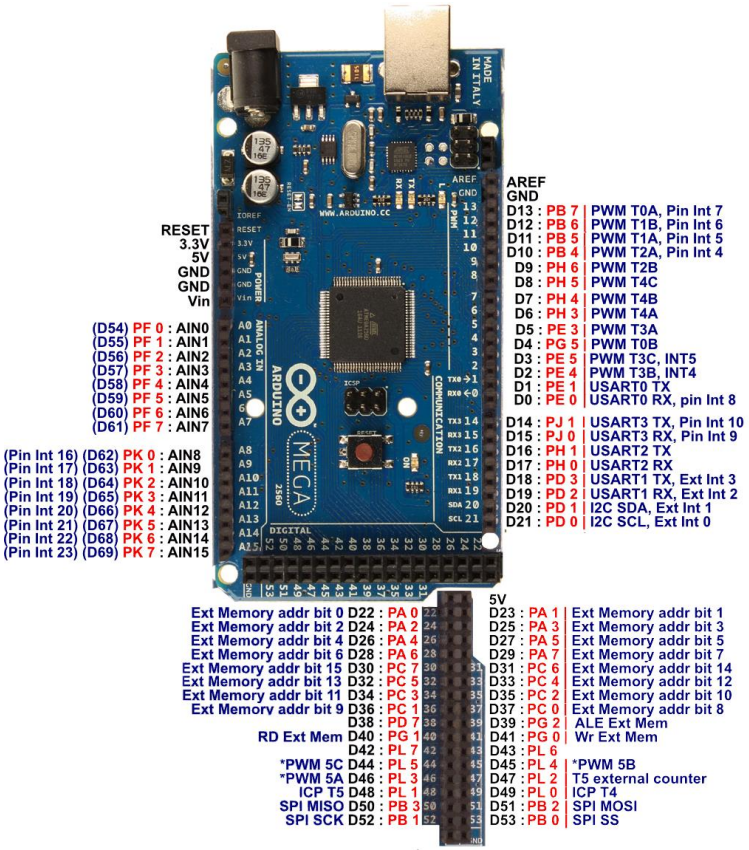
Εικόνα 5-8 Πλακέτα Arduino Mega2560Rev3

Τα χαρακτηριστικά της πλακέτας ArduinoMega2560 πλεονεκτούν έναντι αυτών της Uno, σε εφαρμογές που χρειαζόμαστε να ελέγξουμε πολλές περιφερειακές συσκευές ταυτόχρονα, καθότι λόγω της πληθώρας θυρών μπορούμε να τις συνδέσουμε απευθείας στην πλακέτα, δίχως την προσθήκη κάποιας shield για επαύξηση θυρών. Επίσης, μπορούμε να γράψουμε κώδικα χρησιμοποιώντας πολλές μεταβλητές (καθολικές ή τοπικές) και συναρτήσεις, που με το Uno δεν θα ήταν εφικτό, λόγω περιορισμένης χωρητικότητας στη μνήμη του, σε αντίθεση με το Mega2560 που διαθέτει ικανότητα αποθήκευσης του 3πλάσιου αριθμού τοπικών μεταβλητών, ένεκα των επαυξημένων δυνατοτήτων των μνημών του και κυρίως του επεξεργαστή του *ATmega2560* έναντι του *ATmega328* του Uno. Στον παρακάτω Πίνακα 5-2, παραθέτουμε τα χαρακτηριστικά του ArduinoMega2560Rev3:[24]

Τεχνικά Χαρακτηριστικά ArduinoMega2560	
Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
PWM Digital I/O Pins	15
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Πίνακας 5-2 Τεχνικά Χαρακτηριστικά ArduinoMega2560 [24]

Οι θύρες του ArduinoMega έχουν όλα τα χαρακτηριστικά που αναφέρθηκαν και στο Uno (I2C-TWI, SPI, interrupts, PWM, AREF). Επιπλέον, το Mega διαθέτει έναν πολυδιακόπτη επαναφοράς (ressetable polyfuse) για προστασία από υπερεντάσεις, μέσω της σειριακής θύρας, αν αυτή ξεπεράσει τα 500mA. Στην Εικόνα 5-9 παρατίθεται η αναλυτική περιγραφή των θυρών και οι λειτουργίες που επιτελούν.



Εικόνα 5-9 Αναλυτική Περιγραφή των Θυρών του ArduinoMega2560

5.2 *Προγραμματισμός ArduinoIDE*

Δομή

Το Arduino προγραμματίζεται στο περιβάλλον ArduinoIDE (Εικόνα 8-1) και η γλώσσα κάθε προγράμματος αποτελείται από δύο βασικές ρουτίνες ώστε να έχει την παρακάτω γενική δομή:

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Η αρχική ρουτίνα *setup()* εκτελείται μια φορά μόνο κατά την εκκίνηση του προγράμματος, ενώ η βασική ρουτίνα *loop()* περιέχει τον βασικό κορμό του προγράμματος και η εκτέλεσή της επαναλαμβάνεται συνέχεια σαν ένας βρόχος while (true). [21]

Για να φορτωθεί ο κώδικας από τον Η/Υ στο Arduino κάνουμε compile και μετά load. Όταν ο κώδικας έχει φορτωθεί στο Arduino, τρέχουν μια φορά οι εντολές στο setup() και μετά τρέχουν συνέχεια οι εντολές του βρόχου loop(). Ο κώδικας παραμένει φορτωμένος ακόμα και αν αποσυνδέσουμε τη USB θύρα και δώσουμε

εξωτερική τροφοδοσία. Για να μηδενίσει το Arduino πρέπει να πατήσουμε το κουμπί *Reset* ή για να αλλάξει πρέπει να κάνουμε load άλλον κώδικα.

- *void setup()*: Μπαίνουν όλες οι εντολές που πρέπει να τρέξουν μία φορά, όταν ενεργοποιείται το Arduino. Συνήθως μπαίνουν αρχικοποιήσεις τιμών των μεταβλητών και οπωσδήποτε ο χαρακτηρισμός των εισόδων/εξόδων, μέσω της εντολής pinMode(), που θα χρησιμοποιηθούν (pins I/O).
- *void loop()*: Μπαίνει το βασικό πρόγραμμά μας. Οι εντολές θα τρέξουν αρχικά και θα ενεργοποιηθεί η loop(), συνεχίζοντας ξανά από την αρχή της. Αυτό συμβαίνει συνεχώς, μέχρι να διακόψουμε την παροχή ρεύματος στο Arduino ή να πατηθεί το πλήκτρο reset.

Στην περίπτωση του *Reset* ξανατρέχει η συνάρτηση setup() μία φορά και ακολούθως η loop() συνεχίζει τον βρόχο, γι' αυτό χρησιμοποιείται σε περίπτωση που έχουν γίνει αλλαγές στο πρόγραμμά.

Μεταβλητές

Γενικά στις γλώσσες προγραμματισμού, ως μεταβλητή (val), ορίζεται ένα γλωσσικό αντικείμενο που μπορεί να λάβει διάφορες τιμές, μία κάθε φορά. Στο ArduinoIDE, καθότι είναι βασισμένο στην C/C++, πρέπει πριν το void loop(), να δηλωθούν και να αρχικοποιηθούν οι μεταβλητές που θα χρησιμοποιηθούν. Ο τύπος δεδομένων μιας μεταβλητής μπορεί να είναι:

- *byte*: αποθηκεύει μια αριθμητική τιμή 8-bit χωρίς δεκαδικά ψηφία, παίρνουν τιμές από 0 μέχρι 255.
- *int*: ακέραιοι, παίρνουν τιμές από -32,768 μέχρι 32,767.
- *lo*ng: μεγάλου μεγέθους ακέραιοι, παίρνουν τιμές από -2,147,483,648 μέχρι 2,147,483,647.
- *float*: πραγματικοί αριθμοί, παίρνουν τιμές από 3.4x10^-38 μέχρι 3.4x10^38.
- *char*: ένας χαρακτήρας (μέγεθος 1 byte).
- *string*: πίνακας χαρακτήρων.
- *boolean*: λογικές μεταβλητές με τιμές 0 και 1 (True ή False).

Βασικές Συναρτήσεις

- Ψηφιακές θύρες:

  pinMode(pin,mode) : Ορίζει τη λειτουργία του pin (είσοδος/έξοδος)

  digitalWrite(pin,val): Γράφει την τιμή εξόδου του pin (LOW/HIGH)

  val = digitalRead(pin): Διαβάζει την τιμή του pin

- Αναλογικές θύρες:

  analogWrite(pin,val): (PWM) Γράφει την τιμή % duty cycle του pin (0-255).

  val = analogRead(pin): Διαβάζει την τιμή του pin (0-1023).

- Χρονιστές:

  time = millis(): Επιστρέφει τον χρόνο σε ms από τότε που άρχισε το πρόγραμμα.

  time = micros(): Επιστρέφει τον χρόνο σε μs από τότε που άρχισε το πρόγραμμα.

delay(val): Κάνει παύση το πρόγραμμα για val ms.

delayMicroseconds(val): Κάνει παύση το πρόγραμμα για val μs.

- Σειριακή θύρα/USB (Universal Serial Bus)

Για να ενεργοποιηθεί η σειριακή θύρα επικοινωνίας αρκεί να δοθεί στη setup() η εντολή *Serial.begin(BaudRate)*, όπου το BaudRate εκφράζει το ρυθμό με τον οποίο θα μεταδίδονται τα bits/sec (9600 συνήθως). π.χ. Serial.begin(9600).

Μπορούμε να χρησιμοποιήσουμε τη σειριακή θύρα στις εφαρμογές για αμφίδρομη επικοινωνία, δηλαδή να στείλουμε και να λάβουμε δεδομένα. Μια εντολή που μας βοηθάει σε αυτό είναι η *print()*, που εκτυπώνει ένα μήνυμα ή τιμές και η *println()* που λειτουργεί ακριβώς το ίδιο αλλά εκτυπώνοντας με αλλαγή γραμμής κάθε φορά, εμφανίζοντας την εκτύπωση στη *Σειριακή Οθόνη ή Παράθυρο (Serial Monitoring).*

val = avaliable() : Επιστρέφει τον αριθμό των bytes που είναι διαθέσιμα στην θύρα

begin(baudrate): Ξεκινάει/Ρυθμίζει την σειριακή θύρα. Συνήθως δίνουμε μόνο το baudrate

print(val): Τυπώνει την τιμή val σε ASCII text

println(val): Τυπώνει την τιμή val σε ASCII text και αλλάζει γραμμή ('\r')

val = read(): Διαβάζει ένα byte απο την σειριακή

readBytes(buffer,length): Διαβάζει length bytes απο την σειριακή και τα σώζει στο buffer.

Επιπλέον Δομές

#define: Για ορισμό ενός ονόματος σε μια σταθερή τιμή

#include: Για φόρτωση μιας βιβλιοθήκης στο πρόγραμμα

/ */ (block comment)*: Σχόλια ενδιάμεσα αστερίσκων

// (single line comment): Σχόλια για μια γραμμή στο πρόγραμμα

Βιβλιοθήκες

- Servo

Επιτρέπει τον έλεγχο σερβοκινητήρων από τον Arduino, μέσω σημάτων PWM.

Βασικές συναρτήσεις δομής:

attach(pin): Αντιστοιχίζει ένα αντικείμενο servo σε μία PWM θύρα του Arduino

write(degrees): Στρέφει τον άξονα του σερβοκινητήρα στις degree μοίρες

- Wire

Για Επικοινωνία με I2C (Wire, I2Cdev) η οποία περιλαμβάνεται στην βασική έκδοση του Arduino IDE, ενσωματώνει τις βασικές λειτουργίες επικοινωνίας I2C με περιφερειακές συσκευές. Θα αναλυθεί περαιτέρω σε επόμενο κεφάλαιο.

Οι υπόλοιπες αρχές προγραμματισμού και δομές ελέγχου όπως if condition, while, for κτλ, είναι όπως στη C/C++.

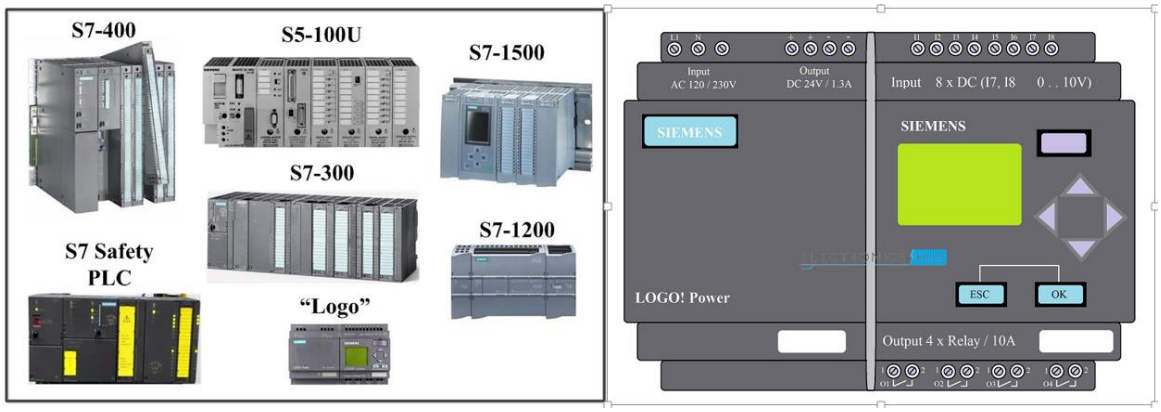
5.3 *Επιλογή Ελέγχου με Arduino*

Για την υλοποίηση ελέγχου στο εξεταζόμενο ΕΣΚ, χρησιμοποιήθηκαν μικρο-ελεγκτές Arduino, οι οποίοι προτιμήθηκαν έναντι άλλων ηλεκτρονικών διατάξεων όπως PLC ή μικρο-επεξεργαστές RaspberryPi, για διαφορετικούς λόγους. Οι βασικότεροι λόγοι προτίμησης των Arduino συνοψίζονται στα παρακάτω:

- Απλότητα στη χρήση του.
- Ευκολία Προγραμματισμού και σύνδεσης με περιφερειακά κυκλώματα και μεταξύ τους.
- Χαμηλό κόστος, ιδιαιτέρως σε σχέση με κάποιο PLC.
- Ευκολότερη πρόσβαση σε επιμέρους εξαρτήματα (shields) και βιβλιοθήκες προγραμμάτων.
- Ποικιλία εφαρμογών σε διαφορετικά συστήματα που απαιτούν έλεγχο.

5.3.1 *Arduino vs PLC*

Οι προγραμματιζόμενοι λογικοί ελεγκτές (Programmable Logic Controllers-PLC) ανήκουν στην ευρύτερη κατηγορία των Ψηφιακών Υπολογιστικών Συστημάτων και χρησιμοποιούνται στον έλεγχο μηχανών κυρίως στη βιομηχανία αλλά και σε κτιριακές εγκαταστάσεις που απαιτούν υψηλή αξιοπιστία ελέγχου και διάγνωση λαθών. Η μορφή των PLC για διάφορα μοντέλα φαίνεται στην Εικόνα 5-10.

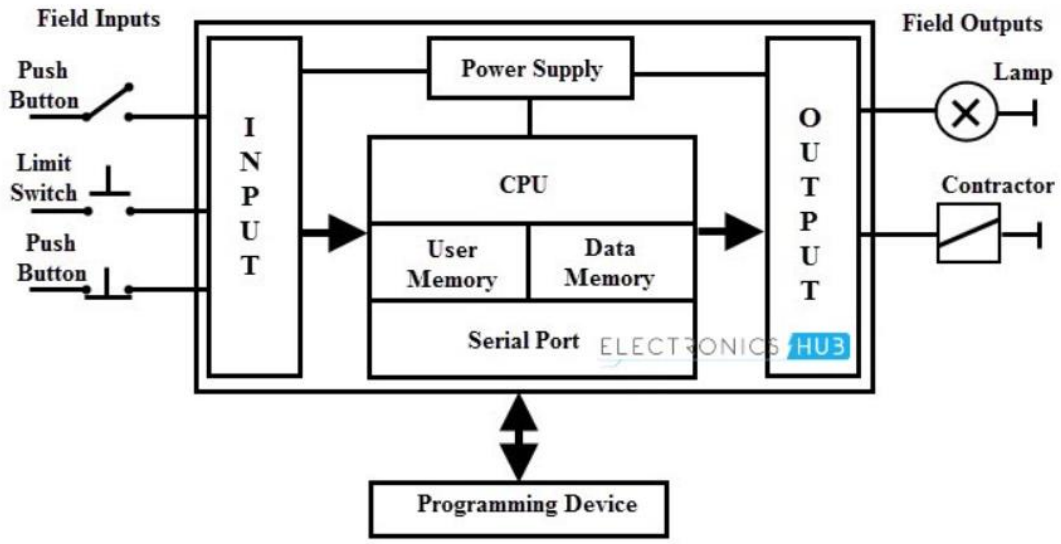


Εικόνα 5-10 Μοντέλα PLC της Siemens [26]

Ένα PLC είναι βασικά ενός «τεράστιος» μικρο-ελεγκτής, ο οποίος διαχειρίζεται και αποθηκεύει πληροφορίες με τη μορφή δύο διαφορετικών καταστάσεων ON και OFF (1 και 0). Οι πληροφορίες αποτελούνται από δυαδικά ψηφία (Bits). Τα Bits σαν στοιχεία πληροφορίας χρησιμοποιούνται είτε αυτόνομα (αναπαριστώντας καταστάσεις ON ή OFF) ή χρησιμοποιούνται σε συνδυασμό με συγκεκριμένη κωδικοποίηση αναπαριστώντας αριθμούς. Ενός, μπορεί να "αντιληφθεί" και να επεξεργασθεί πληροφορίες είτε σε ψηφιακή μορφή (0-1 ή ON-OFF): κλειστή ή ανοιχτή επαφή ή σε αναλογική μορφή: πληροφορία από σύστημα μέτρησης στάθμης, θερμοκρασίας, πίεσης, βάρους. Παρ' όλα αυτά η CPU, που είναι ένα ολοκληρωμένο ψηφιακό κύκλωμα (microchip) "αντιλαμβάνεται" μόνο ψηφιακά δυαδικά σήματα "ON" –

"OFF" ή λογικά 1 και 0. Τα PLC προγραμματίζονται σε γλώσσες προγραμματισμού (συγκεκριμένα συμβολικές γλώσσες ή διαγράμματα), όπου οι γλώσσες και το λογισμικό προγραμματισμού διαφέρουν από εταιρεία σε εταιρεία Οι κύριες γλώσσες που χρησιμοποιούνται για τον προγραμματισμό των PLC είναι η Ladder, η STL (statement list) και η FBD (function block diagram), η κύρια δομή λειτουργίας του φαίνεται στην Εικόνα 5-11, ενώ τα κύρια μέρη του είναι: [8]

- Η κεντρική μονάδα επεξεργασίας CPU
- Οι μονάδες εισόδων/εξόδων (I/O)
- Η μονάδα τροφοδοσίας



Εικόνα 5-11 Δομή Λειτουργίας ενός PLC [26]

Η συνολική εκτίμηση όλων των παραπάνω είναι πως τα PLC αποτελούν αξιόπιστους ελεγκτές, όμως δεν προτιμήθηκαν λόγω του κόστους αρχικής αγοράς τους (3πλάσιο τουλάχιστον) σε σχέση με τους μικρο-ελεγκτές Arduino, αλλά και λόγω της έλλειψης πολλών εισόδων εξόδων για τη σύνδεση τους με αρκετές περιφερειακές συσκευές, όπως έχουμε στην προκειμένη περίπτωση. Κυρίως όμως επειδή η εφαρμογή των PLC σε συστήματα κατεργασιών και γενικά σε βιομηχανικά συστήματα είναι αρκετά διαδεδομένη, ενώ τα Arduino είναι ελεγκτές που μπορούν να εφαρμόσουν παντός τύπου έλεγχο και σε μεγάλο εύρος μηχανημάτων.

5.3.2 Arduino vs RaspberryPi3

Το RaspberryPi λειτουργεί όπως ένας κλασσικός υπολογιστής, καθώς διαθέτει επεξεργαστή, μνήμη, drivers γραφικών και διάφορες εισόδους-εξόδους. Ο χρήστης μπορεί να πλοηγηθεί στο Internet, να παρακολουθήσει βίντεο υψηλής ανάλυσης, να χρησιμοποιήσει κειμενογράφο και υπολογιστικά φύλλα και φυσικά να μάθει γλώσσες προγραμματισμού, όπως η Scratch και η Python. Το Pi συνδέεται σε μια οθόνη υπολογιστή ή μια τηλεόραση και χρησιμοποιεί ένα usb πληκτρολόγιο και ποντίκι, μια SD κάρτα για το λειτουργικό σύστημα και την αποθήκευση δεδομένων, ενώ για τη τροφοδοσία του απαιτείται ένα mini

USB τροφοδοτικό των 5 Volt, διαθέτει θύρα Ethernet, θύρες USB. Χρησιμοποιεί κυρίως λειτουργικά συστήματα που βασίζονται στο Linux, όπως το Raspbian (το δημοφιλέστερο λειτουργικό).[27]

Υπάρχουν διάφορα μοντέλα, με το Pi3 να είναι το τελευταίο (Εικόνα 5-12), όπου στον Πίνακα 5-3 φαίνονται τα κύρια χαρακτηριστικά του:

Χαρακτηριστικά RaspberryPi3
Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
1GB RAM
BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
40-pin extended GPIO
CSI camera port for connecting a Raspberry Pi camera
4 USB 2 ports4 Pole stereo output and composite video port Display
Micro SD port for loading your operating system and storing data
Full size HDMI DSI display port for connecting a Raspberry Pi touchscreen
Upgraded switched Micro USB power source up to 2.5A

Πίνακας 5-3 Χαρακτηριστικά RaspberrPi 3

Εικόνα 5-12 RaspberryPi3 και τα I/O pins του

Η σύγκριση μεταξύ πλεονεκτημάτων - μειονεκτημάτων που παρέχουν οι δύο μικρο-ηλεκτρονικές διατάξεις (Εικόνα 5-13) παρουσιάζεται στον παρακάτω Πίνακα 5-4:

Πλεονεκτήματα	
RaspberryPi	***Arduino***
Εύκολη Σύνδεση στο Διαδίκτυο	Εύκολο στη χρήση
Διαθέτει όλο το λογισμικό Linux	Λογισμικό ανοιχτού κώδικα (IDE), κατάλληλο για real-time εφαρμογές, χωρίς interpreter
Single Board Computer (SPC) δυνατότητες σύνδεσης με Audio-Video-Ethernet-USB-HDMI	Εύκολα Επεκτάσιμο μέσω πρόσθετων περιφερειακών (shields) και βιβλιοθηκών
Προγραμματίζεται με ποικιλία γλωσσών προγραμματισμού	Δεν απαιτεί ιδιαίτερες γνώσεις προγραμματισμού
Μειονεκτήματα	
RaspberryPi	***Arduino***
Πρόσβαση μέσω hardware με άλλα συστήματα δεν είναι real-time	Δεν είναι τόσο ισχυρό σαν το Pi, ως προς το software.
Έλλειψη αρκετής ισχύος για να μεταδώσει επαγωγικά φορτία	Προγραμματίζεται με Arduino και C/C++

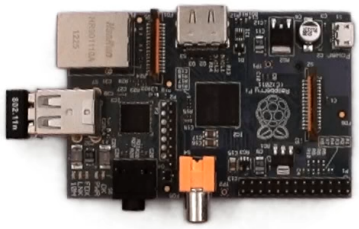
Έλλειψη ενσωματωμένου ADC Το hardware δεν είναι open source	Σύνδεση με το Internet δυσχερής

Πίνακας 5-4 Πλεονεκτήματα - Μειονεκτήματα RaspberryPi - Arduino[28]

Συνοψίζοντας, μπορούμε να πούμε πως αν θέλουμε να συνδέσουμε πολλές συσκευές να επικοινωνούν απλά μεταξύ τους, με χαμηλές απαιτήσεις και σε πραγματικό χρόνο, τότε το Arduino πιθανότατα είναι η καλύτερη επιλογή. Αντίθετα, αν χρειαζόμαστε να «τρέξουμε» πολύπλοκο λογισμικό, απαιτείται να εκτελούνται πολλές λειτουργίες μαζί με σύνδεση στο διαδίκτυο και διαχείριση δεδομένων, τότε ιδανικότερη λύση θα ήταν το RasberryPi.

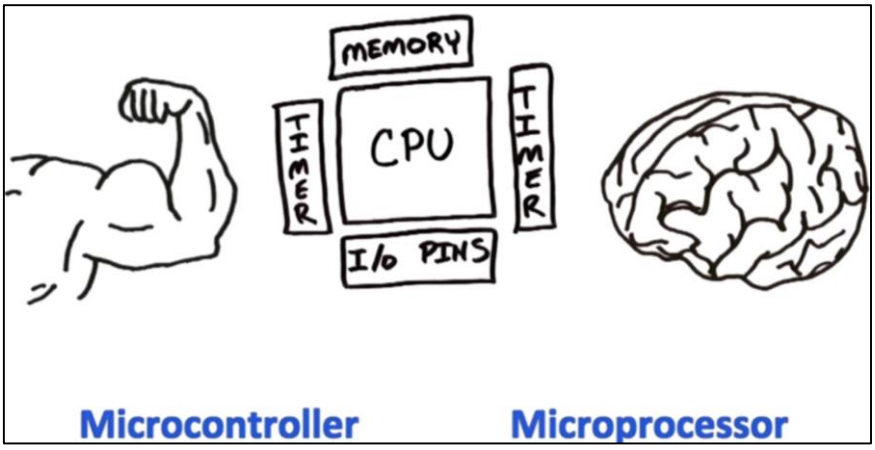
Γενικά, η διαφορά τους έγκειται στο γεγονός πως το RaspberryPi είναι μικρο-επεξεργαστής (micro-processor), ενώ το Arduino μικρο-ελεγκτής (micro-controller). Αυτό γίνεται αντιληπτό και κοιτώντας τα χαρακτηριστικά, όπως αυτά παρατέθηκαν, των δύο μικρο-ηλεκτρονικών διατάξεων, διαπιστώνοντας πως το Arduino αποτελεί μια πιο απλή κατασκευή, τόσο στο hardware, όσο και στο software. Ενώ, το RaspberryPi έχει περισσότερα κοινά με έναν Η/Υ παρά με το Arduino (Εικόνα 5-13).

	Arduino Uno	Raspberry Pi
Processor	AVR ATmega328p	Broadcom ARM1176JZF-S
Clock Speed	16MHz	700MHz
Register Width	8-bit	32-bit
RAM	2k	512 MB
GPIO	20	8
I/O Current Max	40mA	5-10 mA
Power	175 mW	700 mW
Operating System	None	Linux & Others



Εικόνα 5-13 Σύγκριση Χαρακτηριστικών RaspberryPi και Arduino-Uno [29]

Η κύρια διαφορά τους εντοπίζεται στις εισόδους/εξόδους (I/O pins). Οι μικρο-ελεγκτές μπορούν να ελέγχουν απευθείας πολλές περιφερειακές συσκευές, έναντι των μικρο-επεξεργαστών που μπορούν να επεξεργαστούν πολλά δεδομένα – πληροφορία (big data). Οπότε, στην άτυπη σύγκριση RaspberryPi και Arduino δεν υπάρχει καλύτερο και χειρότερο γενικά, αλλά το καθένα είναι ιδανικότερο αναλόγως της εργασίας που θέλουμε να κάνουμε (Εικόνα 5-14).



Εικόνα 5-14 Micro-controller vs Micro-processor [29]

Όλα τα παραπάνω συνηγορούν γιατί στην εφαρμογή της παρούσας εργασίας επιλέχθηκε το Arduino. Καθώς, μπορεί να συνδέσει και να ελέγξει πολλές περιφερειακές συσκευές μαζί, αφού ο έλεγχος που υλοποιήθηκε ήταν απλή εναλλαγή σημάτων εξόδου-εισόδου με High-Low του 1 byte.

5.4 Σειριακή Επικοινωνία – I2C

5.4.1 Τρόποι Επικοινωνίας και Μεταφοράς Δεδομένων

Υπάρχουν δύο τρόποι μεταφοράς δεδομένων η σειριακή και η παράλληλη. Η σειριακή μεταδίδεται bit by bit, ενώ η παράλληλη με blocks δεδομένων. Η ταχύτητα στην παράλληλη επικοινωνία είναι κατά πολύ μεγαλύτερη από τη σειριακή, γι' αυτό χρησιμοποιείται στους Η/Υ, αλλά για συνδέσεις που έχουν μεγάλες αποστάσεις χρησιμοποιείται η σειριακή επειδή η παράλληλη είναι μη βολική.

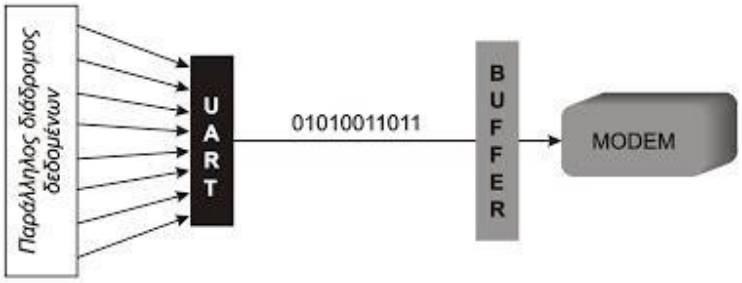
Άλλος διαχωρισμός μετάδοσης της πληροφορίας είναι σε σύγχρονη και ασύγχρονη. Η σύγχρονη πέραν του αποστολέα και του παραλήπτη διαθέτει και ένα ρολόι ή μαζί με τον αποστολέα στέλνεται και ένα χρονικό σήμα, ώστε να ειδοποιήσει τον παραλήπτη. Ενώ, όταν δεν υπάρχει μετάδοση δεδομένων, στέλνεται ένας χαρακτήρας για να εξακριβωθεί η συνέχεια της επικοινωνίας. Αντίθετα, στην ασύγχρονη επικοινωνία δεν υπάρχει ρολόι και ο αποστολέας με τον παραλήπτη συμφωνούν στην ταχύτητα μετάδοσης. Για τον συγχρονισμό χρησιμοποιούν ειδικά bits μαζί με κάθε λέξη.

Η σειριακή επικοινωνία προτιμάται όταν δε θέλουμε να έχουμε πολλές συνδέσεις (καλώδια). Σε αυτήν τα bits της πληροφορίας μεταδίδονται ένα κάθε φορά, στη σειρά, μέσα από έναν αγωγό μεταφοράς δεδομένων. Στην απλούστερη περίπτωση τέτοιας επικοινωνίας χρειαζόμαστε τρεις συνολικά αγωγούς, έναν για την αποστολή δεδομένων, έναν για τη λήψη και έναν που θα βρίσκεται στο δυναμικό αναφοράς των μεταδιδόμενων σημάτων.

UART

Το UART (Universal Asynchronous Receiver/Transmitter) είναι ένα κύκλωμα υπολογιστών το οποίο διαμεσολαβεί στη σειριακή επικοινωνία μεταξύ υπολογιστών ή υπολογιστών και συσκευών ή/και

Ενσωματωμένων υπολογιστικών συστημάτων (Embedded Computer Systems όπως μικροελεγκτές). Η επικοινωνία των UART γίνεται δια μέσου των προτύπων (standards) θυρών RS-232, RS-422 ή RS-485 . Η ταχύτητα επικοινωνίας (ρυθμός μετάδοσης) μπορεί να παραμετροποιηθεί και μετριέται σε baud rate (bit/sec).



Εικόνα 5-15 Σύνδεση μέσω UART [27]

Η UART παίρνει byte δεδομένων και μεταδίδει τα μεμονωμένα bit με διαδοχικό τρόπο. Ουσιαστικά, η συσκευή αλλάζει τα παράλληλα δεδομένα από τον επεξεργαστή σε σειριακά, τα οποία μπορούν να σταλούν μέσω μιας γραμμής. Μια δεύτερη UART (ίσως σε έναν άλλο επεξεργαστή) μπορεί να χρησιμοποιηθεί για να λάβει τη –σειριακή πάντα- πληροφορία. Το UART εκτελεί όλες τις εργασίες, timing, parity check κ.λπ. που απαιτούνται για την επικοινωνία. Η σειριακή μετάδοση ψηφιακής πληροφορίας (bit by bit) μέσω ενός απλού αγωγού είναι πολύ πιο αποδοτική από ό, τι η παράλληλη μετάδοση μέσω πολλαπλών καλωδίων. Τα UART's μεταδίδουν/λαμβάνουν ένα bit τη φορά σε ένα συγκεκριμένο ρυθμό μετάδοσης δεδομένων (π.χ. 9600bps, 115,200bps, κλπ). Αυτή η μέθοδος της σειριακής επικοινωνίας μερικές φορές αναφέρεται ως σειριακή επικοινωνία TTL. [27]

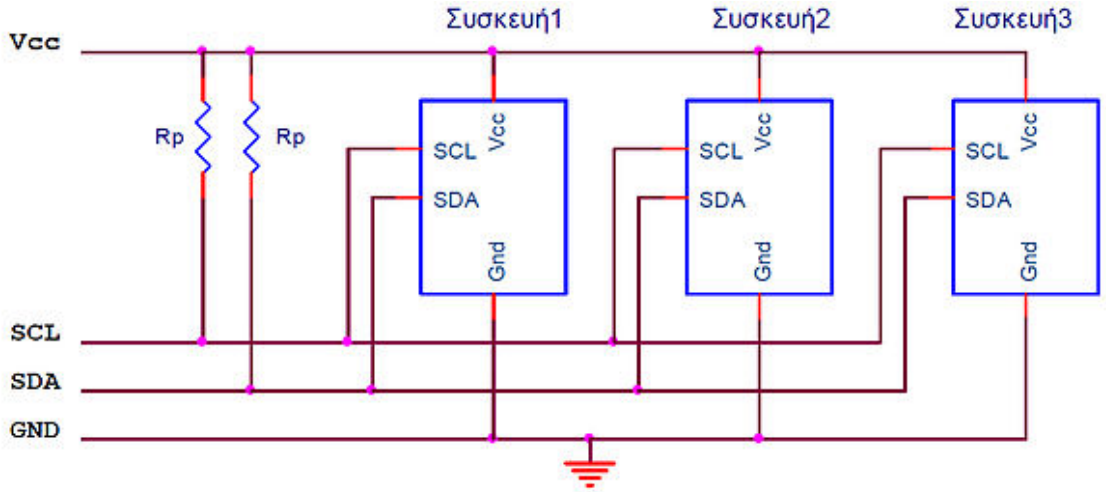
5.4.2 Inter-Integrated Circuit (I2C)

Είναι ένας σειριακός δίαυλος της Philips, που αναφέρεται ως μια διεπαφή δύο καλωδίων (2-Wire Serial Bus). Ουσιαστικά, είναι ένας *multi-master, σειριακός, single-ended* δίαυλος που χρησιμοποιείται για τη σύνδεση περιφερειακών συσκευών με χαμηλή ταχύτητα σε μια μητρική πλακέτα, σε ένα embedded σύστημα, ένα κινητό τηλέφωνο ή άλλη ηλεκτρονική συσκευή.

Το I2C μπορεί να χρησιμοποιηθεί για τη σύνδεση έως και *127 κόμβων μέσω ενός διαύλου με δύο καλώδια δεδομένων*, τα οποία είναι τα SCL και SDA. Η SCL είναι η γραμμή του ρολογιού και χρησιμοποιείται για να συγχρονιστούν όλες οι μεταφορές δεδομένων μέσω του διαύλου I2C. Η SDA είναι η γραμμή δεδομένων και υπάρχει και ένας τρίτος αγωγός γείωσης (GND). Ενδέχεται να υπάρχει ένας αγωγός 5 volts για τη διανομή τροφοδοσίας στις συσκευές.

Και οι δύο SCL και SDA γραμμές είναι ανοικτού επαγωγού (*open drain) drivers*. Αυτό σημαίνει ότι το τσιπ μπορεί να οδηγήσει την έξοδο σε low, αλλά όχι σε high θέση. Για είναι η γραμμή σε θέση high θα πρέπει να υπάρχουν pull-up αντιστάσεις στην τροφοδοσία 5 V. Θα πρέπει να υπάρχει μία αντίσταση μεταξύ της γραμμής SCL και της γραμμής 5V και άλλη μια μεταξύ της γραμμής SDA και της γραμμής 5V. Η τιμή

των αντιστάσεων είναι από 1800 έως 47ΚΩ. Σημειώνεται ότι χρειάζεται μόνο ένα σετ pull-up αντιστάσεων για το σύνολο του I2C διαύλου, και όχι για κάθε συσκευή ξεχωριστά (Εικόνα 5-16).[27]



Εικόνα 5-16 Σύνδεση Συσκευών μέσω I2C

Θεωρητικά, το I2C μπορεί να υποστηρίξει πολλαπλά master, συνήθως υπάρχει ένας master μικρο-ελεγκτής, ενώ οι slaves μπορεί να είναι ολοκληρωμένα κυκλώματα ή άλλοι μικρο-ελεγκτές. Όταν το master επιθυμεί να επικοινωνήσει με ένα slave στέλνει μια σειρά παλμών στις γραμμές SDA και SCL. Τα δεδομένα που αποστέλλονται περιλαμβάνουν μία διεύθυνση που προσδιορίζει το slave με το οποίο το master χρειάζεται να αλληλεπιδράσει. Οι διευθύνσεις πιάνουν 7 bit από ένα byte δεδομένων, το υπολειπόμενο bit καθορίζει εάν το master επιθυμεί να διαβάσει (πάρει δεδομένα από ένα slave) ή να γράψει (στείλει τα δεδομένα σε ένα slave).

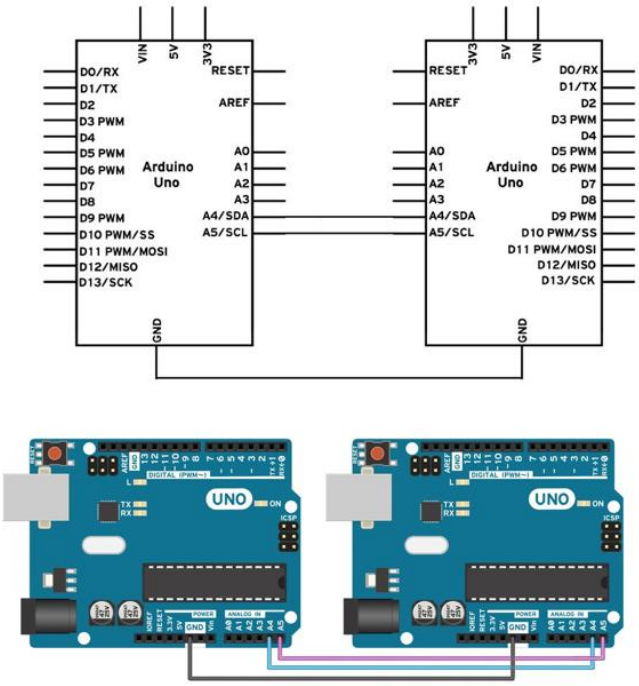
5.5 *Σύνδεση Arduino – Συσκευής μέσω I2C (Βιβλιοθήκη Wire)*

Η σύνδεση μέσω I2C, μπορεί να επιτευχθεί συνδέοντας μικρο-ελεγκτές Arduino μεταξύ τους ή ένα Arduino με διάφορες περιφερειακές συσκευές. Κρίνεται χρήσιμο όταν πρόκειται να συνδεθούν πολλές συσκευές μαζί και μέσω του διαύλου I2C μπορεί να επιτευχθεί συνδέοντας 3 καλώδια από κάθε συσκευή μαζί (SDA, SCL και GND). Όπως είδαμε παραπάνω τα ArduinoMega και τα ArduinoUno διαθέτουν δυνατότητα σύνδεσης I2C, μέσω κατάλληλων θυρών (Πίνακας 5-5):

	Θύρα SDA	Θύρα SCL
ArduinoUnoRev3	A4	A5
ArduinoMega2650Rev3	20	21

Πίνακας 5-5 Θύρες σύνδεσης I2C στα Arduino Mega και Uno

Στο Σχήμα 5-1 φαίνεται η διάταξη και το ηλεκτρονικό σχέδιο σύνδεσης από 2 ArduinoUno μέσω I2C.



Σχήμα 5-1 Σύνδεση δύο Arduino μέσω I2C

Τα Arduino διαθέτουν έτοιμη βιβλιοθήκη για την I2C επικοινωνία, την βιβλιοθήκη *Wire*, όπου παρακάτω περιγράφονται οι συναρτήσεις της: [24]

- *Wire.begin(address):* Καλείται μια φορά μόνο φορά στο πρόγραμμα. Αρχικοποιεί την βιβλιοθήκη Wire και ανοίγει τον δίαυλο μεταξύ I2C των συσκευών. Βάσει αυτής δηλώνονται οι συσκευές ως Master ή ως Slave, γράφοντας τον αριθμό που αντιπροσωπεύει την slave συσκευή (μια 7-bit διεύθυνση) ή τίποτα αν είναι master.
- *Wire.requestFrom(address, quantity,stop):* Μόνο για master συσκευή. Χρησιμοποιείται για να ζητήσει bytes από μια slave device, μετά ακολουθεί μια συνάρτηση Wire.available ή Wire.read. Το address αναφέρεται σε ποιο slave να επικοινωνήσει. Το quantity αναφέρεται στα πόσα bytes να περάσουν στη slave συσκευή. Το stop αναφέρεται στο ότι by default ελευθερώνεται ο δίαυλος I2C μετά το μήνυμα. (Αντιστοιχεί στο slave το Wire.onRequest()).
- *Wire.beginTransmission(address):* Μόνο για master συσκευή. Ξεκινάει μια μετάδοση από το master στο slave address, μετά ακολουθούν οι συναρτήσεις Wire.write() και Wire.endTransimission(). (Αντιστοιχεί στο slave το Wire.onReceive()).
- *Wire.endTransmission(stop):* Μόνο για master συσκευή και πρέπει να έχει γραφτεί πρώτα η Wire.beginTransmission(), τερματίζοντας τη μετάδοση I2C. Το stop by default ελευθερώνει τον δίαυλο I2C.
- *Wire.write(val):* Στο slave εντός της συνάρτησης Wire.onRequest() στέλνει δεδομένα από slave σε master device, αφού πρώτα το έχει ζητήσει το master. Στο master μεταξύ των Wire.beginTransmission() και Wire.endTransmission() στέλνει δεδομένα (val) από master σε slave device.

- *Wire.available():* Στο master εντός της συνάρτησης Wire.requestFrom(). Στο slave εντός της συνάρτησης Wire.onReceive(). Πάντα πριν την Wire.read(), καθώς εξασφαλίζει bytes για να αποθηκευτεί η τιμή που θα διαβάσει η Wire.read().
- *Wire.read():* Διαβάζει 1 byte. Στο master εντός του requestFrom() που μεταδόθηκε από slave. Στο slave εντός του WireonReceive() που μεταδόθηκε από master στο slave.
- *Wire.setClock();* Τροποποιεί την συχνότητα του ρολογιού για επικοινωνία I2C.
- *Wire.onReceive(handler):* Μόνο για slave device. Φτιάχνει μια συνάρτηση, η οποία εκτελείται στο slave, όταν μεταδώσει δεδομένα ο master μέσω της συνάρτησης Wire.beginTransmission(). Όπου handler το όνομα της συνάρτησης.
- *Wire.onRequest(handler):*Μόνο για slave device. Φτιάχνει μια συνάρτηση, η οποία εκτελείται στο slave, όταν ζητήσει δεδομένα ο master μέσω της συνάρτησης Wire.requestFrom(). Όπου handler το όνομα της συνάρτησης.

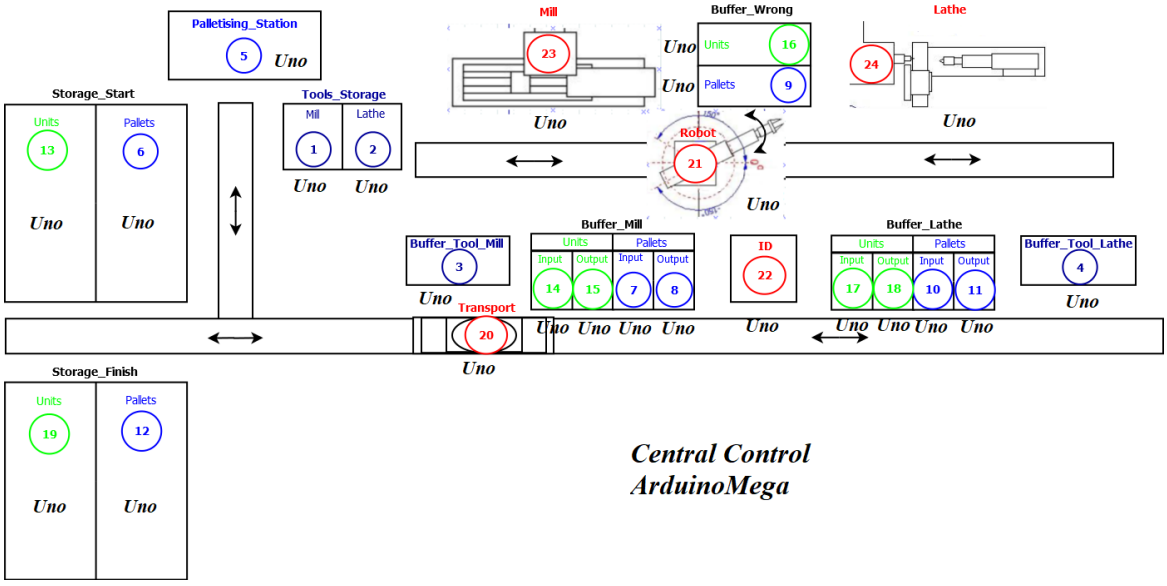
6

Έλεγχος του ΕΣΚ μέσω Μικρο-ελεγκτών Arduino

6.1 Έλεγχος Master - Slaves μέσω I2C

Το ΕΣΚ γνωρίζουμε από το 2ο Κεφάλαιο, πως αποτελείται από 24 σταθμούς, οι οποίοι πρέπει να ελεγχθούν από έναν κεντρικό ελεγκτή. Κατόπιν αυτού, οι 24 σταθμοί αποτελούν τους τοπικούς ελεγκτές, τους οποίους πρέπει να ελέγξει ο κεντρικός ελεγκτής. Επιλέχθηκε η σύνδεση I2C, όπου η master συσκευή θα είναι 1 ArduinoMega2560 που θα προσομοιάζει τον κεντρικό ελεγκτή και οι slave συσκευές που θα είναι 24 ArduinoUno και θα προσομοιάζουν τους τοπικούς ελεγκτές (Σχήμα 6-1). Επιλέχθηκε, η σειριακή επικοινωνία μέσω I2C, λόγω:

- Της απλότητας σύνδεσης, μέσω 3 καλωδίων (SDS, SCL, GND)
- Της έτοιμης βιβλιοθήκη Wire(), που παρέχεται στις πλακέτες Arduino
- Της δυνατότητας σύνδεσης πολλών περιφερειακών συσκευών, εδώ μικρο-ελεγκτές Arduino
- Της μετάδοσης απλών σημάτων High/Low (ή 0/1) μεταξύ του master και των slaves



Σχήμα 6-1 ΕΣΚ με Κεντρικό Ελεγκτή (Arduino-Mega) και Τοπικούς Ελεγκτές (Arduino-Uno)

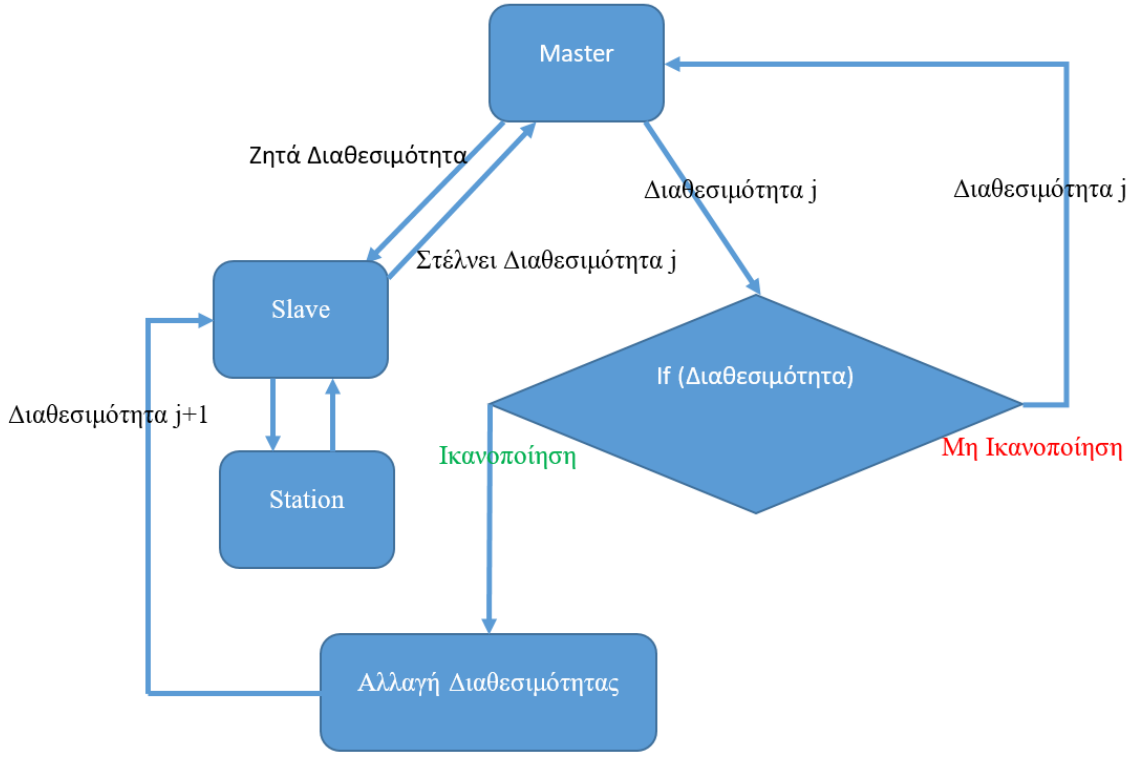
6.1.1 Λογική Ελέγχου

Η μετάδοση σημάτων μεταξύ του κεντρικού και των τοπικών ελεγκτών γίνεται με την εξής λογική κλειστού βρόχου:

- Στην αρχή, ο master ελεγκτής ζητά από τους slaves ελεγκτές να του αναφέρουν, αν είναι διαθέσιμοι για ανάληψη εργασίας ή μη.
- Οι slaves αναφέρουν διαθεσιμότητα προς τον κεντρικό ελεγκτή, αφού επιβεβαιώσουν αν ο σταθμός που ελέγχουν είναι διαθέσιμος ή μη. Αυτό το γνωρίζουν ανά πάσα στιγμή, από το σήμα που λαμβάνουν από το σταθμό σε περίπτωση δέσμευσης, αλλιώς είναι ελεύθεροι για ανάληψη εργασίας (κατάσταση High/Low ή 0/1).
- Ο master ελεγκτής, λαμβάνει τα σήματα από όλους τους τοπικούς ελεγκτές και βάσει αυτών των σημάτων, τα οποία καταχωρεί σε μεταβλητές, ελέγχει τις συνθήκες επιλογής (if conditions), οι οποίες συνθήκες αποτελούν τους κλάδους εισόδου κάθε μετάβασης του δικτύου Petri. Δηλαδή, ο master αφού μάθει ποιοι σταθμοί είναι ελεύθεροι, ελέγχει από τις συνθήκες επιλογής, ποιες δύνανται να ικανοποιηθούν. Σε κάθε loop (κύκλο του προγράμματος) θα πρέπει να ικανοποιούνται συνθήκες, που είναι διαδοχικές (sequence) και μη συγκρουόμενες (conflict).
- Μόλις, ικανοποιείται κάποια συνθήκη επιλογής, τρέχουν οι εντολές εντός αυτής, που αποτελούν τους κλάδους εξόδου των μεταβάσεων του δικτύου Petri. Οι εντολές αυτές είναι σήματα δέσμευσης και αποδέσμευσης των σταθμών του ΕΣΚ, τα οποία αποστέλλονται από τον master προς τους slaves, προκειμένου να ενεργήσουν αναλόγως.
- Αφού ελεγχθούν όλες οι συνθήκες επιλογής και εκτελεστούν οι εντολές των συνθηκών που ικανοποιήθηκαν, οι slaves πλέον έχουν λάβει τα σήματα δέσμευσης και αποδέσμευσης από τον master και ενεργούν αναλόγως του σήματος.
- Οι slaves, κατόπιν ειδοποιούν τους σταθμούς μέσω σήματος, μόνο στη περίπτωση που οι σταθμοί ήταν ελεύθεροι, όταν τους ζήτησε διαθεσιμότητα ο master και κατόπιν δεσμεύτηκαν από αυτόν, λόγω εκτέλεσης των εντολών κάποιας συνθήκης που ικανοποιήθηκε. Ενώ, στην αντίθετη περίπτωση, όπου οι σταθμοί ήταν ήδη δεσμευμένοι και ο αντίστοιχος slave ειδοποίησε τον master κατά τον έλεγχο διαθεσιμότητας, παραμένουν ως έχουν.
- Οπότε, στο τέλος του loop οι slaves έχουν αποστείλει σήμα δέσμευσης/αποδέσμευσης ή τίποτα στον σταθμό τους και έχουν πλέον γνώση της καινούργιας κατάστασης διαθεσιμότητας του σταθμού τους. Ενώ, παράλληλα όσοι σταθμοί δεσμεύτηκαν, έχουν ξεκινήσει την διεργασία τους.
- Γυρίζοντας πάλι στην αρχή του loop, ο master κάνει έλεγχο διαθεσιμότητας στους 24 τοπικούς ελεγκτές και συνεχίζει ο έλεγχος από τους τοπικούς ελεγκτές προς τους σταθμούς τους.

Η παραπάνω λογική, για να εφαρμοστεί πρέπει να δοθεί μεγάλη προσοχή στον έλεγχο ικανοποίησης των συνθηκών των δομών επιλογής, διότι αν ικανοποιηθούν εντολές, που έχουν σύγκρουση τότε η μια από τις δύο δεν θα εκτελεστεί. Επειδή η ροή παραγωγής έχει μια προκαθορισμένη διαδρομή, αν αυτή διακοπεί μπορεί να καθηλώσει και όλα υπόλοιπα προϊόντα (παλέτες/τεμάχια) που έπονται. Δηλαδή, δε σημαίνει πως αν δεν ικανοποιηθεί μια συνθήκη σε ένα loop, τότε θα ικανοποιηθεί στο επόμενο, καθώς μπορεί να μην

ικανοποιηθεί ποτέ. Αυτό συμβαίνει, επειδή οι συνθήκες επιλογής είναι πολυπαραμετρικές και έχουν πολλούς όρους που πρέπει να ικανοποιούνται ταυτόχρονα. Τα παραπάνω συνοψίζονται στο Παρακάτω Διάγραμμα Ροής του Σχήματος 6-2, όπου η Διαθεσιμότητα j είναι η αρχική, ενώ η Διαθεσιμότητα j+1 είναι αυτή που εκτελείται από την ικανοποίηση των συνθηκών της δομής επιλογής.



Σχήμα 6-2 Διάγραμμα Ροής της Λογικής Ελέγχου που εφαρμόστηκε στο ΕΣΚ

Γενικά, η λογική του προγράμματος στηρίζεται πολύ στον έλεγχο ικανοποίησης των συνθηκών στη δομή επιλογής (if condition). Διότι, είναι αυτές που επηρεάζουν τη διαθεσιμότητα του κάθε σταθμού του ΕΣΚ και λειτουργούν όπως οι μεταβάσεις στα δίκτυα Petri. Εκεί, οι μεταβάσεις για να ενεργοποιηθούν έπρεπε οι θέσεις να είχαν τόσα κουπόνια όσα το βάρος του αντίστοιχου κλάδου εισόδου (από τις θέσεις στις μεταβάσεις), τότε ενεργοποιούνταν και έστελναν τόσα κουπόνια στις θέσεις όσα τα βάρη του κλάδου εξόδου που τις σύνδεαν. Με την ίδια λογική μόλις ο κεντρικός ελεγκτής «διαβάσει» τις εισόδους του, που εδώ είναι σήματα high/low από τους τοπικούς ελεγκτές, τα καταχωρεί σε μεταβλητές και τα συγκρίνει στις συνθήκες των δομών επιλογής, συνδεόμενες με λογικά AND ή/και OR μεταξύ τους, αν είναι high/low. Σε περίπτωση που κάποια συνθήκη ικανοποιηθεί, τότε εκτελούνται οι εντολές εντός της δομής επιλογής, οι οποίες είναι οι μεταβλητές που καταχωρήθηκαν τα σήματα και αλλάζουν τιμή high/low. Όλα τα παραπάνω, μπορούν να συνοψιστούν στην εξής δομή, όπου P είναι οι θέσεις του δικτύου Petri:

If (Pin = ……High/low) {

Pout = ……High/low

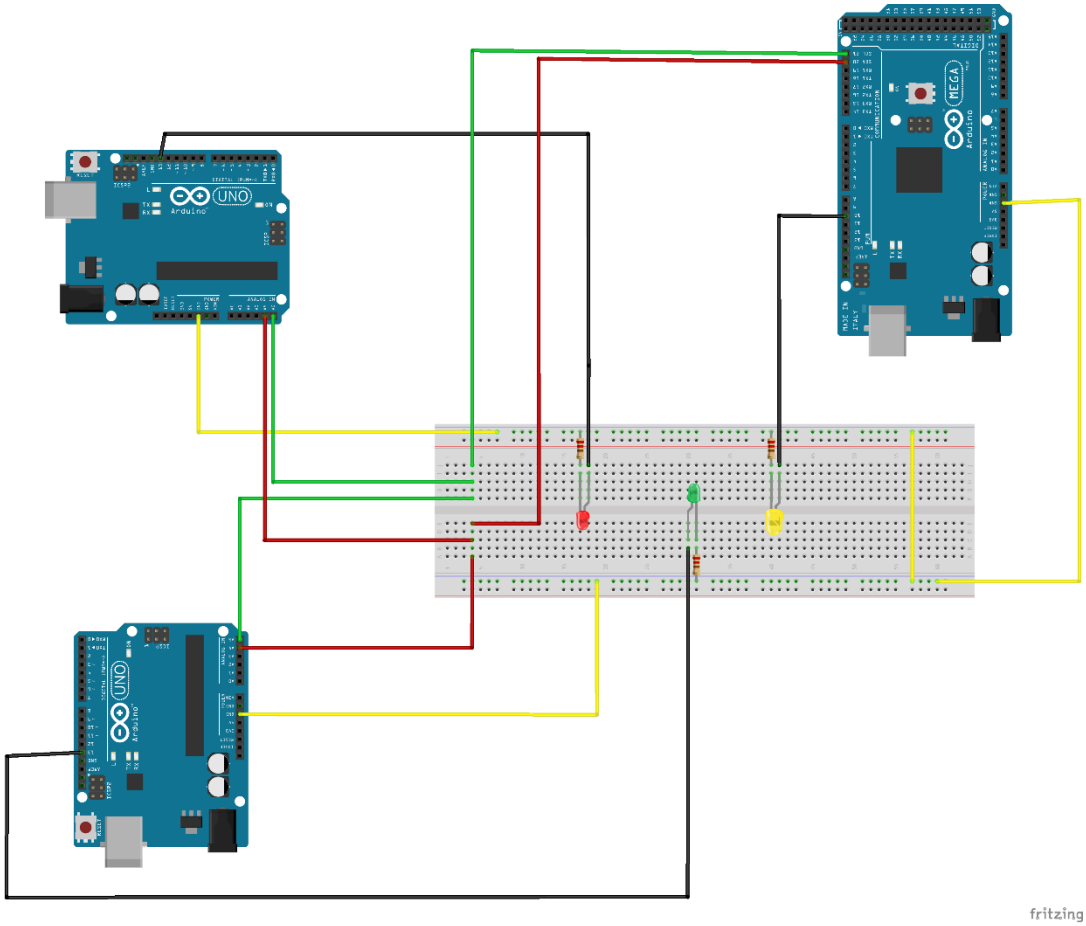
}

6.1.2 Εφαρμογή ArduinoMega (master) – 2 ArduinoUno (slaves)

Η υλοποίηση που πραγματοποιήθηκε στο ΕΣΚ, είναι η σύνδεση του Κεντρικού ελεγκτή και των Τοπικών ελεγκτών μέσω του I2C. Για τον κεντρικό ελεγκτή επιλέχθηκε το ArduinoMega, λόγω των χαρακτηριστικών του, οι οποίοι παρατέθηκαν στο προηγούμενο κεφάλαιο (πολλά I/O pins, αυξημένες δυνατότητες μνήμης και επεξεργαστή), καθώς πρόκειται να ελέγχει 24 ArduinoUno. Για τους τοπικούς ελεγκτές επιλέχθηκαν τα ArduinoUno, διότι το καθένα ελέγχει μόνο έναν σταθμό μέσω σημάτων εισόδου-εξόδου και επίσης, συνδέεται με τον κεντρικό ελεγκτή για τον ίδιο σκοπό. Οπότε, ένα απλό ArduinoUno καλύπτει τις απαιτήσεις του συστήματος τοπικών ελεγκτών.

Επειδή, η υλοποίηση με 24 Arduino ως τοπικούς ελεγκτές καθίσταται ανέφικτη, λόγω του υψηλού κόστους υλοποίησης στα πλαίσια της εργασίας, υλοποιήθηκε η υλοποίηση με 1 ArduinoMega ως κεντρικό και 2 ArduinoUno ως slaves. Εξάλλου, οι αρχές που θα εφαρμοστούν στον έλεγχο 2 Arduino, μπορούν να επεκταθούν και στον έλεγχο των 24 Arduino, με γνώμονα πως και το πρωτόκολλο I2C μπορεί να υποστηρίξει τη σύνδεση 127 κόμβων μέσω ενός διαύλου (SCL, SDA). Παράλληλα, έγινε και η υλοποίηση ενός ArduinoMega με 24 εισόδους (διακόπτες) και 24 εξόδους (leds), όπου εκεί δομήθηκαν οι if_conditions, το οποίο θα παρουσιαστεί παρακάτω. Τα 2 slave devices (ArduinoUnos) προσομοιάζουν τον ελεγκτή ενός Robot και ενός Κέντρου Κατεργασιών (Mill).

Κατόπιν των παραπάνω, στο Σχήμα 6-3 παρουσιάζεται η συνδεσμολογία (μέσω του προγράμματος fritzing) της σύνδεσης I2C μεταξύ των 3 Arduino, όπου βραχυκυκλώνονται οι κλάδοι των Serial Data (SDA), βάσει των οποίων γίνεται η μεταφορά των δεδομένων, δηλαδή η μεταφορά σήματος High/Low μεταξύ master και slaves για την αναφορά ή αλλαγή διαθεσιμότητας του slave. Επίσης, φαίνονται οι βραχυκυκλωμένοι κλάδοι των SerialClock (SCL), που χρησιμοποιείται για τον συγχρονισμό μετάδοσης και λήψης δεδομένων. Και τέλος, βραχυκυκλώνονται και οι γειώσεις (GND).



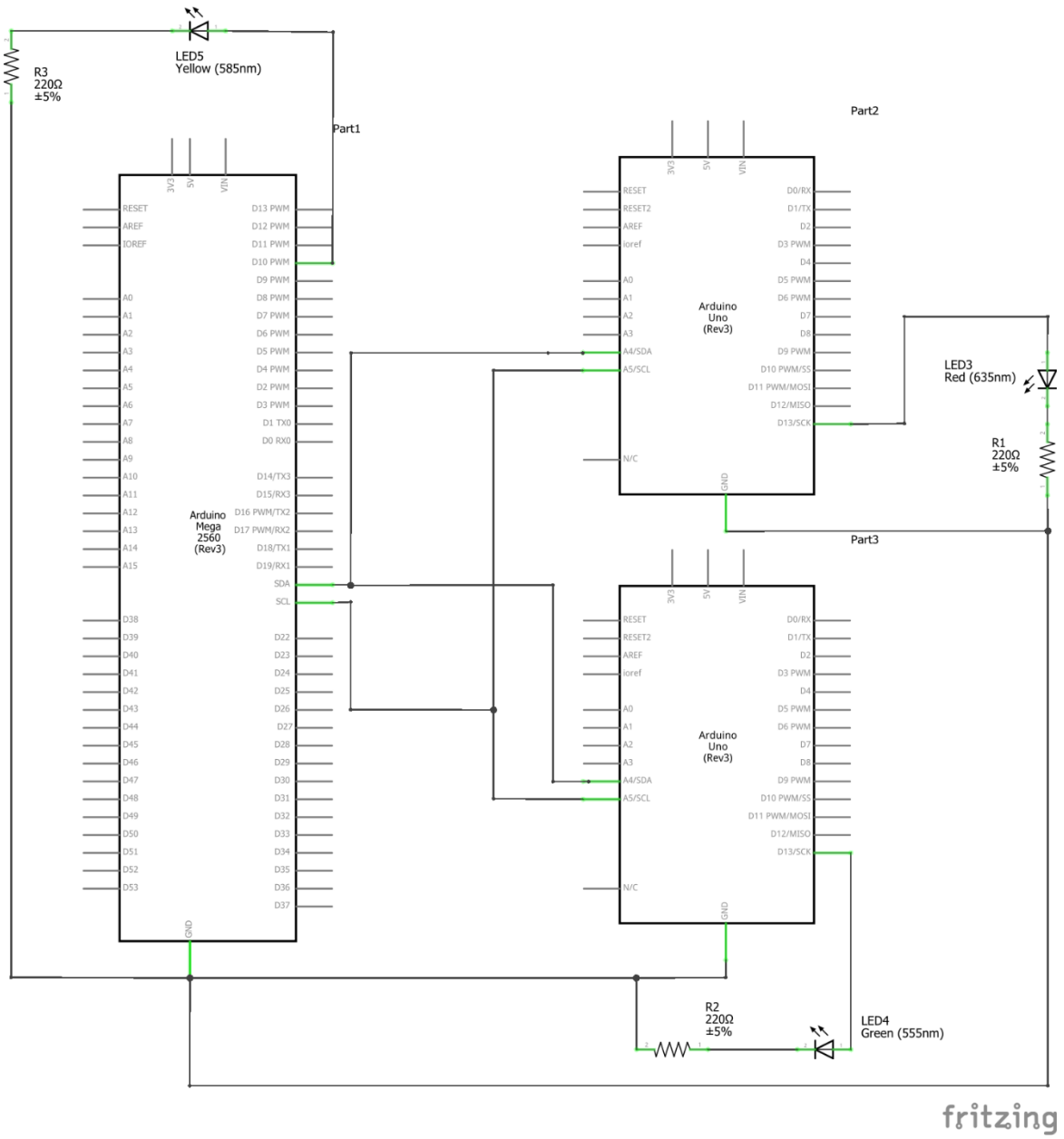
Σχήμα 6-3 Συνδεσμολογία I2C με 1 ArdulinoMega και 2 ArduinoUno

Πέραν των παραπάνω αναγκαίων συνδέσεων για την σύνδεση μέσω I2C, χρησιμοποιήθηκαν τα καλώδια jumpers, ιδανικά για χρήση στην breadboard, φωτοδίοδοι (leds) και αντιστάσεις 220Ω για την προστασία των leds. Τα παραπάνω leds χρησιμοποιήθηκαν, προκειμένου παράλληλα με την παρακολούθηση της σειριακής να ελέγχουμε αν και πότε υπάρχει μετάδοση και λήψη των σημάτων μεταξύ του master και των slaves, καθώς και πότε αλλάζει loop. Η αντιστοίχιση του χρωματισμού των φωτοδιόδων με τα Arduino παρουσιάζονται στον παρακάτω Πίνακα 6-1:

Controller	***Arduino***	***Pin***	***Led***
Central Controller	ArduinoMega	Pin10	Κίτρινο μεγάλο led
Robot’s Controller	ArduinoUno	Pin13	Πράσινο μικρό led
Mill’s Controller	ArduinoUno	Pin13	Κόκκινο μικρό led

Πίνακας 6-1 Εξαρτήματα I2C σύνδεσης

Το ηλεκτρονικό σχέδιο της παραπάνω διάταξης παρουσιάζεται παρακάτω στο Σχήμα 6-4, στο οποίο είναι εμφανείς οι συνδέσεις που πραγματοποιήθηκαν, καθώς και τα χαρακτηριστικά των εξαρτημάτων που συνθέτουν το κύκλωμα.

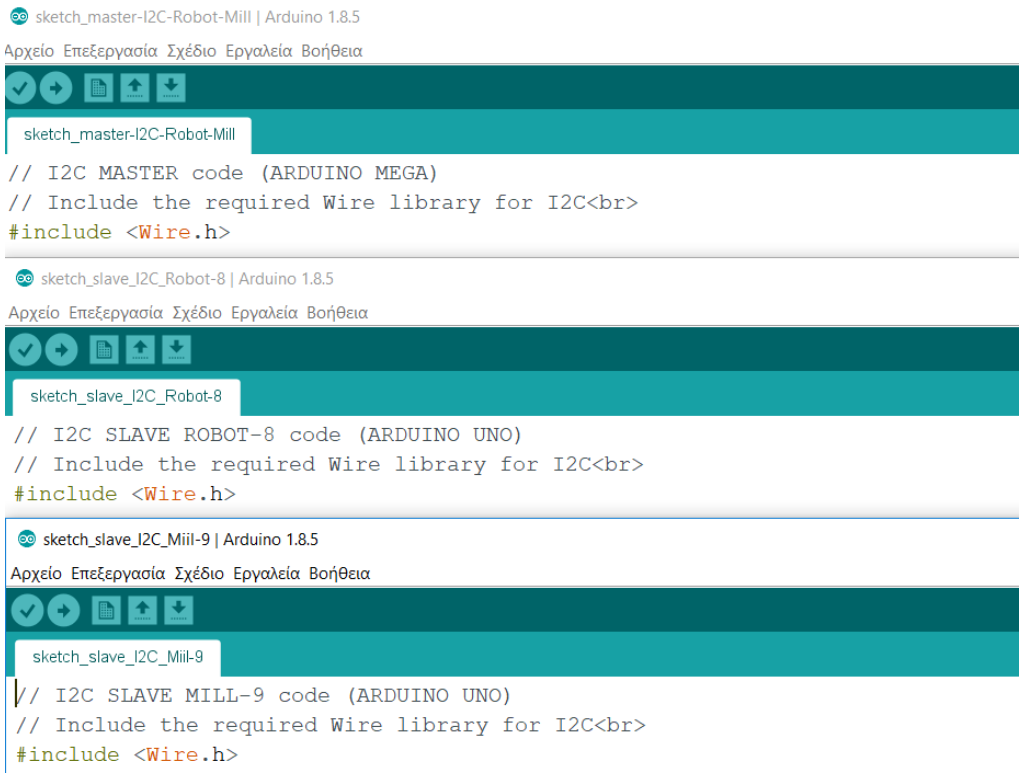


Σχήμα 6-4 Ηλεκτρονικό Σχέδιο I2C Κυκλώματος

6.1.3 Πρόγραμμα Ελέγχου με ArduinoMega2560 - 2 ArduinoUno μέσω I2C

Για τη σύνταξη του κώδικα (Παράρτημα 9.2) χρησιμοποιήθηκε η βιβλιοθήκη Wire() και οι κοινές εντολές προγραμματισμού Arduino. Ουσιαστικά, πρόκειται για 3 κώδικες, έναν για τον master και δύο πανομοιότυπους για τα slaves. Οι 3 κώδικες έχουν πάρει την ονομασία των μηχανών που ελέγχουν, έτσι έχουμε τον master και για τους slaves τον robot και τον mill. Αρχικά, και στους 3 κώδικες φορτώνεται η βιβλιοθήκη Wire (Εικόνα 6-1).

Master Code



Εικόνα 6-1 Φόρτωση βιβλιοθήκης Wire()

Έπειτα, στον κώδικα του master (Παράρτημα 9.2.1) ορίζουμε τον αριθμό διεύθυνσης για κάθε slave, ώστε ο master να ορίζει σε ποιον slave θα κάνει μετάδοση για να ζητήσει ή να στείλει στοιχεία, αφού μόνο ο master καθορίζει την επικοινωνία σε μια σύνδεση I2C. Ύστερα, ορίζονται οι 2 μεταβλητές "Robot" και "Mill", που θα χρησιμοποιηθούν στις if conditions, και είναι αυτές που αντιστοιχούν στη διαθεσιμότητα των 2 slaves (Εικόνα 6-2). Η τιμή 'k' που δίνουμε για αρχικοποίηση των μεταβλητών δεν παίζει κάποιο ρόλο, διότι θα πάρουν μετά τις τιμές 'H' ή 'L' μέσω των slaves. Οι μεταβλητές παίρνουν τιμές χαρακτήρων, οπότε όταν λέμε HIGH/LOW δεν είναι δυαδικοί αριθμοί 0 και 1, αλλά τοποθετήθηκαν έτσι για να ταιριάζουν με την έννοια που ερμηνεύονται, καθώς θα μπορούσαμε να βάλουμε 2 οποιουσδήποτε άλλους χαρακτήρες. Τέλος, και στα 3 προγράμματα (master και 2 slaves) LOW σημαίνει πως είναι διαθέσιμη η συσκευή ή έχει κουπόνι (όπως ερμηνεύεται στα δίκτυα Petri) και HIGH σημαίνει δεσμευμένη.

```
// Name the addresses of the 2 Slaves Arduino Uno (not 0<Address<7)
const byte RobotAd=8;
const byte MillAd=9;

/*Initialize the variables, the char doesn't matter,
  Just to compile the code 'cause of if condition*/
char Robot = 'k';
char Mill = 'k';
```

Εικόνα 6-2 Master Code Μεταβλητές των slaves για if conditions

Κατόπιν, έχουμε το void setup() του master, όπου ορίζουμε να ξεκινήσει η I2C επικοινωνία, μέσω της Wire.begin(), χωρίς αριθμό εντός παρενθέσεως, διότι πρόκειται για τον master, ενώ οι 2 slaves θα βάλουν τους αριθμούς που ορίσαμε παραπάνω. Επίσης, εκκινούμε την σειριακή επικοινωνία (Εικόνα 6-3).

```
void setup() {
  // Start the I2C Bus as Master
  Wire.begin(); // don't need a number in parenthesis, 'cause it's the MASTER ARDUINO MEGA

  // Open Serial Port and set data rate = 9600 bps (baud)
  Serial.begin(9600);
}
```

Εικόνα 6-3 Master Code Έναρξη Wire και Serial

Ορίζουμε το led ως έξοδο στο pin10, το οποίο έχουμε βάλει για τον έλεγχο εισόδου – εξόδου στο βρόχο void loop(), και το έχουμε θέσει να είναι ανοικτό όταν μπαίνουμε στο loop, και να σβήνει πριν βγει. Έπειτα, ζητάει ο master από τον slave Robot, μέσω της συνάρτησης Wire.requestFrom(8,1) να του στείλει την διαθεσιμότητα του (High/Low), η οποία είναι χαρακτήρας ('H'/'L') μεγέθους 1 byte και την αποθηκεύει στην μεταβλητή "Robot", τύπου χαρακτήρα, που ορίστηκε παραπάνω. Στην εντολή Wire.requestFrom(8,1) το 8 αντιστοιχεί στην slave συσκευή που ορίστηκε με αυτόν τον αριθμό, που είναι η Robot και το 1 αντιστοιχεί στο 1 byte που μεταφέρεται από την slave συσκευή στο master. Η δομή επανάληψης while(Wire.available()) «πιστοποιεί», κατά κάποιο τρόπο, πως υφίσταται η I2C επικοινωνία μεταξύ των συσκευών και δεν έχει διακοπεί για κάποιο λόγο.

Μετέπειτα διαβάζει μέσω της εντολής Wire.read(), τη διαθεσιμότητα που στέλνει ο slave και την αποθηκεύει στη μεταβλητή 'Robot'. Μετά εκτυπώνει αυτή τη διαθεσιμότητα για να έχουμε εικόνα των ενεργειών που εκτελούνται μέσω της σειριακής. Την ίδια γραμμή εντολών χρησιμοποιούμε και για τον έτερο slave mill, με μόνη διαφορά αντί για 8 βάζουμε 9. Επίσης, εκτυπώνεται ότι ο master διάβασε τους slaves (Εικόνα 6-4).

```
void loop() {

  // LED for checking the loop
  pinMode(10, OUTPUT);
  digitalWrite(10, HIGH);
  delay (1000);

  // The Master asks the slave Robot (RobotAd = 8) if it is available or not
  Wire.requestFrom(8,1);
  while (Wire.available()){
    Robot = Wire.read();
    Serial.print("1. The Robot read from request = ");
    Serial.println(Robot);
  }
  delay (2000);

  // The Master asks the slave Mill (MillAd = 8) if it is available or not
  Wire.requestFrom(9,1);
  while (Wire.available()){
    Mill = Wire.read();
    Serial.print("2. The Mill read from request = ");
    Serial.println(Mill);
  }
  delay (2000);

  Serial.println("3. End of Request from Slaves to Master");
```

Εικόνα 6-4 Master Code request διαθεσιμότητα από slaves

Μετέπειτα, ελέγχεται αν ικανοποιείται η συνθήκη, από τις μεταβλητές που διάβασε πρωτύτερα, και αν ικανοποιείται, τότε εκτελούνται οι εντολές εντός των αγκύλων. Εκτυπώνεται ότι τα slaves δεσμεύονται, καθώς ήταν διαθέσιμα = LOW, αλλάζει την τιμή στις μεταβλητές από LOW σε HIGH και τις εκτυπώνει. Οπότε, τώρα είναι δεσμευμένα (HIGH) τα slaves και θα πρέπει να σταλθεί αυτή πληροφορία στους slaves προκειμένου να ενεργήσουν (Εικόνα 6-5).

```
Serial.println("3. End of Request from Slaves to Master");
delay (2000);

// value = 'L' in the char variables, means they have token (LED = LOW).
// But the token doesn't mean always available, but as it has been defined in the Petri Net
if (Robot == 'L' && Mill == 'L'){
  // Robot = 1 and Mill = 1 are available, so do the milling process and convert them to
  // Robot = 0 and Mill = 0
  Serial.println("4. Engage Robot and Mill ");
  Robot = 'H';
  Mill = 'H';
  Serial.print("Robot= ");
  Serial.println(Robot);
  Serial.print("Mill= ");
  Serial.println(Mill);
  delay (2000);
```

Εικόνα 6-5 Master Code Έλεγχος Ικανοποίησης if condition και εντολές αυτής

Τέλος, αν ικανοποιήθηκαν οι συνθήκες τότε άλλαξαν οι διαθεσιμότητες και στέλνονται οι αλλαγές στις μεταβλητές των slaves μέσω της συνάρτησης Wire.beginTransmission(), από το master και στα 2 slaves devices και εκτυπώνεται πως τα δεσμεύει. Δηλαδή, τα δεσμεύει για εργασία, όπως π.χ. θα συνέβαινε αν υπήρχε τεμάχιο προς κατεργασία στο Buffer του Mill και για να φορτωθεί δεσμεύονται το Robot για να το μεταφέρει και το Mill για να το επεξεργαστεί. Εκτός συνθήκης επιλογής απενεργοποιούμε το led για να ξέρουμε πως είμαστε στο τέλος του βρόχου (Εικόνα 6-6). Η εντολή Wire.beginTransmission(αριθμός slave) ουσιαστικά ξεκινά την μετάδοση προς το slave, για να στείλει μέσω της εντολής Wire.write() την τιμή της μεταβλητής Robot στο slave και έπειτα τελειώνει τη μετάδοση με την εντολή Wire.endTransmission(). Θα μπορούσαμε να πούμε πως μέσω της Wire.requestFrom() ο master ζητά από τον slave (receives data), ενώ μέσω της Wire.beginTransmission() ο master στέλνει προς τον slave (transmits data). Αλλά πάντα και στις 2 περιπτώσεις (receive/transmit) αυτός που ξεκινά την «κλήση» είναι ο master προς τον slave και ποτέ από μόνος του ο slave δεν καλεί τον master.

```
    Serial.println("5. Begin Transmition to Slaves from Master");
    // Send signals to slaves, only when Robot and Miil are available ('L') to be occupied ('H')
    // Sending to Robot the order of engagement to Robot Address ('H')
    Wire.beginTransmission(8);
    Wire.write(Robot);
    Wire.endTransmission();

    // Sending to Mill the order of engagement to Mill Address ('H')
    Wire.beginTransmission(9);
    Wire.write(Mill);
    Wire.endTransmission();

  }

  delay (2000);
  // Change the LED in each loop
  digitalWrite(10, LOW);
  delay (2000);
```

Εικόνα 6-6 Master Code Έναρξη Μετάδοσης προς slaves για δέσμευση τους

Ο βρόχος στο master συνεχίζει να τρέχει, έως ότου να αποδεσμευθούν και τα 2 slaves (γίνουν LOW = διαθέσιμα), τότε το master θα τα ξαναδεσμεύσει και ούτω κάθε εξής.

Slave Code

Όσον αφορά την εκτέλεση του κώδικα στα slaves, έχουμε ακριβώς τις ίδιες ενέργειες στους τοπικούς ελεγκτές των Robot και Mill, με τα μόνα που αλλάζουν να είναι οι ονομασίες των μεταβλητών. Οπότε, παρατίθεται η προσομοίωση για τον ένα μόνο κώδικα του Robot (Παράρτημα 9.2.2) και ισχύουν τα ίδια ακριβώς και για τον κώδικα του Mill (Παράρτημα 9.2.3).

Αρχικά, όπως και στον master φορτώνεται η βιβλιοθήκη Wire() και ορίζεται ο αριθμός διεύθυνσης ή ταυτότητας για τον slave RobotAd = 8 (για τον MillAd = 9), ώστε μέσω του αριθμού αυτού να οριστεί ως slave συσκευή στη σύνδεση I2C το Arduino, στο οποίο θα φορτωθεί ο υπόψη κώδικας. Ύστερα, ορίζεται η μεταβλητή (τύπου χαρακτήρα) 'Robot', η οποία αποτελεί τοπική μεταβλητή για τον κώδικα του συγκεκριμένου Arduino, και γίνεται αρχικοποίηση της μεταβλητής Robot = 'L' (δηλαδή L = διαθέσιμη) (Εικόνα 6-7). Στην πραγματικότητα, αυτός ο ορισμός δεν θα υπήρχε, καθότι ο τοπικός ελεγκτής θα «διάβαζε» το σήμα από τον σταθμό που ελέγχει, εδώ συγκεκριμένα τον σταθμό robot, και θα το καταχωρούσε/μετέφραζε ως έναν χαρακτήρα αντίστοιχης ερμηνείας του σήματος ('H'/'L') στην μεταβλητή 'Robot'.

sketch_slave_I2C_Robot-8

```
// I2C SLAVE ROBOT-8 code (ARDUINO UNO)
// Include the required Wire library for I2C<br>
#include <Wire.h>

// Name the addresses of the Slave Arduino Uno(not 0<Address<7)
const byte RobotAd=8;

// Initialize the variables L = 1 (available) and H = 0 (occupied = process)
char Robot = 'L';
```

Εικόνα 6-7 Slave Code Φόρτωση Wire - Ορισμός ταυτότητας - Αρχικοποίηση μεταβλητής

Έπειτα, το πρόγραμμα εισέρχεται στη ρουτίνα void setup(), η οποία στις slave devices είναι πιο σημαντική από την void loop(). Εδώ, ξεκινάει η σύνδεση I2C με την εντολή Wire.begin(8), όπου 8 είναι ο αριθμός

διεύθυνσης του slave, και η οποία καθορίζει το Robot ως slave. Μετά, ορίζεται το pin13 ως έξοδος στο led και το θέτουμε ως κλειστό (LOW), το οποίο θα χρησιμεύσει στον έλεγχο εκτέλεσης των εντολών της «*if condition*», αλλά και να δούμε πόσες φορές επικοινωνεί ο master με τον slave, από όταν θα ξεκινήσει η I2C επικοινωνία. Εν συνεχεία, εκτελούνται 2 συναρτήσεις η Wire.onRequest() και η Wire.onReceive(), όπου η κάθε μια εκτελείται, όταν την καλεί η master συσκευή με αντίστοιχες συναρτήσεις. Έτσι, η Wire.onRequest() στον slave απαντά στην Wire.requestFrom(8,1) του master. Ενώ, η Wire.onReceive() στον slave απαντά στη Wire.beginTransmission(8) του master, όπου 8 είναι ο αριθμός ταυτότητας του Robot. Οι 2 αυτές συναρτήσεις ορίζονται όπως και στη γλώσσα C, ξεχωριστά από το κυρίως πρόγραμμα σαν υπορουτίνες. Κατόπιν των παραπάνω, ανοίγει η σειριακή θύρα με τον Η/Υ, για να μπορέσουμε να εκτυπώσουμε στο σειριακό παράθυρο (serial monitoring). (Εικόνα 6-8).

sketch_slave_I2C_Robot-8

```
void setup() {
  // Start the I2C Bus as Slave-8
  Wire.begin(8); // need a number in parenthesis, 'cause it's a SLAVE ARDUINO UNO

  // LED is LOW when the Robot is available (Robot = 1)
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);

  // Send the value of Robot to master(answer to requestFrom) by write
  Wire.onRequest(requestEvent);
  // Receive the value of Robot by master(when Transmission) by read
  Wire.onReceive(receiveEvent);

  // Open Serial Port and set data rate = 9600 bps (baud)
  Serial.begin(9600);

  delay (1000);
}
```

Εικόνα 6-8 Slave Code Εκκίνηση I2C σύνδεσης - Καθορισμός συναρτήσεων στο void setup()

Το void loop(), που αποτελεί τη βασική συνάρτηση εδώ δεν εκτελείται, διότι ο κώδικας των slave devices εκτελείται 1 φορά και έπειτα καλείται από τον master μέσω κάποιας συνάρτησης να λάβει ή να μεταδώσει δεδομένα ενός byte. Αν είχαμε βάλει κάποια συνάρτηση ή κάποια μεταβλητή εντός του βρόχου (loop) αυτή θα εκτελούταν επαναληπτικά. Οπότε, όταν θα καλούσε ο master μέσω μια συνάρτησης, που αυτή είναι στο loop του master, την αντίστοιχη συνάρτηση - απάντηση στον slave που και αυτή είναι εντός του loop του slave, τότε θα είχαμε τη συνάρτηση του slave που είναι εντός του loop, να εκτελείται επαναλαμβανόμενα, το οποίο θα οδηγούσε το πρόγραμμα σε αέναο βρόχο. Εδώ, ο χρονιστής delay(100) δεν επηρεάζει τον κώδικα, καθώς δεν αποτελεί εντολή εκτέλεσης ενεργειών αλλά μια χρονική καθυστέρηση, καθώς το πρόγραμμα σταματάει την εκτέλεσή του για 100 ms. Οπότε, στο void loop() δεν διαβάζεται τίποτα και το πρόγραμμα συνεχίζει, γι' αυτό δεν είναι ορθό να υπάρχουν συναρτήσεις εντός του void loop() στις slave devices.

```
void loop() {
  // put your main code here, to run repeatedly:
  delay(100);
}
```

Εικόνα 6-9 Slave Code void loop() σε Slave device

Στο τέλος, έχουμε τον ορισμό των δύο συναρτήσεων - υπορουτινών που ονομάστηκαν requestEvent() για τη χρήση της από την Wire.onRequest() και την receiveEvent(), για τη χρήση της από την Wire.onReceive(). Η υπορουτίνα requestEvent() μέσω της εντολής Wire.write(Robot) και με την συνάρτηση Wire.onRequest() μεταδίδει την τιμή της τοπικής μεταβλητής 'Robot' από τον slave στον master, όπου εκεί διαβάζεται από την εντολή Wire.read() που βρίσκεται εντός της συνάρτησης Wire.requestFrom(8,1) του master (Εικόνα 6-10). Άρα, γίνεται μετάδοση δεδομένων από τον slave στον master, αφού πρώτα το ζητήσει ο master και έπειτα εκτυπώνει ότι έγινε η μετάδοση αυτή.

```
// Transmission to master
void requestEvent(){
  Wire.write(Robot);
  Serial.println("1. Robot Transmits to Master");
  delay (2000);
}
```

Εικόνα 6-10 Slave Code Υπορουτίνα requestEvent για μετάδοση data από τον slave στον master

Από την άλλη πλευρά η υπορουτίνα receiveEvent() χρησιμοποιείται από τον slave, για να διαβάσει την τιμή της μεταβλητής, που δηλώνει τη δέσμευση ή μη του robot, από τον master. Οπότε, μέσω της εντολής Wire.read() και με την συνάρτηση Wire.onReceive() να λαμβάνει την τιμή της μεταβλητής από τον master, όπου εστάλη από την εντολή Wire.Write() που βρίσκεται μεταξύ των εντολών Wire.beginTransimission(8) και Wire.endTransimission(). Μετέπειτα, γίνεται εκτύπωση ότι έγινε λήψη από τον master προς τον slave και μετά πηγαίνει για έλεγχο στην if condition. Όπου, αν η μεταβλητή που έστειλε ο master και αποθηκεύτηκε στον slave στην τοπική μεταβλητή Robot = 'H' είναι να δεσμευθεί ο σταθμός robot, λόγω της εκτέλεσης των εντολών της if condition του master, τότε ικανοποιείται η if condition και του slave δεσμεύοντας το ρομπότ. Με την δέσμευση του ρομπότ ανάβει το led του slave και εκτυπώνει ανάλογο μήνυμα δέσμευσης.

Έπειτα, αφού δεσμεύτηκε ο σταθμός robot, στην πραγματικότητα θα σταματούσε εκεί η υπορουτίνα, διότι θα τέλειωνε το «τρέξιμο» του κώδικα του slave και το robot θα παρέμενε δεσμευμένο. Για να αποδεσμευθεί το robot, θα έπρεπε ο τοπικός ελεγκτής slave να «διαβάσει» σήμα ολοκλήρωσης/αποδέσμευσης από τον σταθμό-robot. Αυτό το σήμα θα διαβαζόταν μέσω μιας εντολής digitalWrite(), η οποία θα βρισκόταν εντός της υπορουτίνας void requestEvent(). Σε κάθε loop του master θα ζητούσε την τιμή της μεταβλητής διαθεσιμότητας του robot, μέσω της εντολής Wire.requestFrom(8,1), έπειτα στον slave ο τοπικός ελεγκτής θα διάβαζε το σήμα από τον σταθμό και θα έστελνε αντίστοιχη τιμή της μεταβλητής διαθεσιμότητας του robot (Εικόνα 6-11) .

Για να προσομοιωθεί η παραπάνω πραγματική διαδικασία, χρησιμοποιήθηκε ο χρονιστής καθυστέρησης του προγράμματος delay(5000), ο οποίος προσομοιώνει την χρονική διάρκεια δέσμευσης του robot και μετά το αποδεσμεύει, κάνοντας το πάλι διαθέσιμο, εκτυπώνοντας αντίστοιχο μήνυμα (Εικόνα 6-11).

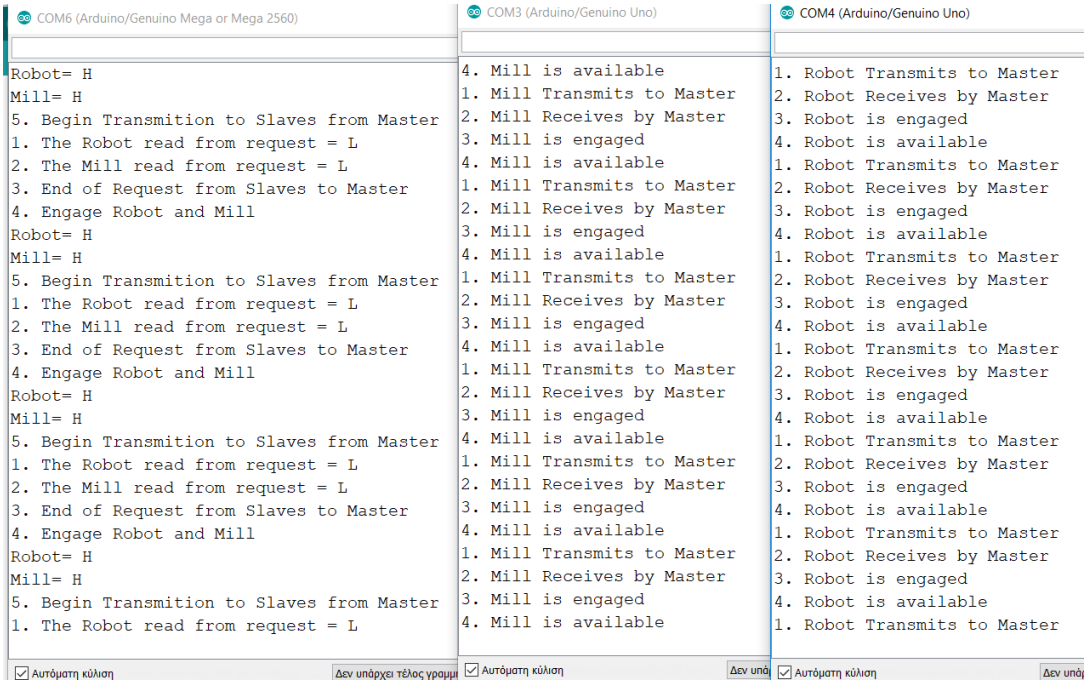
Την ίδια ακριβώς δομή κώδικα διαθέτει και το Arduino που προγραμματίστηκε ως τοπικός ελεγκτής του Mill, με μόνη διαφορά την ονομασία των μεταβλητών διαθεσιμότητας του σταθμού.

```
//Received by master
void receiveEvent(){
  while (Wire.available()){
    Robot = Wire.read();
    Serial.println("2. Robot Receives by Master");
    delay (2000);
    /* the if condition is unnecessary, 'cause the function receiveEvent
    takes place, only when Robot = 0, after entering the if condtion
    in the master's code*/
    if (Robot == 'H'){
      // Change the LED to HIGH, 'cause Robot = 0 means occupied
      digitalWrite(13, HIGH);
      Serial.println("3. Robot is engaged");
      /* Here delay is the most significant point, 'cause it changes the state
      of availability of the Robot. Normally here, a signal would be sent by the
      the Robot to its controller, when the processing was done */
      delay(5000);
      // Make available the Robot again
      Robot = 'L';
      digitalWrite(13, LOW);
      Serial.println("4. Robot is available");
    }
  }
}

/*---------- END OF CODE SLAVE ROBOT-8 ----------*/
```

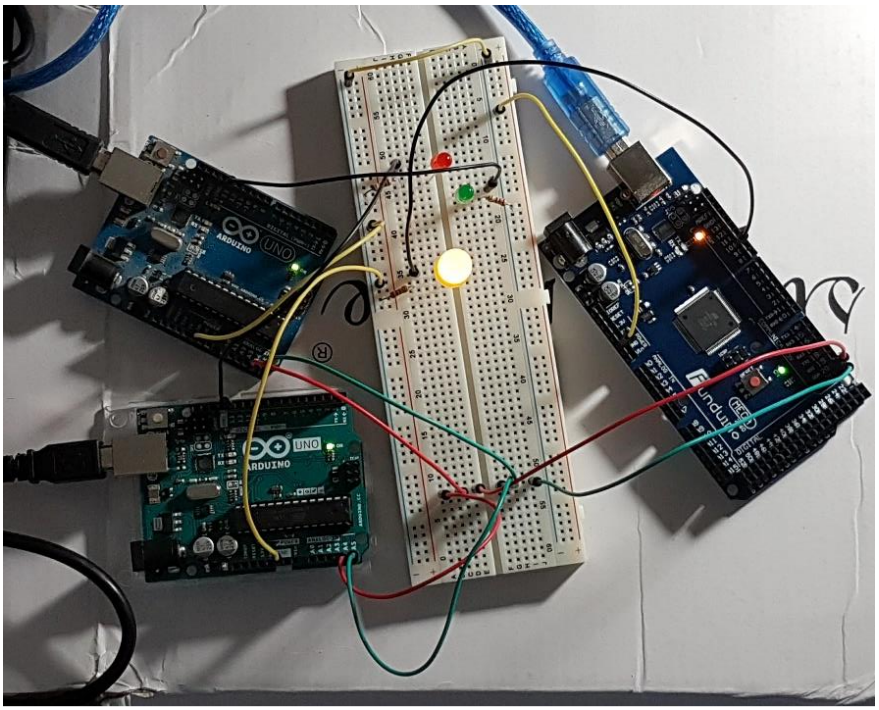
Εικόνα 6-11 Slave Code Υπορουτίνα receiveEvent για μετάδοση data από τον master στον slave

Στην παρακάτω Εικόνα 6-12 φαίνονται το σύνολο των εκτυπώσεων στα παράθυρα σειριακής και των 3 Arduino (Αριστερά του master-ArduinoMega, Μέση του slave(Mill) - ArduinoUno, Δεξιά του slave(Robot) - ArduinoUno).



Εικόνα 6-12 Εκτυπώσεις στη Σειριακή των 3 Arduinos με τη σειρά Master-Slave(Mill)-Slave(Robot)

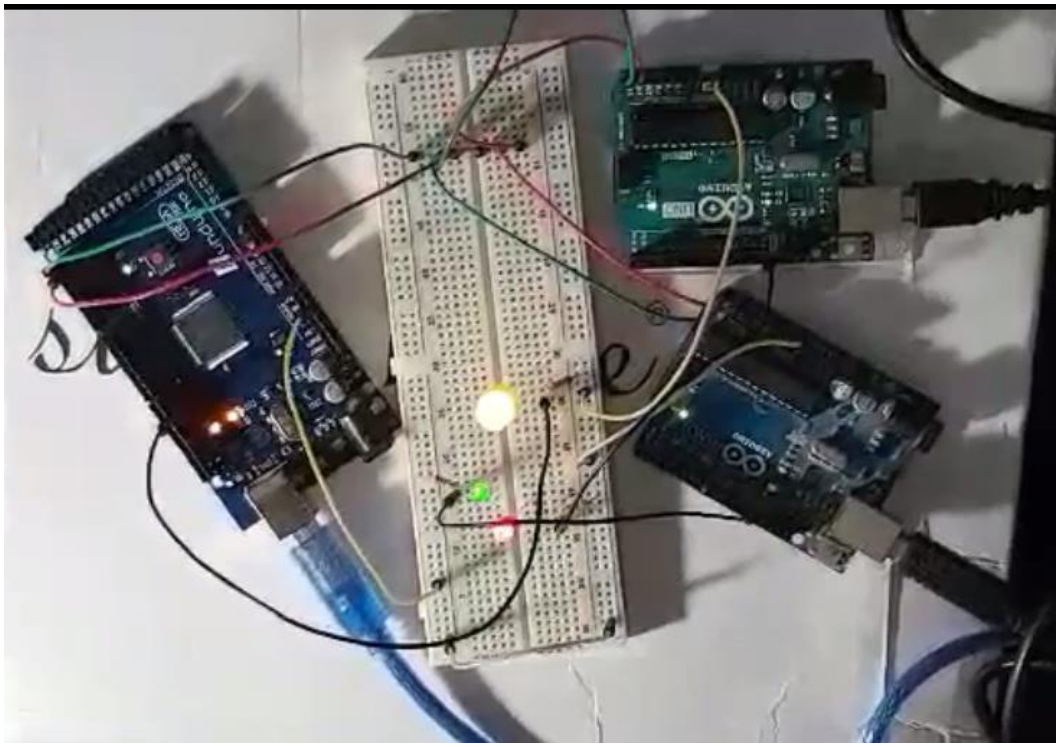
Στην παρακάτω Εικόνα 6-13 παρουσιάζεται η hardware διάταξη της σύνδεσης των 3 Arduino μέσω I2C, κατά την φάση εισόδου στο void loop() του master.



Εικόνα 6-13 Hardware Διάταξη της I2C επικοινωνίας master (ArduinoMega) - 2 slaves (ArduinoUno)

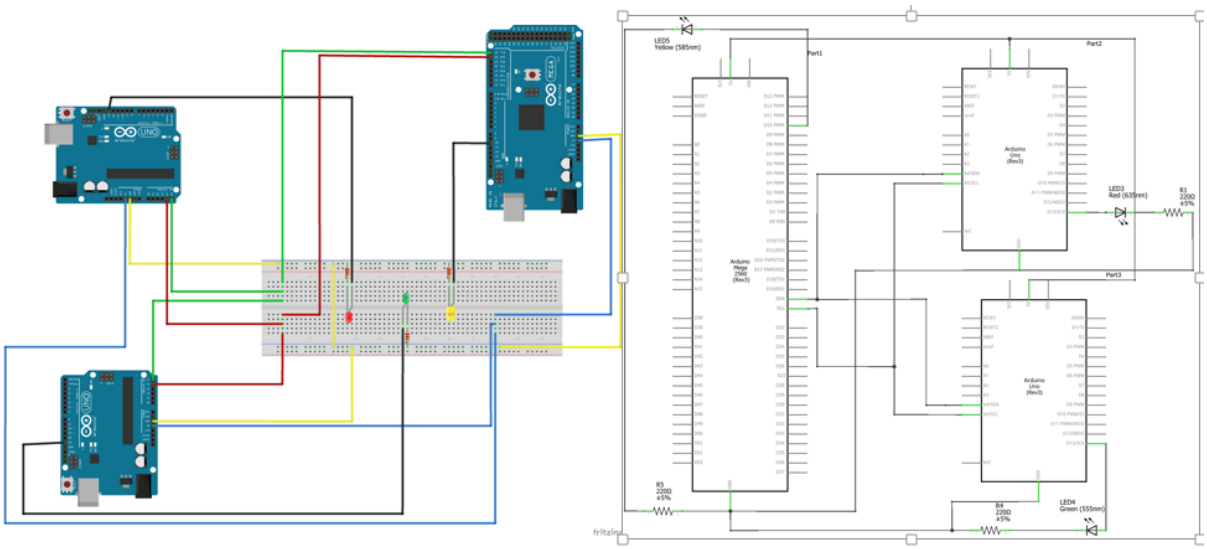
Κατά την εκτέλεση της υλοποίησης της σύνδεσης I2C με 3 Arduinos, παρουσιάστηκε μόνο ένα πρόβλημα. Όταν έτρεχαν οι 3 κώδικες στα 3 Arduino, κατά την διαδικασία δέσμευσης των 2 σταθμών (Robot = 'H' και Mill = 'H') – καθυστέρησης 5 sec – αποδέσμευσης (Robot = 'L' και Mill = 'L'), δεν εφαρμοζόταν καθόλου καθυστέρηση, αλλά τα leds αναβόσβηναν στιγμιαία. Αυτό παρατηρείται στην I2C επικοινωνία στις slave συσκευές, διότι δεν λαμβάνεται υπόψη η εντολή delay() από τις slave συσκευές. Επειδή, η επικοινωνία Master-Slaves γίνεται μέσω συναρτήσεων, οι συναρτήσεις των Wire.onRequest() και Wire.onReceive() είναι ISR (Interrupt Service Routine) και εκτελούνται άμεσα.

Παρόλα αυτά στο Σχήμα 6-14 παρακάτω, παρατηρούνται τα leds των δύο slaves/τοπικών ελεγκτών (κόκκινο και πράσινο) να είναι αναμμένα (HIGH), που σημαίνει πως οι σταθμοί που ελέγχουν οι τοπικοί ελεγκτές είναι δεσμευμένοι, δηλαδή το Mill και το Robot. Επίσης, το led του master/κεντρικού ελεγκτή (μεγάλο κίτρινο) είναι αναμμένο, που ερμηνεύεται πως βρισκόμαστε εντός του loop του master.



Εικόνα 6-14 Δέσμευση και των δύο slaves-τοπικών ελεγκτών

Τέλος, όπως αναφέρθηκε και στο 5ο κεφάλαιο, αφού φορτώσουμε τους κώδικες στα Arduinos μπορούμε να αποσυνδέσουμε τα usb και μέσω τροφοδοσίας από ένα Arduino να παίρνουν και τα άλλα δύο, εκτελώντας τα προγράμματα κανονικά. Η διάταξη και το ηλεκτρονικό σχέδιο της υπόψη υλοποίησης φαίνονται στο παρακάτω Σχήμα 6-5.



Σχήμα 6-5 I2C Επικοινωνία μεταξύ των 3 Arduino με παροχή τάσης 5V - χωρίς σύνδεση USB

6.2 Προσομοίωση Ελέγχου με ArduinoMega2560 – Διακόπτες – Φωτοδιόδους

6.2.1 Αιτίες Υλοποίησης

Ο κεντρικός έλεγχος που εφαρμόστηκε στο προηγούμενο υποκεφάλαιο 6.1, υλοποιούταν με 2 slaves συσκευές, οι οποίοι προσομοίαζαν τοπικούς ελεγκτές ενός ΕΣΚ, ως μια πιο οικονομική λύση. Για να προσομοιωθεί, όμως το ΕΣΚ με 24 σήματα εισόδου/εξόδου που θα λαμβάνει ο κεντρικός ελεγκτής χρησιμοποιήθηκε ένας μικρο-ελεγκτής ArduinoMega2560Rev3 με 24 διακόπτες (switches) και 24 φωτοδίοδους (leds). Δηλαδή, οι τοπικοί ελεγκτές ArduinoUno αντικαταστάθηκαν από διακόπτες. Οι παραδοχές που βασίστηκε αυτή η υλοποίηση είναι οι παρακάτω:

Αντί να ζητά ο κεντρικός ελεγκτής (master) σήματα εισόδου από τους slaves, μέσω της συνάρτησης Wire.requestFrom(), εξετάστηκε η δυνατότητα ο κεντρικός ελεγκτής να διαβάζει μια είσοδο (Input pins) από την τροφοδοσία (+5V) του ίδιου, όπου μεταξύ της τροφοδοσίας και της εισόδου θα παρεμβάλετε ένας διακόπτης 2 θέσεων (On/Off). Η λογική έγκειται στις 2 διαφορετικές καταστάσεις εισόδου που δημιουργούνται από τον διακόπτη και τις οποίες μπορεί να διαβάσει το Arduino-Κεντρικός Ελεγκτής, μέσω της εντολής digitalRead(). Η digitalRead() διαβάζει 2 θέσεις HIGH/LOW, οπότε με κλειστό το διακόπτη περνά τάση και διαβάζει HIGH, ενώ με ανοικτό το διακόπτη δεν περνά τάση και διαβάζει LOW.

Με τη χρήση διακοπτών δύναται ο ελεγκτής να διαβάσει 2 καταστάσεις, το πρόβλημα όμως εντοπίζεται στο πως θα ειδοποιείται ο ελεγκτής να αλλάζει κατάσταση (HIGH/LOW=ON/OFF) στους διακόπτες, καθότι οι διακόπτες πλέον προσομοιάζουν τους τοπικούς ελεγκτές/σταθμούς. Οι τοπικοί ελεγκτές στη σύνδεση I2C, μέσω της συνάρτησης του master Wire.beginTransmission(slave_address) και της συνάρτησης του slave Wire.onReceive() λάμβαναν τα σήματα εξόδου του master, τα οποία αφορούσαν δεσμεύσεις και αποδεσμεύσεις των slaves, αναλόγως των συνθηκών που ικανοποιούνταν στο πρόγραμμα του master. Επίσης, τα σήματα εξόδου του κεντρικού ελεγκτή αποτελούσαν εντολές για την αλλαγή κατάστασης τοπικών ελεγκτών των σταθμών και στην προκειμένη περίπτωση των διακοπτών.

Κατόπιν των παραπάνω, προτιμήθηκε τα σήματα εξόδου να δημιουργούνται μέσω οπτικής επαφής αρχικά και έπειτα ο χρήστης, αναλόγως τις εντολές που θα λαμβάνει/βλέπει για την αλλαγή κατάστασης των διακοπτών να τις αλλάζει χειροκίνητα. Η οπτική επαφή υλοποιήθηκε με 2 τρόπους για περισσότερη αξιοπιστία, μέσω φωτοδιόδων, μια για κάθε διακόπτη, και μέσω της σειριακής παρακολούθησης (serial monitoring). Με τη χρήση αυτών των θεωρήσεων μπορούμε να γράψουμε το πρόγραμμα Arduino, συμπεριλαμβάνοντας και τους 24 σταθμούς του ΕΣΚ στη δομή του, ως προς το κομμάτι της ικανοποίησης συνθηκών και της εκτέλεσης των αντίστοιχων εντολών. Οπότε, υλοποιήθηκε ο κώδικας με τις παραδοχές που αναφέρθηκαν παραπάνω και περιέχει κατά κύριο λόγο σαν if_conditions (κανόνες) τις εισόδους των μεταβάσεων των δικτύων Petri και σαν εντολές εκτέλεσης των if_conditions, τις εξόδους από τις ενεργοποιημένες μεταβάσεις των PN.

Βάσει όλων των παραπάνω υλοποιήθηκε ο κώδικας του Παραρτήματος 9.3, ο οποίος περιέχει όλους τους σταθμούς του ΕΣΚ και υλοποιεί τους κανόνες διακίνησης κουπονιών των δικτύων Petri. Συναφώς, εξάγεται το συμπέρασμα πως η δομή (if_conditions και εκτελέσεις εντολών τους) του κώδικα του Παραρτήματος 9.3 μαζί με τις διαδικασίες μετάδοσης/λήψης σημάτων του κωδίκων του Παραρτήματος 9.2 συνθέτουν την ακριβή υλοποίηση του κεντρικού ελέγχου στο εξεταζόμενο Ευέλικτο Σύστημα Κατεργασιών.

Στον Πίνακα 6-2, παρατηρούμε την αντιστοιχία μεταξύ της υλοποίησης σημάτων ελέγχου (I/O) μέσω I2C και αυτής με Διακόπτες και Leds.

	Arduino I2C		***Arduino Switches and Leds***
	Master	***Slave***	
Inputs στον Κεντρικό Ελεγκτή	Wire.requestFrom(SlaveAd) Val = Wire.read()	Wire.onRequest (requestEvent) Wire.write(val)	digitalRead(switch)
Outputs από τον Κεντρικό Ελεγκτή	Wire.beginTransmission (SlaveAd) Wire.write(val) Wire.endTransmission()	Wire.onReceive (receiveEvent) Val = Wire.read()	digitalWrite(Led) and Serialprint(switch)

Πίνακας 6-2 Πίνακας Αντιστοιχίας Σημάτων I/O μεταξύ των υλοποιήσεων Ελέγχου

Η υλοποίηση της κατασκευής του ελεγκτή με διακόπτες και leds, είχε δυσκολίες στην εφαρμογή λόγω των απαιτήσεων που διαθέτει:

- 24 είσοδοι/διακόπτες και 24 έξοδοι/leds, οπότε απαιτούνται 48 digital I/O pins.
- Ο χρόνος αντίδρασης για την αλλαγή ON/OFF των διακοπτών, καθώς τρέχει ο κώδικας στο void loop().
- Πολλές τοπικές μεταβλητές για την πλήρη αναπαράσταση του δικτύου Petri, συνυπολογίζοντας και τις βοηθητικές μεταβλητές, απαιτούν επαυξημένες δυνατότητες μικρο-ελεγκτή Arduino.
- Απαίτηση πολλών μικρών εξαρτημάτων σε μικρό χώρο.

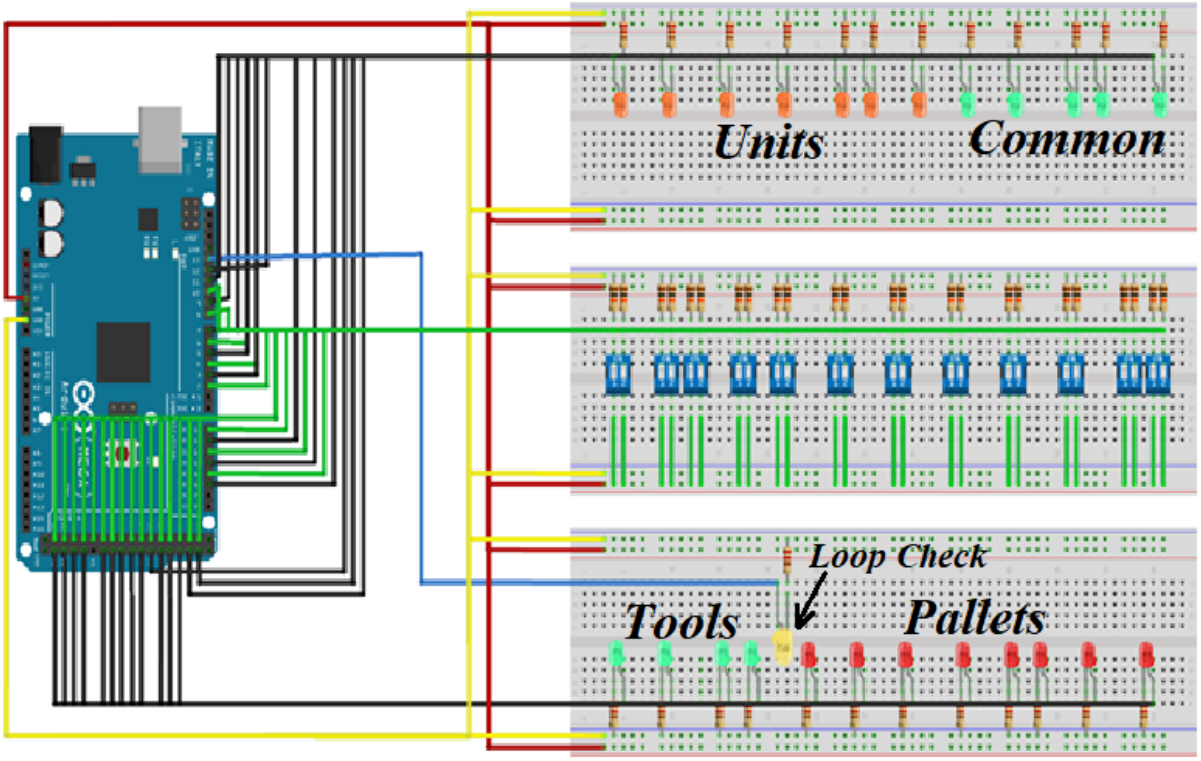
Βάσει των παραπάνω δυσχερειών καταλήξαμε στις παρακάτω λύσεις:

- Χρησιμοποίηση του ArduinoMega2560 ως κεντρικού ελεγκτή με δυνατότητες 54 digital pins I/O και αυξημένες δυνατότητες μνήμης και επεξεργαστή.
- Χρήση ενός μεγάλου και 2 μικρότερων breadboards.
- Χρησιμοποίηση ευρέως του χρονιστή delay() με χρόνο 15 sec, όταν δίνετε εντολή αλλαγής κατάστασης του διακόπτη, το οποίο όμως καθυστερεί την εξέλιξη ολοκλήρωσης του προγράμματος κατά πολύ.
- Ευρύτατη χρησιμοποίηση της σειριακής παρακολούθησης και της εντολής Serial.println(), προκειμένου να γνωρίζει ο χρήστης κατά την προσομοίωση σε ποιο ακριβώς σημείο βρίσκεται.

6.2.2 *Προσομοίωση Ελέγχου – Επεξήγηση Προγράμματος*

Το σχέδιο της δομής υλοποίησης του κεντρικού ελεγκτή ArduinoMega2560 με διακόπτες και leds φαίνεται στο παρακάτω Σχήμα 6-6 και στην Εικόνα 6-15. Για την υλοποίηση του οποίου χρησιμοποιήθηκαν:

- 24 Διακόπτες 2 διακριτών καταστάσεων.
- 24 Leds χρωματισμού ανάλογου της ομάδας που ανήκουν, όπως αυτές ορίστηκαν στο 2ο Κεφάλαιο.
- 24 Αντιστάσεις 220 Ω για την προστασία κάθε led.
- 24 Pull-down αντιστάσεις 10 ΚΩ για τους διακόπτες, ώστε να μην κάνουν floating (όταν είναι ανοικτός ο διακόπτης, χωρίς pull-down resistor δεν διαβάζει ούτε LOW, ούτε HIGH).
- 1 Led μεγάλο κίτρινο, που είναι HIGH μόλις μπαίνει στο loop και γίνεται LOW στο τέλος του loop.



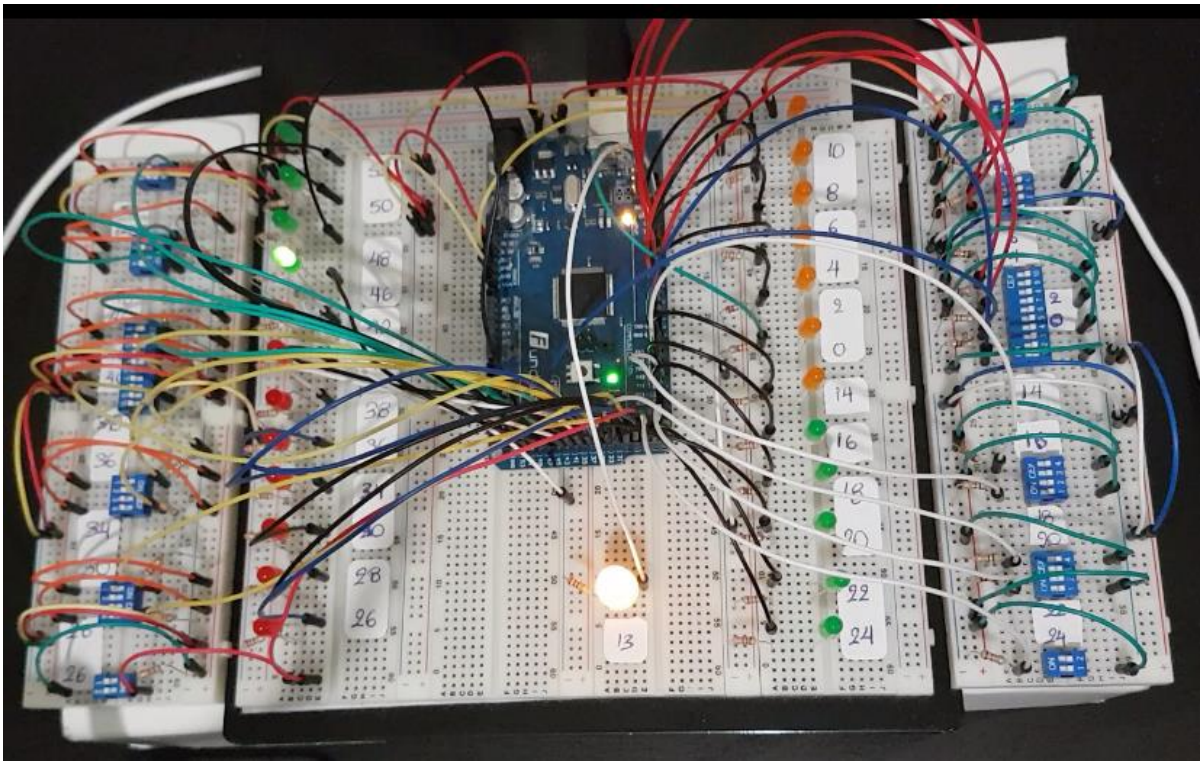
Σχήμα 6-6 Δομή της υλοποίησης Ελέγχου με ένα Κεντρικό Ελεγκτή - Διακόπτες - Leds

Για τον κώδικα του Παρατήματος 9.3, επειδή είναι αρκετά μεγάλος, θα επεξηγηθούν ορισμένα μέρη του και αντίστοιχες προσομοιώσεις που έγιναν. Εξάλλου, όπως και στα δίκτυα Petri οι διεργασίες των παλετών και των τεμαχίων είναι πανομοιότυπες, οπότε θα γίνει μια περιληπτική ανάλυση του βασικού μέρους του κώδικα void loop().

Αρχικά, κάνουμε αρχικοποιήσεις σε 3 είδη μεταβλητών:

- Στους διακόπτες/σταθμούς που αποτελούν τις εισόδους, τους οποίους τοποθετούμε σε αντίστοιχα Pins εισόδου, βάσει του Πίνακα 6-3.
- Τις μεταβλητές που θα διαβάζουν τους σταθμούς των διακοπτών των σταθμών, μέσω της εντολής digitalRead() και αρχικά τους θέτουμε ίσους με 0. Έχουν το όνομα του σταθμού/εισόδου με προσθετικό μπροστά το switch-.
- Στα Leds (φωτοδίοδοι) που αποτελούν τις εξόδους, τα οποία τοποθετούμε σε αντίστοιχα Pins εξόδου, έχουν ίδιο όνομα με το αυτό του σταθμού/εισόδου με προσθετικό μπροστά το Led-.

Στον Πίνακα 6-3 και στην Εικόνα 6-15 παρουσιάζονται οι θέσεις εισόδων/διακοπτών που φαίνονται στο Σχήμα 6-6 σε χρωματισμό ανάλογο του χρώματος του led της ομάδας που ανήκουν. Οι θέσεις εξόδων/leds έχουν οριστεί στο πρόγραμμα σε pins εξόδου και τα οποία αντιστοιχούν σε κάθε είσοδο/διακόπτη, όπου κάθε Led είναι δίπλα ακριβώς από τον διακόπτη στον οποίο αναφέρεται και έχει τον ίδιο αριθμό πάνω στην πλακέτα (Εικόνα 6-15 και 6-16).



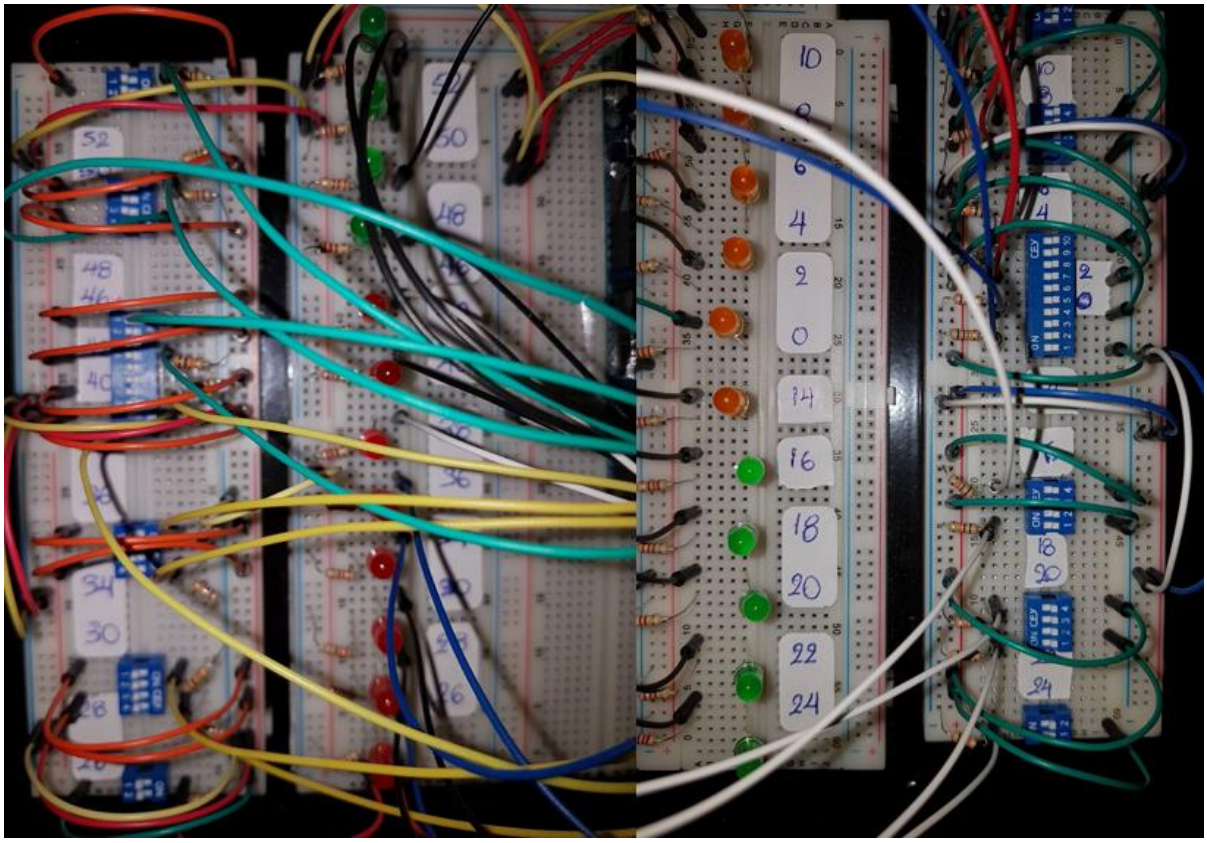
Εικόνα 6-15 Hardware της υλοποίησης Ελέγχου με ένα Κεντρικό Ελεγκτή - Διακόπτες – Leds

Ο Πίνακας 6-3 σε συνδυασμό με την Εικόνα 6-16 (πιο ευκρινής από την 6-15) είναι ότι χρειάζεται ο χρήστης, κατά την προσομοίωση του προγράμματος προκειμένου να αλλάζει κατάσταση στον διακόπτη (ON/OFF), αναλόγως την οδηγία που θα παίρνει από το παράθυρο της σειριακής και την αλλαγή κατάστασης του Led (HIGH/LOW). Η κατάσταση default, είναι αυτή που πρέπει να βρίσκονται οι διακόπτες στην αρχή της προσομοίωσης.

Α/Α	*Είσοδοι – Διακόπτες (Σταθμοί)*	*Αριθμός Pin*	*Είδος*	*Κατάσταση default*
1	Tool_Millω(μεταβλητό)	52	Εργαλεία M	OFF
2	BufferTool_Mill	50	-//-	ON
3	Tool_Lathe (μεταβλητό)	48	Εργαλεία L	OFF
4	BufferTool_Lathe	46	-//-	ON
5	Pallet_station(μεταβλητό)	42	Παλέτες	OFF
6	Pallet_Storage	40	-//-	OFF
7	Pallet_BufferMillIN	38	-//-	OFF
8	Pallet_BufferMillOUT	36	-//-	OFF
9	Wrong_Pallet	34	-//-	OFF
10	Pallet_BufferLatheIN	30	-//-	OFF
11	Pallet_BufferLatheOUT	28	-//-	OFF
12	Pallet_Finish	26	-//-	OFF

Α/Α	*Είσοδοι – Διακόπτες (Σταθμοί)*	*Αριθμός Pin*	*Είδος*	*Κατάσταση default*
13	Unit_Storage (μεταβλητό)	10	Τεμάχια	OFF
14	Unit_BufferMillIN	8	-//-	OFF
15	Unit_BufferMillOUT	6	-//-	OFF
16	Wrong_Unit	4	-//-	OFF
17	Unit_BufferLatheIN	2	-//-	OFF
18	Unit_BufferLatheOUT	0 (44)	-//-	OFF
19	Unit_Finish	14	-//-	OFF
20	Transport	24	Κοινό	OFF
21	Robot	22	-//-	OFF
22	ID_pos	20 (32)	-//-	OFF
23	Mill	18	-//-	OFF
24	Lathe	16	-//-	OFF

Πίνακας 6-3 Αντιστοιχία Διακοπτών/Εισόδων (Σταθμοί) σε Pins του Arduino



Εικόνα 6-16 Αντιστοίχιση Διακοπτών/Εισόδου σε Pins Εισόδου και Leds

Έπειτα, στο πρόγραμμα ορίζονται κάποιες *βοηθητικές μεταβλητές*, οι οποίες επιτρέπουν την ορθή συνέχεια των εργασιών που εκτελούνται ανά είδος εργαλείο/παλέτα/τεμάχιο. Αυτές αναπαριστούν τις διεργασίες που γίνονται κατά την εκτέλεση του προγράμματος και είναι ίσες με 0 αν η διεργασία δεν έχει πραγματοποιηθεί ή ίσες με 1 αν πραγματοποιείται. Στην ουσία, μπαίνουν σαν συνθήκη στην δομή επιλογής κάθε επόμενης διεργασίας, προκειμένου να εξασφαλίσουν ότι έγινε η προηγούμενη διεργασία και να διατηρήσουν την ορθή σειρά πραγματοποίησης των διεργασιών (Εικόνα 6-17). Από αυτές οι πιο

σημαντικές υπογραμμίζονται στην Εικόνα 6-17, καθότι συμμετέχουν σε αρκετές συνθήκες των δομών επιλογής.

```
// Initialize Proceedings - Processings (Places of the Petri Net)
int uploadingTool = 0; // NO uploading
int downloadingTool = 0; // NO downloading
int uploadingPallettoStorage = 0; // NO uploading
int downloadingPallettoStorage = 0; // NO downloading
int uploadingPallettoBufferMiillIN = 0; // NO uploading
int downloadingPallettoBufferMiillIN = 0; // NO downloading
int pallets = 1; // The pallets that are produced in the Pallet_station
int counterPallet = 1; // For making the Pallet_Station = High, when there are no more Pallets
int uploadingPallettoIDpos = 0; // NO uploading
int downloadingPallettoMill = 0; // NO downloading
int Milling = 0; // NO Process
int HoldOnPalletMilling = 0;
int uploadingPallettoBufferMillOUT = 0; // NO uploading
int downloadingPallettoBufferLatheIN = 0; // NO downloading
int uploadingPallettoLathe = 0; // NO uploading
int Turning = 0; // NO Process
int HoldOnPalletTurning = 0;
int downloadingPallettoBufferLatheOUT = 0; // NO downloading
int uploadingPallettoFinish = 0; // NO uploading
int downloadingPallettoFinish = 0; // NO downloading
int endofPallets = 0; // Begin Uints' Processing
int enterUnits = 1; // For Entering Units and not entering again
int units = 1; // The units that are loaded in the Unit_Storage
int counterUnit = 1; // For making the Unit_Storage = High, when there are no more Units
int uploadingUnittoBufferMillIN = 0; // NO uploading
```

Εικόνα 6-17 Βοηθητικές Μεταβλητές εξασφάλισης ορθής συνέχειας των Διεργασιών

Ύστερα, μπαίνουμε στο void setup(), όπου ανοίγουμε τη σειριακή θύρα, μέσω της εντολής: Serial.begin(9600), την οποία χρειαζόμαστε προκειμένου να εμφανίζονται οι εκτυπώσεις στο παράθυρο της σειριακής και οι οποίες αποτελούν εντολές ενεργειών προς το χρήστη να αλλάξει κατάσταση τους διακόπτες παράλληλα και με το άναψε/σβήσε των leds. Έπειτα, ορίζονται ποια pins θα είναι είσοδοι/έξοδοι μέσω της εντολής:

pinMode(αριθμός pin= διακόπτης/led, INPUT/OUTPUT)

δηλαδή τους διακόπτες και τα leds αντίστοιχα (Εικόνα 6-18). Πριν την έξοδο από το void set up(), ορίζουμε τις μόνους διακόπτες που πρέπει από default να είναι ON, πριν την έναρξη του προγράμματος, το οποίο είχε καθοριστεί από τη δομή του δικτύου Petri και γι' αυτό θέτουμε τα leds τους HIGH και τους διακόπτες ON. Αυτοί είναι διακόπτες των BufferToolMill και BufferToolLathe, διότι κατά την έναρξη της προσομοίωσης του PN, δε διαθέτουν κουπόνι.

```
void setup() {
  // For Printing Commands in Serial Monitorin
  Serial.begin(9600);

  // put your setup code here, to run once:
  // Define the pinModes as INPUT - OUTPUT
  pinMode(Tool_Mill,INPUT);
  pinMode(LedTool_Mill,OUTPUT);
  pinMode(Tool_Lathe,INPUT);
  pinMode(LedTool_Lathe,OUTPUT);
  pinMode(LedMill,OUTPUT);
  pinMode(Lathe,INPUT);
  pinMode(LedLathe,OUTPUT);

  pinMode(13,OUTPUT);

  // Initialize BufferTools Mill & Lathe, t
  digitalWrite(LedBufferTool_Mill, HIGH);
  digitalWrite(LedBufferTool_Lathe, HIGH);
  delay (10000);
```

Εικόνα 6-18 Επεξήγηση Προγράμματος void setup()

Στη δομή επανάληψης void loop() ορίζεται η λειτουργία όλου του κώδικα, σε αυτήν υπάρχουν όλες οι ενέργειες που δύναται να γίνουν στο ΕΣΚ και είναι δομημένες σε μια σειρά. Η λογική του βρόχου είναι:

- Πρώτα διαβάζονται όλες οι είσοδοι/διακόπτες και καταχωρούνται στις μεταβλητές switch- με το ίδιο όνομα των εισόδων, μέσω της εντολής digitalRead(), της μορφής:

switchStation = digitalRead(Station), όπου Station είναι οι Είσοδοι/Διακόπτες

- Έπειτα, όλες οι διεργασίες που επιτελούνται στο ΕΣΚ είναι εντός δομών επιλογών (if conditions), και είναι χωρισμένες σε 3 ιεραρχικά μέρη, πρώτα οι διεργασίες για τα εργαλεία, μετά για παλέτες και στο τέλος για τα units, όπως και στο PN η σειρά εκτέλεσης των ενεργειών είναι η ίδια. Οι διακόπτες/σταθμοί (Πίνακας 6-3) και οι βοηθητικές μεταβλητές (Εικόνα 6-17) αποτελούν τις θέσεις στο υλοποιημένο δίκτυο Petri του 4ου Κεφαλαίου και είναι οι συνθήκες που πρέπει να ικανοποιούνται προκειμένου να εκτελούνται οι εντολές εντός των δομών επιλογής.
- Επιπλέον, τηρείται μια γενική αρχή για την κατάσταση των διακοπτών (ON/OFF) και των leds (HIGH/LOW), με βάση το υλοποιημένο δίκτυο Petri:

«Αν η θέση έχει κουπόνι Token = 1, τότε το Led = LOW και switch = OFF,

ανεξαρτήτως αν το κουπόνι σημαίνει διαθέσιμος ή δεσμευμένος ή αριθμό κάποιου είδους.»

- Οπότε, κάθε φορά που εισέρχεται το πρόγραμμα στο void loop(), διαβάζει τις εισόδους/διακόπτες, των οποίων η κατάσταση αποθηκεύεται στις μεταβλητές switch-Σταθμός.
- Έπειτα, διατρέχονται οι δομές επιλογής (if conditions), όπου όποιες ικανοποιούν τις συνθήκες, εισέρχεται και εκτελεί τις εντολές που βρίσκονται εντός τους,
- Οι εντολές εντός των if conditions, ουσιαστικά δίνουν εντολή στον χρήστη να αλλάξει την κατάσταση των διακοπτών (ON/OFF), αλλάζοντας τα leds (HIGH/LOW) και εκτυπώνοντας αντίστοιχη εντολή στο σειριακό παράθυρο.
- Επίσης, εκτυπώνονται στο σειριακό παράθυρο, οι διεργασίες που πραγματοποιήθηκαν, πέραν των εντολών προς το χρήστη.
- Άρα, ο χρήστης βλέποντας το led να ανάβει (HIGH) ή να σβήνει (LOW) και διαβάζοντας το εκτυπωμένο μήνυμα στη σειριακή για αλλαγή κατάστασης του διακόπτη (ON/OFF) έχει 15 sec να αλλάξει τον διακόπτη, μέσω της εντολής delay(15000), που υπάρχει μετά από κάθε εντολή αλλαγής κατάστασης διακόπτη, προκειμένου να έχει τον απαραίτητο χρόνο να αλλάξει τον εκάστοτε διακόπτη.

Μοντελοποιώντας τον Πίνακα 6-3, με βάση το PN οι μόνοι διακόπτες/είσοδοι που μπορούμε να αλλάξουμε την αρχική σήμανση (ON/OFF) είναι 4: οι ToolMill, ToolLathe, Pallet Station και Unit Storage. Καθώς, οι άλλες μεταβλητές μοντελοποιούν θέσεις που ήταν αποθήκες ή πόροι, όπου έχοντας κουπόνι δήλωναν διαθεσιμότητα ή μη και είναι καθορισμένες.

Επιπλέον, μπορούμε να παρέμβουμε και στις βοηθητικές μεταβλητές *pallets* και *units*, στις οποίες ορίζουμε τον αριθμό προς κατεργασία παλετών και τεμαχίων αντίστοιχα (Εικόνα 6-17).

Οι άλλες υπογραμμισμένες μεταβλητές της Εικόνας 6-17, λειτουργούν όπως οι αποτρεπτικοί κλάδοι στα PN. Μπαίνουν στις συνθήκες των δομών επιλογής, αναλόγως την διεργασία, της οποίας την συνέχιση θέλουν να διακόψουν. Και ειδικότερα:

- *counterPallet = 1*, είναι ένας μετρητής, όπου όταν γίνει 0, δίνει εντολή να γίνουν τα παρακάτω:

LedPallet_station = High

switch42=ON, όπου Pallet_station = 42

όταν δεν υπάρχουν άλλες παλέτες για επεξεργασία, δηλαδή όταν pallets = counterPallet (Εικόνα 6-19).

```
// Pallet_Station = HIGH when counterPallet = Pallets
if (counterPallet == pallets) {
  digitalWrite(LedPallet_station, HIGH);
  Serial.println("Pallet_station has no more pallets (Token=0 - switch42=ON) ");
  delay (15000);
}
// Increase counterPallet by one in each loop
counterPallet = counterPallet + 1;
```

Εικόνα 6-19 Λειτουργία του countePallet

- *counterUnit = 1*, αντίστοιχα για τα τεμάχια (Εικόνα 6-20), όπως με το counterPallet, δίνει εντολή:

LedUnit_Storage = High

Switch10=ON, όπου Unit_Storage = 10

```
// Unit_Storage = HIGH when counterUnit = units
if (counterUnit == units) {
  digitalWrite(LedUnit_Storage, HIGH);
  Serial.println("Unit_Storage has no more units (Token=0 - switch10=ON) ");
  delay (15000);
}
// Increase counterUnit by one in each loop
counterUnit = counterUnit + 1;
```

Εικόνα 6-20 Λειτουργία του counterUnit

- Τα endofPallets = 1 και enterUnits = 0, είναι για να εξασφαλίσουν πως όταν τέλειωσε η επεξεργασία όλων των παλετών, το πρόγραμμα θα εισέρχεται μόνο στις συνθήκες για διεργασίες τεμαχίων (Εικόνα 6-21).

```
/*C. Start of Units' Processing
   If there are not any Tools for transport, and Pallets for Processing, go on with UNITS*/
// Transport Unit from Unit_Storage to Unit_BufferMillIN until units. Pallet_station,
// Tool_Mill and Tool_Lathe simulate the deterant branches
if (switchTool_Mill == HIGH && switchTool_Lathe == HIGH && switchPallet_station == HIGH && enterUnits == 1){
  Serial.println("Begin Units Processing!!! No Tools for Transportation nor Pallets for Processing!");
  delay (10000);
  endofPallets = 1;
  enterUnits = 0; // For not entering again in this if_condition!!!
}
delay (5000);

// Transport Unit from Unit_Storage to Unit_BufferMillIN
if (switchUnit_Storage == LOW && switchTransport == LOW && switchUnit_BufferMillIN == LOW && endofPallets == 1){
  Serial.println("Unit is uploaded from Unit_Storage to Transport");
  delay (10000);
  uploadingUnittoBufferMillIN = 1;
```

Εικόνα 6-21Λειτουργία των endofPallets και enterUnits

Τέλος, σημειώνεται πως και εδώ υπάρχει led, για τον έλεγχο εισόδου/εξόδου από τον βρόχο του void loop(), το οποίο είναι συνδεδεμένο στο pin13 του ArduinoMega2560, και είναι σε θέση HIGH όταν εισέρχεται το πρόγραμμα στο βρόχο, και αλλάζει σε θέση LOW, όταν εξέρχεται του βρόχου.

Έλεγχος Λάθος Παλέτας

Μία ακόμη σημαντική σημείωση για το πρόγραμμα αποτελεί, η διάγνωση του λανθασμέν-ης/ου παλέτας/τεμαχίου, για το οποίο δίνεται η δυνατότητα στο χρήστη να επιλέγει αν είναι ορθό ή λανθασμένο, μέσω του διακόπτη *ID_pos = 20*. Έτσι, για την αναγνώριση λάθος παλέτας (το ίδιο ακριβώς γίνεται και για λανθασμένο τεμάχιο) έχουμε:

- Το πρόγραμμα βρίσκεται εντός της συνθήκης κατά την οποία μεταφέρεται παλέτα από το Pallet_BufferMiilIn στο ID_pos για έλεγχο ταυτοποίησης, μέσω του robot, οπότε εκτελούνται οι εντολές της Εικόνας 6-22. Όπου δεσμεύεται το Robot (switch22 = ON και LedRobot = HIGH), ενώ αποδεσμεύεται το Pallet_BufferMiilIn (switch38 = OFF και LedPallet_BufferMiilIn = LOW).

```
// Transport Pallet from PalletBufferMillIN to Pallet_CheckID
if (downloadingPallettoBufferMiillIN == 1 && switchMill == LOW && switchRobot == LOW && switchPallet_BufferMillIN == HIGH){
  downloadingPallettoBufferMiillIN = 0;
  Serial.println("Pallet is uploaded from PalletBufferMillIN to ID_pos by Robot for: --- ID CHECK PROCESSING!!!---");
  delay (10000);
  // Engage Robot for transfering the pallet to CheckID
  digitalWrite(LedRobot, HIGH);
  Serial.println("Robot is engaged (Token=0 - switch22=ON)");
  delay (15000);
  digitalWrite(LedPallet_BufferMillIN, LOW);
  Serial.println("Pallet_BufferMillIN is available (Token=1 - switch38=OFF)");
  delay (15000);
```

Εικόνα 6-22 Εισαγωγή στη διεργασία μεταφοράς παλέτας από Pallet_BufferMillIn στο ID_pos

- Έπειτα, προχωρούμε στον έλεγχο της παλέτας (Εικόνα 6-23), όπου εδώ εκτυπώνεται στη σειριακή μήνυμα προκειμένου να επιλέξει ο χρήστης αν θα το θέσει ως λανθασμένο (switch20 = ON) ή ως σωστό (switch20 = OFF) και μετά από 15,000 sec διαβάζει την εντολή που του δώσαμε. Ύστερα, εισέρχεται στην if condition, αν είναι λανθασμένο, και δηλώνει μέσω του LedID_pos = HIGH και αντίστοιχου μηνύματος στη σειριακή, πως είναι λανθασμένο και πως πρέπει να μεταβεί στο Wrong_pallet. Μετέπειτα, γίνεται ένας έλεγχος, μέσω της δομής επανάληψης while, αν είναι διαθέσιμη η αποθήκη λάθος παλετών, από την οποία βγαίνει μόνο όταν το switch34 = OFF, όπου βάσει του Πίνακα 6-3 Wrong_Pallet = 34.

```
/* START OF PALLET'S ID CHECK PROCESSING*/
// This message will not be seen in the regular code, just here to decide the ID !!!
Serial.println("CHANGE ID_pos, switch20=ON for Wrong Pallet OR switch20=OFF for Correct Pallet");
delay (15000);
switchID_pos = digitalRead(ID_pos);
if (switchID_pos == HIGH){
  digitalWrite(LedID_pos, HIGH);
  Serial.println("The Pallet's ID is WRONG, so switch20(ID_pos)=ON !!!");
  Serial.println("MOVE THE PALLET TO BUFFER WRONG_PALLET");
  delay (10000);
  switchWrong_Pallet = digitalRead(Wrong_Pallet);
  while (switchWrong_Pallet == HIGH){
    digitalWrite(LedWrong_Pallet, HIGH);
    Serial.println("Wrong_Pallet is engaged, release it rapidly by turning switch34=OFF and LedWrong_Pallet=LOW");
    delay (10000);
    switchWrong_Pallet = digitalRead(Wrong_Pallet);
  }
```

Εικόνα 6-23 Έλεγχος λάθος παλέτας (1)

- Έπειτα, και ενώ είμαστε ακόμα εντός της επιλογής if (switchID_pos ==HIGH), αν το switchWrong_Pallet == LOW εισερχόμαστε στην if condition, όπου μεταφέρεται η λάθος παλέτα στο Wrong_Pallet, ελευθερώνεται η θέση ID_pos και το Robot, ενώ δεσμεύεται η αποθήκη Wrong_Pallet (switch34 = ON) και μετά από 15 sec ελευθερώνεται θεωρώντας πως απομακρύνθηκε από την αποθήκη λανθασμένων παλετών (Εικόνα 6-24).

```
    delay (10000);
    switchWrong_Pallet = digitalRead(Wrong_Pallet);
  }
  if (switchWrong_Pallet == LOW){
    digitalWrite(LedWrong_Pallet, LOW);
    Serial.println("Transfer the pallet to Wrong_Pallet");
    delay(10000);
    // Free ID_pos and Robot
    digitalWrite(LedID_pos, LOW);
    Serial.println("The wrong id pallet has been shifted, so ID_pos=LOW - switch20=OFF !!!");
    delay (15000);
    digitalWrite(LedRobot, LOW);
    Serial.println("Robot is available (Token=1 - switch22=OFF)");
    delay (15000);
    // Engage Wrong_Pallet
    digitalWrite(LedWrong_Pallet,HIGH);
    Serial.println("Wrong_Pallet is engaged (Token=0 - switch34=ON), Release it !!!");
    delay (10000); // Here we remain 10 sec and release the wrong pallet! Not in Reality !!!
    digitalWrite(LedWrong_Pallet,LOW);
    Serial.println("OK! Wrong_Pallet has been released !!!");
    Serial.println("Wrong_Pallet is available (Token=1 - switch34=OFF)");
    delay (10000);
  }
```

Εικόνα 6-24 Έλεγχος λάθος παλέτας (2)

- Τέλος, αν *switchID_pos == HIGH*, τότε εκτελούνται οι εντολές εντός του else, όπου μας ενημερώνει πως αποδεσμεύεται το Robot (switch20 = OFF), δεσμεύεται το Mill (switch18 = ON) και βγαίνει από τη συνθήκη. Στο τέλος, εκτυπώνεται μήνυμα όπου μας ενημερώνει ότι πραγματοποιήθηκε ο έλεγχος των παλετών, που ανήκει στην αρχική if condition, όταν εισήλθε το πρόγραμμα για την μεταφορά του pallet από το Pallet_BufferMiilIn στο CheckID_pos (Εικόνα 6-25).

```
  else {
    uploadingPallettoIDpos = 1; // --SOS-- Only when the pallet's ID is correct, we go on the procedure !!!!
    Serial.println("THE PALLET IS CORRECT (so ID-pos switch20=OFF), GO ON!!!");
    delay (10000);
    // Only when pallet's id is correct, we engage Mill and Robot remains engaged
    digitalWrite(LedMill, HIGH);
    Serial.println("Mill is engaged (Token=0 - switch18=ON");
    delay (15000);
  }
  Serial.println("END OF PALLET'S ID CHECK");
}/* END OF PALLET'S ID CHECK PROCESSING*/
```

Εικόνα 6-25 Έλεγχος λάθος παλέτας (3)

Προσομοίωση Λειτουργίας Προγράμματος

Για να γίνουν κατανοητά τα παρακάτω ακολουθεί μια προσομοίωση για ενεργοποίηση της 1ης συνθήκης που αφορά μεταφορά εργαλείου Mill, με αρχική σήμανση αυτή του Πίνακα 6-3:

Αρχικά, εισερχόμαστε στο void loop() και ανοίγει το Led13,που είναι για τον έλεγχο εισόδου στο βρόχο. Έπειτα, διαβάζονται όλες οι είσοδοι/διακόπτες. Κατόπιν, εκτυπώνεται μήνυμα στη σειριακή πως διαβάστηκαν όλες οι είσοδοι. Ενώ, υπάρχει και σε σχόλια η παρακάτω αρχή αντιστοίχισης (Εικόνα 6-26):

Κουπόνι (Token) = 1 => Led = LOW => switch = OFF

Κουπόνι (Token) = 0 => Led = HIGH => switch = ON

```
void loop() {
  // put your main code here, to run repeatedly:
  // Led for checking the loops
  digitalWrite(13, HIGH);
  delay (5000);

  //********** Read the signal in all Inputs **********
  switchTool_Mill = digitalRead(Tool_Mill); // default is OFF = LED LOW (1 token)
  switchTool_Lathe = digitalRead(Tool_Lathe); // default is OFF = LED LOW (1 token)
  switchBufferTool_Mill = digitalRead(BufferTool_Mill); // default is ON = LED HIGH (0 token)
  switchBufferTool_Lathe = digitalRead(BufferTool_Lathe); // default is ON = LED HIGH (0 token)

  switchPallet_station = digitalRead(Pallet_station); // default is OFF = LED LOW (1 token)
  switchPallet_Storage = digitalRead(Pallet_Storage); // default is OFF = LED LOW (1 token)
  switchPallet_BufferMillIN = digitalRead(Pallet_BufferMillIN); // default is OFF = LED LOW (1 token)
  switchPallet_BufferMillOUT = digitalRead(Pallet_BufferMillOUT); // default is OFF = LED LOW (1 token)
  // switchWrong_Pallet = digitalRead(Wrong_Pallet); // EXCEPTION!!! by default is OFF = LED LOW (1 toke
  switchPallet_BufferLatheIN = digitalRead(Pallet_BufferLatheIN);
  switchPallet_BufferLatheOUT = digitalRead(Pallet_BufferLatheOUT);
  switchPallet_Finish = digitalRead(Pallet_Finish);

  switchUnit_Storage = digitalRead(Unit_Storage);
  switchUnit_BufferMillIN = digitalRead(Unit_BufferMillIN);
  switchUnit_BufferMillOUT = digitalRead(Unit_BufferMillOUT);
switchMill = digitalRead(Mill);
switchLathe = digitalRead(Lathe);
//**********End of Reading the signal in Inputs**********
Serial.println("--------The contoller read all the Inputs--------");
delay (10000);

/* In general, LOW means that the Place has token = 1, Independently if it's engaged or available
Token = 1 means LED = LOW and Switch = OFF ----- Otherwise Token = 0 means LED = HIGH and Switch = ON*/
```

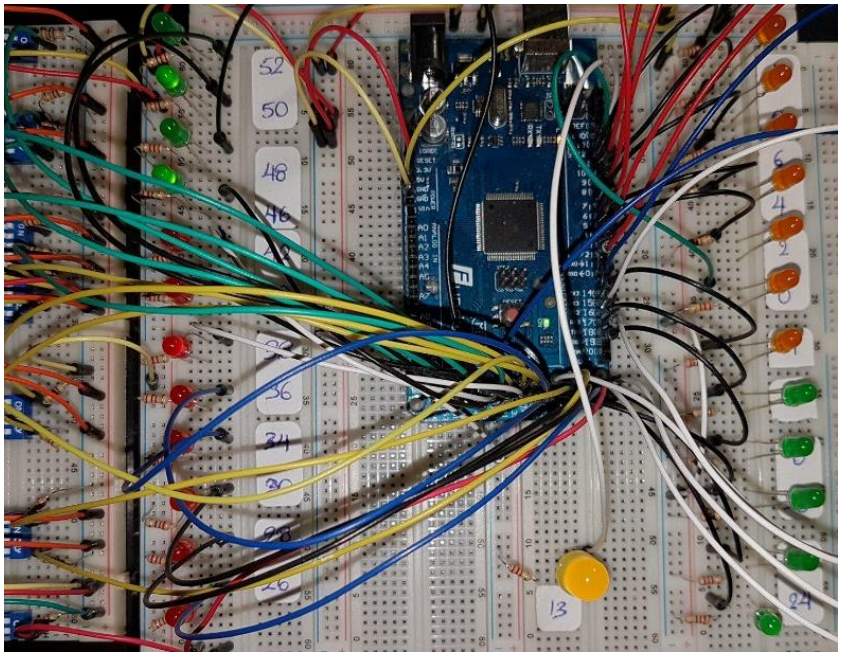
Εικόνα 6-26 Εισαγωγή Προγράμματος στο void loop()- Προσομοίωση Προγράμματος

Μετέπειτα βάσει της Εικόνας 6-27, εισερχόμαστε στην if condition για την μεταφορά εργαλείου Mill από Tool_Mill στην ενδιάμεση αποθήκη BufferTool_Mill, προκειμένου να φορτωθούν στη Mill μηχανή. Βάσει της αρχικής σήμανσης η συνθήκη ικανοποιείται, οπότε θα εκτελεστούν οι εντολές που περιλαμβάνει.

```
/* A. Loading Tools to Buffer_Tools with the Transport*/
// Loading Tool_Mill to Mill Machine
if (switchTool_Mill == LOW && switchBufferTool_Mill == HIGH && switchTransport == LOW){
  digitalWrite(LedBufferTool_Mill, LOW); // Available BufferTool_Mill, by default is LED = HIGH (token=
  Serial.println("BufferTool_Mill (Token=1 - switch50=OFF) is available");
  delay (15000);
  Serial.println("Tool_Mill is uploaded to Transport");
  delay (10000);
  uploadingTool = 1;
  digitalWrite(LedTool_Mill, HIGH);
  Serial.println("Tool_Mill is empty (Token=0 - switch52=ON) ");
  delay (15000);
  digitalWrite(LedTransport, HIGH);
  Serial.println("Transport is engaged (Token=0 - switch24=ON) ");
  delay (15000);
  // Read again BufferTool_Mill and Transport
  switchBufferTool_Mill = digitalRead(BufferTool_Mill);
  switchTransport = digitalRead(Transport);
    if (uploadingTool == 1 && switchBufferTool_Mill == LOW && switchTransport == HIGH){
      Serial.println("Tool_Mill is downloaded to BufferTool_Mill");
      delay (10000);
      downloadingTool = 1;
      uploadingTool = 0;
      digitalWrite(LedBufferTool_Mill, HIGH);
      Serial.println("BufferTool_Mill (Token=0 - switch50=ON) is engaged");
      delay (15000);
      downloadingTool = 0;
      // Free the Transport
      digitalWrite(LedTransport, LOW);
      Serial.println("Transport is available (Token=1 - switch24=OFF) ");
      delay (15000);
    }
  }
  else {
    Serial.println("BufferTool_Mill is LOW or Transport is engaged or Tool_Mill is empty of Tools");
  }
  delay (5000);
  // End of Loading Tool_Mill to Mill Machine
```

Εικόνα 6-27 Ικανοποίηση 1ης Συνθήκης – Προσομοίωση Προγράμματος

Ενώ, οι ενδείξεις led στο hardware του ελεγκτή φαίνονται στην παρακάτω Εικόνα 6-28.



Εικόνα 6-28 Προσομοίωση Ελεγκτή (1)

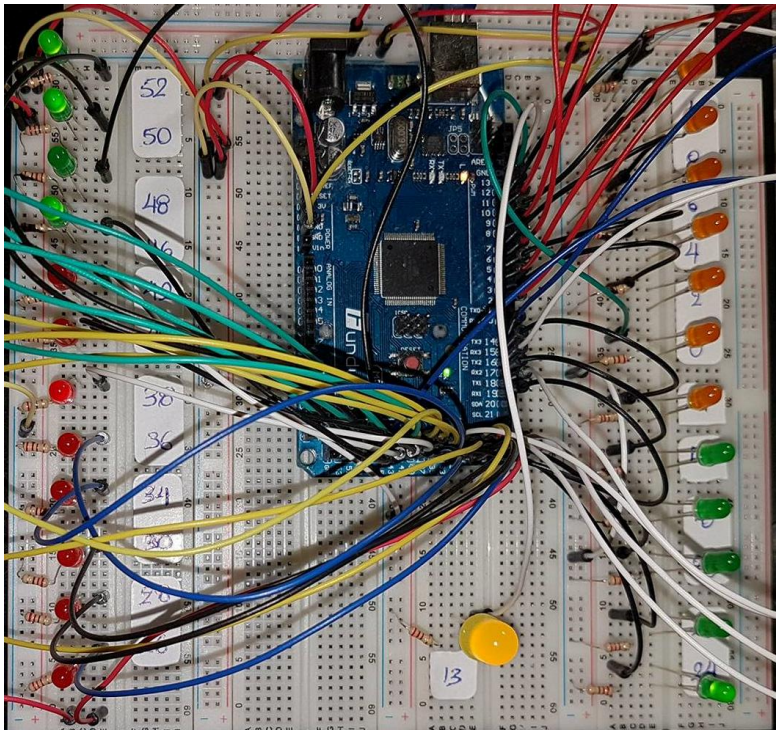
Επίσης, παρατηρούμε τα παρακάτω στην hardware εγκατάσταση και στο σειριακό παράθυρο του Η/Υ:

- Βλέπουμε το Led BufferTool_Mill = LOW και παίρνουμε μήνυμα switch50 = OFF (Αποδέσμευση).
- Παίρνουμε μήνυμα ότι φορτώνεται το Tool_Mill στο Transport
- Η βοηθητική μεταβλητή uploadingTool = 1
- Βλέπουμε LedTool_Mill = HIGH και παίρνουμε μήνυμα switch52 = ON, διότι Token = 0 (Διάθεση κουπονιών = 0 πλέον). Οπότε, θα παραμείνουν ο διακόπτης στο ON και το Led στο HIGH.
- Βλέπουμε LedTransport = HIGH και παίρνουμε μήνυμα switch24 = ON, διότι Token = 0 (Δέσμευση).
- Ξαναδιαβάζει τις εισόδους, διότι άλλαξαν κατάσταση. Αυτό γίνεται, επειδή είναι διαδοχική η παρακάτω ενέργεια, να μην βγαίνουμε από το loop, ώστε να ξαναδιαβαστούν οι είσοδοι.

Οι συνθήκες της if condition ικανοποιούνται (εδώ υπάρχει η σημασία της βοηθητικής μεταβλητής), οπότε:

- Παίρνουμε μήνυμα το Tool_Mill να εκφορτωθεί στο BufferTool_Mill
- Οι βοηθητικές μεταβλητές downloadingTool = 1 και uploadingTool = 0
- Βλέπουμε το Led BufferTool_Mill = HIGH και παίρνουμε μήνυμα switch50 = ON, (Δέσμευση) και παραμένει έτσι, καθώς η ενδιάμεση αποθήκη εργαλείων Mill δεν αποδεσμεύεται.

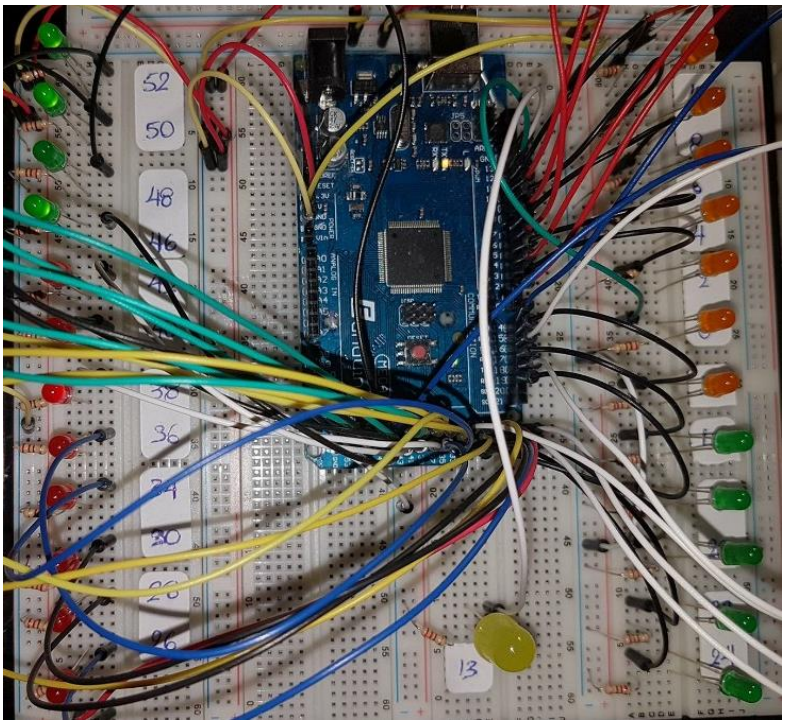
Οπότε, σε αυτή τη φάση, η εικόνα της hardware κατασκευής φαίνεται στην Εικόνα 6-29.



Εικόνα 6-29 Προσομοίωση Ελεγκτή (2)

- Η βοηθητική μεταβλητή downloadingTool = 1
- Βλέπουμε LedTransport = LOW και παίρνουμε μήνυμα switch24 = OFF, διότι Token = 0 (Αποδέσμευση).
- Αν δεν ικανοποιείται η αρχική if condition, τότε η else μας εκτυπώνει αντίστοιχο μήνυμα πως για ποιους λόγους δεν έγινε.

Οπότε, κλείνοντας ο βρόχος η hardware κατασκευή φαίνεται στην Εικόνα 6-30, ενώ οι εκτυπώσεις που έχουμε πάρει στο σειριακό παράθυρο φαίνονται στην Εικόνα 6-31.



Εικόνα 6-30 Προσομοίωση Ελεγκτή (3)

```
--------The contoller read all the Inputs--------
BufferTool_Mill (Token=1 - switch50=OFF) is available
Tool_Mill is uploaded to Transport
Tool_Mill is empty (Token=0 - switch52=ON)
Transport is engaged (Token=0 - switch24=ON)
Tool_Mill is downloaded to BufferTool_Mill
BufferTool_Mill (Token=0 - switch50=ON) is engaged
Transport is available (Token=1 - switch24=OFF)
BufferTool_Lathe is LOW or Transport is engaged or Tool_Lathe is empty of Tools
No pallets from Pallet_station to Pallet_Storage or Transport is engaged
No pallets from Pallet_Storage to Pallet_BufferMillIN or Transport is engaged
No units from Unit_Storage to Unit_BufferMillIN or Transport is engaged
------------  PRINTING IN EACH LOOP ------------
```

Εικόνα 6-31 Εκτυπώσεις στην Σειριακή κλείνοντας τον βρόχο (loop) στην Προσομοίωση

7

Συμπεράσματα – Επεκτάσεις

7.1 Συμπεράσματα

Από τον σχεδιασμό και την υλοποίηση ελέγχου μέσω σημάτων εισόδου – εξόδου διαπιστώθηκαν χρήσιμα συμπεράσματα, τόσο ως προς το αποτέλεσμα που ήταν επιτυχές, αλλά και ως προς την διαδικασία που ακολουθήθηκε.

Αρχικά, εξακριβώθηκε πως δύναται να ελεγχθεί το ΕΣΚ με χρησιμοποίηση των μικρο-ελεγκτών Arduino, με αξιόπιστη λύση τη σύνδεση του κεντρικού ελεγκτή με τους επιμέρους τοπικούς ελεγκτές, μέσω του διαύλου επικοινωνίας I^2C. Η υλοποίηση της υπόψη σύνδεσης και η χρησιμοποίηση της έτοιμης βιβλιοθήκης Wire του Arduino, πλεονεκτεί από την υλοποίηση μιας σύνδεσης μέσω πολλών κοινών καλωδίων και εντολών καθορισμού των ενεργειών του κάθε ελεγκτή καθότι:

- Ορίζεται απευθείας ο κεντρικός ελεγκτής ως master και οι υπόλοιποι ως slaves, χωρίς να χρειάζεται για κάθε ενέργεια να γράφουμε γραμμές κώδικα για τον καθορισμό των σχέσεων μεταξύ των συνεργαζόμενων συσκευών.
- Η σύνδεση με κάθε συσκευή γίνεται μέσω 3 καλωδίων (SDA, SCL, GND), οπότε δεν χρειάζεται να διαθέτει ο master πληθώρα θυρών εισόδου – εξόδου, αλλά πρέπει να έχει ικανό επεξεργαστή και μνήμη. Όπου εδώ, ο microcontroller ATmega2560 του ArduinoMega2560Rev3 καλύπτει τις ανάγκες για την υλοποίηση 24 τοπικών ελεγκτών.
- Η ταχύτητα διαβίβασης της πληροφορίας, μέσω του σειριακού διαύλου I2C, είναι ικανοποιητική για την συγκεκριμένη λειτουργία, διότι δεν απαιτείται μεγάλος όγκος πληροφορίας σε μικρά χρονικά διάστημα που να καταστούν δυσχερής τη σύνδεση, αφού μεταφέρονται απλά σήματα High/Low και οι χρόνοι αντίδρασης δεν είναι ακαριαίοι.
- Το κυρίως πρόγραμμα εκτελείται στον master, στο οποίο καθορίζεται η σειρά των ενεργειών ενώ στους slave το πρόγραμμα είναι ακριβώς ίδιο. Οπότε, οποιαδήποτε αλλαγή συμβαίνει, ως προς την σειρά των ενεργειών (if conditions), επεμβαίνουμε στο πρόγραμμα του master, χωρίς να χρειάζονται επιπλέον αλλαγές και στα αντίστοιχα προγράμματα των slaves.

Επίσης, η αντιστοίχιση των εντολών από ένα πρόγραμμα ελέγχου μεταξύ των Arduino με σύνδεση I2C, σε ένα πρόγραμμα ενός ελεγκτή που προσομοιάζει τους τοπικούς ελεγκτές με διακόπτες, πραγματοποιήθηκε με πετυχημένο τρόπο. Μία σύνδεση I2C με 1 κεντρικό ελεγκτή και 24 τοπικούς ελεγκτές, μπορεί να επιτευχθεί με σχετική ευκολία, διότι έχουμε έτοιμο το πρόγραμμα για έναν ελεγκτή με 24 διακόπτες/εισόδους, που προσομοιάζουν τους τοπικούς ελεγκτές και αυτό που απομένει είναι να - αντιστοιχίσουμε τις εντολές, σύμφωνα και με τον παρακάτω Πίνακα 7-1.

	Arduino I2C		Arduino Switches and Leds
	Master	Slave	
Inputs στον Κεντρικό Ελεγκτή	Wire.requestFrom(SlaveAd) Val = Wire.read()	Wire.onRequest (requestEvent) Wire.write(val)	digitalRead(switch)
Outputs από τον Κεντρικό Ελεγκτή	Wire.beginTransmission (SlaveAd) Wire.write(val) Wire.endTransmission()	Wire.onReceive (receiveEvent) Val = Wire.read()	digitalWrite(Led) and Serialprint(switch)

Πίνακας 7-1 Αντιστοίχιση Εισόδων - Εξόδων Προγραμμάτων Arduino I2C - Swithes/Leds

Επίσης, τα δίκτυα του Petri αποτέλεσαν ένα χρήσιμο εργαλείο μοντελοποίησης, καθώς μοντελοποιώντας τις θέσεις εισόδου στις μεταβάσεις, ως συνθήκες ικανοποίησης σε δομές επιλογής "if" και τις θέσεις εξόδου από τις μεταβάσεις ως εντολές που εκτελούνται εντός της δομής επιλογής, κατέστη δυνατό να μεταφράσουμε ένα δίκτυο Petri σε ένα υλοποιήσιμο πρόγραμμα ελέγχου Arduino. Σύμφωνα με την παρακάτω αρχή:

```
If (Pin = ……High/low) {
    Pout = ……High/low
}
```

Σχήμα 7-1 Εξαγωγή Εντολών Προγράμματος από τους Κανόνες Ενεργοποίησης του PN

Πέραν των ορθών αποτελεσμάτων που εξήχθησαν κατά τη διάρκεια του σχεδιασμού και της υλοποίησης της εφαρμογής ελέγχου στο ΕΣΚ, παρατηρήθηκαν και δυσχέρειες, όπως παρακάτω:

- Η κατασκευή του δικτύου Petri δεν ήταν εύκολη, κυρίως λόγω των πολλών ενδιάμεσων θέσεων που υπήρχαν, καθιστώντας το δίκτυο ιδιαίτερα εκτενές και πολύπλοκο ως προς τη δομή του.
- Επίσης, το υπόψη δίκτυο Petri υλοποιεί την προσομοίωση μιας συγκεκριμένης σειράς κατεργασιών (φρεζάρισμα – τόρνευση) από τις έξι πιθανές. Αυτό έγινε κυρίως ως απόρροια του τρόπου λειτουργίας των κλασσικών δικτύων Petri, καθώς στα κλασσικά PN θα ήταν δυνατόν να εκτελούμε και τις 6 διαφορετικές περιπτώσεις κατεργασιών, αλλά η δομή του PN θα ήταν κατά πολύ μεγαλύτερη.
- Δεν υπήρξε απόλυτη ταύτιση του δικτύου Petri με το πρόγραμμα ελέγχου, αλλά έπρεπε να γίνουν παρεμβάσεις. Π.χ. Στα PN δεν μπορούσε να προβλεφθεί, κατά τη διάρκεια της προσομοίωσης, η αναγνώριση λανθασμένου τεμαχίου. Το οποίο όμως, προβλέφθηκε και υλοποιήθηκε στην κατασκευή του προγράμματος ελέγχου, διότι κατά το «τρέξιμο» του προγράμματος ο χρήστης μπορεί να παρέμβει στις εντολές που διαβάζει το πρόγραμμα και να τις εισάγει ως εισόδους.

- Επιπλέον, δεν μπορεί να υπάρξει στο υφιστάμενο πλαίσιο, κάποιου είδους τυποποίηση της διαδικασίας μετατροπής του PN σε πρόγραμμα C/C++, Python ή κάποιο άλλο. Είναι εφαρμόσιμη, μια αρχική μετατροπή των PN σε διάγραμμα Ladder του PLC, αλλά και εκεί χρειάζεται να παρέμβει ο χρήστης, ώστε να το κάνει υλοποιήσιμο. [8]
- Η μετάβαση από το PN σε πρόγραμμα δεν ήταν τόσο δύσκολη όσο χρονοβόρα, διότι στην παρούσα εργασία η προσομοίωση του PN χρησιμοποιήθηκε περισσότερο σαν ένα είδος Έμπειρου Συστήματος (Expert System), ώστε να μεταφράζονται ορθά οι κανόνες ενεργοποίησης σε Δομές Επιλογής (if conditions), βάσει του Πίνακα 7-2.

Το γενικό συμπέρασμα είναι πως η υλοποίηση ελέγχου με Arduino σε FMS υλοποιήθηκε με επιτυχία, διότι πέραν του προγραμματιστικού μέρους (software), παράλληλα έγινε και το κατασκευαστικό μέρος (hardware) του ελεγκτή. Τέλος, παρατηρήθηκε κατά τη διάρκεια ανεύρεσης παρόμοιων εφαρμογών, πως αν και υπάρχει πληθώρα εφαρμογών με Arduino, τα οποία αποτελούν έναν ευρέως διαδεδομένο μικρο-ελεγκτή, δεν παρατηρήθηκε αντίστοιχη εφαρμογή στον έλεγχο κυττάρων κατεργασιών ή και απλών μηχανών κατεργασίας.

7.2 Μελλοντικές Επεκτάσεις

Οι εφαρμογές που πραγματοποιήθηκαν μπορούν να βελτιωθούν ή να αποτελέσουν κορμό για διάφορες επεκτάσεις, όπως:

- Η βελτιστοποίηση της σειράς εκτέλεσης των διεργασιών μέσα στο πρόγραμμα, με στόχο την αύξηση της ταχύτητας παραγωγής ή της μείωσης του χρόνου παραγωγής ή της αύξησης των τελικών παραγόμενων τεμαχίων και παλετών, με τη χρήση γενετικών αλγορίθμων, αφού κάθε περίπτωση μπορεί να δημιουργήσει μια αντικειμενική συνάρτηση.
- Η υλοποίηση της εφαρμογής με 24 Arduino ως τοπικούς ελεγκτές και τη σύνδεση αυτών με τους σταθμούς που συνθέτουν το Ευέλικτο Σύστημα Κατεργασιών υπό έναν κεντρικό ελεγκτή Arduino.
- Βελτίωση της υπόψη εφαρμογής, μειώνοντας τους σταθμούς που συνθέτουν το FMS ή μειώνοντας τις διεργασίες που επιτελούνται με κάποιο τρόπο ομαδοποίησης τους. Π.χ. uploading και downloading => Μια διεργασία.
- Την κατά το δυνατόν παραμετροποίηση του προγράμματος, ώστε να προστεθούν και οι άλλες 5 περιπτώσεις σειράς κατεργασιών σε αυτό.
- Εκτέλεση προσομοιώσεων του προγράμματος ελέγχου, για ακραίες περιπτώσεις εισαγωγής δεδομένων (εισόδων), ώστε να εξακριβωθεί η δυνατότητα υλοποίησης αυτών, αλλά και η διαπίστωση πιθανών λαθών, που μπορεί να οδηγήσουν σε λάθη εκτέλεσης εντολών.

8

Βιβλιογραφία

[1] Ι. Γιαννάτσης, Β. Δεδούσης, and Β. Κανελλιδης, *Σύγχρονες Τεχνολογίες Κατασκευής με τη βοήθεια Η / Υ*. Ελληνικά Ακαδημαϊκά Ηλεκτρονικά Συγγράμματα και Βοηθήματα, 2015.

[2] A. V. Ranson, "Group Technology," p. 30, 1972.

[3] G. Halevi, *Expectations and Disappointments of Industrial Innovations*. 2017.

[4] Γ.-Χ. Βοσνιάκος, "Σημειώσεις - Συστήματα Κατεργασιών." 2016.

[5] C. H. Chu and M. Tsai, "A comparison of three array-based clustering techniques for manufacturing cell formation," *Int. J. Prod. Res.*, vol. 28, no. 8, pp. 1417–1433, 1990.

[6] M. Zhou and K. Venkatesh, *Modelling, Simulation and Control of FMS, A Petri Net Approach*. World Scientific, 1999.

[7] J. Lenz, "Flexible Manufacturing Systems," *Manuf. Eng. Handb.*, 2015.

[8] Κ. Παπάζογλου, *Διπλωματική εργασία: Μοντελοποίηση ευέλικτου κυττάρου κατεργασιών με δίκτυα Coloured Petri και μετατροπή σε γλώσσα προγραμματιζόμενων λογικών ελεγκτών*. 2014, p. 158.

[9] Λ. Αλεξόπουλος, "Σημειώσεις - Βιολογική Μηχανική (Bioinformatics)." pp. 1–68, 2016.

[10] Ε. Παπαδόπουλος and Κ. Κυριακόπουλος, "Σημειώσεις - Εισαγωγή στην Ρομποτική," 2014, pp. 1–8.

[11] Γ.-Χ. Βοσνιάκος, *Σχεδιασμός Συστημάτων Ελέγχου για Συστήματα Κατεργασιών*. 2000.

[12] J. L. Peterson, "Petri Nets," *ACM Comput. Surv.*, vol. 9, no. 3, pp. 223–252, 1977.

[13] Ξ. Γωγουβίτης, Δ*ιπλωματική εργασία: Εισαγωγή στα δίκτυα Petri (Petri-nets, PN)*. 2002, pp. 1–19.

[14] M. Zhou and F. DiCesare, *Petri net synthesis for discrete event control of manufacturing systems*. SPRINGER SCIENCE+BUSINESS MEDIA, LLC, 1993.

[15] X. Li, "Discrete Event System Modeling and Simulation," p. 38, 2000.

[16] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems," *Int. J. Softw. Tools Technol. Transf.*, vol. 9, no. 3–4, pp. 213–254, 2007.

[17] H. Mayr, "Foundamentals of Modeling." pp. 1–28, 2002.

[18] F. Dicesare, G. Harhalakis, J. M. Proth, and M. Silva, *Practice of Petri Nets in Manufacturing Practice*. Melbourne: Chapman and Hall, 1993.

[19] T. H. Koh, *Koh , Tae Hoon Control of a Flexible Manufacturing Cell using Signal Interpreted Petri Nets May 2004*. 2004, p. 63.

[20] G. Frey and M. Minas, "PLC Programming with Signal Interpreted Petri Nets," pp. 440–441, 2000.

[21] Θ. Παπιγγιώτης, *Διπλωματική Εργασία: Έλεγχος και προγραμματισμός ρομποτικής διάταξης τύλιξης νήματος για κατασκευή απλών τεμαχίων από σύνθετα υλικά*. 2017, p. 107.

[22] Κ. Κυριακόπουλος, "Σημειώσεις: Έλεγχος με Μικροϋπολογιστές (embedded systems)." p. 20,

2016.

[23] Ε. Πουλάκης, *Προγραμματίζοντας με τον μικροελεγκτή Arduino*. 2015.

[24] "Arduino." [Online]. Available: available: https://www.arduino.cc.

[25] Grobotronics, "Τι είναι το Arduino?" [Online]. Available: http://learning.grobotronics.com/el/getting-started/arduino-uno/.

[26] Anusha, "Programmable Logic Controller." [Online]. Available: https://www.electronicshub.org/programmable-logic-controllers/.

[27] Κ. Γεωργιάδης, *Διπλωματική Εργασία: Σειριακή Επικοινωνία Δύο Raspberry Pi*. 2015, p. 104.

[28] C. McFadden, "Raspberry Pi and Arduino: What's the Difference and Which Is Best for Your Project?," 2018. [Online]. Available: https://interestingengineering.com/raspberry-pi-and-arduino-whats-the-difference-and-which-is-best-for-your-project.

[29] AddOhms, "Comparing the Arduino Uno and Raspberry Pi | AddOhms #7," 2013. [Online]. Available: https://www.youtube.com/watch?time_continue=183&v=7vhvnaWUZjE.

9

Παραρτήματα

9.1 Προγράμματα για Σχεδίαση και Προσομοιώσεις

Για τα σχέδια της εργασίας χρησιμοποιήθηκαν τα παρακάτω λογισμικά προγράμματα:

- Το **Fritzing** για την σχεδίαση των πλακετών (breadboards) και των αντίστοιχων ηλεκτρονικών σχεδίων. (http://fritzing.org/home/)
- Το **PIPEv4.3.0** για τη σχεδίαση και την προσομοίωση των δικτύων Petri (PN). (pipe2.sourceforge.net/)
- Το **Arduino1.8.5** για τους κώδικες των μικρο-ελεγκτών Arduino. (https://www.arduino.cc/en/Main/Software/)

9.2 Προγράμματα Arduino Master – Slaves μέσω I2C

9.2.1 Πρόγραμμα ArduinoMega2560 για Master

```
// I2C MASTER code (ARDUINO MEGA)
// Include the required Wire library for I2C<br>
#include <Wire.h>

// Name the addresses of the 2 Slaves Arduino Uno (not 0<Address<7)
const byte RobotAd=8;
const byte MillAd=9;

/*Initialize the variables, the char doesn't matter,
  Just to compile the code 'cause of if condition*/
char Robot = 'k';
char Mill = 'k';

void setup() {
  // Start the I2C Bus as Master
  Wire.begin(); // don't need a number in parenthesis, 'cause it's the MASTER ARDUINO
MEGA
```

```
  // Open Serial Port and set data rate = 9600 bps (baud)
  Serial.begin(9600);
}

void loop() {

  // LED for checking the loop
  pinMode(10, OUTPUT);
  digitalWrite(10, HIGH);
  delay (1000);

  // The Master asks the slave Robot (RobotAd = 8) if it is available or not
  Wire.requestFrom(8,1);
  while (Wire.available()){
    Robot = Wire.read();
    Serial.print("1. The Robot read from request = ");
    Serial.println(Robot);
  }
  delay (2000);

  // The Master asks the slave Mill (MillAd = 8) if it is available or not
  Wire.requestFrom(9,1);
  while (Wire.available()){
    Mill = Wire.read();
    Serial.print("2. The Mill read from request = ");
    Serial.println(Mill);
  }
  delay (2000);

  Serial.println("3. End of Request from Slaves to Master");
  delay (2000);

  // value = 'L' in the char variables, means they have token (LED = LOW).
  // But the token doesn't mean always available, but as it has been defined in the
Petri Net
  if (Robot == 'L' && Mill == 'L'){
    // Robot = 1 and Mill = 1 are available, so do the milling process and convert
them to
    // Robot = 0 and Mill = 0
    Serial.println("4. Engage Robot and Mill ");
    Robot = 'H';
    Mill = 'H';
    Serial.print("Robot= ");
    Serial.println(Robot);
    Serial.print("Mill= ");
    Serial.println(Mill);
    delay (2000);

    Serial.println("5. Begin Transmition to Slaves from Master");
    // Send signals to slaves, only when Robot and Miil are available ('L') to be
occupied ('H')
    // Sending to Robot the order of engagement to Robot Address ('H')
    Wire.beginTransmission(8);
    Wire.write(Robot);
    Wire.endTransmission();

    // Sending to Mill the order of engagement to Mill Address ('H')
    Wire.beginTransmission(9);
    Wire.write(Mill);
    Wire.endTransmission();
```

```
  }

  delay (2000);
  // Change the LED in each loop
  digitalWrite(10, LOW);
  delay (2000);
}

/*---------- END OF CODE MASTER-MAIN CONTROLLER ----------*/
```

9.2.2 *Πρόγραμμα ArduinoUno για Slave – Robot (8)*

```
// I2C SLAVE ROBOT-8 code (ARDUINO UNO)
// Include the required Wire library for I2C<br>
#include <Wire.h>

// Name the addresses of the Slave Arduino Uno(not 0<Address<7)
const byte RobotAd=8;

// Initialize the variables L = 1 (available) and H = 0 (occupied = process)
char Robot = 'L';

void setup() {
  // Start the I2C Bus as Slave-8
  Wire.begin(8); // need a number in parenthesis, 'cause it's a SLAVE ARDUINO UNO

  // LED is LOW when the Robot is available (Robot = 1)
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);

  // Send the value of Robot to master(answer to requestFrom) by write
  Wire.onRequest(requestEvent);
  // Receive the value of Robot by master(when Transmission) by read
  Wire.onReceive(receiveEvent);

  // Open Serial Port and set data rate = 9600 bps (baud)
  Serial.begin(9600);

  delay (1000);
}

void loop() {
  // put your main code here, to run repeatedly:
  delay(100);
}

// Transmission to master
void requestEvent(){
  Wire.write(Robot);
  Serial.println("1. Robot Transmits to Master");
  delay (2000);
}

//Received by master
void receiveEvent(){
  while (Wire.available()){
    Robot = Wire.read();
    Serial.println("2. Robot Receives by Master");
    delay (2000);
```

```
    /* the if condition is unnecessary, 'cause the function receiveEvent
    takes place, only when Robot = 0, after entering the if condtion
    in the master's code*/
    if (Robot == 'H'){
      // Change the LED to HIGH, 'cause Robot = 0 means occupied
      digitalWrite(13, HIGH);
      Serial.println("3. Robot is engaged");
      /* Here delay is the most significant point, 'cause it changes the state
      of availability of the Robot. Normally here, a signal would be sent by the
      the Robot to its controller, when the processing was done */
      delay(5000);
      // Make available the Robot again
      Robot = 'L';
      digitalWrite(13, LOW);
      Serial.println("4. Robot is available");
    }
  }
}

/*---------- END OF CODE SLAVE ROBOT-8 ----------*/
```

9.2.3 *Πρόγραμμα ArduinoUno για Slave – Mill (9)*

```
// I2C SLAVE MILL-9 code (ARDUINO UNO)
// Include the required Wire library for I2C<br>
#include <Wire.h>

// Name the addresses of the Slave Arduino Uno(not 0<Address<7)
const byte MillAd=9;

// Initialize the variables L = 1 (available) and H = 0 (occupied = process)
char Mill = 'L';

void setup() {
  // Start the I2C Bus as Master
  Wire.begin(9); // need a number in parenthesis, 'cause it's a SLAVE ARDUINO UNO

  // LED is LOW when the Mill is available (Mill = 1)
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);

  // Send the value of Mill to master(answer to requestFrom) by write
  Wire.onRequest(requestEvent);
  // Receive the value of Mill by master(when Transmission) by read
  Wire.onReceive(receiveEvent);

  // Open Serial Port and set data rate = 9600 bps (baud)
  Serial.begin(9600);

  delay (1000);
}

void loop() {
  // put your main code here, to run repeatedly:
  delay(100);
}

// Transmission to master
```

```
void requestEvent(){
  Wire.write(Mill);
  Serial.println("1. Mill Transmits to Master");
  delay (2000);
}

//Received by master
void receiveEvent(){
  while (Wire.available()){
    Mill = Wire.read();
    Serial.println("2. Mill Receives by Master");
    delay (2000);
    /* the if condition is unnecessary, 'cause the function receiveEvent
    takes place, only when Mill = 0, after entering the if condtion
    in the master's code*/
    if (Mill == 'H'){
      // Change the LED to HIGH, 'cause Mill=0 means occupied
      digitalWrite(13, HIGH);
      Serial.println("3. Mill is engaged");
      /* Here delay is the most significant point, 'cause it changes the state
      of availability of the Mill. Normally here, a signal would be sent by the
      the Mill machine to its controller, when the processing was done */
      delay(5000);
      // Make available the Mill again
      Mill = 'L';
      digitalWrite(13, LOW);
      Serial.println("4. Mill is available");
    }
  }
}

/*---------- END OF CODE SLAVE ROBOT-8 ----------*/
```

9.3 *Πρόγραμμα ArduinoMega2560 – Διακοπτές – Φωτοδιόδοι*

```
/* Project: Design and Implementation of the Controller in a Flexible Manufacturing
System(FMS)
   Dissertation of Konstantinos Petropoulos (2018)
   Implementation of the Petri Net by 24 LEDs and Switches*/


// ********** Initialize Button Switches**********
// Tool Places (4)
int switchTool_Mill = 0; //1
int switchTool_Lathe = 0; //2
int switchBufferTool_Mill = 0; //3
int switchBufferTool_Lathe = 0; //4
// Pallet Places (8)
int switchPallet_station = 0; //5
int switchPallet_Storage = 0; //6
int switchPallet_BufferMillIN = 0; //7
int switchPallet_BufferMillOUT = 0; //8
int switchWrong_Pallet = 0; //9
int switchPallet_BufferLatheIN = 0; //10
int switchPallet_BufferLatheOUT = 0; //11
int switchPallet_Finish = 0; //12
// Unit Places (7)
int switchUnit_Storage = 0; //13
```

```
int switchUnit_BufferMillIN = 0; //14
int switchUnit_BufferMillOUT = 0; //15
int switchWrong_Unit = 0; //16
int switchUnit_BufferLatheIN = 0; //17
int switchUnit_BufferLatheOUT = 0; //18
int switchUnit_Finish = 0; //19
// Common Places (5)
int switchTransport = 0; //20
int switchRobot = 0 ; //21
int switchID_pos = 0; //22
int switchMill = 0; //23
int switchLathe = 0; //24
// **********End of Initialize Button Switches**********

// **********Definiton of PinModes**********
// pinModes for Inputs (Switches)
int Tool_Mill = 52;
int BufferTool_Mill = 50;
int Tool_Lathe = 48;
int BufferTool_Lathe = 46;

int Pallet_station = 42;
int Pallet_Storage = 40;
int Pallet_BufferMillIN = 38;
int Pallet_BufferMillOUT = 36;
int Wrong_Pallet = 34;
int Pallet_BufferLatheIN = 30;
int Pallet_BufferLatheOUT = 28;
int Pallet_Finish = 26;

int Unit_Storage = 10;
int Unit_BufferMillIN = 8;
int Unit_BufferMillOUT = 6;
int Wrong_Unit = 4;
int Unit_BufferLatheIN = 2;
int Unit_BufferLatheOUT = 44; //0
int Unit_Finish = 14;

int Transport = 24;
int Robot = 22;
int ID_pos = 32; //20
int Mill = 18;
int Lathe = 16;

// pinModes for Outputs (LEDs)
int LedTool_Mill = 53;
int LedBufferTool_Mill = 51;
int LedTool_Lathe = 49;
int LedBufferTool_Lathe = 47;

int LedPallet_station = 43;
int LedPallet_Storage = 41;
int LedPallet_BufferMillIN = 39;
int LedPallet_BufferMillOUT = 37;
int LedWrong_Pallet = 35;
int LedPallet_BufferLatheIN = 31;
int LedPallet_BufferLatheOUT = 29;
int LedPallet_Finish = 27;

int LedUnit_Storage = 11;
int LedUnit_BufferMillIN = 9;
```

```
int LedUnit_BufferMillOUT = 7;
int LedWrong_Unit = 5;
int LedUnit_BufferLatheIN = 3;
int LedUnit_BufferLatheOUT = 12; //1
int LedUnit_Finish = 15;

int LedTransport = 25;
int LedRobot = 23;
int LedID_pos = 33; //21
int LedMill = 19;
int LedLathe = 17;
// **********End of Definiton of PinModes**********

// Initialize Proceedings - Processings (Places of the Petri Net)
int uploadingTool = 0; // NO uploading
int downloadingTool = 0; // NO downloading
int uploadingPallettoStorage = 0; // NO uploading
int downloadingPallettoStorage = 0; // NO downloading
int uploadingPallettoBufferMiillIN = 0; // NO uploading
int downloadingPallettoBufferMiillIN = 0; // NO downloading
int pallets = 1; // The pallets that are produced in the Pallet_station
int counterPallet = 1; // For making the Pallet_Station = High, when there are no
more Pallets
int uploadingPallettoIDpos = 0; // NO uploading
int downloadingPallettoMill = 0; // NO downloading
int Milling = 0; // NO Process
int HoldOnPalletMilling = 0;
int uploadingPallettoBufferMillOUT = 0; // NO uploading
int downloadingPallettoBufferLatheIN = 0; // NO downloading
int uploadingPallettoLathe = 0; // NO uploading
int Turning = 0; // NO Process
int HoldOnPalletTurning = 0;
int downloadingPallettoBufferLatheOUT = 0; // NO downloading
int uploadingPallettoFinish = 0; // NO uploading
int downloadingPallettoFinish = 0; // NO downloading
int endofPallets = 0; // Begin Uints' Processing
int enterUnits = 1; // For Entering Units and not entering again
int units = 1; // The units that are loaded in the Unit_Storage
int counterUnit = 1; // For making the Unit_Storage = High, when there are no more
Units
int uploadingUnittoBufferMillIN = 0; // NO uploading
int downloadingUnittoBufferMillIN = 0; // NO downloading
int uploadingUnittoIDpos = 0; // NO uploading
int downloadingUnittoMill = 0; // NO downloading
int HoldOnUnitMilling = 0;
int uploadingUnittoBufferMillOUT = 0; // NO uploading
int downloadingUnittoBufferLatheIN = 0; // NO downloading
int uploadingUnittoLathe = 0; // NO uploading
int HoldOnUnitTurning = 0;
int downloadingUnittoBufferLatheOUT = 0; // NO downloading
int uploadingUnittoFinish = 0; // NO uploading
int downloadingUnittoFinish = 0; // NO downloading

void setup() {
  // For Printing Commands in Serial Monitoring
  Serial.begin(9600);

  // put your setup code here, to run once:
  // Define the pinModes as INPUT - OUTPUT
  pinMode(Tool_Mill,INPUT);
  pinMode(LedTool_Mill,OUTPUT);
```

```
  pinMode(Tool_Lathe,INPUT);
  pinMode(LedTool_Lathe,OUTPUT);
  pinMode(BufferTool_Mill,INPUT);
  pinMode(LedBufferTool_Mill,OUTPUT);
  pinMode(BufferTool_Lathe,INPUT);
  pinMode(LedBufferTool_Lathe,OUTPUT);

  pinMode(Pallet_station,INPUT);
  pinMode(LedPallet_station,OUTPUT);
  pinMode(Pallet_Storage,INPUT);
  pinMode(LedPallet_Storage,OUTPUT);
  pinMode(Pallet_BufferMillIN,INPUT);
  pinMode(LedPallet_BufferMillIN,OUTPUT);
  pinMode(Pallet_BufferMillOUT,INPUT);
  pinMode(LedPallet_BufferMillOUT,OUTPUT);
  pinMode(Wrong_Pallet,INPUT);
  pinMode(LedWrong_Pallet,OUTPUT);
  pinMode(Pallet_BufferLatheIN,INPUT);
  pinMode(LedPallet_BufferLatheIN,OUTPUT);
  pinMode(Pallet_BufferLatheOUT,INPUT);
  pinMode(LedPallet_BufferLatheOUT,OUTPUT);
  pinMode(Pallet_Finish,INPUT);
  pinMode(LedPallet_Finish,OUTPUT);

  pinMode(Unit_Storage,INPUT);
  pinMode(LedUnit_Storage,OUTPUT);
  pinMode(Unit_BufferMillIN,INPUT);
  pinMode(LedUnit_BufferMillIN,OUTPUT);
  pinMode(Unit_BufferMillOUT,INPUT);
  pinMode(LedUnit_BufferMillOUT,OUTPUT);
  pinMode(Wrong_Unit,INPUT);
  pinMode(LedWrong_Unit,OUTPUT);
  pinMode(Unit_BufferLatheIN,INPUT);
  pinMode(LedUnit_BufferLatheIN,OUTPUT);
  pinMode(Unit_BufferLatheOUT,INPUT);
  pinMode(LedUnit_BufferLatheOUT,OUTPUT);
  pinMode(Unit_Finish,INPUT);
  pinMode(LedUnit_Finish,OUTPUT);

  pinMode(Transport,INPUT);
  pinMode(LedTransport,OUTPUT);
  pinMode(Robot,INPUT);
  pinMode(LedRobot,OUTPUT);
  pinMode(ID_pos,INPUT);
  pinMode(LedID_pos,OUTPUT);
  pinMode(Mill,INPUT);
  pinMode(LedMill,OUTPUT);
  pinMode(Lathe,INPUT);
  pinMode(LedLathe,OUTPUT);

  pinMode(13,OUTPUT);

  // Initialize BufferTools Mill & Lathe, they are NOT available (Engaged) by default
(token=0)
  digitalWrite(LedBufferTool_Mill, HIGH);
  digitalWrite(LedBufferTool_Lathe, HIGH);
  delay (10000);
}

void loop() {
  // put your main code here, to run repeatedly:
```

```
  // Led for checking the loops
  digitalWrite(13, HIGH);
  delay (5000);

  //********** Read the signal in all Inputs **********
  switchTool_Mill = digitalRead(Tool_Mill); // default is OFF = LED LOW (1 token)
  switchTool_Lathe = digitalRead(Tool_Lathe); // default is OFF = LED LOW (1 token)
  switchBufferTool_Mill = digitalRead(BufferTool_Mill); // default is ON = LED HIGH
(0 token)
  switchBufferTool_Lathe = digitalRead(BufferTool_Lathe); // default is ON = LED HIGH
(0 token)

  switchPallet_station = digitalRead(Pallet_station); // default is OFF = LED LOW (1
token)
  switchPallet_Storage = digitalRead(Pallet_Storage); // default is OFF = LED LOW (1
token)
  switchPallet_BufferMillIN = digitalRead(Pallet_BufferMillIN); // default is OFF =
LED LOW (1 token)
  switchPallet_BufferMillOUT = digitalRead(Pallet_BufferMillOUT); // default is OFF =
LED LOW (1 token)
  // switchWrong_Pallet = digitalRead(Wrong_Pallet); // EXCEPTION!!! by default is
OFF = LED LOW (1 token) // It's not needed here !!!
  switchPallet_BufferLatheIN = digitalRead(Pallet_BufferLatheIN);
  switchPallet_BufferLatheOUT = digitalRead(Pallet_BufferLatheOUT);
  switchPallet_Finish = digitalRead(Pallet_Finish);

  switchUnit_Storage = digitalRead(Unit_Storage);
  switchUnit_BufferMillIN = digitalRead(Unit_BufferMillIN);
  switchUnit_BufferMillOUT = digitalRead(Unit_BufferMillOUT);
  // switchWrong_Unit = digitalRead(Wrong_Unit); // // EXCEPTION!!! by default is OFF
= LED LOW (1 token) // It's not needed here !!!
  switchUnit_BufferLatheIN = digitalRead(Unit_BufferLatheIN);
  switchUnit_BufferLatheOUT = digitalRead(Unit_BufferLatheOUT);
  switchUnit_Finish = digitalRead(Unit_Finish);

  switchTransport = digitalRead(Transport);
  switchRobot = digitalRead(Robot);
  // switchID_pos = digitalRead(ID_pos); // EXCEPTION!!! by default is OFF = LED LOW
(1 token) // It's not needed here !!!
  switchMill = digitalRead(Mill);
  switchLathe = digitalRead(Lathe);
  //**********End of Reading the signal in Inputs**********
  Serial.println("--------The contoller read all the Inputs--------");
  delay (10000);

  /* In general, LOW means that the Place has token = 1, Independently if it's
engaged or available
  Token = 1 means LED = LOW and Switch = OFF ----- Otherwise Token = 0 means LED =
HIGH and Switch = ON*/


  /* A. Loading Tools to Buffer_Tools with the Transport*/
  // Loading Tool_Mill to Mill Machine
  if (switchTool_Mill == LOW && switchBufferTool_Mill == HIGH && switchTransport ==
LOW){
    digitalWrite(LedBufferTool_Mill, LOW); // Available BufferTool_Mill, by default
is LED = HIGH (token=0)
    Serial.println("BufferTool_Mill (Token=1 - switch50=OFF) is available");
    delay (15000);
    Serial.println("Tool_Mill is uploaded to Transport");
    delay (10000);
```

```
    uploadingTool = 1;
    digitalWrite(LedTool_Mill, HIGH);
    Serial.println("Tool_Mill is empty (Token=0 - switch52=ON) ");
    delay (15000);
    digitalWrite(LedTransport, HIGH);
    Serial.println("Transport is engaged (Token=0 - switch24=ON) ");
    delay (15000);
    // Read again BufferTool_Mill and Transport
    switchBufferTool_Mill = digitalRead(BufferTool_Mill);
    switchTransport = digitalRead(Transport);
    if (uploadingTool == 1 && switchBufferTool_Mill == LOW && switchTransport ==
HIGH){
      Serial.println("Tool_Mill is downloaded to BufferTool_Mill");
      delay (10000);
      downloadingTool = 1;
      uploadingTool = 0;
      digitalWrite(LedBufferTool_Mill, HIGH);
      Serial.println("BufferTool_Mill (Token=0 - switch50=ON) is engaged");
      delay (15000);
      downloadingTool = 0;
      // Free the Transport
      digitalWrite(LedTransport, LOW);
      Serial.println("Transport is available (Token=1 - switch24=OFF) ");
      delay (15000);
    }
  }
  else {
    Serial.println("BufferTool_Mill is LOW or Transport is engaged or Tool_Mill is
empty of Tools");
  }
  delay (5000);
  // End of Loading Tool_Mill to Mill Machine

  // Loading Tool_Lathe to Lathe Machine
  if (switchTool_Lathe == LOW && switchBufferTool_Lathe == HIGH && switchTransport ==
LOW){
    digitalWrite(LedBufferTool_Lathe, LOW); // Available BufferTool_Lathe, by default
is LED = HIGH (token=0)
    Serial.println("BufferTool_Lathe (Token=1 - switch46=OFF) is available");
    delay (15000);
    Serial.println("Tool_Lathe is uploaded to Transport");
    delay (10000);
    uploadingTool = 1;
    digitalWrite(LedTool_Lathe, HIGH);
    Serial.println("Tool_Lathe is empty (Token=0 - switch48=ON) ");
    delay (15000);
    digitalWrite(LedTransport, HIGH);
    Serial.println("Transport is engaged (Token=0 - switch24=ON) ");
    delay (15000);
    // Read again BufferTool_Lathe and Transport
    switchBufferTool_Lathe = digitalRead(BufferTool_Lathe);
    switchTransport = digitalRead(Transport);
    if (uploadingTool == 1 && switchBufferTool_Lathe == LOW && switchTransport ==
HIGH){
      Serial.println("Tool_Lathe is downloaded to BufferTool_Lathe");
      delay (10000);
      downloadingTool = 1;
      uploadingTool = 0;
      digitalWrite(LedBufferTool_Lathe, HIGH);
      Serial.println("BufferTool_Lathe (Token=0 - switch46=ON) is engaged");
      delay (15000);
```

```
      downloadingTool = 0;
      digitalWrite(LedTransport, LOW);
      Serial.println("Transport is available (Token=1 - switch24=OFF) ");
      delay (15000);
    }
  }
  else {
    Serial.println("BufferTool_Lathe is LOW or Transport is engaged or Tool_Lathe is
empty of Tools");
  }
  delay (5000);
  // End of Loading Tool_Lathe to Lathe Machine
  /* End of Loading Tools to Buffer_Tools with the Transport*/


  /* B. Start of Pallets' Processing
     If there are not any tools for transport,go on with PALLETS*/
  // Transport Pallet from Pallet_Station to Pallet_Storage until pallets, Tool_Mill
and Tool_Lathe simulate the deterant branches
  if (switchTool_Mill == HIGH && switchTool_Lathe == HIGH && switchPallet_station ==
LOW && switchPallet_Storage == LOW && switchTransport == LOW){
    Serial.println("No Tools for Transportaion. Begin Pallets Processing");
    uploadingPallettoStorage = 1;
    Serial.println("Pallet is uploaded from Pallet_Station to Transport");
    delay (10000);
    // Pallet_Station = HIGH when counterPallet = Pallets
    if (counterPallet == pallets) {
      digitalWrite(LedPallet_station, HIGH);
      Serial.println("Pallet_station has no more pallets (Token=0 - switch42=ON) ");
      delay (15000);
    }
    // Increase counterPallet by one in each loop
    counterPallet = counterPallet + 1;
    digitalWrite(LedPallet_Storage, HIGH);
    Serial.println("Pallet_Storage is engaged (Token=0 - switch40=ON) ");
    delay (15000);
    digitalWrite(LedTransport, HIGH);
    Serial.println("Transport is engaged (Token=0 - switch24=ON) ");
    delay (15000);
    switchTransport = digitalRead(Transport);
    switchPallet_Storage = digitalRead(Pallet_Storage);
    if (uploadingPallettoStorage == 1 && switchTransport == HIGH &&
switchPallet_Storage == HIGH){
      downloadingPallettoStorage = 1;
      uploadingPallettoStorage = 0;
      Serial.println("Pallet is downloaded from Transport to Pallet_Storage");
      delay (10000);
      digitalWrite(LedTransport, LOW);
      Serial.println("Transport is available (Token=1 - switch24=OFF) ");
      delay (15000);
    }
  }
  else {
    Serial.println("No pallets from Pallet_station to Pallet_Storage or Transport is
engaged");
  }
  delay (5000);

  // Transport Pallet from Pallet_Storage to PalletBufferMillIN
  if (downloadingPallettoStorage == 1 && switchPallet_Storage == HIGH &&
switchTransport == LOW && switchPallet_BufferMillIN == LOW) {
```

```
    uploadingPallettoBufferMiillIN = 1;
    downloadingPallettoStorage = 0;
    Serial.println("Pallet is uploaded from Pallet_Storage to Transport");
    delay (10000);
    digitalWrite(LedPallet_BufferMillIN, HIGH);
    Serial.println("Pallet_BufferMillIN is engaged (Token=0 - switch38=ON");
    delay (15000);
    digitalWrite(LedTransport, HIGH);
    Serial.println("Transport is engaged (Token=0 - switch24=ON)");
    delay (15000);
    digitalWrite(LedPallet_Storage, LOW);
    Serial.println("Pallet_Storage is available (Token=1 - switch40=OFF)");
    delay (15000);
    switchTransport = digitalRead(Transport);
    switchPallet_BufferMillIN = digitalRead(Pallet_BufferMillIN);
    if (uploadingPallettoBufferMiillIN == 1 &&  switchTransport == HIGH &&
switchPallet_BufferMillIN == HIGH){
      downloadingPallettoBufferMiillIN = 1;
      uploadingPallettoBufferMiillIN = 0;
      Serial.println("Pallet is downloaded from Transport to Pallet_BufferMillIN");
      delay (10000);
      digitalWrite(LedTransport, LOW);
      Serial.println("Transport is available (Token=1 - switch24=OFF) ");
      delay (15000);
    }
  }
  else {
    Serial.println("No pallets from Pallet_Storage to Pallet_BufferMillIN or
Transport is engaged");
  }
  delay (5000);

  // Transport Pallet from PalletBufferMillIN to Pallet_CheckID
  if (downloadingPallettoBufferMiillIN == 1 && switchMill == LOW && switchRobot ==
LOW && switchPallet_BufferMillIN == HIGH){
    downloadingPallettoBufferMiillIN = 0;
    Serial.println("Pallet is uploaded from PalletBufferMillIN to ID_pos by Robot
for: --- ID CHECK PROCESSING!!!---");
    delay (10000);
    // Engage Robot for transfering the pallet to CheckID
    digitalWrite(LedRobot, HIGH);
    Serial.println("Robot is engaged (Token=0 - switch22=ON)");
    delay (15000);
    digitalWrite(LedPallet_BufferMillIN, LOW);
    Serial.println("Pallet_BufferMillIN is available (Token=1 - switch38=OFF)");
    delay (15000);
    /* START OF PALLET'S ID CHECK PROCESSING*/
    // This message will not be seen in the regular code, just here to decide the ID
!!!
    Serial.println("CHANGE ID_pos, switch20=ON for Wrong Pallet OR switch20=OFF for
Correct Pallet");
    delay (15000);
    switchID_pos = digitalRead(ID_pos);
    if (switchID_pos == HIGH){
      digitalWrite(LedID_pos, HIGH);
      Serial.println("The Pallet's ID is WRONG, so switch20(ID_pos)=ON !!!");
      Serial.println("MOVE THE PALLET TO BUFFER WRONG_PALLET");
      delay (10000);
      switchWrong_Pallet = digitalRead(Wrong_Pallet);
      while (switchWrong_Pallet == HIGH){
        digitalWrite(LedWrong_Pallet, HIGH);
```

```
        Serial.println("Wrong_Pallet is engaged, release it rapidly by turning
switch34=OFF and LedWrong_Pallet=LOW");
        delay (10000);
        switchWrong_Pallet = digitalRead(Wrong_Pallet);
      }
      if (switchWrong_Pallet == LOW){
        digitalWrite(LedWrong_Pallet, LOW);
        Serial.println("Transfer the pallet to Wrong_Pallet");
        delay(10000);
        // Free ID_pos and Robot
        digitalWrite(LedID_pos, LOW);
        Serial.println("The wrong id pallet has been shifted, so ID_pos=LOW -
switch20=OFF !!!");
        delay (15000);
        digitalWrite(LedRobot, LOW);
        Serial.println("Robot is available (Token=1 - switch22=OFF)");
        delay (15000);
        // Engage Wrong_Pallet
        digitalWrite(LedWrong_Pallet,HIGH);
        Serial.println("Wrong_Pallet is engaged (Token=0 - switch34=ON), Release it
!!!");
        delay (10000); // Here we remain 10 sec and release the wrong pallet! Not in
Reality !!!
        digitalWrite(LedWrong_Pallet,LOW);
        Serial.println("OK! Wrong_Pallet has been released !!!");
        Serial.println("Wrong_Pallet is available (Token=1 - switch34=OFF)");
        delay (10000);
      }
    }
    else {
      uploadingPallettoIDpos = 1; // --SOS-- Only when the pallet's ID is correct, we
go on the procedure !!!!
      Serial.println("THE PALLET IS CORRECT (so ID-pos switch20=OFF), GO ON!!!");
      delay (10000);
      // Only when pallet's id is correct, we engage Mill and Robot remains engaged
      digitalWrite(LedMill, HIGH);
      Serial.println("Mill is engaged (Token=0 - switch18=ON");
      delay (15000);
    }
    Serial.println("END OF PALLET'S ID CHECK");
  }/* END OF PALLET'S ID CHECK PROCESSING*/

  // Transport Pallet from Pallet ID_pos to Mill, with Robot and Mill already engaged
  if (uploadingPallettoIDpos == 1 && switchRobot == HIGH && switchMill == HIGH){
    downloadingPallettoMill = 1;
    uploadingPallettoIDpos = 0;
    Serial.println("Pallet is downloaded from ID_pos to Mill by Robot");
    delay (10000);
    digitalWrite(LedRobot, LOW);
    Serial.println("Robot is available (Token=1 - switch22=OFF");
    delay (15000);

    // Milling Process for Pallet
    Milling = 1; // Start of Milling Process
    downloadingPallettoMill = 0;
    Serial.println("Pallet begins the Milling Process");
    delay (10000); // We  simulate Milling Process
    Milling = 0; // End of Milling Process
    Serial.println("Pallet ends the Milling Process, but Mill is still engaged");
    delay (10000);
    // After the Milling Process, Hold on the Pallet in Mill
```

```
    HoldOnPalletMilling = 1;
  }

  // Transport Pallet from Mill to Pallet_BufferMillOUT
  if (HoldOnPalletMilling == 1 && switchRobot == LOW && switchMill == HIGH &&
switchPallet_BufferMillOUT == LOW){
    uploadingPallettoBufferMillOUT = 1;
    HoldOnPalletMilling = 0;
    Serial.println("Pallet is uploaded from Mill to Pallet_BufferMillOUT by Robot");
    delay (10000);
    // Engage Robot for transfering the pallet to Pallet_BufferMillOUT
    digitalWrite(LedRobot, HIGH);
    Serial.println("Robot is engaged (Token=0 - switch22=ON)");
    delay (15000);
    digitalWrite(LedPallet_BufferMillOUT, HIGH);
    Serial.println("Pallet_BufferMillOUT is engaged (Token=0 - switch36=ON");
    delay (15000);
    // Free Mill
    digitalWrite(LedMill, LOW);
    Serial.println("Mill is available (Token=1 - switch18=OFF");
    delay (15000);
    // Free Robot after transfering the pallet to Pallet_BufferMillOUT
    digitalWrite(LedRobot, LOW);
    Serial.println("Robot is available (Token=1 - switch22=OFF)");
    delay (15000);
  }

  // Transport Pallet from Pallet_BufferMillOUT to Pallet_BufferLatheIN by Robot
  if (uploadingPallettoBufferMillOUT == 1 && switchRobot == LOW &&
switchPallet_BufferMillOUT == HIGH && switchPallet_BufferLatheIN == LOW){
    downloadingPallettoBufferLatheIN = 1;
    uploadingPallettoBufferMillOUT = 0;
    Serial.println("Pallet is downloaded from Pallet_BufferMillOUT to
Pallet_BufferLatheIN");
    delay (10000);
    digitalWrite(LedPallet_BufferLatheIN, HIGH);
    Serial.println("Pallet_BufferLatheIN is engaged (Token=0 - switch30=ON)");
    delay (15000);
    // Engage Robot for transfering the pallet to Pallet_BufferLatheIN
    digitalWrite(LedRobot, HIGH);
    Serial.println("Robot is engaged (Token=0 - switch22=ON");
    delay (15000);
    digitalWrite(LedPallet_BufferMillOUT, LOW);
    Serial.println("Pallet_BufferMillOUT is available (Token=1 - switch36=OFF)");
    delay (15000);
    // Free Robot after transfering the pallet to Pallet_BufferLatheIN
    digitalWrite(LedRobot, LOW);
    Serial.println("Robot is available (Token=1 - switch22=OFF)");
    delay (15000);
  }

  // Transport Pallet from Pallet_BufferLatheIN to Lathe
  if (downloadingPallettoBufferLatheIN == 1 && switchRobot == LOW &&
switchPallet_BufferLatheIN == HIGH && switchLathe == LOW){
    uploadingPallettoLathe = 1;
    downloadingPallettoBufferLatheIN = 0;
    Serial.println("Pallet is uploaded from Pallet_BufferLatheIN to Lathe");
    delay (10000);
    digitalWrite(LedLathe, HIGH);
    Serial.println("Lathe is engaged (Token=0 - switch16=ON");
    delay (15000);
```

```
    // Engage Robot for transfering the pallet to Lathe
    digitalWrite(LedRobot, HIGH);
    Serial.println("Robot is engaged (Token=0 - switch22=ON)");
    delay (15000);
    digitalWrite(LedPallet_BufferLatheIN, LOW);
    Serial.println("Pallet_BufferLatheIN is available (Token=1 - switch30=OFF)");
    delay (15000);
    // Free Robot after transfering the pallet to Pallet_BufferLatheIN
    digitalWrite(LedRobot, LOW);
    Serial.println("Robot is available (Token=1 - switch22=OFF");
    delay (15000);
  }

  // Turning Process for Pallet
  if (uploadingPallettoLathe == 1 && switchLathe == HIGH){
    // Turning Process for Pallet
    Turning = 1; // Start of Turning Process
    uploadingPallettoLathe = 0;
    Serial.println("Pallet begins the Turning Process");
    delay (10000);
    Turning = 0; // End of Turning Process
    Serial.println("Pallet ends the Turning Process, but Lathe is still engaged");
    delay (10000);
    // After the Turning Process, Hold on the Pallet in Lathe
    HoldOnPalletTurning = 1;
  }

  // Transport Pallet from Lathe to Pallet_BufferLatheOUT
  switchRobot = digitalRead(Robot); // Here it is needed again, in the case where we
have multiple units
  if (HoldOnPalletTurning == 1 && switchRobot == LOW && switchLathe == HIGH &&
switchPallet_BufferLatheOUT == LOW){
    downloadingPallettoBufferLatheOUT = 1;
    HoldOnPalletTurning = 0;
    Serial.println("Pallet is downloaded from Lathe to Pallet_BufferLatheOUT by
Robot");
    delay (10000);
    // Engage Robot for transfering the pallet to Pallet_BufferLatheOUT
    digitalWrite(LedRobot, HIGH);
    Serial.println("Robot is engaged (Token=0 - switch22=ON");
    delay (15000);
    digitalWrite(LedPallet_BufferLatheOUT, HIGH);
    Serial.println("Pallet_BufferLatheOUT is engaged (Token=0 - switch28=ON");
    delay (15000);
    // Free Lathe
    digitalWrite(LedLathe, LOW);
    Serial.println("Lathe is available (Token=1 - switch16=OFF");
    delay (15000);
    // Free Robot after transfering the pallet to Pallet_BufferLatheOUT
    digitalWrite(LedRobot, LOW);
    Serial.println("Robot is available (Token=1 - switch22=OFF");
    delay (15000);
  }

  // Transport Pallet from Pallet_BufferLatheOUT to Pallet_Finish
  if (downloadingPallettoBufferLatheOUT == 1 && switchTransport == LOW &&
switchPallet_BufferLatheOUT == HIGH && switchPallet_Finish == LOW){
    uploadingPallettoFinish = 1;
    downloadingPallettoBufferLatheOUT = 0;
    Serial.println("Pallet is uploaded from Pallet_BufferLatheOUT to Transport");
    delay (10000);
```

```
    // Engage Pallet_Finish
    digitalWrite(LedPallet_Finish, HIGH);
    Serial.println("Pallet_Finish is engaged (Token=0 - switch26=ON");
    delay (15000);
    digitalWrite(LedTransport, HIGH);
    Serial.println("Transport is engaged (Token=0 - switch24=ON)");
    delay (15000);
    digitalWrite(LedPallet_BufferLatheOUT, LOW);
    Serial.println("Pallet_BufferLatheOUT is available (Token=1 - switch28=OFF)");
    delay (15000);
    switchTransport = digitalRead(Transport);
    switchPallet_Finish = digitalRead(Pallet_Finish);
    if (uploadingPallettoFinish == 1 &&  switchTransport == HIGH &&
switchPallet_Finish == HIGH){
      downloadingPallettoFinish = 1;
      uploadingPallettoFinish = 0;
      Serial.println("Pallet is downloaded from Transport to Pallet_Finish");
      delay (10000);
      digitalWrite(LedTransport, LOW);
      Serial.println("Transport is available (Token=1 - switch24=OFF) ");
      delay (15000); // Here we remain 15 sec and release the Pallet_Finish! Not in
Reality !!!
      digitalWrite(LedPallet_Finish, LOW);
      Serial.println("Pallet_Finish is available (Token=1 - switch26=OFF");
      Serial.println("OK! Pallet_Finish has been released!!!");
      delay (15000);
      downloadingPallettoFinish = 0;
    }
  }/* End of Pallets' Processing */


  /*C. Start of Units' Processing
     If there are not any Tools for transport, and Pallets for Processing, go on with
UNITS*/
  // Transport Unit from Unit_Storage to Unit_BufferMillIN until units.
Pallet_station, Tool_Mill and Tool_Lathe simulate the deterant branches
  if (switchTool_Mill == HIGH && switchTool_Lathe == HIGH && switchPallet_station ==
HIGH && enterUnits == 1){
    Serial.println("Begin Units Processing!!! No Tools for Transportation nor Pallets
for Processing!");
    delay (10000);
    endofPallets = 1;
    enterUnits = 0; // For not entering again in this if_condition!!!
  }
  delay (5000);

  // Transport Unit from Unit_Storage to Unit_BufferMillIN
  if (switchUnit_Storage == LOW && switchTransport == LOW && switchUnit_BufferMillIN
== LOW && endofPallets == 1){
    Serial.println("Unit is uploaded from Unit_Storage to Transport");
    delay (10000);
    uploadingUnittoBufferMillIN = 1;
    // Unit_Storage = HIGH when counterUnit = units
    if (counterUnit == units) {
      digitalWrite(LedUnit_Storage, HIGH);
      Serial.println("Unit_Storage has no more units (Token=0 - switch10=ON) ");
      delay (15000);
    }
    // Increase counterUnit by one in each loop
    counterUnit = counterUnit + 1;
    digitalWrite(LedUnit_BufferMillIN, HIGH);
```

```
    Serial.println("Unit_BufferMillIN is engaged (Token=0 - switch8=ON)");
    delay (15000);
    digitalWrite(LedTransport, HIGH);
    Serial.println("Transport is engaged (Token=0 - switch24=ON)");
    delay (15000);
    // Unit_Storage for units is like Pallet_Station for pallets, so we don't turn it
LOW.
    // It's always LOW, until there are no more units and we turn it into HIGH!!!
    switchTransport = digitalRead(Transport);
    switchUnit_BufferMillIN = digitalRead(Unit_BufferMillIN);
    if (uploadingUnittoBufferMillIN == 1 && switchTransport == HIGH &&
switchUnit_BufferMillIN == HIGH){
      downloadingUnittoBufferMillIN = 1;
      uploadingUnittoBufferMillIN = 0;
      Serial.println("Unit is downloaded from Transport to UnittoBufferMillIN");
      delay (10000);
      digitalWrite(LedTransport, LOW);
      Serial.println("Transport is available (Token=1 - switch24=OFF) ");
      delay (15000);
    }
  }
  else {
    Serial.println("No units from Unit_Storage to Unit_BufferMillIN or Transport is
engaged");
    delay (5000);
  }

  // Transport Unit from UnitBufferMillIN to Unit_CheckID
  if (downloadingUnittoBufferMillIN == 1 && switchMill == LOW && switchRobot == LOW
&& switchUnit_BufferMillIN == HIGH && endofPallets == 1){
    downloadingUnittoBufferMillIN  = 0;
    Serial.println("Unit is uploaded from UnitBufferMillIN to ID_pos by Robot for: --
- ID CHECK PROCESSING!!!---");
    delay (10000);
    // Engage Robot for transfering the Unit to CheckID
    digitalWrite(LedRobot, HIGH);
    Serial.println("Robot is engaged (Token=0 - switch22=ON)");
    delay (15000);
    digitalWrite(LedUnit_BufferMillIN, LOW);
    Serial.println("Unit_BufferMillIN is available (Token=1 - switch8=OFF)");
    delay (15000);
    /* START OF UNIT'S ID CHECK PROCESSING*/
    // This message will not be seen in the regular code, just here to decide the ID
!!!
    Serial.println("CHANGE ID_pos, switch20=ON for Wrong Unit OR switch20=OFF for
Correct Unit");
    delay (15000);
    switchID_pos = digitalRead(ID_pos);
    if (switchID_pos == HIGH){
      digitalWrite(LedID_pos, HIGH);
      Serial.println("The Unit's ID is WRONG, so switch20(ID_pos)=ON !!!");
      Serial.println("MOVE THE UNIT TO BUFFER WRONG_UNIT");
      delay (10000);
      // In the code, Wrong_Unit has different meaning from the PN
      // HIGH (token=0) means engaged, that it has unit and it's not available, but
in the PN token=1 means that it has unit!!!
      switchWrong_Unit = digitalRead(Wrong_Unit);
      while (switchWrong_Unit == HIGH){
        digitalWrite(LedWrong_Unit, HIGH);
        Serial.println("Wrong_Unit is engaged, release it rapidly by turning
switch4=OFF and LedWrong_Unit=LOW");
```

```
        delay (10000);
        switchWrong_Unit = digitalRead(Wrong_Unit);
      }
      if (switchWrong_Unit == LOW){
        digitalWrite(LedWrong_Unit, LOW);
        Serial.println("Transfer the Unit to Wrong_Unit");
        delay(10000);
        // Free ID_pos and Robot
        digitalWrite(LedID_pos, LOW);
        Serial.println("The wrong id unit has been shifted, so ID_pos=LOW -
switch20=OFF !!!");
        delay (15000);
        digitalWrite(LedRobot, LOW);
        Serial.println("Robot is available (Token=1 - switch22=OFF)");
        delay (15000);
        // Engage Wrong_Unit
        digitalWrite(LedWrong_Unit,HIGH);
        Serial.println("Wrong_Unit is engaged (Token=0 - switch4=ON), Release it
!!!");
        delay (10000); // Here we remain 10 sec and release the wrong unit! Not in
Reality !!!
        digitalWrite(LedWrong_Unit,LOW);
        Serial.println("OK! Wrong_Unit has been released !!!");
        Serial.println("Wrong_Unit is available (Token=1 - switch4=OFF)");
        delay (10000);
      }
    }
    else {
      uploadingUnittoIDpos = 1; // --SOS-- Only when the unit's ID is correct, we go
on the procedure !!!!
      Serial.println("THE UNIT IS CORRECT (so ID-pos switch20=OFF), GO ON!!!");
      delay (10000);
      // Only when unit's id is correct, we engage Mill and Robot remains engaged
      digitalWrite(LedMill, HIGH);
      Serial.println("Mill is engaged (Token=0 - switch18=ON");
      delay (15000);
    }
    Serial.println("END OF UNIT'S ID CHECK");
    delay (5000);
  }/* END OF UNIT'S ID CHECK PROCESSING*/

  // Transport Unit from Unit ID_pos to Mill, with Robot and Mill already engaged
  if (uploadingUnittoIDpos == 1 && switchRobot == HIGH && switchMill == HIGH &&
endofPallets == 1){
    downloadingUnittoMill = 1;
    uploadingUnittoIDpos = 0;
    Serial.println("Unit is downloaded from ID_pos to Mill by Robot");
    delay (10000);
    digitalWrite(LedRobot, LOW);
    Serial.println("Robot is available (Token=1 - switch22=OFF");
    delay (15000);

    // Milling Process for Unit
    Milling = 1; // Start of Milling Process
    downloadingUnittoMill = 0;
    Serial.println("Unit begins the Milling Process");
    delay (10000);  // We  simulate Milling Process
    Milling = 0; // End of Milling Process
    Serial.println("Unit ends the Milling Process, but Mill is still engaged");
    delay (10000);
    // After the Milling Process, Hold on the Unit in Mill
```

```
    HoldOnUnitMilling = 1;
  }

  // Transport Unit from Mill to Unit_BufferMillOUT
  if (HoldOnUnitMilling == 1 && switchRobot == LOW && switchMill == HIGH
&&  switchUnit_BufferMillOUT == LOW &&  endofPallets == 1){
    uploadingUnittoBufferMillOUT = 1;
    HoldOnUnitMilling = 0;
    Serial.println("Unit is uploaded from Mill to Unit_BufferMillOUT by Robot");
    delay (10000);
    // Engage Robot for transfering the Unit to Unit_BufferMillOUT
    digitalWrite(LedRobot, HIGH);
    Serial.println("Robot is engaged (Token=0 - switch22=ON)");
    delay (15000);
    digitalWrite(LedUnit_BufferMillOUT, HIGH);
    Serial.println("Unit_BufferMillOUT is engaged (Token=0 - switch6=ON");
    delay (15000);
    // Free Mill
    digitalWrite(LedMill, LOW);
    Serial.println("Mill is available (Token=1 - switch18=OFF");
    delay (15000);
    // Free Robot after transfering the Unit to Unit_BufferMillOUT
    digitalWrite(LedRobot, LOW);
    Serial.println("Robot is available (Token=1 - switch22=OFF)");
    delay (15000);
  }

  // Transport Unit from Unit_BufferMillOUT to Unit_BufferLatheIN by Robot
  switchRobot = digitalRead(Robot); // Here it is needed again, in the case where we
have multiple units
  if (uploadingUnittoBufferMillOUT == 1 && switchRobot == LOW &&
switchUnit_BufferMillOUT == HIGH && switchUnit_BufferLatheIN == LOW && endofPallets
== 1){
    downloadingUnittoBufferLatheIN = 1;
    uploadingUnittoBufferMillOUT = 0;
    Serial.println("Unit is downloaded from Unit_BufferMillOUT to
Unit_BufferLatheIN");
    delay (10000);
    digitalWrite(LedUnit_BufferLatheIN, HIGH);
    Serial.println("Unit_BufferLatheIN is engaged (Token=0 - switch2=ON)");
    delay (15000);
    // Engage Robot for transfering the unit to Unit_BufferLatheIN
    digitalWrite(LedRobot, HIGH);
    Serial.println("Robot is engaged (Token=0 - switch22=ON)");
    delay (15000);
    digitalWrite(LedUnit_BufferMillOUT, LOW);
    Serial.println("Unit_BufferMillOUT is available (Token=1 - switch6=OFF)");
    delay (15000);
    // Free Robot after transfering the unit to Unit_BufferLatheIN
    digitalWrite(LedRobot, LOW);
    Serial.println("Robot is available (Token=1 - switch22=OFF)");
    delay (15000);
  }

  // Transport Unit from Unit_BufferLatheIN to Lathe
  if (downloadingUnittoBufferLatheIN == 1 && switchRobot == LOW &&
switchUnit_BufferLatheIN == HIGH && switchLathe == LOW && endofPallets == 1){
    uploadingUnittoLathe = 1;
    downloadingUnittoBufferLatheIN = 0;
    Serial.println("Unit is uploaded from Unit_BufferLatheIN to Lathe");
    delay (10000);
```

```
    digitalWrite(LedLathe, HIGH);
    Serial.println("Lathe is engaged (Token=0 - switch16=ON");
    delay (15000);
    // Engage Robot for transfering the unit to Lathe
    digitalWrite(LedRobot, HIGH);
    Serial.println("Robot is engaged (Token=0 - switch22=ON)");
    delay (15000);
    digitalWrite(LedUnit_BufferLatheIN, LOW);
    Serial.println("Unit_BufferLatheIN is available (Token=1 - switch2=OFF)");
    delay (15000);
    // Free Robot after transfering the unit to Unit_BufferLatheIN
    digitalWrite(LedRobot, LOW);
    Serial.println("Robot is available (Token=1 - switch22=OFF");
    delay (15000);
  }

  // Turning Process for Unit
  if (uploadingUnittoLathe == 1 && switchLathe == HIGH && endofPallets == 1){
    Turning = 1; // Start of Turning Process
    uploadingUnittoLathe = 0;
    Serial.println("Unit begins the Turning Process");
    delay (10000);
    Turning = 0; // End of Turning Process
    Serial.println("Unit ends the Turning Process, but Lathe is still engaged");
    delay (10000);
    // After the Turning Process, Hold on the Unit in Lathe
    HoldOnUnitTurning = 1;
  }

  // Transport Unit from Lathe to Unit_BufferLatheOUT
  if (HoldOnUnitTurning == 1 && switchRobot == LOW && switchLathe == HIGH &&
switchUnit_BufferLatheOUT == LOW && endofPallets == 1){
    downloadingUnittoBufferLatheOUT = 1;
    HoldOnUnitTurning = 0;
    Serial.println("Unit is downloaded from Lathe to Unit_BufferLatheOUT by Robot");
    delay (10000);
    // Engage Robot for transfering the unit to Unit_BufferLatheOUT
    digitalWrite(LedRobot, HIGH);
    Serial.println("Robot is engaged (Token=0 - switch22=ON");
    delay (15000);
    digitalWrite(LedUnit_BufferLatheOUT, HIGH);
    Serial.println("Unit_BufferLatheOUT is engaged (Token=0 - switch0=ON)");
    delay (15000);
    // Free Lathe
    digitalWrite(LedLathe, LOW);
    Serial.println("Lathe is available (Token=1 - switch16=OFF)");
    delay (15000);
    // Free Robot after transfering the unit to Unit_BufferLatheOUT
    digitalWrite(LedRobot, LOW);
    Serial.println("Robot is available (Token=1 - switch22=OFF)");
    delay (15000);
  }

  // Transport Unit from Unit_BufferLatheOUT to Unit_Finish
  if (downloadingUnittoBufferLatheOUT == 1 && switchTransport == LOW &&
switchUnit_BufferLatheOUT == HIGH && switchUnit_Finish == LOW && endofPallets == 1){
    uploadingUnittoFinish = 1;
    downloadingUnittoBufferLatheOUT = 0;
    Serial.println("Unit is uploaded from Unit_BufferLatheOUT to Transport");
    delay (10000);
    // Engage Unit_Finish
```

```
    digitalWrite(LedUnit_Finish, HIGH);
    Serial.println("Unit_Finish is engaged (Token=0 - switch14=ON)");
    delay (15000);
    digitalWrite(LedTransport, HIGH);
    Serial.println("Transport is engaged (Token=0 - switch24=ON)");
    delay (15000);
    digitalWrite(LedUnit_BufferLatheOUT, LOW);
    Serial.println("Unit_BufferLatheOUT is available (Token=1 - switch0=OFF)");
    delay (15000);
    switchTransport = digitalRead(Transport);
    switchUnit_Finish = digitalRead(Unit_Finish);
    if (uploadingUnittoFinish == 1 &&  switchTransport == HIGH && switchUnit_Finish
== HIGH){
      downloadingUnittoFinish = 1;
      uploadingUnittoFinish = 0;
      Serial.println("Unit is downloaded from Transport to Unit_Finish");
      delay (10000);
      digitalWrite(LedTransport, LOW);
      Serial.println("Transport is available (Token=1 - switch24=OFF) ");
      delay (15000); // Here we remain 15 sec and release the Unit_Finish! Not in
Reality !!!
      digitalWrite(LedUnit_Finish, LOW);
      Serial.println("Unit_Finish is available (Token=1 - switch14=OFF)");
      Serial.println("OK! Unit_Finish has been released!!!");
      delay (15000);
      downloadingUnittoFinish = 0;
    }
  }/* End of Pallets' Processing */


  // Led for checking the loops
  digitalWrite(13, LOW);
  Serial.println("------------  PRINTING IN EACH LOOP ------------");
  delay(5000);
}


/********** END OF CODE **********/
```