

Βιομηχανική Πληροφορική
Τελική Εργασία 2021
18/01/2021



**Πετρίδης Δημητρίου Παναγιώτης
ΑΕΜ 9286**

Περιεχόμενα

A. Εισαγωγή	4
B. Θέμα Εργασίας	5
Γ. Υλοποίηση σε CoDeSys	6
Initialize	7
delayBeforeStart	7
emptySupply	8
door2	9
door	9
active running	10
full bottle running	10
capped bottle running	11
labeled bottle running	11
watering	11
capping	12
labeling	13
stopConveyerCondition	13
restart	14
terminal	15
alarm_on	15
POU	16
START_STOP	17
GVL_CAPI_INVISIBLE	18
BANK	19
BANKED_BOTTLE_XYPOSITION	20
GVL_LIST	21
VISUALIZATION	22
Προσομοίωση	23
Εισαγωγή αριθμού μπουκαλιών	23
Σε φάση λειτουργίας μηχανών	24
Λίγο πριν λήξει η διαδικασία	24

Λήξη διαδικασίας	25
Ανεπαρκής τροφοδοσία	25
Σενάριο γεμάτης τράπεζας απόθεσης, χωρίς να έχει ολοκληρωθεί η διαδικασία.	26
Γ. Υλοποίηση κατανεμημένου δικτύου υπολογιστών.	26
Πρώτος υπολογιστής	26
Σύνδεση 2ου μπλοκ	28
Σύνδεση 2ου υπολογιστή	29

A. Εισαγωγή

Στη σημερινή εποχή παρατηρείται μια ραγδαία αύξηση των εργοστασίων και γενικότερα των παραγωγικών μονάδων, καθώς οι αυξανόμενες απαιτήσεις σε ποσότητα, ποιότητα και φυσικά ταχύτητα, θέτουν νέες προκλήσεις στο βιομηχανικό τομέα. Η λεγόμενη 4^η βιομηχανική επανάσταση επαναπροσδιόρισε την σχέση ανθρώπου-μηχανής και δημιούργησε νέα πρότυπα στην τεχνολογία κατασκευής, υλοποίησης, διαχείρισης και συντήρησης συστημάτων βιομηχανικού αυτοματισμού και ελέγχου. Σύμφωνα με το πρότυπο **Industry 4.0**, το οποίο εισήχθη σαν ιδέα το 2011 στο **Hannover Messe**, ένα βιομηχανικό σύστημα θα πρέπει να ακολουθεί τις παρακάτω αρχές.

- ❖ **Interconnection** — Η ικανότητα των μηχανών, συσκευών, αισθητήρων και ανθρώπων να συνδέονται και να επικοινωνούν, μεταξύ τους μέσω του Διαδικτύου των Πραγμάτων (**IoT**) ή μέσω του Διαδικτύου των Ανθρώπων (**IoP**).
- ❖ **Information transparency** - Η διαφάνεια που παρέχει η τεχνολογία Industry 4.0 παρέχει στους χειριστές ολοκληρωμένες πληροφορίες για τη λήψη αποφάσεων. Η διασύνδεση επιτρέπει στους χειριστές να συλλέγουν τεράστιες ποσότητες δεδομένων και πληροφοριών από όλα τα σημεία της διαδικασίας κατασκευής, να προσδιορίζουν βασικούς τομείς που μπορούν να επωφεληθούν από τη βελτίωση για να αυξήσουν τη λειτουργικότητα.
- ❖ **Technical assistance** - Η τεχνολογική δυνατότητα συστημάτων που βοηθούν τους ανθρώπους στη λήψη αποφάσεων και στην επίλυση προβλημάτων και η ικανότητα να βοηθούν τους ανθρώπους με δύσκολες ή μη ασφαλείς εργασίες.

❖ **Decentralized decisions** - Η ικανότητα των φυσικών συστημάτων στον κυβερνοχώρο να λαμβάνουν αποφάσεις μόνες τους και να εκτελούν τα καθήκοντά τους όσο το δυνατόν πιο αυτόνομα. Μόνο στην περίπτωση εξαιρέσεων, παρεμβολών ή αντικρουόμενων στόχων, οι εργασίες ανατίθενται σε υψηλότερο επίπεδο.

To **Industry 4.0** έχει σαν βασικό χαρακτηριστικό την ψηφιοποίηση όλης της πληροφορίας που βρίσκεται μέσα σε ένα σύστημα ελέγχου και αυτοματοποίησης βιομηχανίας. Μερικά απλά παραδείγματα από την καθημερινότητα μας που συνιστούν την έννοια του Industry 4.0 είναι:

- Κινητές συσκευές.
- Πλατφόρμες (IoT)
- 3D printing
- Έξυπνοι αισθητήρες
- On-demand διαθεσιμότητα υπολογιστικών πόρων

Άλλες εφαρμογές περιλαμβάνουν μηχανές που μπορούν να προβλέψουν αστοχίες και να ενεργοποιούν διαδικασίες συντήρησης αυτόνομα ή αυτο-οργανωμένο συντονισμό ως αντίδραση σε απροσδόκητες αλλαγές στην παραγωγή.

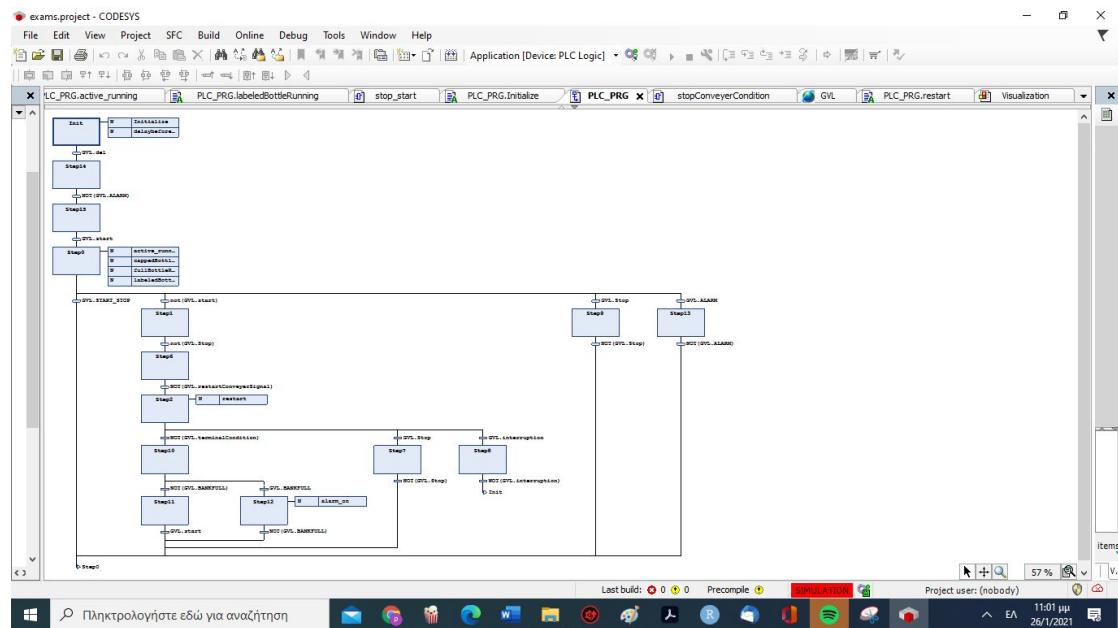
B. Θέμα Εργασίας

Στο Θέμα της παρούσας εργασίας θα υλοποιηθεί ένα πρόγραμμα προγραμματιζόμενου λογικού ελεγκτή PLC που θα λειτουργεί στα πρότυπα του Industry 4.0 για να χρησιμοποιηθεί σε ένα εμφιαλωτήριο νερού. Συγκεκριμένα θα ελέγχει την λειτουργία της μηχανής τροφοδοσίας άδειων φιαλών, του ταινιόδρομου, των μηχανών πλήρωσης, τάπωσης και ετικετοποίησης, καθώς και της μηχανής τράπεζας απόθεσης των έτοιμων φιαλών. Επιπλέον υλοποιήθηκαν ρεαλιστικές γραφικές απεικονίσεις της διεργασίας real-time καθώς και

γραφικά εργαλεία εποπτείας και χειρισμών ελέγχου. Τέλος στο δεύτερο κομμάτι της εργασίας υλοποιήθηκε το Blinker tutorial για το σύστημα επικοινωνίας κατανεμημένων συσκευών εντός βιομηχανικού δικτύου.

Γ. Υλοποίηση σε CoDeSys

Στη παρακάτω εικόνα φαίνεται μια πανοραμική άποψη του pipeline της διεργασίας.



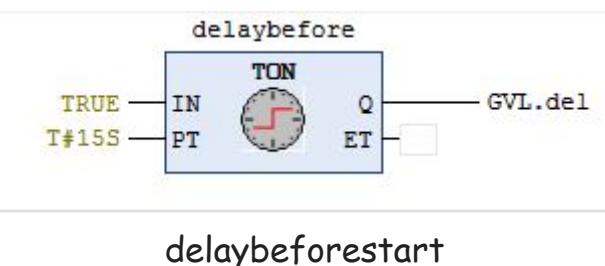
Διακρίνουμε ξεκινώντας από την αρχή το Step Αρχικοποίησης τιμών που απενεργοποιεί τις μηχανές, κρατάει το πλήθος των άδειων φιαλών που βρίσκονται στη μηχανή τροφοδοσίας και κρατάει την χωρητικότητα της τράπεζας απόθεσης. Επίσης γίνονται κάποιες αρχικοποιήσεις που έχουν να κάνουν με το Visualization και όχι με την ουσία του προγράμματος ελέγχου της διεργασίας. Παράλληλα με το action **Initialization** τρέχει και το action **delaybeforestart**, όπου προσομοιώνεται το ζέσταμα των μηχανών διάρκειας 15 δευτερολέπτων μετά το πάτημα του κουμπιού start, καθώς και δίνεται αυτό το χρονικό διάστημα, ώστε ο χειριστής να πληκτρολογήσει τον αριθμό των φιαλών που θέλει να ετοιμάσει.

Initialize

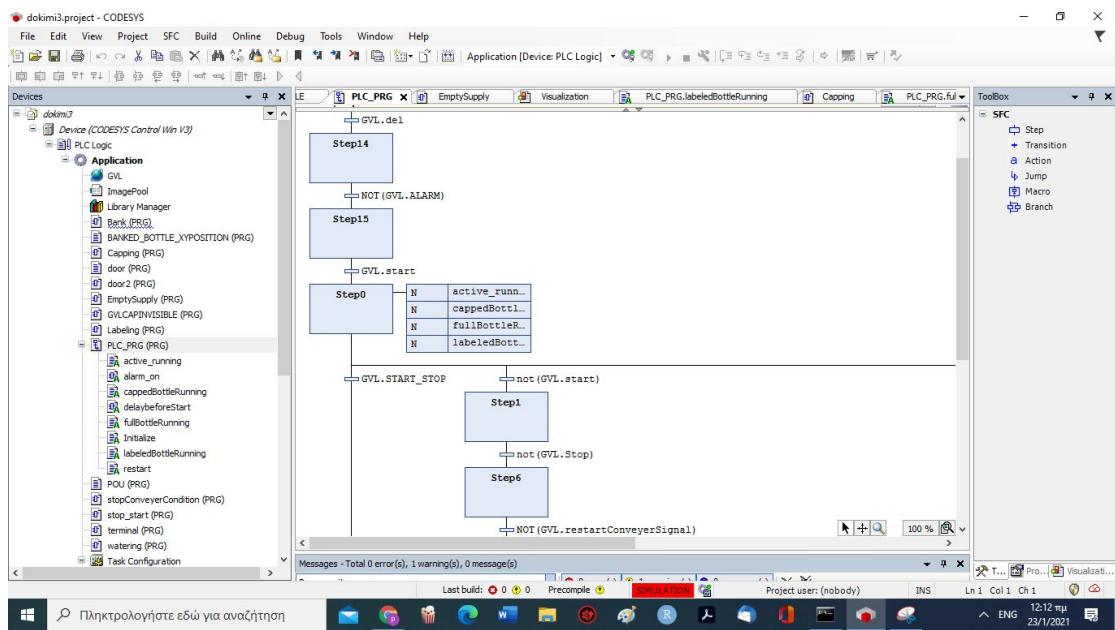
```
1 GVL.colorWater :=16#FFB0E0E6;
2 GVL.FullInvisible:=TRUE;
3 GVL.bottleInvisible:=TRUE;
4 GVL.capMachineSwitch:=FALSE;
5 GVL.CapInvisible:=TRUE;
6 GVL.LabelInvisible:=TRUE;
7 GVL.waterMachineSwitch:=FALSE
8 GVL.labelingMachineSwitch :=F
9 GVL.BankSwitcher:=FALSE;
10 GVL.BottleArsenal:=40;
11 GVL.BANKCAPACITY:=7;
12
```

Initialize

delayBeforeStart

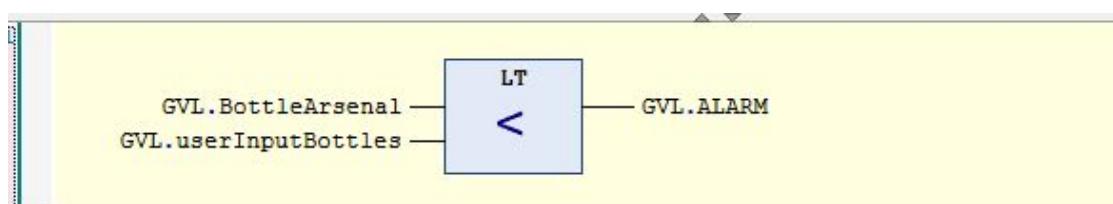


Στη συνέχεια αφού η μεταβλητή *GVL.del* γίνει αληθής, στα step 14, 15 γίνεται ο έλεγχος αν τα μπουκάλια που ζήτησε ο χειριστής καλύπτονται από την τροφοδοσία. Αυτό γίνεται από την μεταβλητή *GVL.ALARM* που τρέχει στην POU *EmptySupply*. Αν δεν αρκούν, τότε δεν ξεκινάει η διαδικασία και ο χειριστής πρέπει να βάλει λιγότερα μπουκάλια.



Τα βήματα ελέγχου διαθεσιμότητας φιαλών

emptySupply



POU emptySupply

Σύγκριση αποθήκης άδειων φιαλών με την είσοδο του χειριστή.

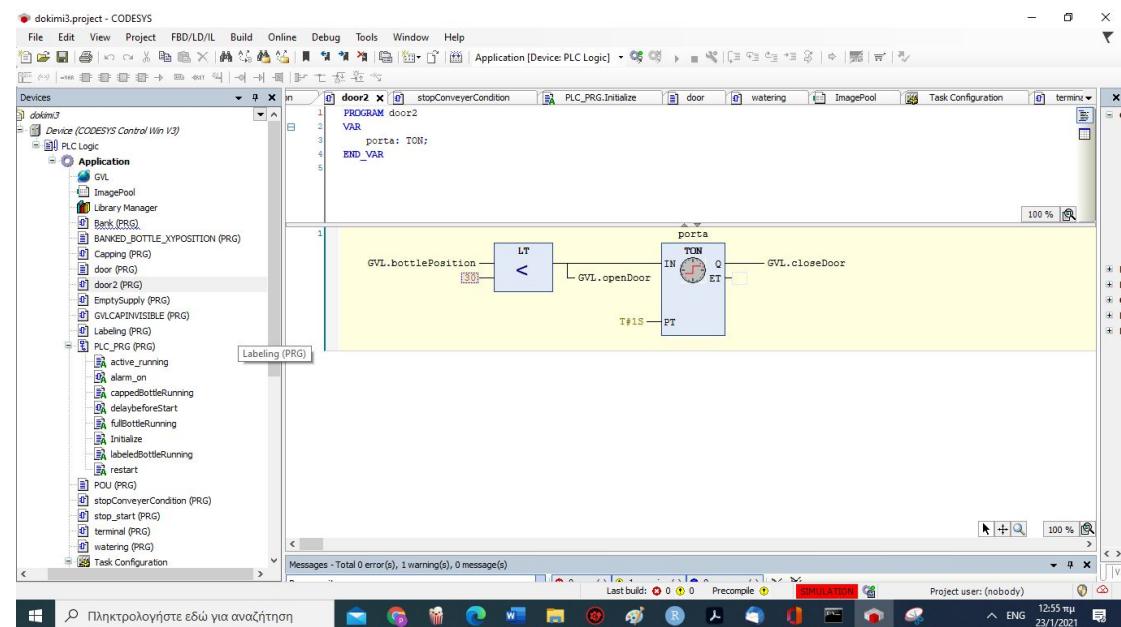
Στη συνέχεια και αφού έχει ενεργοποιηθεί ο διακόπτης GVL.start προχωράμε στο STEP0 όπου εδώ παράγεται η κίνηση των φιαλών πάνω στον ταινιόδρομο. Συγκεκριμένα εκτελούνται τα actions active_running (empty bottle), fullBottlerunning (full bottle), cappedBottlerunning (capped bottle) και labeledBottleRunning (labeled bottle).

Δε θα πρέπει να παραλείψουμε να αναφέρουμε την μοντελοποίηση του "ανοιγοκλειματος" της πόρτας της τροφοδοσίας με άδειες φιάλες.

Αρχικά συγκρίνουμε την θέση του άδειου μπουκαλιού με την θέση 30 (μια καλή θέση ώστε να έχουμε την απαιτούμενη διάρκεια παλμού στην έξοδο του συγκριτή ώστε να τρέξει το χρονοκύκλωμα

καθυστέρησης). Έτσι εδώ αρχικά ενεργοποιείται το GVL.openDoor ανοίγοντας την πόρτα και μετά από χρονοκαθυστέρηση 1 δευτερολέπτου η μεταβλητή στην έξοδο του timer γίνεται αληθής και κλείνει η πόρτα GVL.closeDoor TRUE.

door2



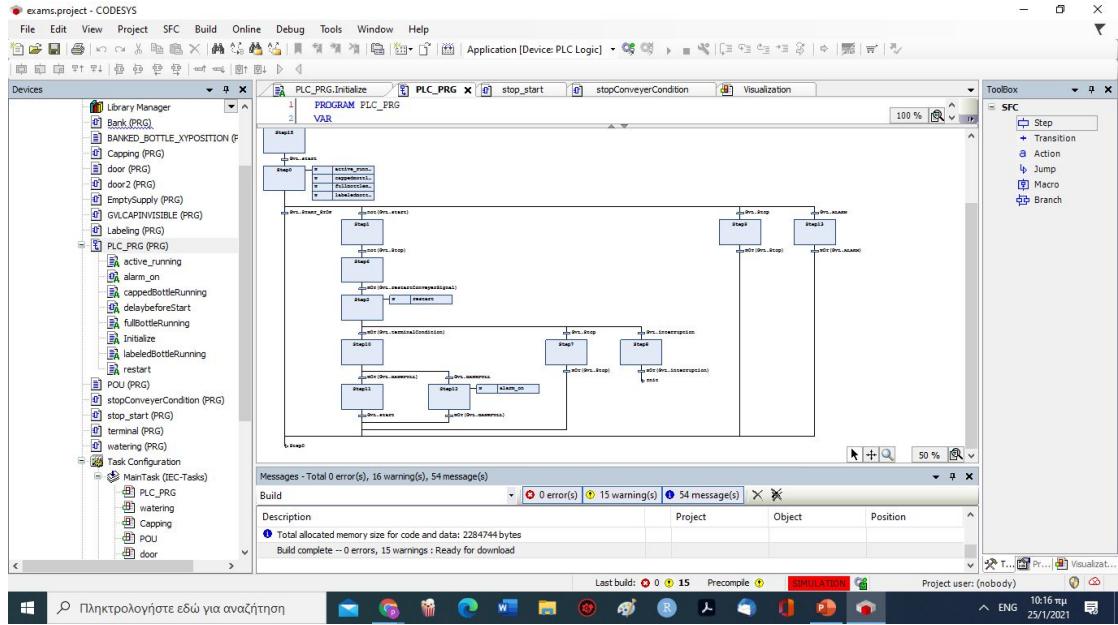
door

```

IF( GVL.openDoor) THEN
    GVL.rotDoor:=-90;
    IF(GVL.closeDoor) THEN
        GVL.rotDoor:=0;
    END_IF
END_IF

```

Εδώ η POU Door θέτει την μεταβλητή GVL.rotDoor μία -90 και μια 0 κάθε φορά που είναι να μπει ένα νέο μπουκάλι. Αυτή η μεταβλητή χρησιμοποιείται για την περιστροφή της πόρτας.



η συνέχεια της διαδρομής.

active running

```

IF( (GVL.userInputBottles>=(GVL.BANKEDBOTTLES+3)) AND (NOT(GVL.BANKFULL)) ) THEN //EDW BANKFULL
GVL.bottlePosition := GVL.bottlePosition +0.2;
END_IF
IF( GVL.START_STOP) THEN
GVL.rotWheel:=GVL.rotWheel+10;
END_IF

```

active running

ο περιορισμός στην IF μπήκε ώστε να σταματήσει να κινείται το γραφικό στοιχείο του άδειου μπουκαλιού όταν από μένουν μόνο 3 μπουκάλια για να τελειώσει η διαδικασία. Δηλαδή όταν προηγούνται 3 μπουκάλια ένα στην πλήρωση ένα στην τάπωση και ένα στην ετικετοποίηση ή να έχει φουλάρει η τράπεζα απόθεσης.

full bottle running

```

1 IF ((watering.InsertedBottles.CV>=1)AND(GVL.userInputBottles>=(2+GVL.BANKEDBOTTLES))AND(NOT(GVL.BANKFULL)) )THEN //
2   GVL.FullBottlePosition:=GVL.FullBottlePosition+0.2;
3 END_IF

```

full bottle running

Ομοίως και εδώ ο περιορισμός για να σταματήσει το γεμάτο μπουκάλι όταν απομένουν 2 μπουκάλια για τη λήξη ή έχει φουλάρει η τράπεζα

απόθεσης. Προηγούνται δηλαδή ενα μπουκαλι στην τάπωση και ένα στην ετικετοποίηση.

capped bottle running

```
1 IF ((watering.InsertedBottles.CV >=2) AND (GVL.userInputBottles>=(GVL.BANKEDBOTTLES+1)) AND (NOT (GVL.BANKFULL))) THEN  
2   GVL.CapBottlePosition:=GVL.CapBottlePosition+0.2;  
3 END_IF
```

capped bottle running

Ομοίως και εδώ για την τάπωση, σταματάει να κινείται το γραφικό του καπακωμένου μπουκαλιού, όταν θέλουμε 1 μπουκάλι μόνο για να λήξει η διαδικασία ή όταν έχει φουλάρει η τράπεζα απόθεσης..

labeled bottle running

```
IF ((watering.InsertedBottles.CV >=3) AND (GVL.userInputBottles>GVL.BANKEDBOTTLES)) THEN  
  GVL.XlabelBottlePosition:=GVL.XlabelBottlePosition+0.2;  
END_IF
```

labeled bottle running

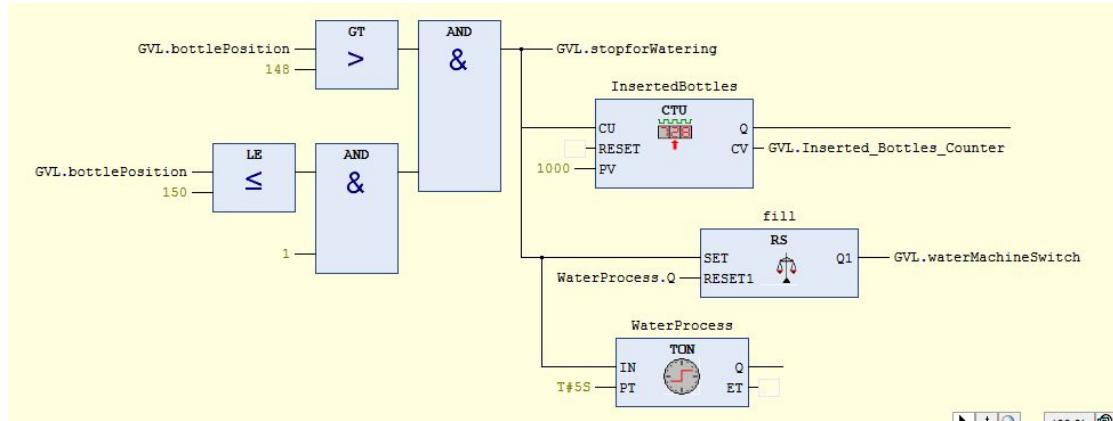
Και τέλος εδώ σταματάει την κίνηση του το γραφικό του ετικετοποιημένου μπουκαλιού όταν έχουν συμπληρωθεί τα ζητούμενα μπουκάλια στην τράπεζα απόθεσης ή έχει φουλάρει η τράπεζα απόθεσης.

Τώρα θα δούμε τις συναρτήσεις POU που μοντελοποιούν την πληρωση, τάπωση και ετικετοποίηση.

watering

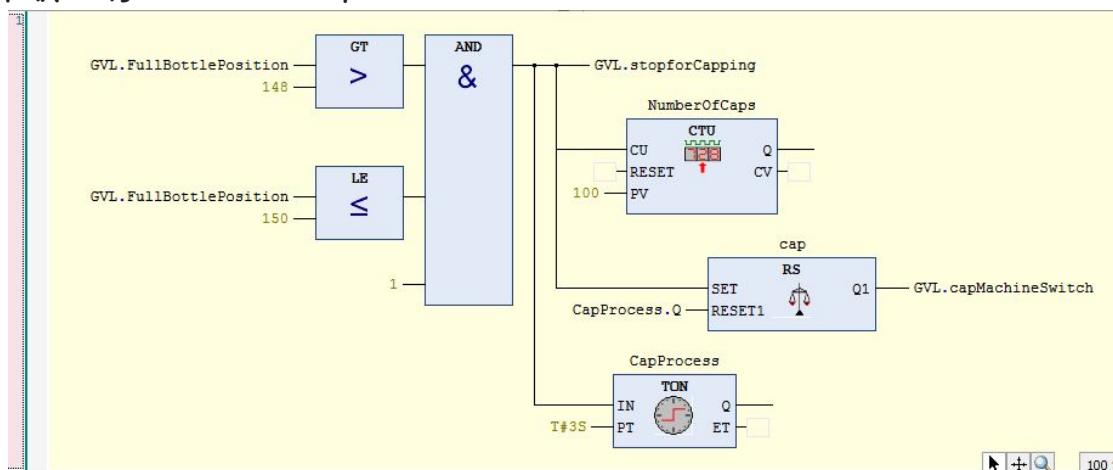
Στην παρακάτω εικόνα βλέπουμε αρχικά δυο συγκριτές που ελέγχουν την θέση του άδειου μπουκαλιού. Αυτή η θέση που τους επαληθέυει είναι η 149, η οποία επιλέχτηκε τέτοια σε συνδιασμό με την ταχύτητα του μπουκαλιού, ώστε η μετακίνηση να διαρκεί 12 δευτερόλεπτα, όσα δηλαδή και στον πραγματικό ταινιόδρομο. Ετσι λοιπόν τα σήματα

πηγαίνουν στην AND και ουσιαστικά αυτό το κομμάτι μοντελοποιεί το φωτοκύτταρο που βρίσκεται κάτω από τη μηχανή πλήρωσης. Δηλαδή το σήμα GVL.stopforWatering. Στη συνέχεια μετράμε με τον απαριθμητή τα μπουκάλια που εισήχθησαν στον ταινιόδρομο και αποθηκέυουμε στην μεταβλητή GVL.INSERTED_BOTTLES_COUNTER. Μετά με τη χρήση ενός φλιπ φλοπ rs και ενός κυκλώματος χρονοκαθυστέρησης μοντελοποιούμε την ενεργοποιηση/απενεργοποίηση της μηχανής πλήρωσης για το διάστημα λειτουργίας της (5 δευτερόλεπτα).



capping

Η ίδια διαδικασία και για το τάπωμα με διαφορά οτι τώρα ελέγχουμε την Θέση του γεμισμένου μπουκαλιού και οτι η διάρκεια ενεργοποίησης της μηχανής ειναι 3 δευτερόλεπτα .

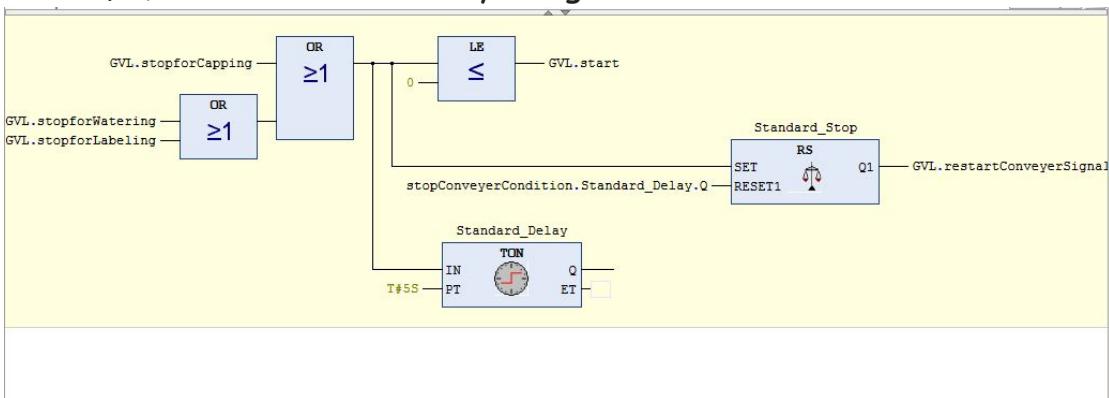


labeling

Η ίδια διαδικασία και για την ετικετοποίηση με τη διαφορά ότι ελέγχουμε την θέση του καπακωμένου μπουκαλιού και ότι η διάρκεια του σήματος ενεργοποίησης της αντίστοιχης μηχανής είναι 2 δευτερόλεπτα.

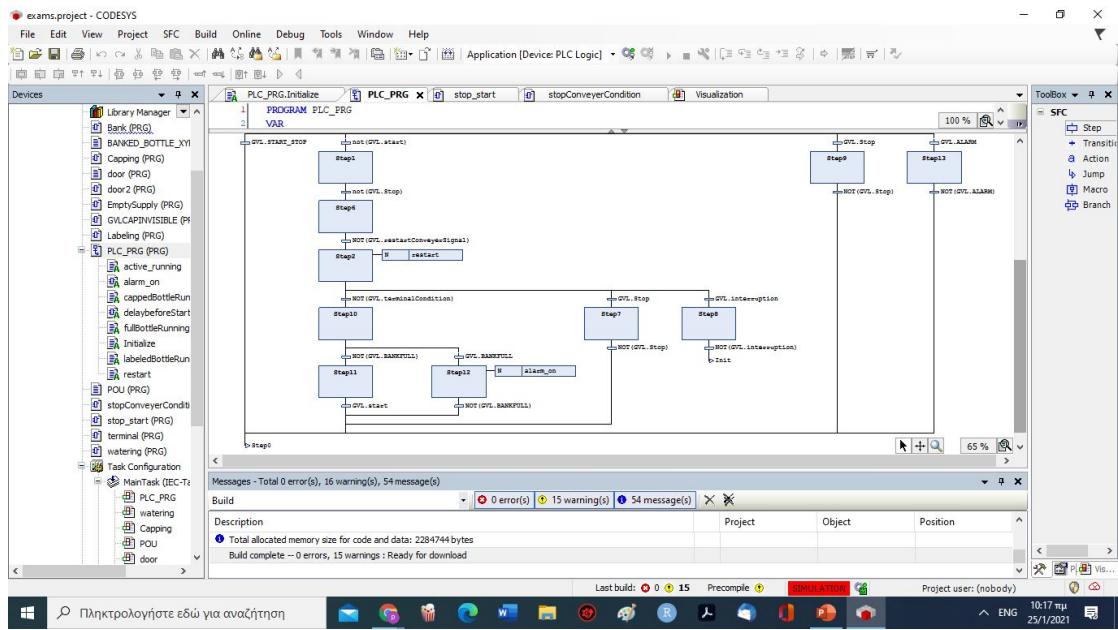
stopConveyerCondition

Στη συνέχεια υλοποιήσαμε την POU *stopConveyerCondition*, όπου σταματάμε τον ταινιόδρομο εάν έχουμε κάποιο μπουκάλι κάτω από κάποια μηχανή. Έτσι εφαρμόζεται χρονοκαθυστέρηση 5 δευτερολέπτων και στη συνέχεια το φλιπ φλοπ ξανα ενεργοποιεί τον ταινιόδρομο *GVL.restartConveyerSignal*.



Παύση ταινιόδρομου για 5 δευτερόλεπτα

Παρατηρούμε λοιπόν, ότι στη 2η κατα σειρά διακλάδωση όταν απενεργοποιηθεί ο ταινιόδρομος για να ανοίξουν οι μηχανές (*GVL.start* FALSE). Τότε ελέγχουμε αν έχει πατηθεί το κουμπί έκτακτης διακοπής *not* (*GVL.Stop*) και μετά περιμένουμε να ενεργοποιηθεί το σήμα που επανεκκινεί τον ταινιόδρομο (*GVL.restartConveyerSignal*). Το σήμα που αντιπροσωπεύει αυτή η μεταβλητή λειτουργεί σαν αντιστροφέας.



restart

Τώρα θα δούμε τι συμβαίνει στο step 2 που εκτελείται το restart action.

```

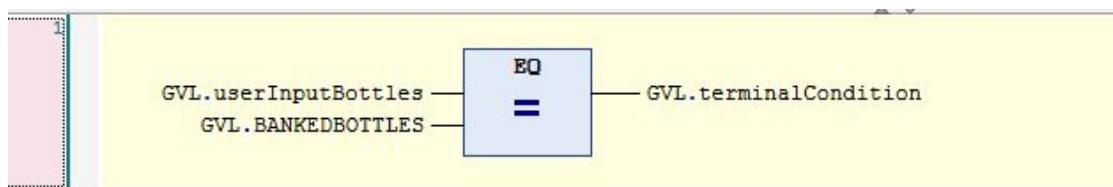
1  GVL.bottlePosition:=0;
2  GVL.start:=TRUE;
3  //GVL.FullInvisible:=FALSE;
4  GVL.FullBottlePosition:=0;
5  GVL.capMove:=0;
6  GVL.CapBottlePosition:=0;
7  //IF (watering.InsertedBottles.CV >1) THEN
8      //GVL.CapInvisible:=FALSE;
9  //END_IF
10
11 GVL.labelXMove:=0;
12 //IF (watering.InsertedBottles.CV >2) THEN
13 //GVL.LabelInvisible:=FALSE;
14 //END_IF
15 GVL.XlabelBottlePosition:=0;
16 GVL.YlabelBottlePosition:=0;
17

```

Εδώ ενεργοποιείται ο ταινιόδρομος `GVL.start:=TRUE` και επιστρέφουν τα μπουκάλια (αδειογεμάτο, καπάκι, ετικέτα) στις αρχικές τους θέσεις. Επίσης τα γραφικά στοιχεία που δίνουν την κίνηση των μηχανών τάπωσης και ετικετοποίησης και δίνονται από τις μεταβλητές `GVL.capMOVE` και `GVL.labelXMove` επιστρέφουν στις αρχικές θέσεις. Αυτό το βήμα δηλαδή έχει να κάνει πάλι με το κομμάτι του visualization.

terminal

Φεύγοντας και από step2 μπορούμε να έχουμε 3 εναλλακτικές διαδρομές. Στη πρώτη περίπτωση ελέγχουμε αν υπάρχει σήμα τερματικού διακόπτη, ότι δηλαδή τα μπουκάλια που μπήκαν στην τράπεζα απόθεσης ισούνται με την είσοδο του χειριστή.



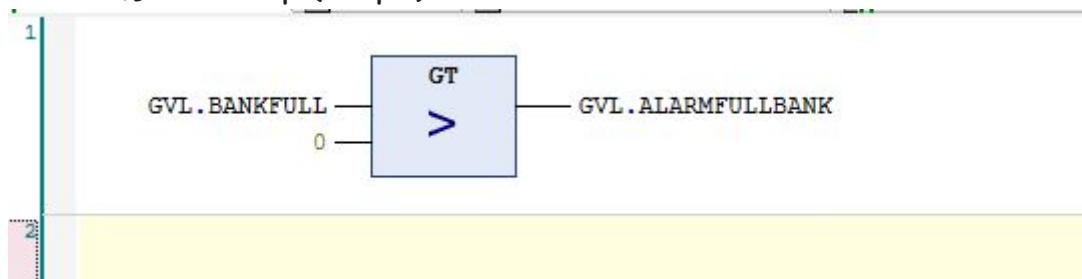
Σήμα τερματικού διακόπτη

alarm_on

Αν δεν ισχύει προχωράμε και ελέγχουμε αν έχουν μπει στη τράπεζα απόθεσης τα ζητούμενα μπουκάλια *not* (*GVL.Bankfull*). Αν όχι, τότε προχωράμε στον επόμενο κύκλο δεδομένου ότι έχουμε σήμα *start* ενεργοποιημένο. Άν δεν έχουμε χώρο τότε ενεργοποιείται η σειρήνα και σταματάει ο ταινιοδρομος αυτό γίνεται στο step12.

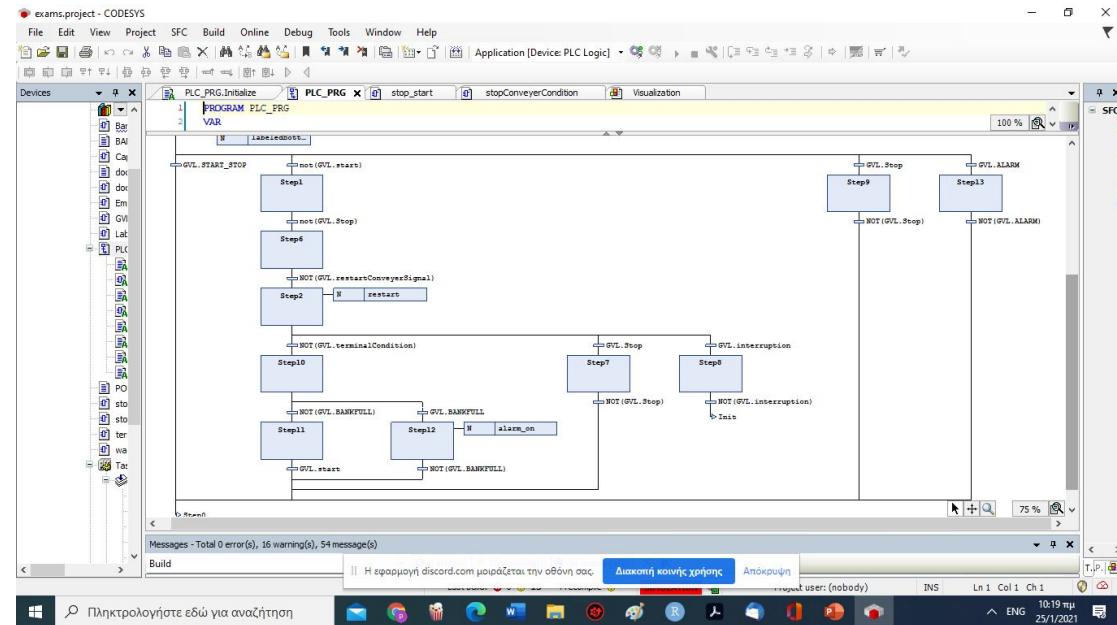
Επιστρέφοντας στις τρείς διακλαδώσεις θα δούμε με ποιόν τρόπο πάμε στην 2η κατα σειρά διακλάδωση. Εκεί πάμε αν έχουμε σήμα έκτακτης διακοπής *GVL.Stop* TRUE.

Τέλος στην 3η διακλάδωση οδηγούμαστε αν έχουμε γεμάτη τράπεζα απόθεσης χωρίς έχουν ετοιμαστεί τα μπουκάλια που θέλει ο χειριστής και ενεργοποιείται η σειρήνα *GVL.ALARMFULLBANK* και η έκτακτη διακοπής *GVL.stop* (step8).



Ενεργοποίηση σηρήνας όταν γεμίσει η τράπεζα

Οι ίδιοι έλεγχοι γίνονται και στο παρακάτω σχήμα δεξιά, ώστε να εξασφαλίσουμε ότι ανα πάσα στιγμή θα εκτελεστούν οι κατάλληλες ενέργειες.



POU

Στη συνάρτηση POU δίνεται η κίνηση των μηχανών όταν ενεργοποιούνται. Και εδώ αφορά το γραφικό κομμάτι.

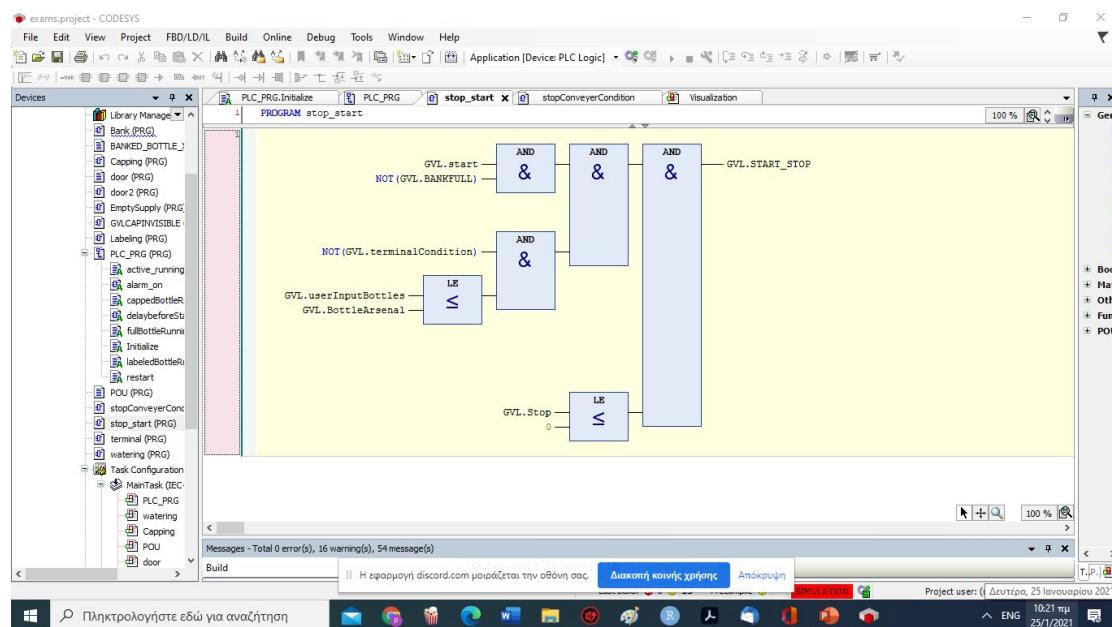
```

1 IF ( GVL.capMachineSwitch =TRUE) THEN
2     GVL.capMove:=40;
3     //GVL.labelMove:=40;
4 END_IF
5 IF(GVL.labelingMachineSwitch=TRUE)THEN
6     GVL.labelMove:=40;
7 END_IF
8 IF (Capping.cap.RESET1) THEN
9     GVL.capMove:=0;
10 END_IF
11 IF (Labeling.label.RESET1) THEN
12     GVL.labelMove:=0;
13 END_IF

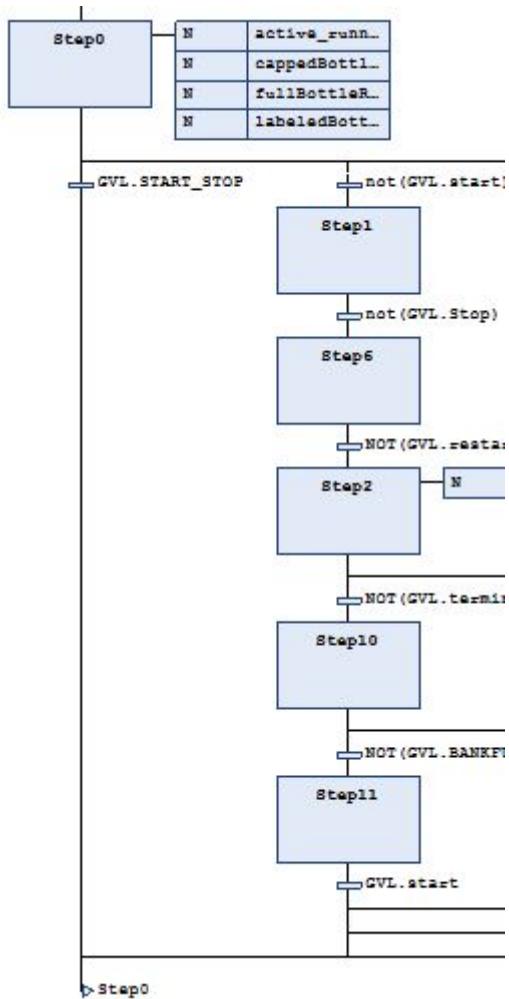
```

START_STOP

Στην POU START_STOP ελέγχουμε αν θα πρέπει να γίνει επανάληψη του step0(δηλαδή εκκίνηση ταινιόδρομου) .Δηλαδή αν το σταρτ είναι ον, αν δεν έχουμε διακοπή από τερματικό διακόπτη(ολοκλήρωση διαδικασίας για τον επιθυμητό αριθμό μπουκαλιών) και αν δεν έχουμε πατήσει το κουμπί έκτακτης διακοπής και αν η αποθήκη τροφοδοσίας έχει επαρκή μπουκάλια και αν η τράπεζα απόθεσης έχει χώρο.

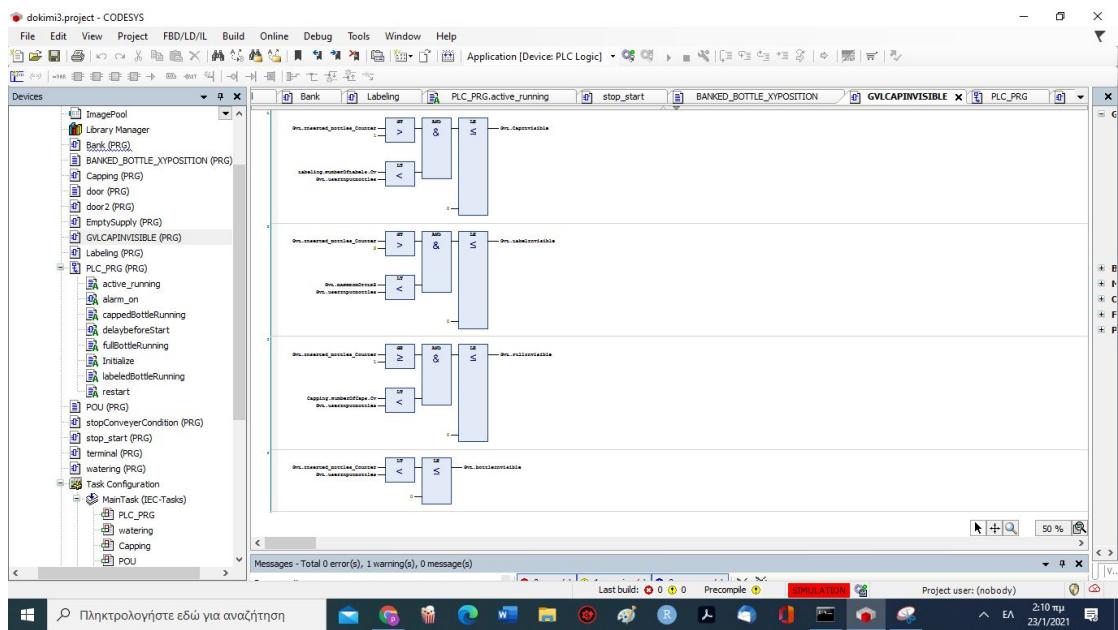


ελεγχος μεταβλητής start_stop



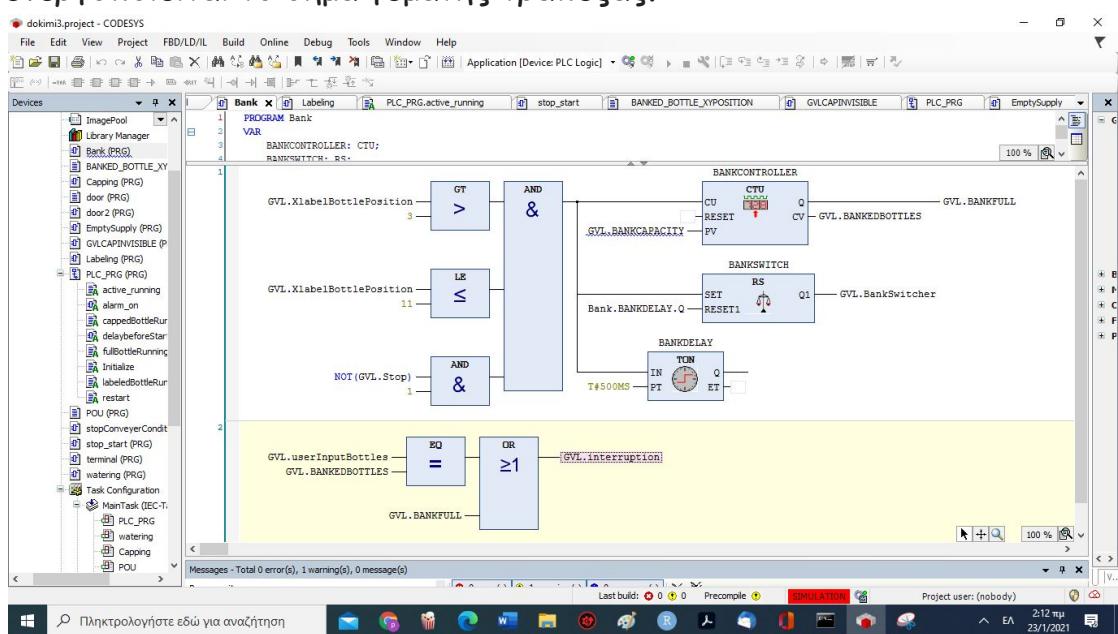
GVL CAP INVISIBLE

Στην POU `GVL CAP INVISIBLE` εξαφανίζουμε τα μπουκάλια από το γραφικό περιβάλλον, όταν δεν τα χρειαζόμαστε πλέον.



BANK

Στην POU BANK ενεργοποιείται η μηχανή της απόθεσης κάθε φορά που φτάνει ένα ετικετοποιημένο μπουκάλι. Επίσης στον απαριθμητή BANKCONTROLLER έχουμε σαν όριο για το Q το BANK CAPACITY δηλαδή όταν γεμίσει τράπεζα το gvl.bankfull γίνεται αληθές και ενεργοποιείται το σήμα γεμάτης τράπεζας.



BANKED_BOTTLE_XYPOSITION

Στην POU BANKED_BOTTLE_XYPOSITION, γραφικά αναπαριστούμε την είσοδο του ετικετοποιημένου μπουκαλιού στην τράπεζα απόθεσης.
Γραφική λειτουργία και εδώ.

The screenshot shows the CODESYS Development Environment interface. The main window displays the PLC Logic editor for the 'BANKED_BOTTLE_XYPOSITION' program. The code is written in a structured text language:

```
PROGRAM BANKED_BOTTLE_XYPOSITION
VAR
END_VAR

IF (Bank.BANKSWITCH.SET) THEN
    GVL_YlabelBottlePosition:=#90;
END_IF
```

The left sidebar shows the project structure under 'dokimi3.project - CODESYS'. The 'PLC Logic' section contains several programs and subroutines, including 'Bank (PRG)', 'Capping (PRG)', 'door (PRG)', 'door2 (PRG)', 'EmptySupply (PRG)', 'GV_CAPINVISIBLE (P)', 'Labeling (PRG)', and 'PLC_PRG (PRG)'. The 'PLC_PRG (PRG)' section lists various functions like 'active_running', 'alarm_on', 'cappedBottleRur', 'delaybeforestart', 'fullBottleRunning', 'Initials', 'labeledBottleRur', and 'restart'. The bottom status bar indicates the date and time: '23/1/2021 2:15 πμ'.

GVL LIST

η Λίστα των μεταβλητών που χρησιμοποιήθηκαν

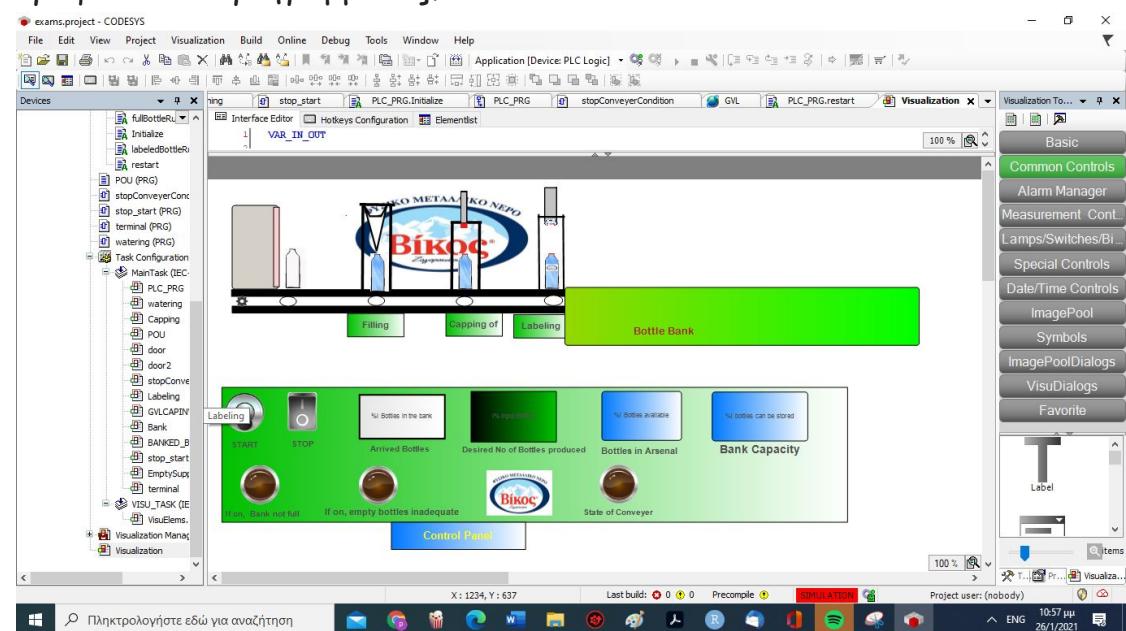
The screenshot shows the CODESYS Development System interface with the following details:

- Project:** dokimi3.project - CODESYS
- File Menu:** File Edit View Project Declarations Build Online Debug Tools Window Help
- Devices:** dokim3, Device (CODESYS Control Win V3), PLC Logic, Application (GVL, ImagePool, Library Manager, Bank (PRG), BOTTLED_BOTTLE_XY, Capping (PRG), door (PRG), door2 (PRG), EmptySupply (PRG), GVL_CAPIVISIBLE (P), Labeling (PRG), PLC_PRG (PRG), active_running, alarm_on, cappedbottleRur, delaybeforeStar, fullbottleRunning, initail, labeledBottleRur, restart, POU (PRG), stopConvergeCondit, stop_start (PRG), terminal (PRG), watering (PRG), Task Configuration).
- Current View:** GVL X
- Code Content:** The code lists various global variables and their initial values or descriptions. Some variables are annotated with Greek comments.
- Messages:** Total 0 error(s), 1 warning(s), 0 message(s)
- System Bar:** Last build: 0 0 0, Precompile: 0, Project user: (nobody), INS, Ln 8 Col 44 Ch 41, 2.17 ms, 23/7/2021.

```
colorWater: IWORD;
start:BOOL; //simi energeopoiisis tainioedcomou
bottleVisible:BOOL;
bottlePosition:REAL;
rotWheel:REAL; //peristrefesi gis gronasis
waterMachineSwitch:BOOL; // diskopis minanis plirosis
fullInvisible:BOOL;
fullInvisiblePosition:REAL;
emptyMachineSwitch:BOOL; // minani kapakosis
capture:INT;
rotDoor:INT; // peristrefsi portas
doorSwitch:BOOL; //diskopis trofodosis me edeies fialeis
openDoor:BOOL; //
closeDoor:BOOL;
stopForCapping:BOOL;
stopForWatering:BOOL;
CapInvincible: BOOL;
CapInvinciblePosition:REAL;
labelInvisible:INT;
labelInvisibleSwitch:BOOL; // diskopis minanis etiketas
stopForLabeling:BOOL;
LabelInvincible: BOOL;
Inserted_Bottles_Counter:WORD;
Label_BottlePosition:REAL;
Vial_BottlePosition:REAL;
Stop:BOOL; //stop me alliws diskopis
BOTTLEULL:BOOL; // i trapeza genise
BOTTLECAPACITY:INT; //xerontikotita trapezas
BOTTLED_BOTTLES:WORD; // posa mpoulalia etemastasan
BottleSwitch:BOOL; // diskopis trofodosis me apothanai
restartCondition:BOOL; // simi epenergeopoiisis tainioedcomou
userInputSwitch:INT; // arithmou sponkalion pou tha genisei o meizistis
BottleArenal:INT; // arithmou mpoulalia trofodosis
ALARM:BOOL; // simi aneparkseis trofodosis me mpsikalisa
START_STOP:BOOL; // start and not/stop
ALARM_STOP:BOOL; // arithmou sponkalion genisei trapeza
del:BOOL; // setmana ekkinisis
terminalCondition:BOOL; // tematiki zinhili diskopis
interruption:BOOL; //diskopi logo genetis trapezas
```

VISUALIZATION

Γραφικά του προγράμματος.



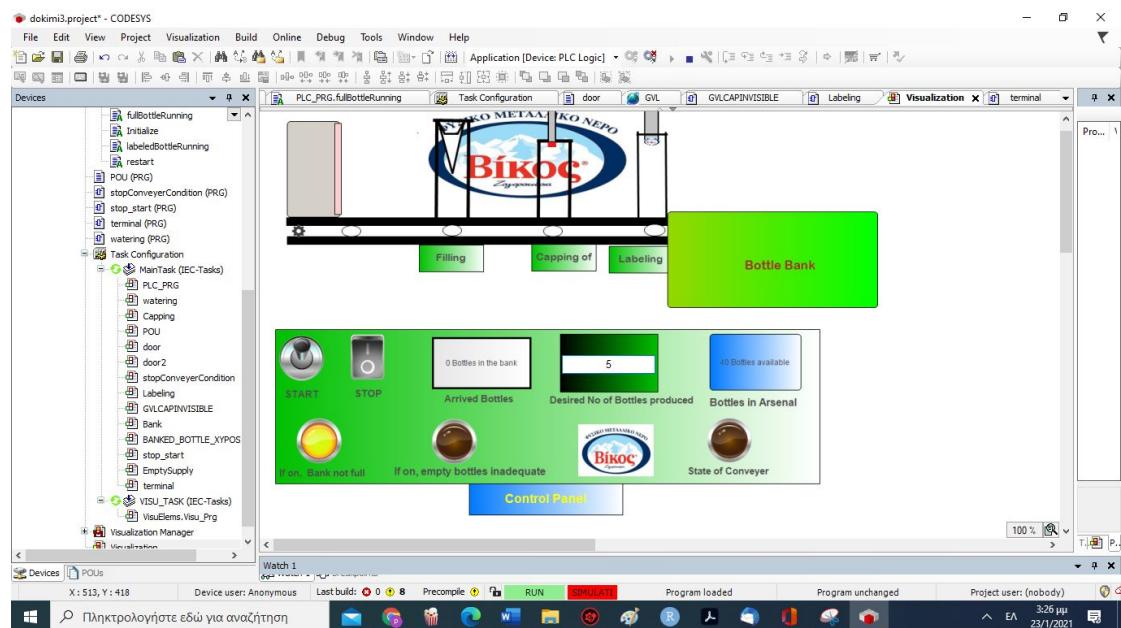
Παρατηρούμε τον πίνακα ελέγχου και έχουμε :

- Τον διακόπτη έναρξης START
- Τον διακόπτη έκτακτης διακοπής STOP
- Την λυχνία για την διαθεσιμότητα της τροφοδοσίας
- Την λυχνία για την διαθεσιμότητα της τράπεζας απόθεσης
- Την λυχνία για την κατάσταση του ταινιόδρομου
- Τον μετρητή για τις φιάλες που ολοκληρώθηκαν

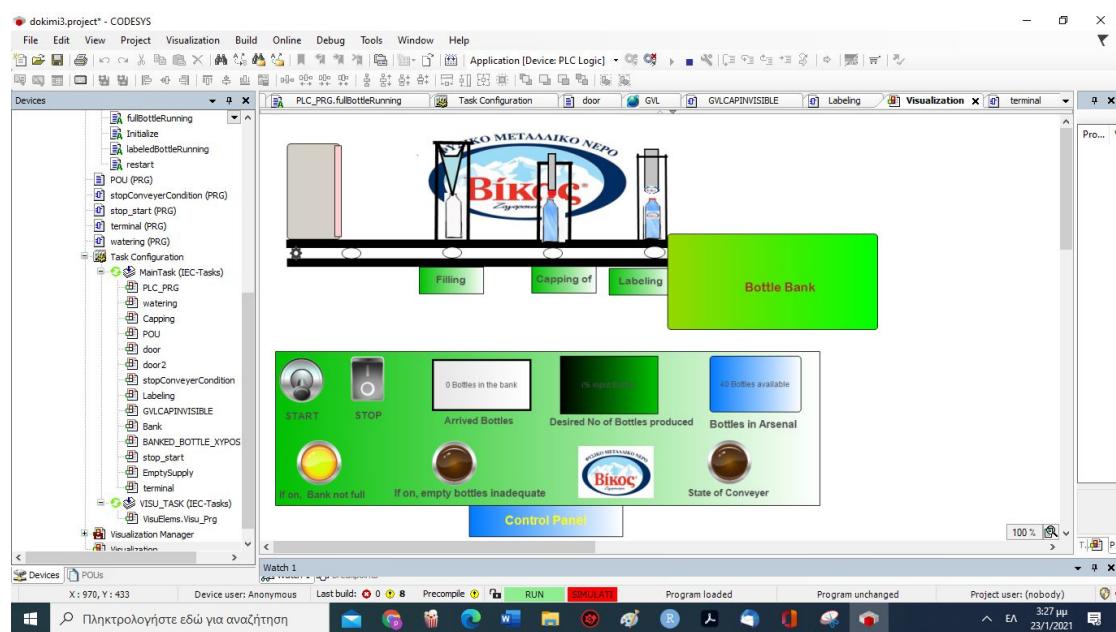
- Το πλαίσιο προσθήκης επιθυμητού αριθμού μπουκαλιών προς εμφιάλωση από τον χειριστή
- Τον μετρητή των φιαλών που βρίσκονται στην αποθήκη τροφοδοσίας κατά την έναρξη της διαδικασίας.
- Τον μετρητή της αρχικής χωρητικότητας της τράπεζας απόθεσης

Προσομοίωση

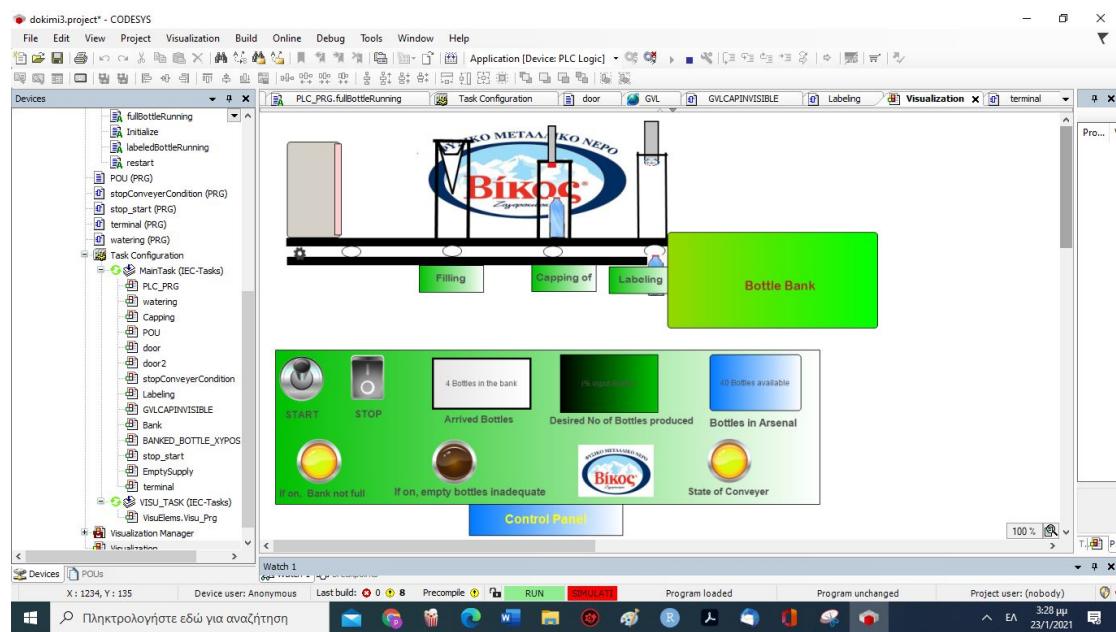
Εισαγωγή αριθμού μπουκαλιών



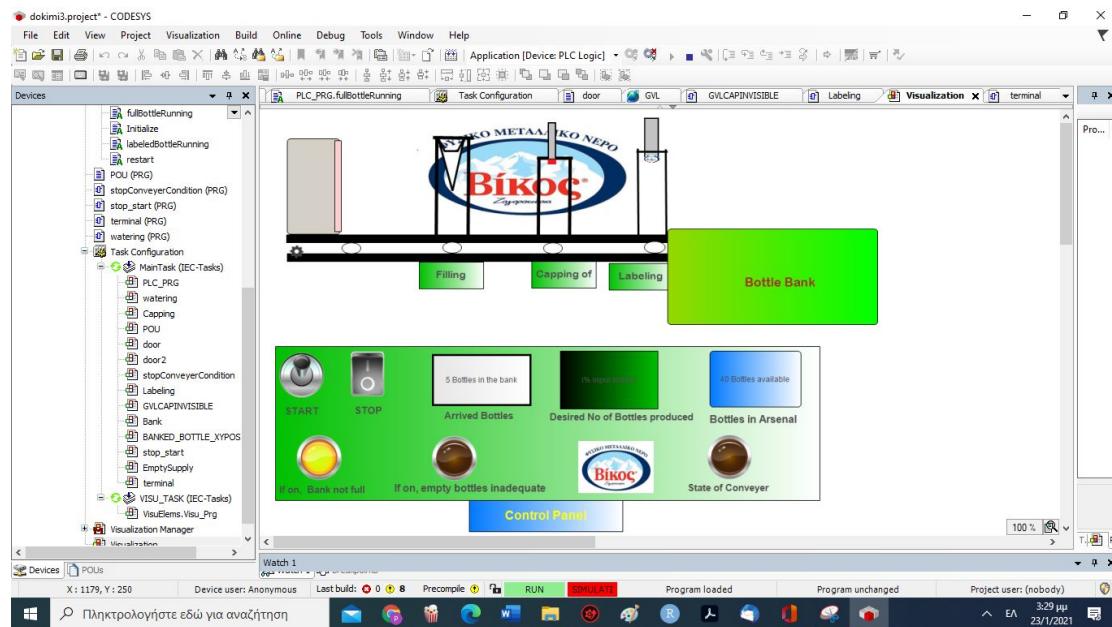
Σε φάση λειτουργίας μηχανών



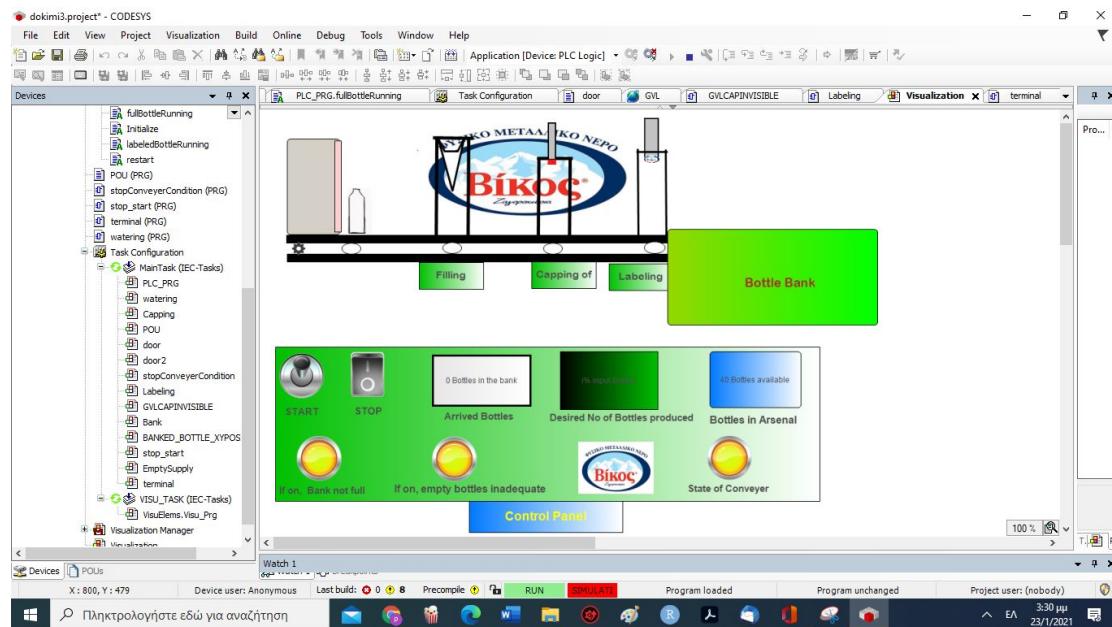
Λίγο πριν λήξει η διαδικασία



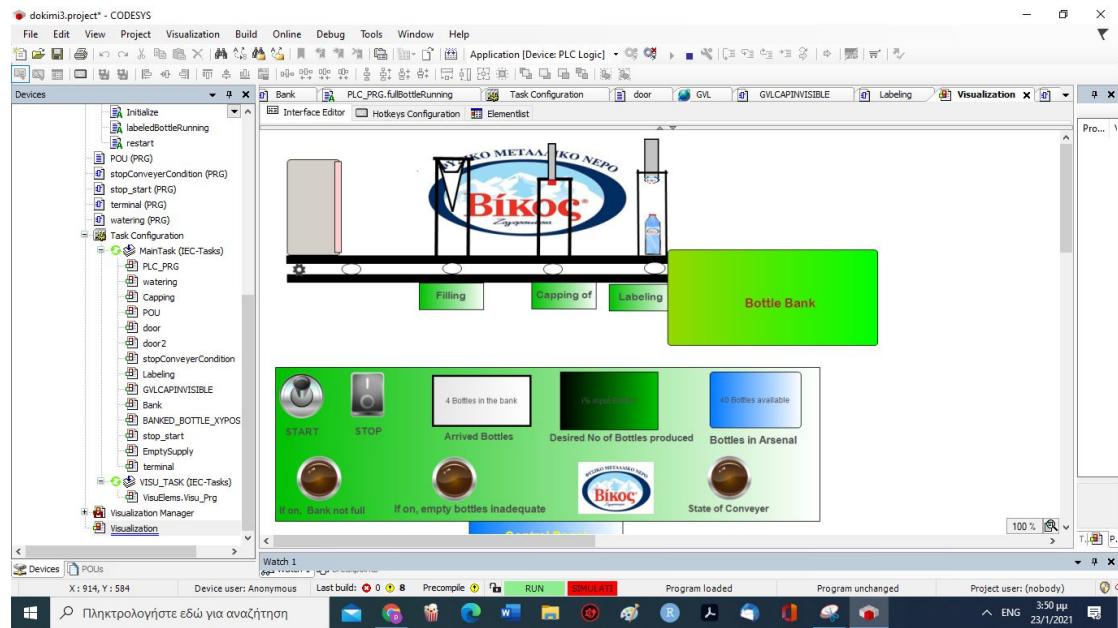
Λήξη διαδικασίας



Ανεπαρκής τροφοδοσία



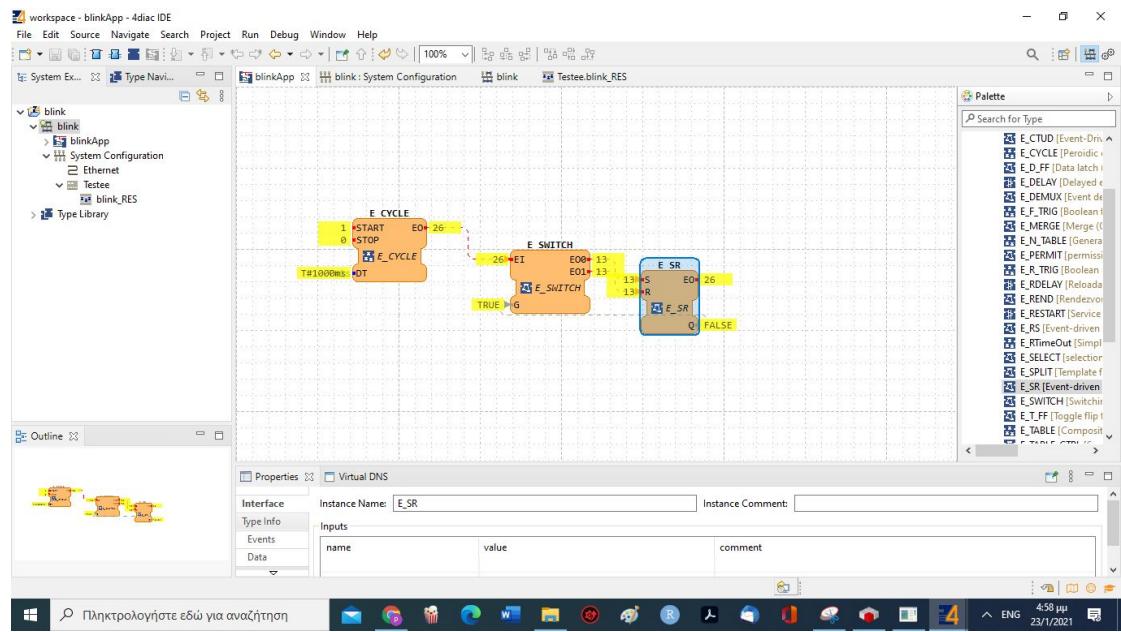
Σενάριο γεμάτης τράπεζας απόθεσης, χωρίς να έχει ολοκληρωθεί η διαδικασία.



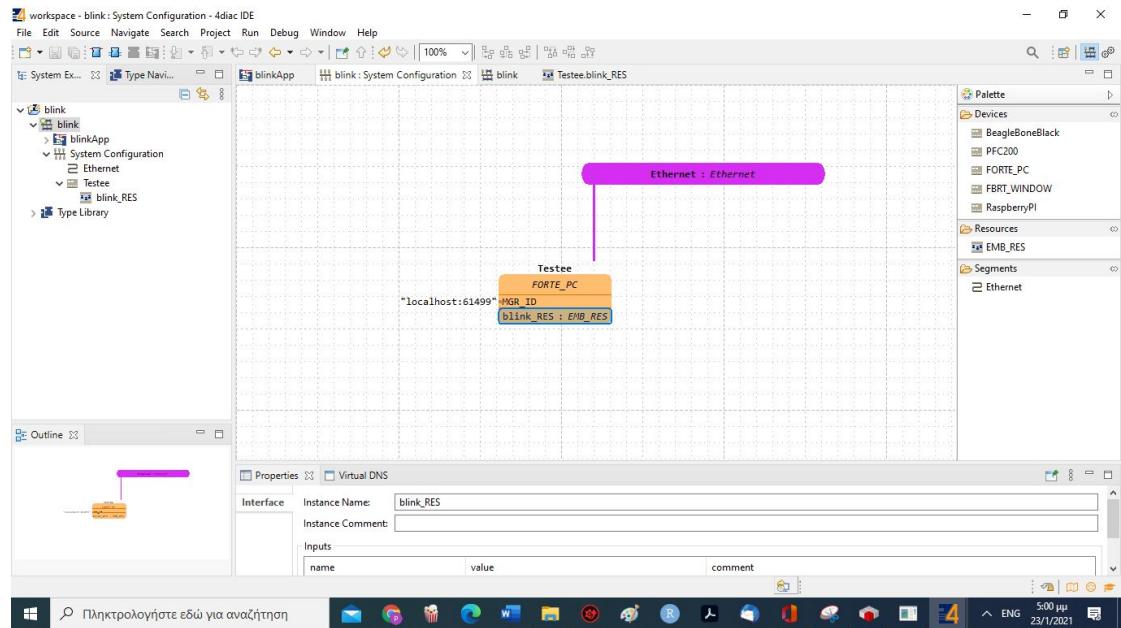
Γ. Υλοποίηση κατανεμημένου δικτύου υπολογιστών.

Πρώτος υπολογιστής

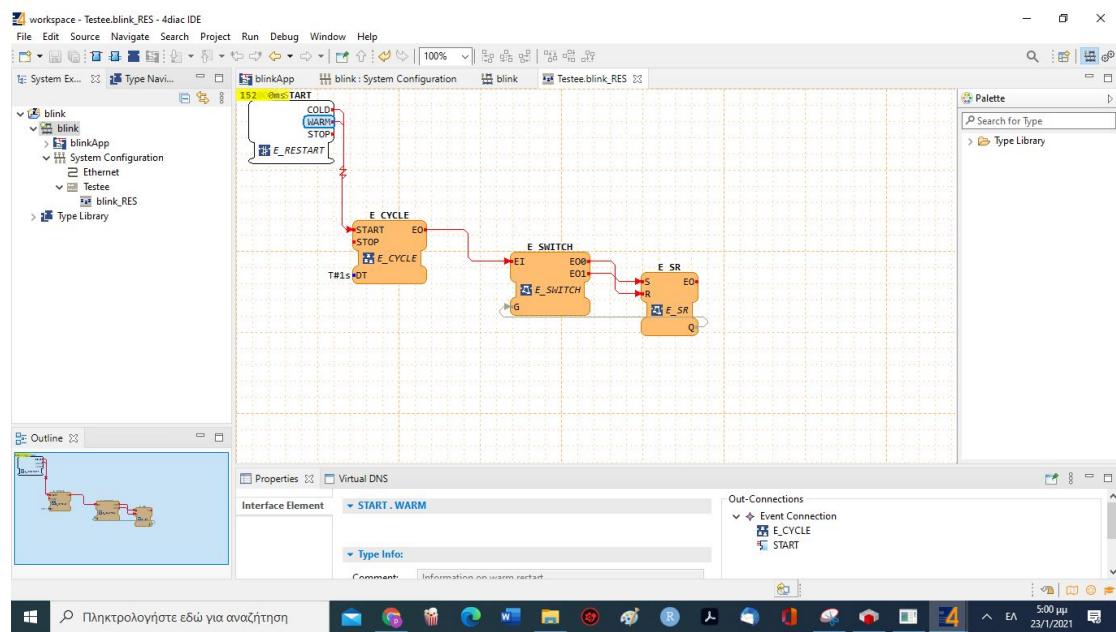
Στο πρώτο σκέλος δημιουργούμε ένα συναρτησιακό Block από 1 γεννήτρια κυκλικών γεγονότων ανα 1 sec, ενός διακόπτη switch ο οποίος στέλνει τις 2 εξόδους του στο Set Reset Flip Flop. Ανάλογα με τις τιμές του G ενεργοποιούνται οι έξοδοι EO0 ή EO1. Οι οποίες με τη σειρά τους ενεργοποιούν το s και το r.



Το δίκτυο ethernet με τον υπολογιστή Testee. Όπου τρέχει το παραπάνω συναρτησιακό μπλοκ. Η πύλη πάνω στην βλέπει αυτός ο υπολογιστής είναι η 61499 του τοπικού δικτύου.

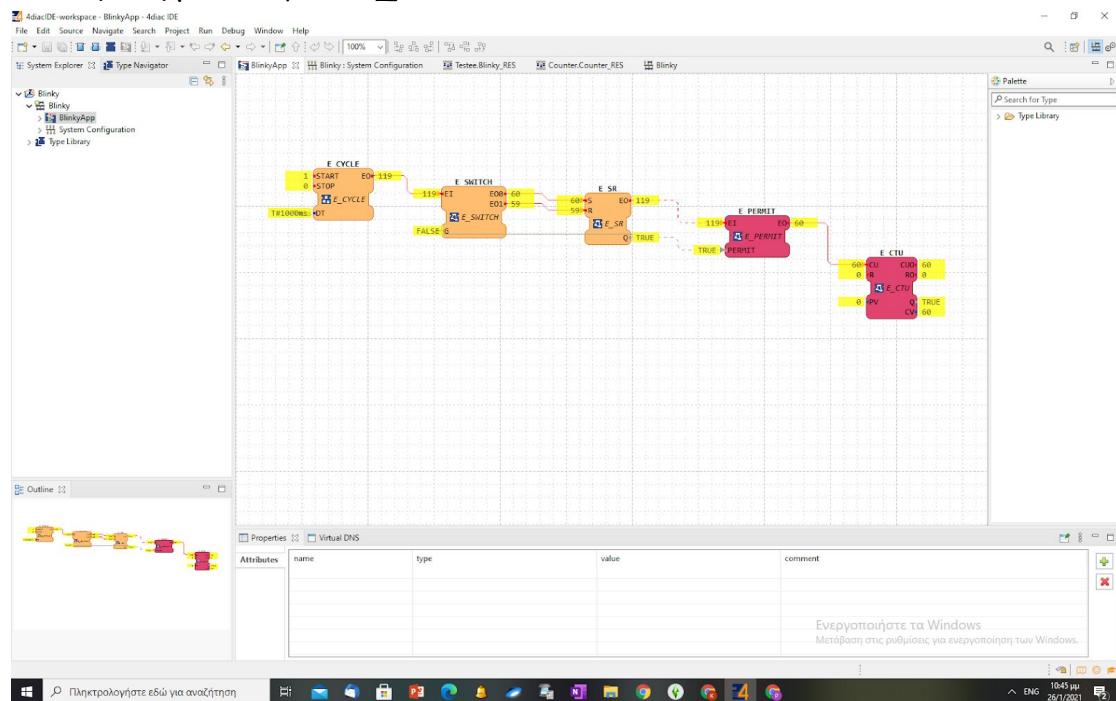


Παρακάτω είναι το EMB_Res του Blinky . Βλέπουμε ότι συνδέεται με ένα κόμβο στο start , cold και warm. Το cold είναι για την πρώτη εκκίνηση ενώ το warm για τις επόμενες.



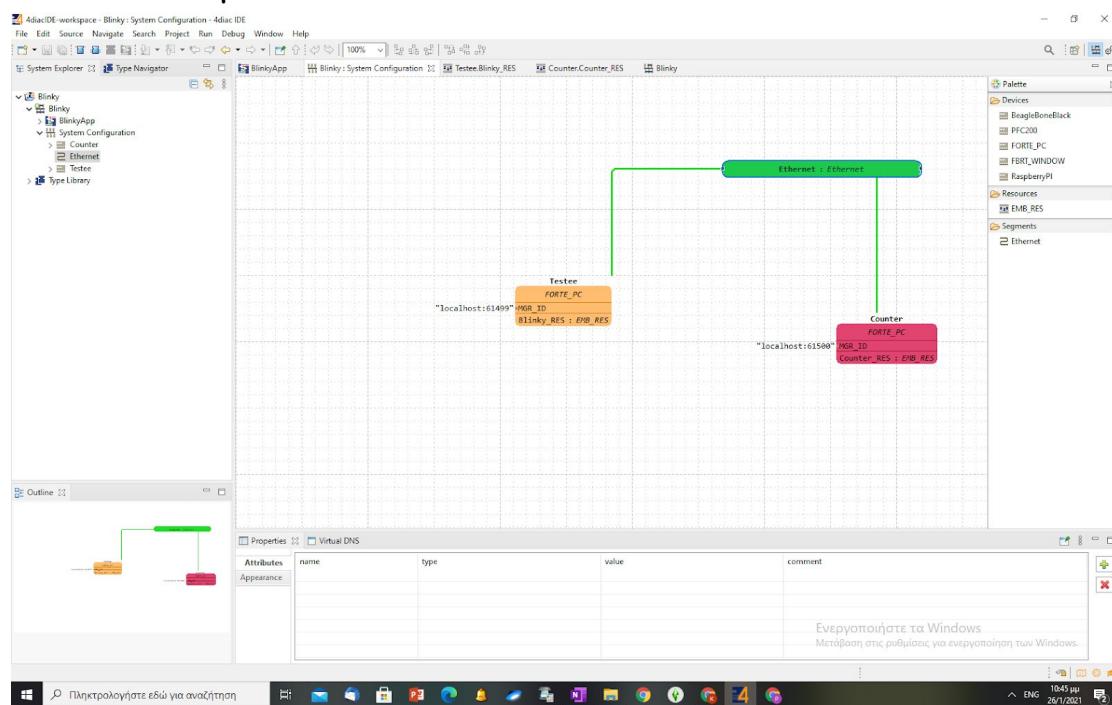
Σύνδεση 2ου μπλοκ

Εδώ προσθέτουμε δυο κόμβους τον permit και τον e_ctu. Ο πρώτος λαμβάνει μέσω του αρχιτεκτονικού προτύπου Publish and Subscribe τα events του 1ου μπλοκ και τα στέλνει στον μετρητή που βρίσκεται στο ίδιο υπολογιστή με αυτό, τον e_ctu.

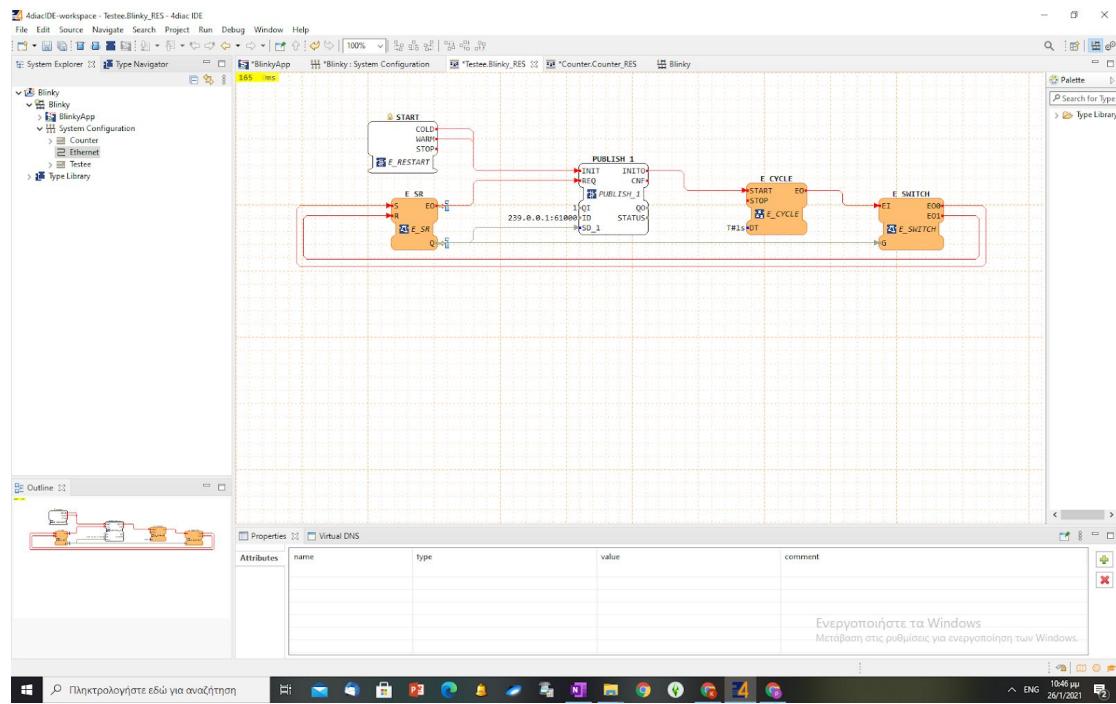


Σύνδεση 2ου υπολογιστή

Εδώ βλέπουμε την σύνδεση του 2ου υπολογιστή στο τοπικό δίκτυο, ο οποίος περιλαμβάνει το permit και ctu. Αυτός βλέπει στην πόρτα 61500 του τοπικού δικτύου, διαφορετική από την 61499, πράγμα απαραίτητο για την σωστή λειτουργία του δικτύου. Η πληροφορία ανταλλάσσεται μέσω του καναλιού ethernet.



Παρακάτω βλέπουμε την τροποποιημένη σύνδεση του Blinky_Res. Εδώ συνδέεσμε έναν κόμβο publish_1 διότι θέλουμε να στείλουμε 1 event κάθε φορά. Στο ID είναι η διεύθυνση στην οποία γίνεται casting η πληροφορία από τον εκδότη, και είναι με πρωτόκολλο UDP. SD_1 είναι τα δεδομένα.



Ομοίως και στη παρακάτω εικόνα βλέπουμε τη μορφή ενός συνδρομητή που λαμβάνει ότι πληροφορία υπάρχει στην διεύθυνση του casting. Το RD_1 είναι τα δεδομένα που εισάγονται στο δεύτερο υπολογιστή από τον 1ο μέσω του δικτύου.

