Ατομική Διπλωματική Εργασία

ΠΡΟΣΟΜΟΙΩΣΗ ΔΙΑΔΙΚΑΣΙΩΝ ΣΕ ΒΙΟΜΗΧΑΝΙΚΑ ΔΙΚΤΥΑ ΕΛΕΓΧΟΥ

Γεώργιος Κουσαρίδης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Προσομοίωση Διαδικασιών σε Βιομηχανικά Δίκτυα Ελέγχου

Γεώργιος Κουσαρίδης

Επιβλέπων καθηγητής
Βάσος Βασιλείου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2022

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, Δρ. Βάσο Βασιλείου για την καθοδήγηση του και για την πολύ καλή συνεργασία που είχαμε κατά την διάρκεια της εκπόνησης της διπλωματικής μου εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω την ερευνήτρια Κρίστια Χαριλάου και την μεταδιδάκτορα του τμήματος Χριστιάνα Ιωάννου για την πολύτιμη βοήθεια που μου πρόσφεραν για την ολοκλήρωση της εργασίας αυτής.

Τέλος θα ήθελα να ευχαριστήσω την οικογένεια μου και τους φίλους μου για την στήριξη και συμπαράσταση τους καθ' όλη την διάρκεια των σπουδών μου

Περίληψη

Το σύστημα βιομηχανικού ελέγχου (ICS) είναι ένας συλλογικός όρος που χρησιμοποιείται για να περιγράψει διαφορετικούς τύπους συστημάτων ελέγχου και συναφών οργάνων, τα οποία περιλαμβάνουν τις συσκευές, τα συστήματα, τα δίκτυα και τα στοιχεία ελέγχου που χρησιμοποιούνται για τη λειτουργία ή/και την αυτοματοποίηση βιομηχανικών διεργασιών. Σήμερα, οι συσκευές και τα πρωτόκολλα που χρησιμοποιούνται σε ένα ICS χρησιμοποιούνται σχεδόν σε κάθε βιομηχανικό τομέα και κρίσιμης σημασίας υποδομές όπως η βιομηχανία, οι μεταφορές, η ενέργεια και οι βιομηχανίες επεξεργασίας νερού.

Τα συστήματα βιομηχανικού ελέγχου αναφέρονται γενικά στη σύνδεση διαφόρων ηλεκτρονικών εξοπλισμών που χρησιμοποιούνται σε εργοστάσια, μονάδες ελέγχου διεργασιών και αυτοματοποιημένες εγκαταστάσεις για την παρακολούθηση και τον έλεγχο της κατασκευής και άλλων εφαρμογών. Ο ηλεκτρονικός εξοπλισμός περιλαμβάνει κυρίως μηχανές όπως ρομπότ, υπολογιστές, εργαλειομηχανές, προγραμματιζόμενους λογικούς ελεγκτές (PLC), αισθητήρες, ενεργοποιητές (π.χ. ρελέ, βαλβίδες, κινητήρες) και όργανα μέτρησης. Ωστόσο, ένα κυρίαρχο στοιχείο σε όλα τα συστήματα βιομηχανικού ελέγχου είναι τα δίκτυα επικοινωνίας που είναι υπεύθυνα για τη σύνδεση όλου του εξοπλισμού και των συσκευών με ηλεκτρικές διεπαφές και πρωτόκολλα επικοινωνίας για τη λειτουργία των συστημάτων. Κατά τη διάρκεια των δεκαετιών, η χρήση του εξοπλισμού επικοινωνίας έχει αυξηθεί σημαντικά στα βιομηχανικά συστήματα ελέγχου χάρη στην ανάπτυξη προηγμένων τεχνολογιών πληροφοριών και επικοινωνιών προκειμένου να βελτιωθούν οι επιδόσεις του δικτύου επικοινωνίας καθώς και να βελτιστοποιηθούν οι λειτουργίες παρακολούθησης και ελέγχου.

Στην παρούσα διπλωματική εργασία θα γίνει μια εισαγωγή στο Industrial Internet όπως επίσης και στα ασύρματα δίκτυα αισθητήρων. Ακολούθως, θα γίνει αναφορά στη σημασία των SCADA συστημάτων, στα προβλήματα που μπορούν να συμβούν αλλά και στους τρόπους αντιμετώπισης αυτών των προβλήματων στο κομμάτι της ασφάλειας. Τέλος, θα παρουσιαστεί η υλοποιήση του συστήματος επεξεργασίας νερού χρησιμοποιώντας τον προσομοιωτή SCADASim, τα πειραματικά αποτελέσματα της προσομοίωσης αυτής και τα τελικά συμπεράσματα που θα εξαχθούν από την διπλωματική αυτή.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Γενική Εισαγωγή του Θέματος

Το Βιομηχανικό Διαδίκτυο των Πραγμάτων (IIoT), γνωστό και ως Industrial Internet, είναι το δίκτυο μιας πληθώρας βιομηχανικών συσκευών που συνδέονται με τεχνολογίες επικοινωνιών που οδηγεί τα συστήματα για τη σύλληψη και τη μετακίνηση δεδομένων, την αίσθηση των αλλαγών στη θερμοκρασία, τη ροή ή τον όγκο, την αυτοματοποίηση διαδικασιών για αποτελεσματικότητα, ακρίβεια και ασφάλεια, να παρακολουθούν, να συλλέγουν, να αναλύουν νέες γνώσεις. Αυτές οι πληροφορίες μπορούν στη συνέχεια να βοηθήσουν στην εξυπνότερη και ταχύτερη λήψη επιχειρηματικών αποφάσεων για τις βιομηχανικές εταιρείες. [1]

1.2 Προβλήματα στο Βιομηχανικό Διαδίκτυο των Πραγμάτων (Industrial IoT)

Το Βιομηχανικό Διαδίκτυο των Πραγμάτων (IIoT) συνοδεύεται από μερικές μοναδικές παγίδες λόγω της κυβερνο-φυσικής σύνδεσής του. Με άλλα λόγια, όταν κάτι πάει στραβά στον ψηφιακό κόσμο, οι συνέπειες μπορεί να εμφανιστούν στον πραγματικό κόσμο.

Για παράδειγμα, ακατάλληλες ενσωματώσεις σκεφτείτε ελαττωματικούς αλγόριθμους ή λανθασμένους υπολογισμούς, θα μπορούσαν να προκαλέσουν ζημιά στην εγκατάσταση, το τελικό προϊόν ή τις πρώτες ύλες σας. Οι συσκευές μπορεί να υπερθερμανθούν, να εκραγούν ή να δυσλειτουργήσουν με τρόπο που να βλάψει τα προϊόντα, τον εξοπλισμό ή ακόμα και να τραυματίσει τους εργαζομένους σας.

Επίσης, η προσθήκη περισσότερων συσκευών συνδεδεμένων στο δίκτυο στο περιβάλλον μας αυξάνει την επιφάνεια επίθεσης. Αυτό δημιουργεί περισσότερες ευκαιρίες για τους εγκληματίες του κυβερνοχώρου να χακάρουν το σύστημα – είτε ως πράξη δολιοφθοράς, είτε ως προσπάθεια συλλογής λύτρων είτε ως πολιτική επίθεση. Μερικές φορές, οι χάκερ αποκτούν ακόμη και πρόσβαση σε μηχανήματα τυχαία. [2]

Πρόσφατες επιθέσεις σε συστήματα SCADA υπογραμμίζουν την ανάγκη για ισχυρότερη ασφάλεια SCADA. Είναι σημαντικό να αναλύσουμε τους κινδύνους ασφαλείας και να αναπτύξουμε κατάλληλες λύσεις ασφάλειας για την προστασία τέτοιων συστημάτων. Ωστόσο, ένα βασικό πρόβλημα είναι η έλλειψη κατάλληλων εργαλείων μοντελοποίησης για την αξιολόγηση της ασφάλειας των συστημάτων SCADA. Όπως είναι ευρέως αποδεκτό στις ακαδημαϊκές και βιομηχανικές κοινότητες, δεν είναι πρακτικό να διεξάγονται πειράματα ασφαλείας σε ζωντανά συστήματα. Ένα εργαλείο προσομοίωσης μοντελοποίησης θα επέτρεπε την προσομοίωση συστημάτων SCADA με το πλεονέκτημα της δοκιμής διαφορετικών λύσεων επίθεσης και ασφάλειας. Για την καλύτερη κατανόηση της προστασίας των συστημάτων SCADA, είναι σημαντικό να αναλυθούν οι κίνδυνοι ασφάλειας τέτοιων συστημάτων και να αναπτυχθούν κατάλληλες λύσεις για την προστασία τους από κακόβουλες επιθέσεις. Ένα βασικό πρόβλημα στην έρευνα και ανάπτυξη λύσεων ασφαλείας για συστήματα SCADA είναι η έλλειψη κατάλληλων εργαλείων μοντελοποίησης για την αξιολόγηση της ασφάλειας τέτοιων συστημάτων. Ένα πλαίσιο προσομοίωσης SCADA θα επέτρεπε τη δημιουργία απλών μοντέλων συστημάτων SCADA με το πρόσθετο πλεονέκτημα της δοκιμής πραγματικών επιθέσεων και της δοκιμής διαφορετικών λύσεων ασφαλείας.

1.3 Ασφάλεια στο Βιομηχανικό Διαδίκτυο των Πραγμάτων

Οι ανησυχίες για την ασφάλεια στο Βιομηχανικό Διαδίκτυο των Πραγμάτων πηγάζουν από την αυξημένη επιφάνεια επίθεσης και την ανάγκη για απομακρυσμένη πρόσβαση. Καθώς, περισσότερες συσκευές και αισθητήρες έρχονται στο διαδίκτυο, δημιουργούν περισσότερα κανάλια επικοινωνίας, αποθήκες δεδομένων, θύρες και τελικά σημεία. Αυτή η αυξημένη διασυνδεσιμότητα αντιπροσωπεύει περισσότερες ευπάθειες εάν αφεθεί απροστάτευτη.

Μια επισκόπηση των βιομηχανικών IoT όσον αφορά στην ασφάλεια αναλύεται σε τρεις κατηγορίες: τοπικά δίκτυα, επεξεργασία δεδομένων και διαχείριση τελικού σημείου (end points).

Στην ασφάλεια τοπικών δικτύων IIoT (LAN), οι κατασκευαστές και άλλοι βιομηχανικοί χρήστες IoT θα πρέπει να λαμβάνουν σοβαρά υπόψη την ασφάλεια ακόμη και εντός των ορίων του LAN τους. Πολλές συσκευές IIoT δεν κατασκευάστηκαν με τη βέλτιστη ασφάλεια. Η προτεραιότητα της ασφάλειας σε ένα LAN IIoT απαιτεί όλες οι συσκευές να προστατεύονται από μη εξουσιοδοτημένη πρόσβαση, ανεξάρτητα από τη λειτουργία της λειτουργικής τεχνολογίας τους.

Ασφαλής μετάδοση δεδομένων, διαφορετικά δημιουργείται μια άλλη πιθανή ευπάθεια για τα εργοστάσια παραγωγής IIoT λόγω της αυξημένης κοινής χρήσης δεδομένων σε δίκτυα συσκευών και εγκαταστάσεων IIoT. Καθώς ο όγκος των ευφυών μηχανημάτων συνεχίζει να αυξάνεται, δημιουργούνται περισσότεροι χώροι αποθήκευσης δεδομένων και πύλες που πρέπει να ασφαλιστούν. Μια παραβίαση ευαίσθητων δεδομένων θα μπορούσε να οδηγήσει σε κινδύνους για την ασφάλεια, δυσλειτουργία του εξοπλισμού ή χρόνο διακοπής της αυτο-άρνησης παροχής υπηρεσιών.

Τέλος, για να αυξήσουμε το επίπεδο προστασίας, συνιστάται η δημιουργία ισχυρών κωδικών πρόσβασης και πρόσβασης ελέγχου ταυτότητας δύο παραγόντων. Η κρυπτογραφημένη επικοινωνία θα μας βοηθήσει να προστατεύσουμε το απόρρητο των ψηφιακών δεδομένων. Είναι επίσης σημαντικό να διασφαλίσουμε ότι δεν θα ξεχάσουμε να κλείσουμε τις θύρες εγκαίρως και ότι προστατεύουμε την υποδομή IoT από την υπέρβαση του buffer. Προηγμένες υπηρεσίες όπως συστήματα αναγνώρισης φωνής, βιομετρικά στοιχεία ή η αρχή του ελάχιστου προνομίου (PoLP) μπορούν να χρησιμοποιηθούν για τον εντοπισμό και την πρόληψη πιθανών κινδύνων. [3]

1.4 Σκοπός

Το SWaT (Secure Water Treatment) είναι ένα βασικό πλεονέκτημα για τους ερευνητές που στοχεύουν στο σχεδιασμό ασφαλών κυβερνοφυσικών συστημάτων (CPS). Το WADI είναι μια φυσική επέκταση του SWaT, που περιλαμβάνει δύο υπερυψωμένες δεξαμενές, έξι δεξαμενές καταναλωτών, δύο δεξαμενές ακατέργαστου νερού και μια δεξαμενή επιστροφής. Έρχεται επίσης εξοπλισμένο με συστήματα δοσομέτρησης χημικών, ενισχυτικές αντλίες και βαλβίδες, όργανα και αναλυτές.

Σε αντίθεση με μια μονάδα συστήματος επεξεργασίας νερού που συνήθως περιέχεται σε μια ασφαλή τοποθεσία, ένα σύστημα διανομής περιλαμβάνει πολυάριθμους αγωγούς που εκτείνονται σε μια μεγάλη και συχνά λιγότερο ασφαλή περιοχή. Αυτό αυξάνει σημαντικά τον κίνδυνο φυσικών επιθέσεων σε ένα δίκτυο διανομής.

Σκοπός της παρούσας διπλωματικής εργασίας είναι η προσομοίωση διαδικασιών του συστήματος επεξεργασίας νερού δείχνοντας έτσι τη κανονική λειτουργία του.

1.5 Δομή Διπλωματικής Εργασίας

Η παρούσα Διπλωματική Εργασία αποτελείται από έξι κεφάλαια και είναι τα ακόλουθα:

Στο πρώτο κεφάλαιο γίνεται η γενική εισαγωγή στο θέμα, στο βιομηχανικό διαδίκτυο των πραγμάτων καθώς επίσης στα προβλήματα και την ασφάλεια και αναφέρεται ο σκοπός της διπλωματικής εργασίας.

Στο δεύτερο κεφάλαιο παρουσιάζονται τα ασύρματα δίκτυα αισθητήρων κι άλλες σχετικές δουλειές που έχουν γίνει.

Στο τρίτο κεφάλαιο γίνεται εισαγωγή στα συστήματα SCADA, διάφορες επιθέσεις που μπορούν να εμφανισθούν καθώς επίσης τρόπους αντιμετώπισης για την ασφάλεια αυτών των συστημάτων και διάφορους προσομοιωτές που χρησιμοποιούνται σε αυτού του είδους συστήματα.

Στο τέταρτο κεφάλαιο γίνεται λεπτομερής περιγραφή της μεθοδολογίας που έχει ακολουθηθεί καθώς και το πειραματικό περιβάλλον που έχει χρησιμοποιηθεί. Γίνεται αναφορά στον προσομοιωτή SCADASim εξηγώντας τον τρόπο λειτουργίας του όπως επίσης και στα στάδια του συστήματος επεξεργασίας νερού.

Στο πέμπτο κεφάλαιο παρουσιάζονται τα πειραματικά αποτελέσματα της προσομοίωσης.

Στο έκτο κεφάλαιο παρουσιάζονται τα τελικά συμπεράσματα και η μελλοντική δουλειά.

Κεφάλαιο 2

Ασύρματα Δίκτυα Αισθητήρων

2.1 Εισαγωγή

Τα ασύρματα δίκτυα αισθητήρων αποτελούνται από μια μεγάλη συλλογή οργανωμένων κόμβων οι οποίοι είναι ενωμένοι μεταξύ τους ασύρματα, σχηματίζοντας ένα δίκτυο. Οι κόμβοι αυτοί, είναι αισθητήρες μικρού μεγέθους και σχετικά χαμηλού κόστους. Οι αισθητήρες μπορούν να παρέχουν διάφορες πληροφορίες σχετικά με το περιβάλλον που βρίσκονται όπως θερμοκρασία, υγρασία, πίεση και κίνηση. Ο κάθε αισθητήρας έχει ενσωματωμένο έναν πομποδέκτη, έναν μικροεπεξεργαστή, μνήμη και μια πηγή ενέργειας, συνήθως μια μπαταρία. Τα συστατικά αυτά επιτρέπουν την εκτέλεση των λειτουργιών του αισθητήρα. Ένας αισθητήρας μπορεί να συλλέγει, να αποστέλλει, να λαμβάνει και να επεξεργάζεται δεδομένα, όπως επίσης και να επικοινωνεί με άλλους αισθητήρες. Η επικοινωνία μεταξύ τους γίνεται αυτόνομα χρησιμοποιώντας διάφορα πρωτόκολλα επικοινωνίας. [4]

2.2 Άλλες προσεγγίσεις/προτάσεις

Η ανίχνευση ανωμαλιών και εισβολής σε συστήματα βιομηχανικού ελέγχου (που ονομάζονται επίσης φυσικά συστήματα στον κυβερνοχώρο) έχουν μελετηθεί εκτενώς. Ένας αριθμός περιεκτικών ερευνών είναι αφιερωμένος στην ταξινόμηση τεχνικών και μεθοδολογιών σε αυτόν τον τομέα. [5] [6]

Μια πολύ γνωστή προσέγγιση για την ανίχνευση εισβολών στο ICS βασίζεται στη μοντελοποίηση και την προσομοίωση του συστήματος [7], [8]. Πρακτικά προβλήματα με αυτήν την προσέγγιση είναι η ανάγκη για ακριβή γνώση του σχεδιασμού και των διαμορφώσεων του συστήματος, καθώς και η ανάγκη για ακριβή μοντελοποίηση της περίπλοκης φυσικής συμπεριφοράς του συστήματος.

Κεφάλαιο 3

SCADA Συστήματα

3.1 Εισαγωγή

Τα βιομηχανικά Συστήματα Ελέγχου (ICS) είναι υπεύθυνα για τη διαχείριση και την επίβλεψη των βιομηχανικών διεργασιών που εκτελούνται από υποδομές ζωτικής σημασίας σε βιομηχανίες όπως ηλεκτρική, νερό, φυσικό αέριο. Τα συστήματα εποπτικού ελέγχου και συλλογής δεδομένων (SCADA) είναι Συστήματα Βιομηχανικού Ελέγχου (ICS) που έχουν γίνει η ύψιστη τεχνολογία για την παρακολούθηση και τον έλεγχο μεγάλης κλίμακας βιομηχανικών και ζωτικών υποδομών τις τελευταίες δεκαετίες και χρησιμοποιούνται ευρέως από τις βιομηχανίες για την παρακολούθηση και τον έλεγχο διαφορετικών διαδικασιών όπως το πετρέλαιο και το φυσικό αέριο, σωληνώσεις, διανομή νερού, δίκτυα ηλεκτρικής ενέργειας κ.λ.π. Αυτά τα συστήματα παρέχουν αυτοματοποιημένο έλεγχο και απομακρυσμένη παρακολούθηση των υπηρεσιών που χρησιμοποιούνται στην καθημερινή ζωή. Για παράδειγμα, χρησιμοποιούνται για την παρακολούθηση και τη ρύθμιση των επιπέδων νερού σε δεξαμενές, την πίεση των σωλήνων και τη διανομή νερού. Ένα τυπικό σύστημα SCADA περιλαμβάνει στοιχεία όπως σταθμούς εργασίας υπολογιστή, Ανθρώπινη Μηχανή Διεπαφής (HMI), Προγραμματιζόμενοι Λογικοί Ελεγκτές (PLC), αισθητήρες και ενεργοποιητές. Ιστορικά, αυτά τα συστήματα είχαν ιδιωτικά και αποκλειστικά δίκτυα. Ωστόσο, λόγω της ευρείας εμβέλειας ανάπτυξης της απομακρυσμένης διαχείρισης, τα ανοιχτά δίκτυα IP (π.χ. Διαδίκτυο) χρησιμοποιούνται πλέον για συστήματα SCADA επικοινωνίες. Αυτό εκθέτει τα συστήματα SCADA στον κυβερνοχώρο και τα καθιστά ευάλωτα σε κυβερνοεπιθέσεις μέσω του Διαδικτύου.

3.2 Επιθέσεις στα SCADA συστήματα

Γνωρίζουμε ότι είναι σημαντικό να κατανοούμε και να γνωρίζουμε τις πραγματικές απειλές και τα τρωτά σημεία που υπάρχουν στα συστήματα SCADA. Εξάλλου, δεν μπορούμε να υπερασπιστούμε το δίκτυό μας από κάτι για το οποίο δεν γνωρίζουμε τίποτα.

Ο σκοπός μιας κυβερνοεπίθεσης σε ένα σύστημα SCADA μπορεί να κυμαίνεται από έναν χάκερ που προσπαθεί να αποδείξει ότι μπορεί να περάσει τις άμυνές σας, μέχρι έναν τρομοκράτη που θέλει να βλάψει έναν μεγάλο αγωγό μεταφοράς πετρελαιοειδών. Ενδεχομένως κάποιος να οργανώσει μια επίθεση για κατασκοπευτικούς (βιομηχανικούς) σκοπούς ή για να δημιουργήσει «ψευδείς» πληροφορίες στο σύστημα SCADA. Οι πιο σοβαρές απειλές είναι αυτές που σκοπεύουν είτε να απενεργοποιήσουν σοβαρά το σύστημα (το οποίο θα μπορούσε να περιλαμβάνει τη δημιουργία ψευδών δεδομένων έτσι ώστε οι χειριστές να μην γνωρίζουν τα προβλήματα που αναπτύσσονται) είτε εκείνες που προσπαθούν να δώσουν εντολή στο σύστημα να προκαλέσει ζημιά στη διεργασία ή τον εξοπλισμό που ελέγχεται με αποστολή ακατάλληλες εντολές ελέγχου. [10]

Έτσι, για να έχουμε μια καλύτερη εικόνα για τα περιστατικά hacking SCADA, ας ρίξουμε μια ματιά σε ένα χρονοδιάγραμμα των πρόσφατων κυβερνοεπιθέσεων στα συστήματα SCADA:

Παραδείγματα απειλών για τα συστήματα SCADA, περιλαμβάνουν μια επίθεση σε σύστημα επεξεργασίας λυμάτων στο Maroochy Shire του Κουίνσλαντ, όπου 800.000 λίτρα ακατέργαστων λυμάτων απελευθερώθηκαν για να χυθούν σε τοπικά πάρκα και ποτάμια, προκαλώντας θάνατο της θαλάσσιας ζωής, δυσωδία και αποχρωματισμό του νερού [11], ο πυρηνικός σταθμός Davis-Besse στο Oak Harbor του Οχάιο, δέχτηκε επίθεση από τον ιό τύπου worm διακομιστή Slammer SQL, ο οποίος απενεργοποίησε ένα σύστημα παρακολούθησης ασφάλειας του πυρηνικού σταθμού για σχεδόν 5 ώρες [12]. Πιο πρόσφατα, ανακαλύφθηκε το Stuxnet [13] μια απειλή ειδικά γραμμένη για να στοχεύει συστήματα βιομηχανικού ελέγχου.

3.3 Ασφάλεια στα SCADA Συστήματα

Πρόσφατες επιθέσεις σε συστήματα SCADA υπογραμμίζουν την ανάγκη για ισχυρότερη ασφάλεια SCADA. Είναι σημαντικό να αναλύσουμε τους κινδύνους ασφαλείας και να αναπτύξουμε κατάλληλες λύσεις ασφάλειας για την προστασία τέτοιων συστημάτων. Ωστόσο, ένα βασικό πρόβλημα είναι η έλλειψη κατάλληλων εργαλείων μοντελοποίησης για την αξιολόγηση της ασφάλειας των συστημάτων SCADA. Όπως είναι ευρέως αποδεκτό στις ακαδημαϊκές και βιομηχανικές κοινότητες, δεν είναι πρακτικό να διεξάγονται πειράματα ασφαλείας σε ζωντανά συστήματα. Ένα εργαλείο προσομοίωσης μοντελοποίησης θα επέτρεπε την προσομοίωση συστημάτων SCADA με το πλεονέκτημα της δοκιμής διαφορετικών λύσεων επίθεσης και ασφάλειας. Για την καλύτερη κατανόηση της προστασίας των

συστημάτων SCADA, είναι σημαντικό να αναλυθούν οι κίνδυνοι ασφάλειας τέτοιων συστημάτων και να αναπτυχθούν κατάλληλες λύσεις για την προστασία τους από κακόβουλες επιθέσεις. Ένα βασικό πρόβλημα στην έρευνα και ανάπτυξη λύσεων ασφαλείας για συστήματα SCADA είναι η έλλειψη κατάλληλων εργαλείων μοντελοποίησης για την αξιολόγηση της ασφάλειας τέτοιων συστημάτων. Ένα πλαίσιο προσομοίωσης SCADA θα επέτρεπε τη δημιουργία απλών μοντέλων συστημάτων SCADA με το πρόσθετο πλεονέκτημα της δοκιμής πραγματικών επιθέσεων και της δοκιμής διαφορετικών λύσεων ασφαλείας.

Πιο κάτω βλέπουμε μερικούς τρόπους που μπορούμε ν' ακολουθήσουμε για να εξασφαλίσουμε ασφάλεια στα Συστήματα SCADA: [14]

- Να προσδιορίσουμε όλες τις συνδέσεις σε δίκτυα SCADA.

- Αξιολογούμε και ενισχύουμε την ασφάλεια τυχόν υπολειπόμενων συνδέσεων στο δίκτυο SCADA.

- Μην βασίζόμαστε σε ιδιόκτητα πρωτόκολλα για την προστασία του συστήματός σας, να εφαρμόσουμε κατάλληλους μηχανισμούς ελέγχου ταυτότητας και εξουσιοδότησης.

- Να καθιερώσουμε ισχυρούς ελέγχους σε οποιοδήποτε μέσο που χρησιμοποιείται ως backdoor στο δίκτυο SCADA.

- Εφαρμογή εσωτερικών και εξωτερικών συστημάτων ανίχνευσης εισβολών και καθιέρωση 24ωρης παρακολούθησης συμβάντων.

- Πραγματοποιήση τεχνικών ελέγχων των συσκευών και δικτύων SCADA, καθώς και οποιωνδήποτε άλλων συνδεδεμένων δικτύων, για να εντοπίσουμε προβλήματα ασφάλειας.

- Δημιουργία SCADA «Red Teams» για να εντοπίσουν και να αξιολογήσουν πιθανά σενάρια επίθεσης.

- Να καθιερώσουμε μια στρατηγική προστασίας δικτύου που βασίζεται στην αρχή της άμυνας σε βάθος μαζί με μια διαδικασία διαχείρισης κινδύνου.

- Να προσδιορίσουμε με σαφήνεια τις απαιτήσεις και τις διαδικασίες κυβερνοασφάλειας που χρησιμοποιούνται για την τακτική αξιολόγηση του δικτύου για γνωστά τρωτά σημεία.

- Δημιουργία αντιγράφων ασφαλείας του συστήματος και σχέδια αποκατάστασης καταστροφών.

- Να Καθιερώσουμε πολιτικές και πραγματοποιήστε εκπαίδευση για να ελαχιστοποιήσετε την πιθανότητα το οργανωτικό προσωπικό να αποκαλύψει κατά λάθος ευαίσθητες πληροφορίες σχετικά με το σχεδιασμό του συστήματος SCADA, τις λειτουργίες ή τους ελέγχους ασφαλείας που υπάρχουν.

3.4 Εργαλεία Προσομοίωσης

Οι προσομοιωτές SCADA χρησιμοποιούνται για να βοηθήσουν ως προς τον σχεδιασμό, τη βελτιστοποίησης πόρων, της εκπαίδευσης προσωπικού, των ελέγχων ασφαλείας, της πρόβλεψης ζήτησης και της ευρωστίας σε σενάρια καταστροφών.

Οι προσομοιωτές δικτύου, προσπαθούν να αντιγράψουν και να μοντελοποιήσουν πραγματικά δίκτυα. Παρά το γεγονός ότι τον τρόπο που οι προσομοιωτές δικτύου δεν είναι άψογοι, μπορούν να δώσουν μια ζωτικής σημασίας κατανόηση του δοκιμαζόμενου δικτύου και το πώς οι αλλαγές θα επηρεάσουν τη λειτουργία του. Μερικοί απ' αυτούς είναι: [15]

- Ο NS2 (Network Simulator 2) ο οποίος είναι ένας προσομοιωτής ανοιχτού κώδικα που βασίζεται σε συμβάντα που έχει σχεδιαστεί για έρευνα σε δίκτυα επικοινωνίας υπολογιστών. Μπορεί να προσομοιώσει υπάρχοντα πρωτόκολλα δικτύου όπως πρωτόκολλα TCP, UDP, δρομολόγησης και πολλαπλής διανομής τόσο σε ενσύρματα όσο και σε ασύρματα δίκτυα. Επιτρέπει τη δημιουργία διαφόρων σεναρίων επικοινωνίας χρησιμοποιώντας συγκεκριμένα πρωτόκολλα και προσομοίωση της συμπεριφοράς αυτών των πρωτοκόλλων υπό διαφορετικές συνθήκες. Το NS2 είναι ιδιαίτερα σχετικό με τις βιομηχανικές προσομοιώσεις, καθώς υπάρχει μια τάση ενθυλάκωσης ιδιόκτητων πακέτων πρωτοκόλλων σε TCP που εκτελούνται μέσω δικτύων Ethernet.

- Ο NS3 (Network Simulator 3) έχει δανειστεί έννοιες και υλοποιήσεις από αρκετούς προσομοιωτές ανοιχτού κώδικα. Υποστηρίζει εικονικοποίηση, ενοποίηση λογισμικού και δοκιμαστικής κλίνης και εστιάζει την προσοχή του στον ρεαλισμό υποστηρίζοντας βασικές διεπαφές όπως πρίζες και συσκευές δικτύου, πολλαπλές διεπαφές ανά κόμβο και χρήση διευθύνσεων IP.

- Ο προσομοιωτής OPNET είναι ένα εργαλείο για την προσομοίωση της συμπεριφοράς και της απόδοσης οποιουδήποτε τύπου δικτύου. Βασίζεται σε διακριτό μηχανισμό συμβάντων συστήματος που προσομοιώνει τη συμπεριφορά του συστήματος μοντελοποιώντας τα συμβάντα των σεναρίων που έχει ρυθμίσει ο χρήστης. Το κύριο χαρακτηριστικό του είναι ότι παρέχει διάφορες πραγματικές δυνατότητες διαμόρφωσης δικτύου που κάνουν το περιβάλλον

προσομοίωσης κοντά στην πραγματικότητα. Τέλος, επιτρέπει την ανάπτυξη των δικτύων, των πρωτοκόλλων και των μορφών πακέτων του χρήστη παρέχοντας τα απαραίτητα εργαλεία προγραμματισμού και τεκμηρίωση.

Κεφάλαιο 4

Υλοποίηση

4.1 Εισαγωγή

Στην παρούσα διπλωματική εργασία έχουν προσομοιωθεί οι διαδικασίες του συστήματος επεξεργασίας νερού κι έχουν υλοποιηθεί στον προσομοιωτή SCADASim. Στο κεφάλαιο αυτό κάνουμε μια λεπτομερή περιγραφή στον προσομοιωτή SCADASim αναφέροντας τα βασικά του στοιχεία και τα πλεονέκτηματα τα οποία διακατέχει για τέτοιου είδους συστήματα κι επίσης, γίνεται περιγραφή των σταδίων του συστήματος επεξεργασίας νερού για καλύτερη κατανόηση ως προς τον τρόπο λειτουργίας του.

4.2 Εργαλείο Προσομοίωσης SCADASim

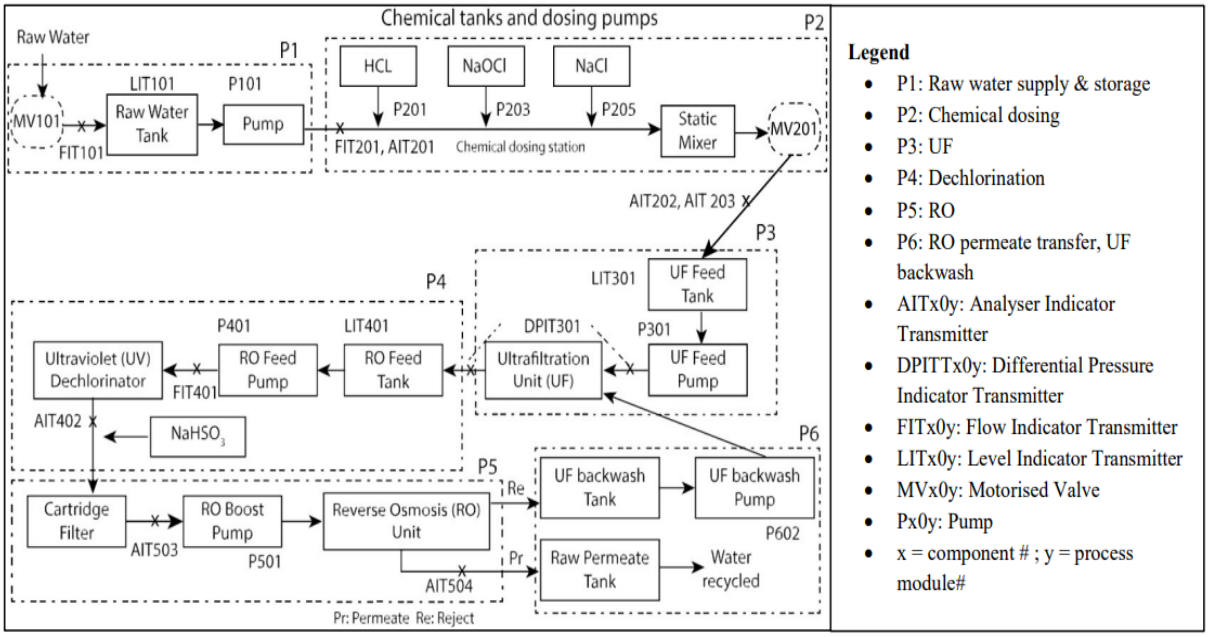
Ο προσομοιωτής SCADASim, είναι ένα πλαίσιο για την κατασκευή προσομοιώσεων SCADA. Παρέχει ένα αρθρωτό εργαλείο μοντελοποίησης SCADA που επιτρέπει την επικοινωνία σε πραγματικό χρόνο με εξωτερικές συσκευές χρησιμοποιώντας πρωτόκολλα SCADA. Προσθέτει αξία για τη σύνδεση πραγματικών και προσομοιωμένων συσκευών και επίσης προσφέροντας μια λύση plug-n-play που υποστηρίζει πολλαπλά πρωτόκολλα και εξωτερικές εφαρμογές για τη δημιουργία προσομοιώσεων SCADA.

Το SCADASim παρέχει ένα σύνολο μονάδων που αντιπροσωπεύουν στοιχεία SCADA όπως RTU, PLC, MTU και πρωτόκολλα (όπως Modbus/TCP και DNP3). Τέτοιες ενότητες και πρωτόκολλα μπορούν εύκολα να επεκταθούν, να συνδυαστούν σε άλλες ενότητες και να χρησιμοποιηθούν σε οποιαδήποτε προσομοίωση SCADASim.

Το SCADASim υποστηρίζει τέσσερις κύριους τύπους επιθέσεων: άρνηση υπηρεσίας, man-in-the-middle, υποκλοπή και πλαστογράφηση.

Ένα βασικό πλεονέκτημα του SCADASim είναι η υψηλή του επεκτασιμότητα, ενώ ενσωματώνει και πραγματικές μονάδες υλικού και πρωτότυπου λογισμικού. Επιπλέον, η ενσωμάτωση υπαρχόντων εξωτερικών στοιχείων σε προσομοιώσεις απλοποιεί την αξιολόγηση αυτών των στοιχείων, καθώς οι τοπολογίες, τα μοτίβα κυκλοφορίας και άλλες παράμετροι μπορούν να αλλάξουν εύκολα μέσα στο εργαλείο. [16]

4.3 Στάδια του Συστήματος Ασφάλειας Επεξεργασίας Νερού



Εικόνα 4.1: SWaT's six-stage processes (iTrust Centre for Research in Cyber Security)

Το SWaT είναι ένα βασικό πλεονέκτημα για τους ερευνητές που στοχεύουν στο σχεδιασμό ασφαλών κυβερνοφυσικών συστημάτων (CPS). Η δοκιμαστική βάση αποτελείται από μια σύγχρονη διαδικασία επεξεργασίας νερού έξι σταδίων που μιμείται στενά μια μονάδα επεξεργασίας του πραγματικού κόσμου. Το πρώτο στάδιο της φυσικής διαδικασίας ξεκινά με λήψη ακατέργαστου νερού, ακολουθούμενη από χημική δοσολογία (Στάδιο 2) και φιλτράρισμα μέσω συστήματος υπερδιήθησης (Ultra-Filtration - Στάδιο 3), αποχλωρίωση με χρήση λαμπτήρων UV (Στάδιο 4) και στη συνέχεια τροφοδοσία σε σύστημα αντίστροφης όσμωσης (Reverse Osmosis) (Στάδιο 5). Μια διαδικασία αντίστροφης πλύσης (Στάδιο 6) καθαρίζει τις μεμβράνες σε UF χρησιμοποιώντας το διήθημα RO. Κάθε ένα στάδιο που αναφέρονται ως P1 έως P6, ελέγχεται από ένα σύνολο PLCs. [17]

4.3.1 Πρώτο Στάδιο - Raw water supply & storage

Αυτή η διαδικασία αναφέρεται στην παροχή και αποθήκευση ακατέργαστου νερού κι αποτελείται όπως βλέπουμε από τον κάτω πίνακα, από δύο αισθητήρες (sensors) που είναι οι FIT101 (Flow Indicator Transmitter) και LIT101 (Level Transmitter) και τρεις ενεργοποιητές (actuators), MV101 (Motorized Valve) και δύο αντλίες όπου η δεύτερη λειτουργεί ως εφεδρική αντλία.

Συγκεκριμένα στις εικόνες 4.2 και 4.3 βλέπουμε τις εξής λειτουργίες:

Ο MV101 ελέγχει τη ροή του νερού στη δεξαμενή ακατέργαστου νερού. Στην αρχή, η βαλβίδα έχει την τιμή 1 για να δείξει ότι εκείνη τη στιγμή είναι κλειστή, δεν λειτουργεί. Στη συνέχεια, όταν η ροή του νερού ξεκινά προς τη δεξαμενή νερού, η τιμή του αλλάζει σε 2 για να δείξει ότι η βαλβίδα είναι ανοιχτή.

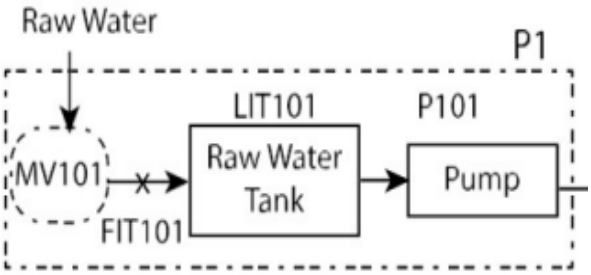
Ο FIT101 μετρά την εισροή στη δεξαμενή ακατέργαστου νερού. Στην αρχή, όπως η Μηχανοκίνητη Βαλβίδα, το FIT101 είναι κλειστό αλλά με την τιμή 0. Όταν το MV101 αλλάξει την τιμή του σε 2 τότε ο Πομπός Ροής αρχίζει να μετρά την εισροή νερού.

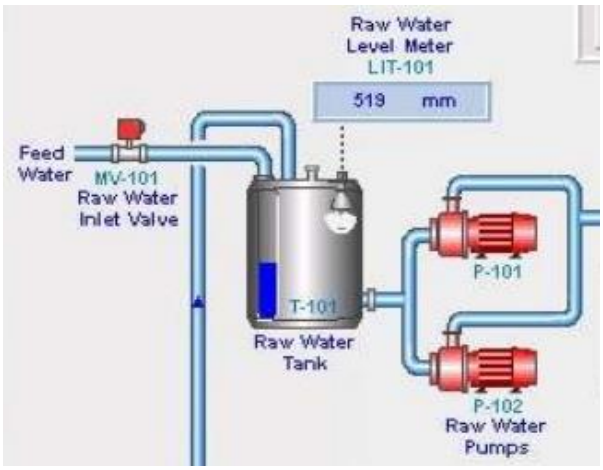
Ο LIT101 μετρά τη στάθμη της δεξαμενής ακατέργαστου νερού. Όταν το MV101 είναι ανοιχτό, τότε η στάθμη του δοχείου νερού αυξάνεται μέχρι να φτάσει στη μέγιστη τιμή στάθμης. Μετά από κάποιο χρονικό διάστημα, όταν η στάθμη του νερού φτάσει στη μέγιστη τιμή, τότε η P-101 (αντλία) από την τιμή 1 (κλειστή) μεταβαίνει στο 2 για να αντλήσει το νερό και να περάσει μέσα από αυτό και η στάθμη της δεξαμενής νερού αρχίζει να μειώνεται.

Ο κύριος ρόλος του **P101** είναι να αντλεί νερό από τη δεξαμενή ακατέργαστου νερού στο δεύτερο στάδιο (δεύτερη διαδικασία).

Πίνακας περιγραφής των field devices του πρώτου σταδίου

Name	Type	Description
FIT-101	Sensor	Flow meter; Measures inflow into raw water tank.
LIT-101	Sensor	Level Transmitter; Raw water tank level.
MV-101	Actuator	Motorized valve; Controls water flow to the raw water tank.
P-101	Actuator	Pump; Pumps water from raw water tank to second stage.
P-102 (backup)	Actuator	Pump; Pumps water from raw water tank to second stage.





Εικόνα 4.2 **Εικόνα 4.3**

Για την υλοποίηση του πρώτου σταδίου του συστήματος όπως και των υπολοίπων που θα δούμε στην συνέχεια, ο κύριος μας στόχος ήταν η επικοινωνία μεταξύ των συσκευών PLCs (Programmable Logic Controllers) όπου το κάθε PLC αντιστοιχεί σε μια συσκευή πεδίου (field device). Γενικά, το PLC επικοινωνεί με το σταθμό εργασίας και τις συσκευές πεδίου χρησιμοποιώντας τα πρωτόκολλα επικοινωνίας SCADA (π.χ. Modbus, S7comm) κι ελέγχει μια βιομηχανική διαδικασία.

Μια συνομιλία ξεκινά πάντα από έναν master στο δίκτυο Modbus. Ένας master Modbus στέλνει ένα μήνυμα και ανάλογα με τα περιεχόμενα του μηνύματος, ο slave (PLC) ερμηνεύει το μήνυμα και απαντά σε αυτό. Η φυσική υποτελής διευθυνσιοδότηση στην κεφαλίδα του μηνύματος χρησιμοποιείται για να ορίσει ποια εξαρτημένη συσκευή πρέπει να ανταποκρίνεται σε ένα μήνυμα. Όλοι οι άλλοι κόμβοι στο δίκτυο Modbus αγνοούν το μήνυμα εάν το πεδίο διεύθυνσης δεν ταιριάζει με τη δική τους διεύθυνση.

Modbus Protocol Devices

Οι συσκευές πρωτοκόλλου Modbus θα περιλαμβάνουν συνήθως έναν "χάρτη καταχωρητή" που περιγράφει από πού μπορούν να γραφτούν και να διαβαστούν τα δεδομένα διαμόρφωσης, εισόδου και εξόδου. Το μοντέλο δεδομένων Modbus έχει μια απλή δομή που περιγράφεται σε τέσσερις βασικούς τύπους δεδομένων:

1. Discrete Inputs

2. Coils Outputs

3. Input Registers (Input Data)

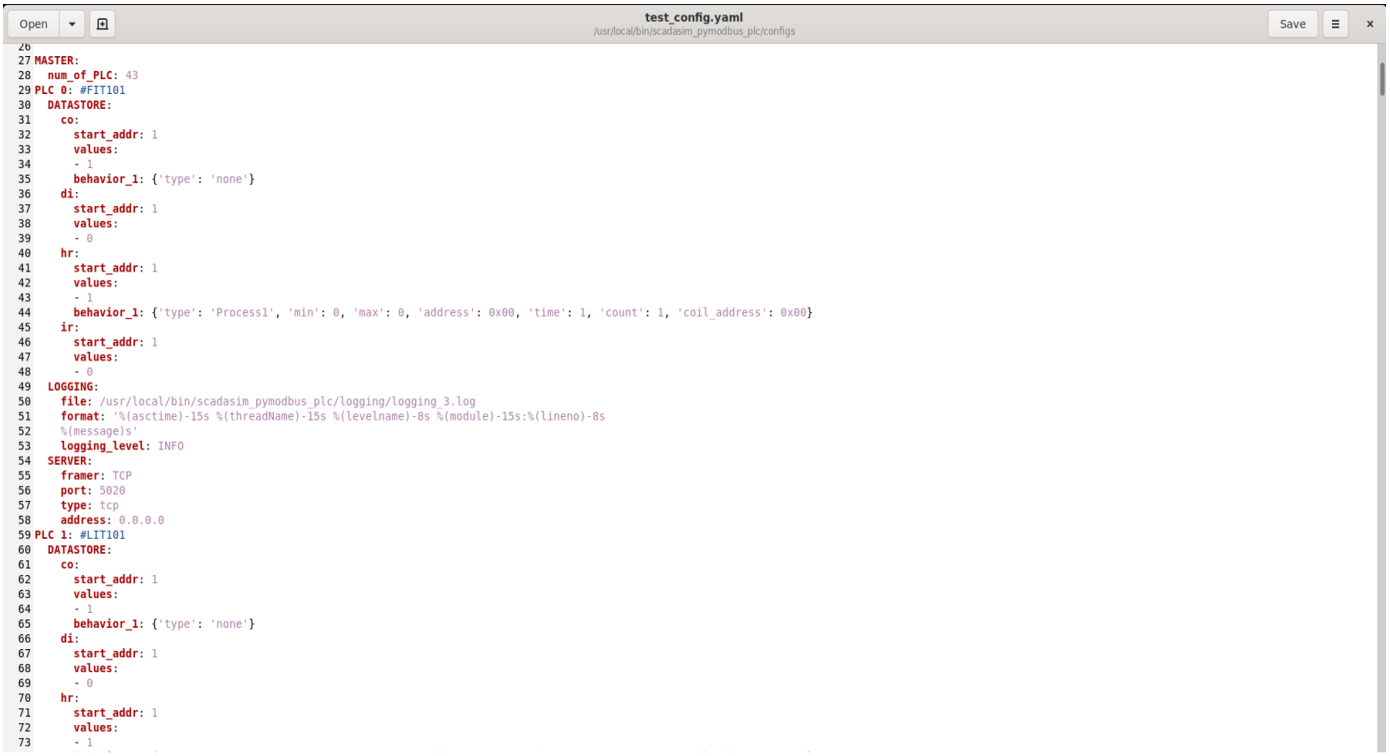
4. Holding Registers (Output Data)

Τα δεδομένα του Modbus διαβάζονται και γράφονται συχνότερα ως "καταχωρητές". Στην παρούσα διπλωματική εργασία, χρησιμοποιήσαμε τους Holding και Coil registers οι οποιοί μπορούν και να διαβάσουν αλλά και να γράψουν δεδομένα, για τους sensors και actuators αντίστοιχα. Ο τύπος του καταχωρητή που απευθύνεται σε ένα αίτημα Modbus καθορίζεται από τον κωδικό λειτουργίας (function code). Ο πιο συνηθισμένος κωδικός είναι ο 3 για "ανάγνωση και για γράψιμο του holding register" και μπορεί να διαβάζουν 1 ή περισσότερους.

Η υλοποίηση του συστήματος είχε ως βάσει ένα κώδικα στην γλώσσα προγραμματισμού python2.7 από το GitHub και στη συνέχεια κτίσαμε πάνω σ' αυτόν τροποποιώντας και προσθέτωντας νέες λειτουργίες. Αρχικά, διαβάζουμε μέσω ενός csv αρχείου τα δεδομένα του συστήματος για όλα τα στάδια. Στη συνέχεια, για κάθε στάδιο δημιουργήσαμε την αντίστοιχη συνάρτηση η οποία περιέχει τις ακόλουθες παραμέτρους:

Για το πρώτο στάδιο έχουμε τη συνάρτηση με το όνομα "Process1" στην οποία θα εξηγήσουμε πιο κάτω τον τρόπο υλοποιήσης της έτσι ώστε να διαβάζει τα δεδομένα του συστήματος από το csv αρχείο χρησιμοποιώντας του δύο καταχωρητές hr και co τους οποίους αναφέραμε πιο πάνω. Το κάθε στάδιο περιλαμβάνει ένα αριθμό από field devices (sensors and actuators) από τα οποία το καθένα αντιστοιχεί σ' ένα thread με τη δικιά του διεύθυνση. Αυτές οι διευθύνσεις έχουν ανατεθεί σ'ενα yaml αρχείο με τ' όνομα "test_config" όπου εκεί περιλαμβάνονται όλα τα threads (field devices) του συστήματος τα

οποία το κάθε thread αντιστοιχεί και σ' ένα PLC device. Επίσης, στο yaml αρχείο δηλώνουμε σε ποια συνάρτηση από τις έξι που δημιουργήσαμε (έξι στάδια του συστήματος) θα μεταφερθεί το συγκεκριμένο thread μαζί με τη συγκεκριμένη διεύθυνση έτσι ώστε να δημιουργηθούν τα σωστά αποτελέσματα των δεδομένων για κάθε field device. Για καλύτερη κατανόηση, πιο κάτω θα δούμε μερικά screenshots του yaml αρχείου δείχνοντας αυτά που εξηγήσαμε μόλις τώρα.



Επιπρόσθετα, βλέπουμε από τις πιο κάτω εικόνες ότι σην αρχή της συνάρτησης όπως και στις υπόλοιπες, δημιουργήσαμε αρκετούς πίνακες για τα field devices με αποτέλεσμα να αποθηκεύουμε τα νέα δεδομένα όταν συμβαίνει μια αλλαγή κατάστασης στο σύστημα. Όπως γνωρίζουμε ήδη στο πρώτο στάδιο, έχουμε 4 threads (field devices) κι αν η διεύθυνση στην οποία ελέγχουμε είναι η ίδια, τότε αρχίζουμε να τυπώνουμε τα αποτελέσματα του συγκεκριμένου field device δηλώνοντας την στήλη στην οποία θα παίρνουμε τα αποτέσματα από το csv αρχείο. Στη συγκεκριμένη περίπτωση, όσον αφορά για τους sensors του πρώτου στάδιου (FIT101 και LIT101) μπορούν να συμβούν τρείς αλλαγές κατάστασης. Ας δούμε πρώτα τις καταστάσεις του FIT101: η πρώτη είναι όταν το water flow δεν περνά ακόμη από τη σωλήνα και οι τιμές στις οποίες διαβάζει είναι πολύ χαμηλές, η δεύτερη όταν μετά από κάποιο χρονικό διάστημα κι εφόσον ανοίξει η βαλβίδα (MV101) αρχίσει και περνά νερό από τη σωλήνα δείχνοντας τη normal κατάσταση του water flow και τέλος όταν φτάσει στο σημείο ο FIT101 να διαβάζει τις μέγιστες τιμές. Αυτές τις τρείς καταστάσεις μπορούμε να τις δούμε και στις πιο κάτω εικόνες στο κομμάτι της υλοποιήσης.

Open ▾ helper.py /usr/local/bin/scadasim_pymodbus_plc/plc Save

```
686
687
688
689 def Process1(MIN, MAX, time, address, slave_id, count, context, log, my_backup, coil_address, df, n, FlagFIT101, FLIT101, FlagP101):
690
691     MaxFIT101 = []
692     MinFIT101 = []
693     NormalFIT101 = []
694     FIT101 = []
695
696
697     MaxLIT101 = []
698     MinLIT101 = []
699     NormalLIT101 = []
700     LIT101 = []
701
702     ON_MV101 = []
703     OFF_MV101 = []
704     P_MV101 = []
705     MV101 = []
706
707     ON_P101 = []
708     OFF_P101 = []
709     P101 = []
710
711     global FlagMV101
712
713     message1 = message()
714     message2 = message()
715     message3 = message()
716     message4 = message()
717
718
719     FlagMV101 = False
720
721
722     row,col=df.shape
723
724     schedule.every(1).minutes.do(func, context, slave_id, address, count, MAX, MIN, FIT101,  MinFIT101, NormalFIT101, MaxLIT101, MinLIT101, NormalLIT101, ON_P101, OFF_P101, ON_MV101, OFF_MV101, P_MV101,
    FlagFIT101, FLIT101, FlagP101, message1, message2, message3, message4)
725
726     for lines in range(2,row-1):
727
728         if(address == 0): # Checking the address of the thread
729
730
731                 with open("/usr/local/bin/scadasim_pymodbus_plc/plc/FIT101.txt", "a") as f:
732                         value = float(df['FIT101'][lines]) # Data for the column 'FIT101' from the csv file
733
734                         f.write(str(value)) # The data will be written in a txt file.
735                         f.write("\n")
736
```

Open ▾ helper.py /usr/local/bin/scadasim_pymodbus_plc/plc Save

```
719     FlagMV101 = False
720
721
722     row,col=df.shape
723
724     schedule.every(1).minutes.do(func, context, slave_id, address, count, MAX, MIN, FIT101,  MinFIT101, NormalFIT101, MaxLIT101, MinLIT101, NormalLIT101, ON_P101, OFF_P101, ON_MV101, OFF_MV101, P_MV101,
    FlagFIT101, FLIT101, FlagP101, message1, message2, message3, message4)
725
726     for lines in range(2,row-1):
727
728         if(address == 0): # Checking the address of the thread
729
730
731                 with open("/usr/local/bin/scadasim_pymodbus_plc/plc/FIT101.txt", "a") as f:
732                         value = float(df['FIT101'][lines]) # Data for the column 'FIT101' from the csv file
733
734                         f.write(str(value)) # The data will be written in a txt file.
735                         f.write("\n")
736
737
738
739                         if (value >= (MAX-0.05)): # if the value is close enough to the range of the maximum value of FIT101
740
741                                 FIT101.append(value) # we collect these values
742                                 sleep(1)
743
744                                 if (float(df['FIT101'][lines+1])<(MAX-0.05)):
745
746                                         now  = datetime.now() # To show the exact datetime
747
748                                         current_time = now.strftime("%H:%M:%S") # TO show the exact time
749
750                                         #FlagFIT101 = True
751
752                                         FIT101.append(df['FIT101'][lines+1])
753                                         write_hr_register(context[0], slave_id, address, FIT101) # Adding the values from the table FIT101 to the holding register
754                                         values = read_hr_register(context[0], slave_id, address, len(FIT101)) # To read those values from the table FIT101
755
756
757                                         if (df['FIT101'][lines+1]<=(MIN+0.05) : # To check if the status for the sensor FIT101 has changed
758                                                 print("Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is almost empty  --> "+ str(values[len(values)-1]))
759
760                                                 message1.message = "Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is almost empty  --> "+
    str(values[len(values)-1])
761
762                                         else:
763                                                 print("Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is normal  --> "+ str(values[len(values)-1]))
764
765                                                 message1.message = "Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is normal  --> "+
    str(values[len(values)-1])
766
767
```

Open ▾ | helper.py — /usr/local/bin/scadasim_pymodbus_plc/plc | Save

```
771
772
773                     if (value <= (MIN+0.05)): # if the value is close enough to the range of the minimum value of FIT101
774
775                             MinFIT101.append(value) # To collect these values which are low
776                             sleep(1)
777
778
779                             if (float(df['FIT101'][lines+1])>(MIN+0.05)):
780
781                                     now  = datetime.now()
782
783                                     current_time = now.strftime("%H:%M:%S")
784
785
786                                     MinFIT101.append(df['FIT101'][lines+1])
787                                     write_hr_register(context[0], slave_id, address, MinFIT101) # Adding the values from the table MinFIT101 to the holding register
788                                     values = read_hr_register(context[0], slave_id, address, len(MinFIT101)) # To read those values from the table MinFIT101
789
790                                     if (df['FIT101'][lines+1])>=(MAX-0.05) : # To check if the status for the sensor FIT101 has changed
791                                             print("Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is almost full  --> "+ str(values[len(values)-1]))
792                                             message1.message = "Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is almost full  --> "+
  str(values[len(values)-1])
793
794                                     else:
795                                             print("Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is normal  --> "+ str(values[len(values)-1]))
796
797                                             message1.message = "Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is normal  --> "+
  str(values[len(values)-1])
798
799                                     del MinFIT101[:]
800
801
802
803                     if ((value < (MAX-0.05)) and (value > (MIN+0.05))): # if the value is normal and it's between the minimum and maximum value
804                             NormalFIT101.append(value) # To collect these values which are normal
805                             sleep(1)
806
807
808                             if ((float(df['FIT101'][lines+1])<=(MIN+0.05)) or (float(df['FIT101'][lines+1])>=(MAX-0.05)) ):
809
810                                     now  = datetime.now()
811
812                                     current_time = now.strftime("%H:%M:%S")
813
814                                     #FlagFIT101 = True
815
816                                     NormalFIT101.append(df['FIT101'][lines+1])
817                                     write_hr_register(context[0], slave_id, address, NormalFIT101) # Adding the values from the table NormalFIT101 to the holding register
818                                     values = read_hr_register(context[0], slave_id, address, len(NormalFIT101)) # To read these values from the table NOrmalFIT101
819
```

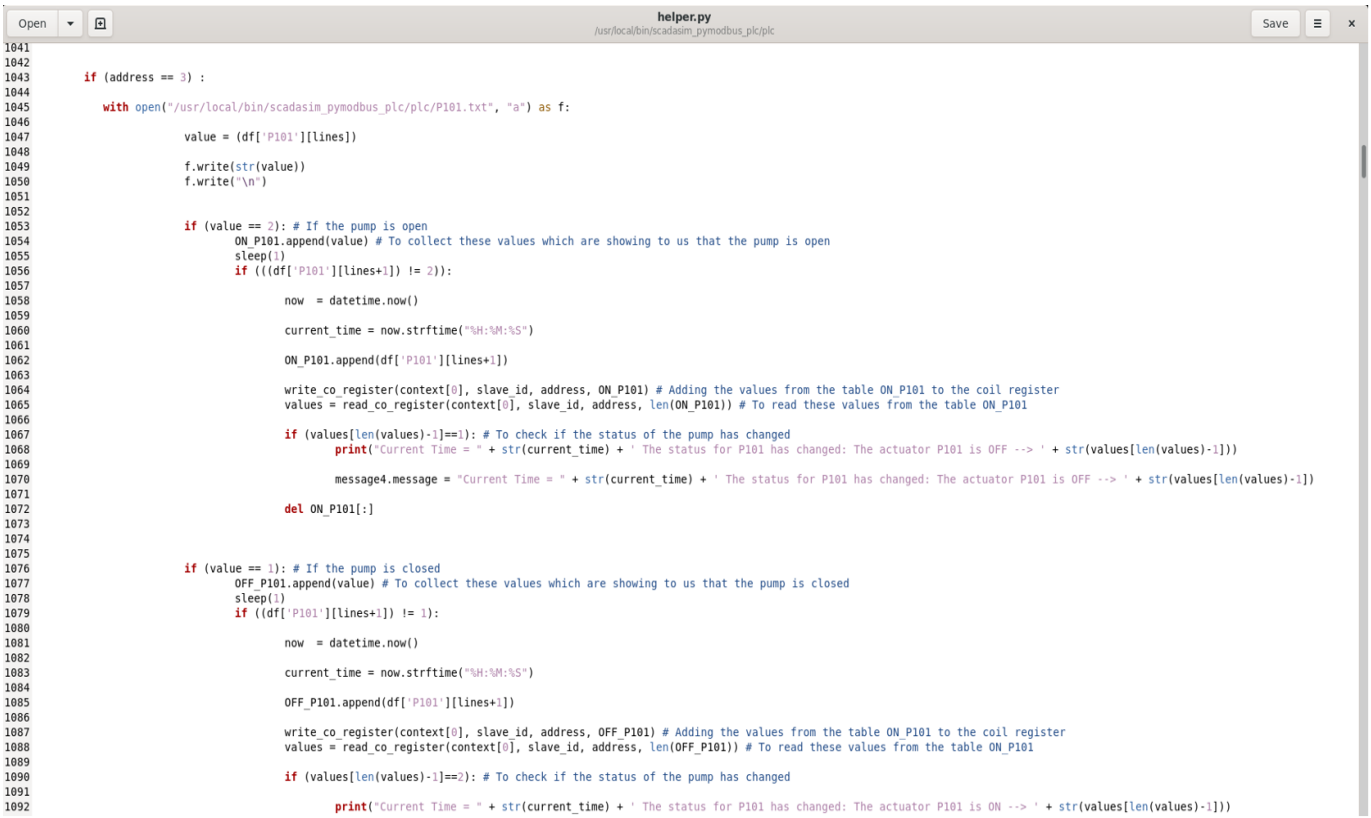
Open ▾ | helper.py — /usr/local/bin/scadasim_pymodbus_plc/plc | Save

```
787                                     write_hr_register(context[0], slave_id, address, MinFIT101) # Adding the values from the table MinFIT101 to the holding register
788                                     values = read_hr_register(context[0], slave_id, address, len(MinFIT101)) # To read those values from the table MinFIT101
789
790                                     if (df['FIT101'][lines+1])>=(MAX-0.05) : # To check if the status for the sensor FIT101 has changed
791                                             print("Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is almost full  --> "+ str(values[len(values)-1]))
792                                             message1.message = "Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is almost full  --> "+
  str(values[len(values)-1])
793
794                                     else:
795                                             print("Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is normal  --> "+ str(values[len(values)-1]))
796
797                                             message1.message = "Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is normal  --> "+
  str(values[len(values)-1])
798
799                                     del MinFIT101[:]
800
801
802
803                      if ((value < (MAX-0.05)) and (value > (MIN+0.05))): # if the value is normal and it's between the minimum and maximum value
804                              NormalFIT101.append(value) # To collect these values which are normal
805                              sleep(1)
806
807
808                              if ((float(df['FIT101'][lines+1])<=(MIN+0.05)) or (float(df['FIT101'][lines+1])>=(MAX-0.05)) ):
809
810                                     now  = datetime.now()
811
812                                     current_time = now.strftime("%H:%M:%S")
813
814                                     #FlagFIT101 = True
815
816                                     NormalFIT101.append(df['FIT101'][lines+1])
817                                     write_hr_register(context[0], slave_id, address, NormalFIT101) # Adding the values from the table NormalFIT101 to the holding register
818                                     values = read_hr_register(context[0], slave_id, address, len(NormalFIT101)) # To read these values from the table NOrmalFIT101
819
820                                     if (df['FIT101'][lines+1])>=(MAX-0.05) : # To check if the status for the sensor FIT101 has changed
821                                             print("Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is almost full  --> "+ str(values[len(values)-1]))
822
823                                             message1.message = "Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is almost full  --> "+
  str(values[len(values)-1])
824
825                                     else:
826                                             print("Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is empty  --> "+ str(values[len(values)-1]))
827
828                                             message1.message = "Current Time = " + str(current_time) + " The status for FIT101 has changed: The pump's water flow is empty  --> "+
  str(values[len(values)-1])
829
830                                     del NormalFIT101[:]
831
832
833
834                 f.close()
```

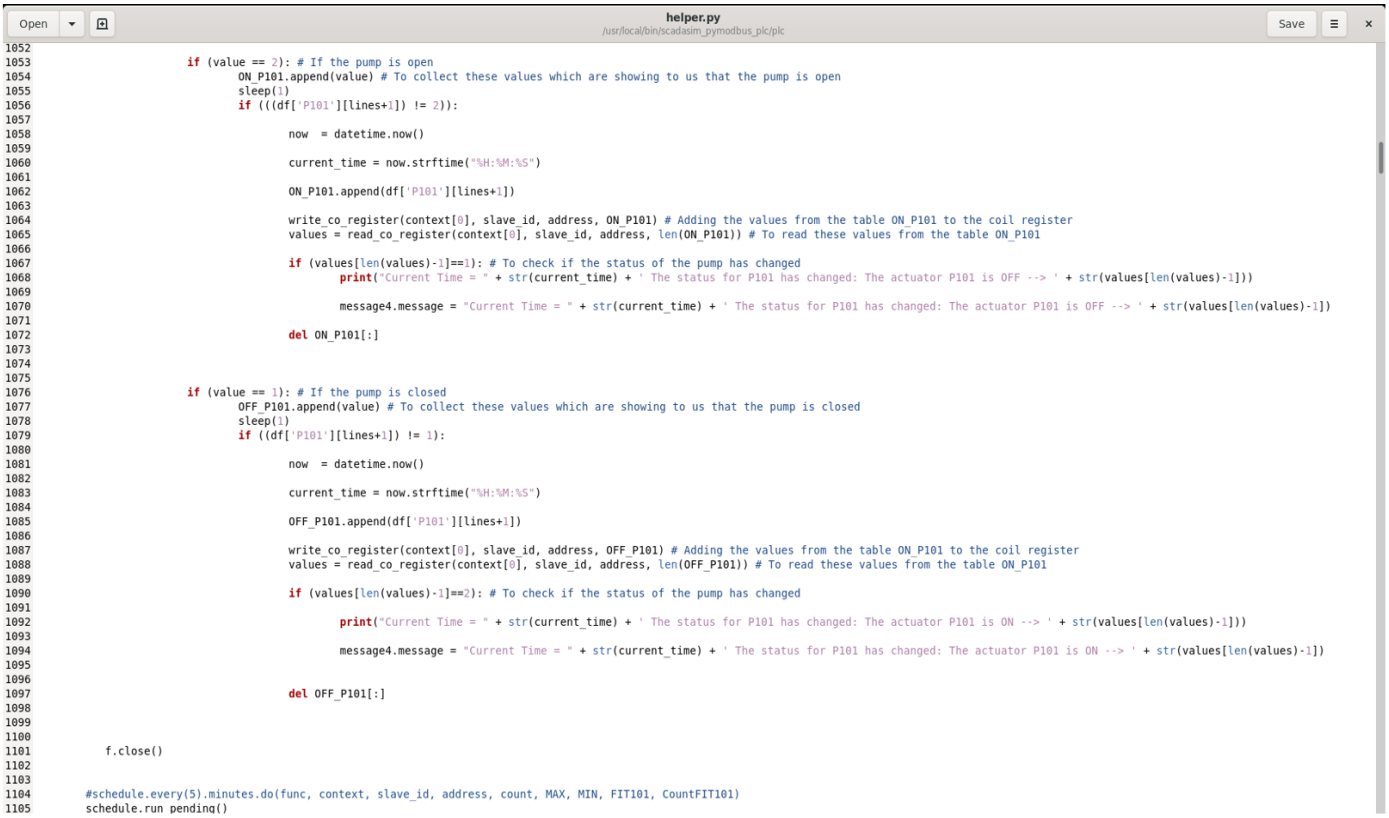
Επίσης, στην αρχή της συνάρτησης “Process1” χρησιμοποιούμε τη έτοιμη μέθοδο της Python την Schedule.every().minutes() η οποία μας εκτελεί κάθε x λεπτά τα αποτελέσματα του κάθε thread (field device) όπως επίσης και τα μηνύματα αλλαγής της κάθε κατάστασης.

Όσο αφορά τους actuators MV101 και P101, ο αριθμός καταστάσεων τους είναι τρείς και δύο αντίστοιχα. Η βαλβίδα στην αρχή είναι κλειστή, στη συνέχεια μετά από κάποιο χρονικό διάστημα ανοίγει για να περάσει το νερό μέσα στη σωλήνα και τέλος, αμέσως πριν κλείσει ξανά γίνεται paused η βαλβίδα για μερικά δευτερόλεπτα. Ενώ, η αντλία P101 αρχικά είναι κλειστή και στη συνέχεια με την πάροδο του χρόνου ανοίγει για να προωθήσει το νερό στο δεύτερο στάδιο. Στις πιο κάτω εικόνες βλέπουμε τις δύο καταστάσεις της αντλίας P101 για καλύτερη κατανόηση. Για τις καταστάσεις του sensor LIT101 και actuator MV101 ως προς την υλοποιήση μπορούμε να τις δούμε στο παράρτημα Α.

Τέλος, με τον ίδιο τρόπο υλοποιήθηκαν και τα υπόλοιπα στάδια του συστήματος τα οποία εξηγούμε στα επόμενα υποκεφάλαια.


helper.py
/usr/local/bin/scadasim_pymodbus_plc/plc

```
1041
1042
1043        if (address == 3) :
1044
1045            with open("/usr/local/bin/scadasim_pymodbus_plc/plc/P101.txt", "a") as f:
1046
1047                        value = (df['P101'][lines])
1048
1049                        f.write(str(value))
1050                        f.write("\n")
1051
1052
1053                        if (value == 2): # If the pump is open
1054                                ON_P101.append(value) # To collect these values which are showing to us that the pump is open
1055                                sleep(1)
1056                                if (((df['P101'][lines+1]) != 2)):
1057
1058                                        now  = datetime.now()
1059
1060                                        current_time = now.strftime("%H:%M:%S")
1061
1062                                        ON_P101.append(df['P101'][lines+1])
1063
1064                                        write_co_register(context[0], slave_id, address, ON_P101) # Adding the values from the table ON_P101 to the coil register
1065                                        values = read_co_register(context[0], slave_id, address, len(ON_P101)) # To read these values from the table ON_P101
1066
1067                                        if (values[len(values)-1]==1): # To check if the status of the pump has changed
1068                                                print("Current Time = " + str(current_time) + ' The status for P101 has changed: The actuator P101 is OFF --> ' + str(values[len(values)-1]))
1069
1070                                                message4.message = "Current Time = " + str(current_time) + ' The status for P101 has changed: The actuator P101 is OFF --> ' + str(values[len(values)-1])
1071
1072                                        del ON_P101[:]
1073
1074
1075
1076                        if (value == 1): # If the pump is closed
1077                                OFF_P101.append(value) # To collect these values which are showing to us that the pump is closed
1078                                sleep(1)
1079                                if ((df['P101'][lines+1]) != 1):
1080
1081                                        now  = datetime.now()
1082
1083                                        current_time = now.strftime("%H:%M:%S")
1084
1085                                        OFF_P101.append(df['P101'][lines+1])
1086
1087                                        write_co_register(context[0], slave_id, address, OFF_P101) # Adding the values from the table ON_P101 to the coil register
1088                                        values = read_co_register(context[0], slave_id, address, len(OFF_P101)) # To read these values from the table ON_P101
1089
1090                                        if (values[len(values)-1]==2): # To check if the status of the pump has changed
1091
1092                                                print("Current Time = " + str(current_time) + ' The status for P101 has changed: The actuator P101 is ON --> ' + str(values[len(values)-1]))
```



4.3.2 Δεύτερο Στάδιο - Chemical dosing

Αυτή η διαδικασία σχετίζεται με τη χημική δοσολογία, αποτελείται όπως βλέπουμε και από τον κάτω πίνακα, από τέσσερις αισθητήρες (sensors) οι οποιοί είναι οι AIT201, AIT202, AIT203 (Analyser Indicator Transmitter) και FIT201 και τέσσσερις ενεργοποιητές (actuators), MV201 (Motorized Valve) και τρείς αντλίες.

Συγκεκριμένα στις εικόνες 4.4 και 4.5 βλέπουμε τις εξής λειτουργίες:

Ο AIT201 είναι ο αναλυτής αγωγιμότητας και μετρά το επίπεδο NaCl.. Όταν η αντλία P101 είναι ανοιχτή, τότε η στάθμη του δοχείου νερού αυξάνεται μέχρι να φτάσει στη μέγιστη τιμή στάθμης. Μετά από κάποιο χρονικό διάστημα, όταν η στάθμη του νερού φτάσει στη μέγιστη τιμή, τότε η P-101 (αντλία) από την τιμή 2 (ανοιχτή) μεταβαίνει στο 1 με αποτέλεσμα η στάθμη της δεξαμενής νερού αρχίζει να μειώνεται.

Ο AIT202 είναι ο αναλυτής αγωγιμότητας και μετρά το επίπεδο HCl.. Όταν η αντλία P203 είναι ανοιχτή, τότε η στάθμη του δοχείου νερού αυξάνεται μέχρι να φτάσει στη μέγιστη τιμή στάθμης. Μετά από κάποιο χρονικό διάστημα, όταν η στάθμη του νερού φτάσει στη μέγιστη τιμή, τότε η P203 (αντλία) από την τιμή 2 (ανοιχτή) μεταβαίνει στο 1 με αποτέλεσμα η στάθμη της δεξαμενής νερού αρχίζει να μειώνεται.

Ο AIT203 είναι ο αναλυτής αγωγιμότητας και μετρά το επίπεδο NaOCl.. Όταν η αντλία P205 είναι ανοιχτή, τότε η στάθμη του δοχείου νερού αυξάνεται μέχρι να φτάσει στη μέγιστη τιμή στάθμης. Μετά από κάποιο χρονικό διάστημα, όταν η στάθμη του νερού φτάσει στη μέγιστη τιμή, τότε η P205

(αντλία) από την τιμή 2 (ανοιχτή) μεταβαίνει στο 1 με αποτέλεσμα η στάθμη της δεξαμενής νερού αρχίζει να μειώνεται.

Ο FIT201 μετρά την εισροή για τα dosing pumps. Στην αρχή, το FIT201 είναι κλειστό λόγω του ότι "επικοινωνεί" με την αντλία P101 η οποία είναι κι αυτή κλειστή στην αρχή. Όταν η αντλία P101 αλλάξει την τιμή της σε 2 τότε ο Πομπός Ροής αρχίζει να μετρά την εισροή νερού.

Ο MV201 ελέγχει τη ροή του νερού στη δεξαμενή ακατέργαστου νερού. Στην αρχή, η βαλβίδα έχει την τιμή 1 για να δείξει ότι εκείνη τη στιγμή είναι κλειστή, δεν λειτουργεί. Στη συνέχεια, όταν η ροή του νερού ξεκινά προς τη δεξαμενή νερού, η τιμή του αλλάζει σε 2 για να δείξει ότι η βαλβίδα είναι ανοιχτή.

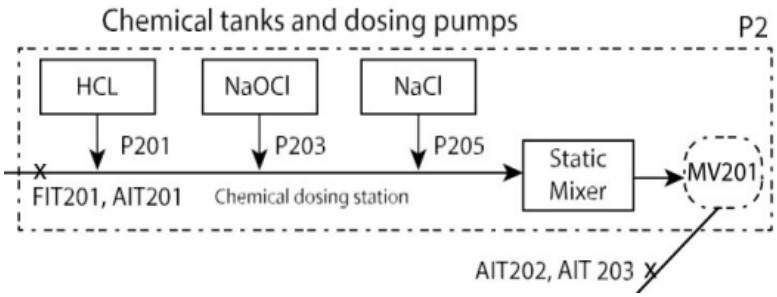
Ο κύριος ρόλος του **P201** είναι να αντλεί νερό για τη δοσολογία του NaCL

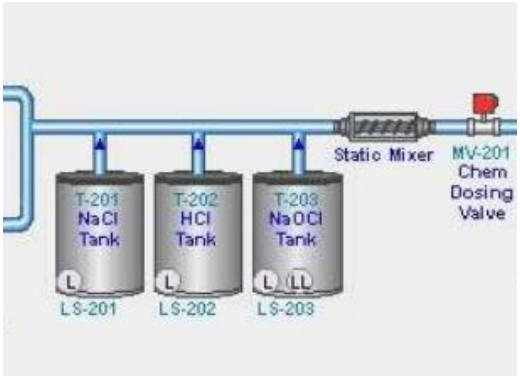
Ο κύριος ρόλος του **P203** είναι να αντλεί νερό για τη δοσολογία του HCL

Ο κύριος ρόλος του **P205** είναι να αντλεί νερό για τη δοσολογία του NaOCL

Πίνακας περιγραφής των field devices του δεύτερου σταδίου

AIT-201	Sensor	Conductivity analyser; Measures NaCl level.
AIT-202	Sensor	
AIT-203	Sensor	ORP analyser; Measures NaOCl level.
FIT-201	Sensor	Flow Transmitter; Control dosing pumps.
MV-201	Actuator	Motorized valve; Controls water flow to the UF feed water tank.
P-201	Actuator	Dosing pump; NaCl dosing pump.
P-202 (backup)	Actuator	Dosing pump; NaCl dosing pump.
P-203	Actuator	Dosing pump; HCl dosing pump.
P-204 (backup)	Actuator	Dosing pump; HCl dosing pump.
P-205	Actuator	Dosing pump; NaOCl dosing pump.
P-206 (backup)	Actuator	Dosing pump; NaOCl dosing pump.





Εικόνα 4.4 **Εικόνα 4.5**

4.3.3 Τρίτο Στάδιο – Ultra-Filtration

Αυτή η διαδικασία έχει ως σκοπό το φιλτράρισμα μέσω συστήματος υπερδιήθησης αποτελείται όπως βλέπουμε και από τον κάτω πίνακα, από τρεις αισθητήρες (sensors) οι οποιοί είναι οι DPIT301, FIT301, LIT301 και πέντε ενεργοποιητές (actuators), MV301, MV302, MV303, MV304 (Motorized Valve) και μία αντλία P302.

Συγκεκριμένα στις εικόνες 4.6 και 4.7 βλέπουμε τις εξής λειτουργίες:

Ο DPIT301 (Differential Pressure Indicating Transmitter) είναι ένας πομπός ένδειξης διαφορικής πίεσης και ελέγχει τη διαδικασία πλύσης (backwash process).

Ο FIT301 Μετρά τη ροή του νερού στο στάδιο του UF (Ultra-Filtration).

Ο LIT301 μετρά τη στάθμη δεξαμενής νερού τροφοδοσίας UF. Όταν το MV301 είναι ανοιχτό, τότε η στάθμη του δοχείου νερού αυξάνεται μέχρι να φτάσει στη μέγιστη τιμή στάθμης. Μετά από κάποιο χρονικό διάστημα, όταν η στάθμη του νερού φτάσει στη μέγιστη τιμή, τότε η P-302 (αντλία) από την τιμή 1 (κλειστή) μεταβαίνει στο 2 για να αντλήσει το νερό και να περάσει μέσα από αυτό και η στάθμη της δεξαμενής νερού αρχίζει να μειώνεται..

Ο MV301 ελέγχει τη διαδικασία UF-Backwash. Στην αρχή, η βαλβίδα έχει την τιμή 1 για να δείξει ότι εκείνη τη στιγμή είναι κλειστή, δεν λειτουργεί. Στη συνέχεια, όταν η ροή του νερού ξεκινά προς τη δεξαμενή νερού, η τιμή του αλλάζει σε 2 για να δείξει ότι η βαλβίδα είναι ανοιχτή.

Ο MV302 ελέγχει το νερό από τη διαδικασία UF στη μονάδα αποχλωρίωσης (Dechloration unit). Στην αρχή, η βαλβίδα έχει την τιμή 1 για να δείξει ότι εκείνη τη στιγμή είναι κλειστή, δεν λειτουργεί. Στη συνέχεια, όταν η ροή του νερού ξεκινά προς τη δεξαμενή νερού, η τιμή του αλλάζει σε 2 για να δείξει ότι η βαλβίδα είναι ανοιχτή.

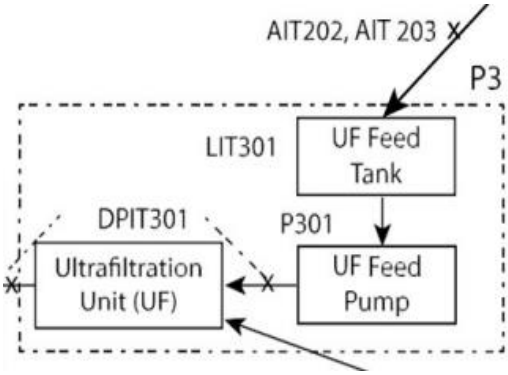
Ο MV303 ελέγχει την αποστράγγιση UF-Backwash. Στην αρχή, η βαλβίδα έχει την τιμή 1 για να δείξει ότι εκείνη τη στιγμή είναι κλειστή, δεν λειτουργεί. Στη συνέχεια, όταν η ροή του νερού ξεκινά προς τη δεξαμενή νερού, η τιμή του αλλάζει σε 2 για να δείξει ότι η βαλβίδα είναι ανοιχτή.

Ο MV304 Ελέγχει την αποστράγγιση UF. Στην αρχή, η βαλβίδα έχει την τιμή 1 για να δείξει ότι εκείνη τη στιγμή είναι κλειστή, δεν λειτουργεί. Στη συνέχεια, όταν η ροή του νερού ξεκινά προς τη δεξαμενή νερού, η τιμή του αλλάζει σε 2 για να δείξει ότι η βαλβίδα είναι ανοιχτή.

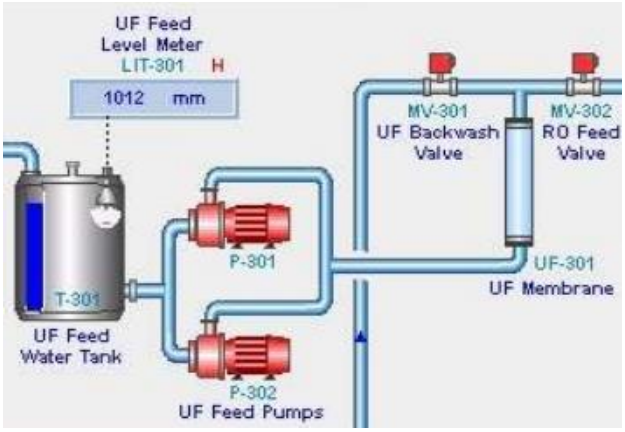
Ο κύριος ρόλος της αντλίας τροφοδοσίας UF **P301** είναι να αντλεί νερό από τη δεξαμενή νερού τροφοδοσίας UF στη δεξαμενή νερού τροφοδοσίας RO μέσω φιλτραρίσματος UF.

Πίνακας περιγραφής των field devices του τρίτου σταδίου

DPIT-301	Sensor	Differential pressure indicating transmitter; Controls the backwash process.
FIT-301	Sensor	Flow meter; Measures the flow of water in the UF stage.
LIT-301	Sensor	Level Transmitter; UF feed water tank level.
MV-301	Actuator	Motorized Valve; Controls UF-Backwash process.
MV-302	Actuator	Motorized Valve; Controls water from UF process to De Chlorination unit.
MV-303	Actuator	Motorized Valve; Controls UF-Backwash drain.
MV-304	Actuator	Motorized Valve; Controls UF drain.
P-301 (backup)	Actuator	UF feed Pump; Pumps water from UF feed water tank to RO feed water tank via UF filtration.
P-302	Actuator	UF feed Pump; Pumps water from UF feed water tank to RO feed water tank via UF filtration.



Εικόνα 4.6



Εικόνα 4.7

4.3.4 Τέταρτο Στάδιο – Dechlorination

Αυτή η διαδικασία βασίζεται στην αποχλωρίωση με χρήση λαμπτήρων UV, αποτελείται όπως βλέπουμε και από τον κάτω πίνακα, από τέσσερις αισθητήρες (sensors) οι οποιοί είναι οι AIT401, AIT402, FIT401, LIT401 και τρείς ενεργοποιητές (actuators), P402, P403 και μία UV401.

Συγκεκριμένα στις εικόνες 4.8 και 4.9 βλέπουμε τις εξής λειτουργίες:

Ο AIT401 είναι ένας μετρητής σκληρότητας νερού RO.

Ο AIT402 είναι ένας ORP (Oxidation reduction potential) μετρητής κι ελέγχει τη δοσολογία NaHSO3 (P203), NaOCl (P205).

Ο FIT401 είναι ένας πομπός ροής κι ελέγχει τον αποχλωριωτή UV.

Ο LIT401 είναι μια στάθμη δεξαμενής νερού τροφοδοσίας RO.

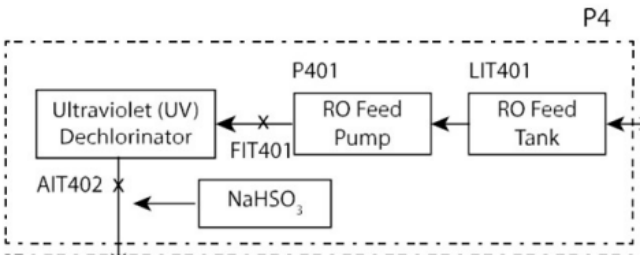
Ο κύριος ρόλος του **P402** είναι να αντλεί νερό από τη δεξαμενή τροφοδοσίας RO στον αποχλωριωτή UV.

Ο κύριος ρόλος του **P403** είναι μια αντλία διθειικού νατρίου.

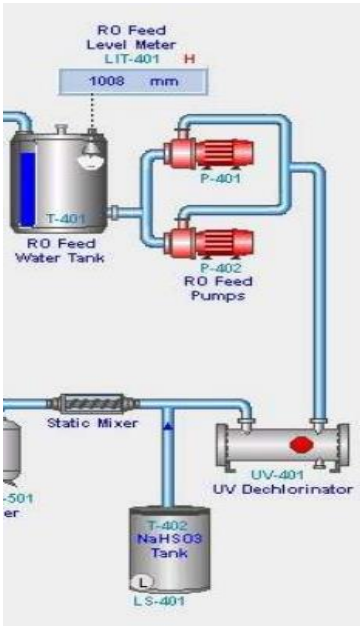
Ο κύριος ρόλος του **UV401** είναι ένας αποχλωριωτής κι αφαιρεί το χλώριο από το νερό.

Πίνακας περιγραφής των field devices του τέταρτου σταδίου

AIT-401	Sensor	RO hardness meter of water.
AIT-402	Sensor	ORP meter; Controls the NaHSO3dosing(P203), NaOCl dosing (P205).
FIT-401	Sensor	Flow Transmitter; Controls the UV dechlorinator.
LIT-401	Sensor	Level Transmitter; RO feed water tank level.
P-401 (backup)	Actuator	Pump; Pumps water from RO feed tank to UV dechlorinator.
P-402	Actuator	Pump; Pumps water from RO feed tank to UV dechlorinator.
P-403	Actuator	Sodium bi-sulphate pump.
P-404 (backup)	Actuator	Sodium bi-sulphate pump.
UV-401	Actuator	Dechlorinator; Removes chlorine from water.



Εικόνα 4.8



Εικόνα 4.9

4.3.5 Πέμπτο Στάδιο – **Reverse Osmosis**

Αυτή η διαδικασία βασίζεται στη τροφοδοσία σε σύστημα αντίστροφης όσμωσης, αποτελείται όπως βλέπουμε και από τον κάτω πίνακα, από έντεκα αισθητήρες (sensors) οι οποιοί είναι οι AIT501, AIT502, AIT503, AIT504, FIT501, FIT502, FIT503, FIT504, PIT501, PIT502 και PIT503 κι ένα ενεργοποιητή (actuators) την αντλία P501.

Συγκεκριμένα στις εικόνες 4.10 και 4.11 βλέπουμε τις εξής λειτουργίες:

Ο AIT501 είναι ένας αναλυτής pH RO και μετρά το επίπεδο HCl.

Ο AIT502 είναι ένας αναλυτής ORP τροφοδοσίας RO και μετρά το επίπεδο NaOCl.

Ο AIT503 είναι ένα αναλυτής αγωγιμότητας τροφοδοσίας RO και μετρά το επίπεδο NaCl.

Ο AIT504 είναι ένας αναλυτής αγωγιμότητας διαπερατών RO; Μετρά το επίπεδο NaCl. (RO permeate conductivity analyser; Measures NaCl level).

Ο FIT501 μετρά την ροή μεμβράνης RO (Reverse Osmosis). Στην αρχή, το FIT501 είναι κλειστό αλλά με την τιμή 0. Μετά από κάποιο χρονικό διάστημα, αρχίζει να μετρά την ροή αυτή.

Ο FIT502 μετρά την ροή διαπερατών. (Flow meter; RO Permeate flow meter)

Ο FIT503 είναι μετρητής ροής. RO Απόρριψη ροόμετρου.

Ο FIT504 μετρά την ροή επανακυκλοφορίας RO. Μετά από κάποιο χρονικό διάστημα, αρχίζει να μετρά την ροή αυτή.

Ο κύριος ρόλος του **P501** είναι να αντλεί αποχλωριωμένο νερό σε RO (Reverse Osmosis).

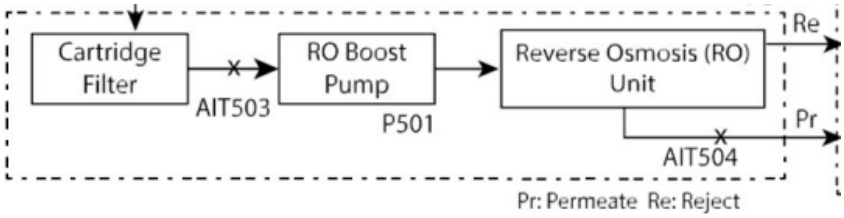
Ο κύριος ρόλος του **PIT501** είναι να μετρά την πίεση τροφοδοσίας RO.

Ο κύριος ρόλος του **PIT502** είναι να μετρά την πίεση διείσδυσης RO.

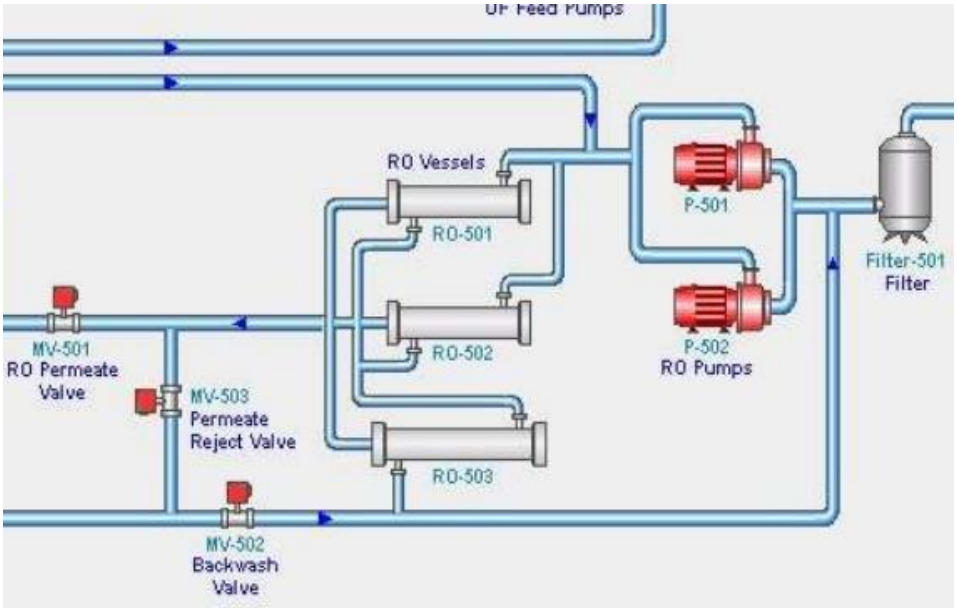
Ο κύριος ρόλος του **PIT503** είναι να απορρίψει την πίεση του RO.

Πίνακας περιγραφής των field devices του πέμπτου σταδίου

AIT-501	Sensor	RO pH analyser; Measures HCl level.
AIT-502	Sensor	RO feed ORP analyser; Measures NaOCl level.
AIT-503	Sensor	RO feed conductivity analyser; Measures NaCl level.
AIT-504	Sensor	RO permeate conductivity analyser; Measures NaCl level.
FIT-501	Sensor	Flow meter; RO membrane inlet flow meter.
FIT-502	Sensor	Flow meter; RO Permeate flow meter.
FIT-503	Sensor	Flow meter; RO Reject flow meter.
FIT-504	Sensor	Flow meter; RO re-circulation flow meter.
P-501	Actuator	Pump; Pumps dechlorinated water to RO.
P-502 (backup)	Actuator	Pump; Pumps dechlorinated water to RO.
PIT-501	Sensor	Pressure meter; RO feed pressure.
PIT-502	Sensor	Pressure meter; RO permeate pressure.
PIT-503	Sensor	Pressure meter; RO reject pressure.



Εικόνα 4.10



Εικόνα 4.11

4.3.6 Έκτο Στάδιο – Reverse Osmosis permeate transfer, UF backwash

Αυτή η διαδικασία έχει ως σκοπό την αντίστροφη πλύση και καθαρίζει τις μεμβράνες σε UF χρησιμοποιώντας το διήθημα RO. Αποτελείται όπως βλέπουμε και από τον κάτω πίνακα, από ένα αισθητήρα (sensors) τον FIT601 και τρείς ενεργοποιητές (actuators) P601, P602 και P603.

Συγκεκριμένα στις εικόνες 4.12 και 4.13 βλέπουμε τις εξής λειτουργίες:

Ο FIT601 μετρά την ροή του UF Backwash.

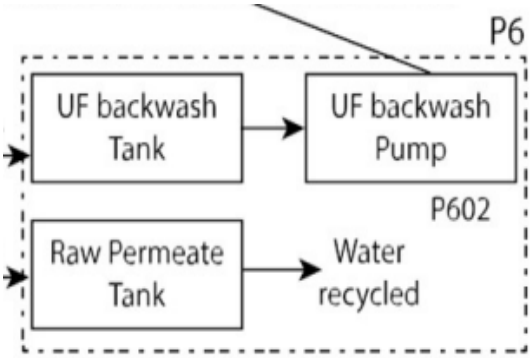
Ο κύριος ρόλος του **P601** είναι να αντλεί νερό από τη δεξαμενή διείσδυσης RO (Reverse Osmosis) στη δεξαμενή ακατέργαστου νερού (δεν χρησιμοποιείται για συλλογή δεδομένων).

Ο κύριος ρόλος του **P602** είναι να αντλεί νερό από τη δεξαμενή πίσω πλύσης UF (back wash tank) στο φίλτρο UF για να καθαρίσει τη μεμβράνη.

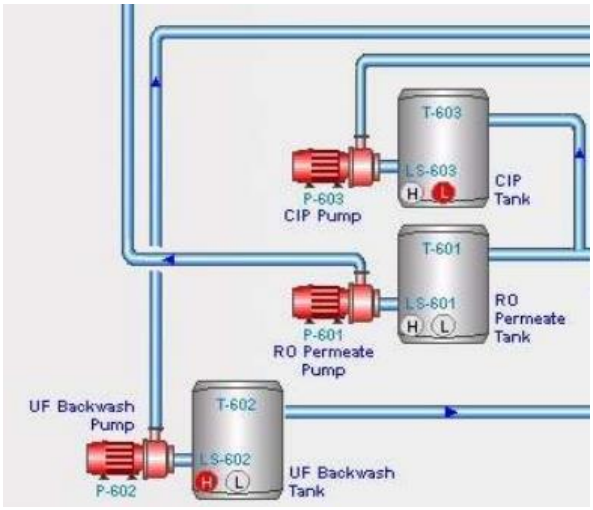
Το **P603** δεν έχει υλοποιηθεί ακόμη.

Πίνακας περιγραφής των field devices του έκτου σταδίου

FIT-601	Sensor	Flow meter; UF Backwash flow meter.
P-601	Actuator	Pump; Pumps water from RO permeate tank to raw water tank (not used for data collection).
P-602	Actuator	Pump; Pumps water from UF back wash tank to UF filter to clean the membrane.
P-603	Actuator	Not implemented in SWaT yet.



Εικόνα 4.12



Εικόνα 4.13

4.4 Validation

Σ’ αυτό το υποκεφάλαιο έχουμε σκοπό να δείξουμε τη πραγματική λειτουργία του προσομοιωτή δημιουργώντας μερικά αποτελέσματα συγκρίνοντας τα μ’ εκείνα του csv αρχείου. Γι αυτό, για λόγους διευκόλυνσης επιλέξαμε το τέταρτο στάδιο του συστήματος χρησιμοποιώντας μερικά field devices τα οποία είναι ο LIT401, ο P402 και τη βαλβίδα MV302 από το τρίτο στάδιο λόγω της επικοινωνίας που έχουν μεταξύ τους αυτές οι δύο διαδικασίες.

Για την υλοποίηση αυτή, είχαμε τρείς συνθήκες στις οποίες έπρεπε να υπολογίσουμε για την κάθεμια το δικό της water flow. Δηλαδή, στην πρώτη συνθήκη έχουμε την κατάσταση όπου η βαλβίδα MV302 είναι ανοιχτή, η αντλία P402 αρχικά είναι κλειστή και η δεξαμενή του LIT401 είναι σχεδόν άδεια και ξαφνικά αρχίζει και αυξάνεται η στάθμη του νερού. Οπότε, για καλύτερη μας διευκόλυνση, διαβάσαμε από το csv αρχείο μόνο τις αρχικές τιμές της βαλβίδας και στη συνέχεια όταν την χρονική στιγμή η βαλβίδα ανοίξει την πρώτη φόρα, τότε ο προσομοιωτής θα πρέπει να ξεκινήσει την προσομοίωση για να βγάλει τα σωστά αποτελέσματα. Ακολούθως, υπολογίζουμε την ροή νερού όταν αυξάνεται προσθέτοντας την αρχική τιμή που θέσαμε στον LIT401 με το average flow το οποίο υπολογίσαμε. Επιπρόσθετα, η δεύτερη συνθήκη είναι να κλείσει η βαλβίδα και τότε ν’ ανοίξει η αντλία P402 για να περάσει το νερό από τη δεξαμενή σ’ αυτή. Ακολουθούμε την ίδια λογική μ’ αυτή της πρώτης συνθήκης μόνο που αντί να προσθέτουμε το average flow το αφαιρούμε επειδή μειώνεται η ροή του νερού σ’ αυτή την χρονική κατάσταση. Τέλος, η τρίτη και τελευταία συνθήκη είναι να είναι ανοιχτή και η βαλβίδα MV302 και η αντλία P402. Χρησιμοποιήσαμε και σ’ αυτό το κομμάτι της υλοποιήσης του δύο καταχωρητές, holding register και coil register για να αποθηκεύουν τη ροή του νερού σε κάθε χρονική στιγμή.

Κεφάλαιο 5

Αποτελέσματα

5.1 Πειραματικά Αποτελέσματα

Σε αυτό το κεφάλαιο παρουσιάζονται πρώτα τα αποτελέσματα που έχουμε δημιουργήσει δείχνοντας την ομαλή λειτουργία για κάθε στάδιο του συστήματος σε πραγματικό χρόνο μέσω του csv αρχείου. Στη συνέχεια, παρουσιάζονται μερικές γραφικές παραστάσεις για να συγκρίνουμε τα πραγματικά αποτελέσματα μ' αυτά του προσομοιωτή για να δείξουμε ότι ο προσομοιωτής βγάζει καλά αποτελέσματα.

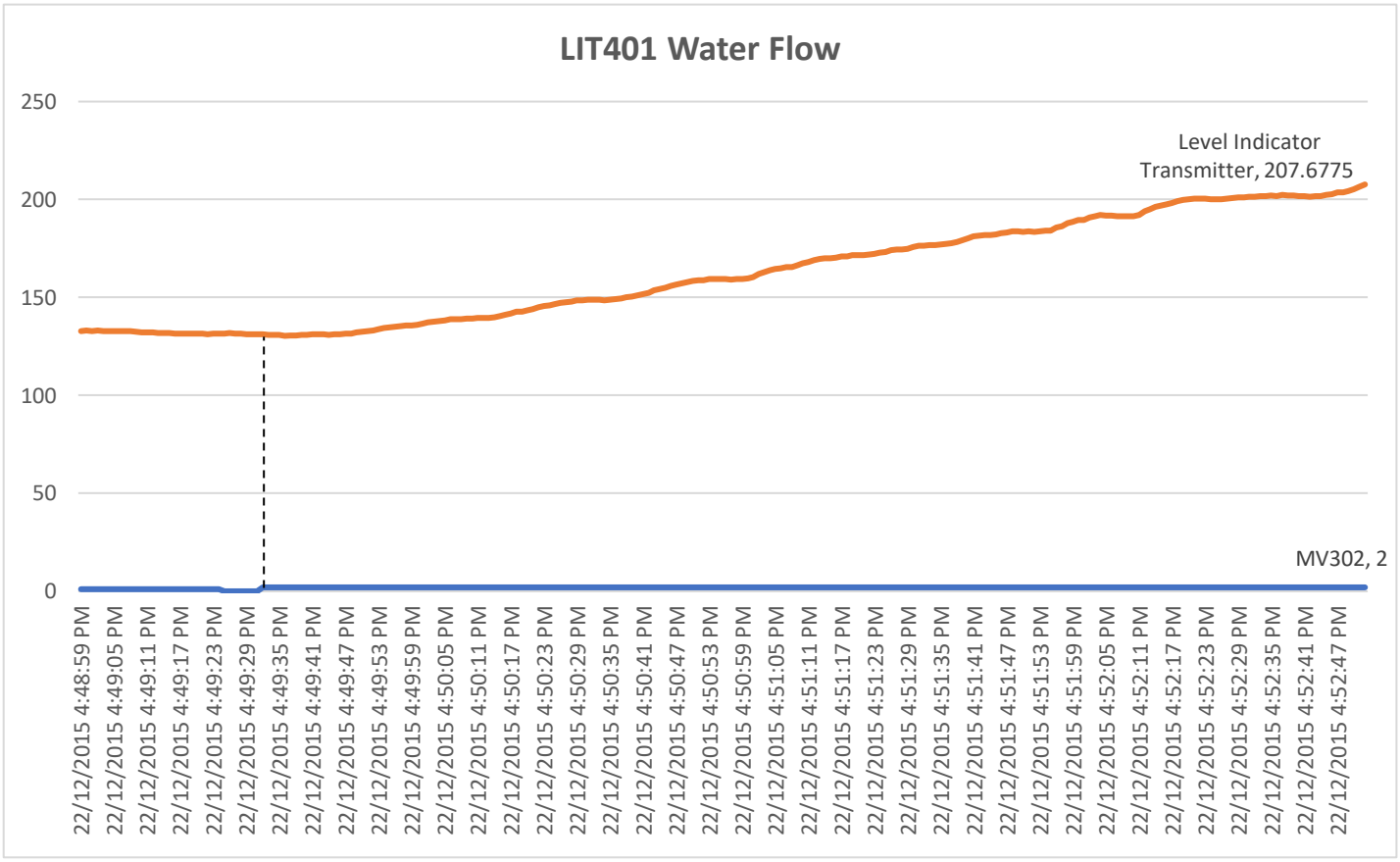
```
16 Current Time = 10:45:54 The actuator P101 current status is OFF --> 1
17
18 Current Time = 10:46:54 The pump's water flow is  very LOW (FIT101) --> 0.0
19
20 Current Time = 10:46:55 The water tank level status is very LOW --> 124.3527
21
22 Current Time = 10:46:55 The actuator P101 current status is OFF --> 1
23
24 Current Time = 10:46:55 The actuator MV101 current status is OFF --> 1
25
26 Current Time = 10:47:55 The pump's water flow is  very LOW (FIT101) --> 0.0
27
28 Current Time = 10:47:55 The water tank level status is very LOW --> 123.1359
29
30 Current Time = 10:47:55 The actuator P101 current status is OFF --> 1
31
32 Current Time = 10:47:55 The actuator MV101 current status is OFF --> 1
33
34 Current Time = 10:48:55 The pump's water flow is  very LOW (FIT101) --> 0.0
35
36 Current Time = 10:48:55 The water tank level status is very LOW --> 123.3714
37
38 Current Time = 10:48:55 The actuator P101 current status is OFF --> 1
39
40 Current Time = 10:48:55 The actuator MV101 current status is OFF --> 1
41
42 Current Time = 10:49:55 The pump's water flow is  very LOW (FIT101) --> 0.0
43
44 Current Time = 10:49:55 The water tank level status is very LOW --> 122.5864
45
46 Current Time = 10:49:55 The actuator P101 current status is OFF --> 1
47
48 Current Time = 10:49:55 The actuator MV101 current status is OFF --> 1
49
50 Current Time = 10:50:55 The pump's water flow is  very LOW (FIT101) --> 0.0
51
52 Current Time = 10:50:55 The water tank level status is very LOW --> 122.1938
53
54 Current Time = 10:50:55 The actuator P101 current status is OFF --> 1
55
56 Current Time = 10:50:55 The actuator MV101 current status is OFF --> 1
57
58 Current Time = 10:51:55 The pump's water flow is  very LOW (FIT101) --> 0.0
59
60 Current Time = 10:51:55 The water tank level status is very LOW --> 121.5658
61
```

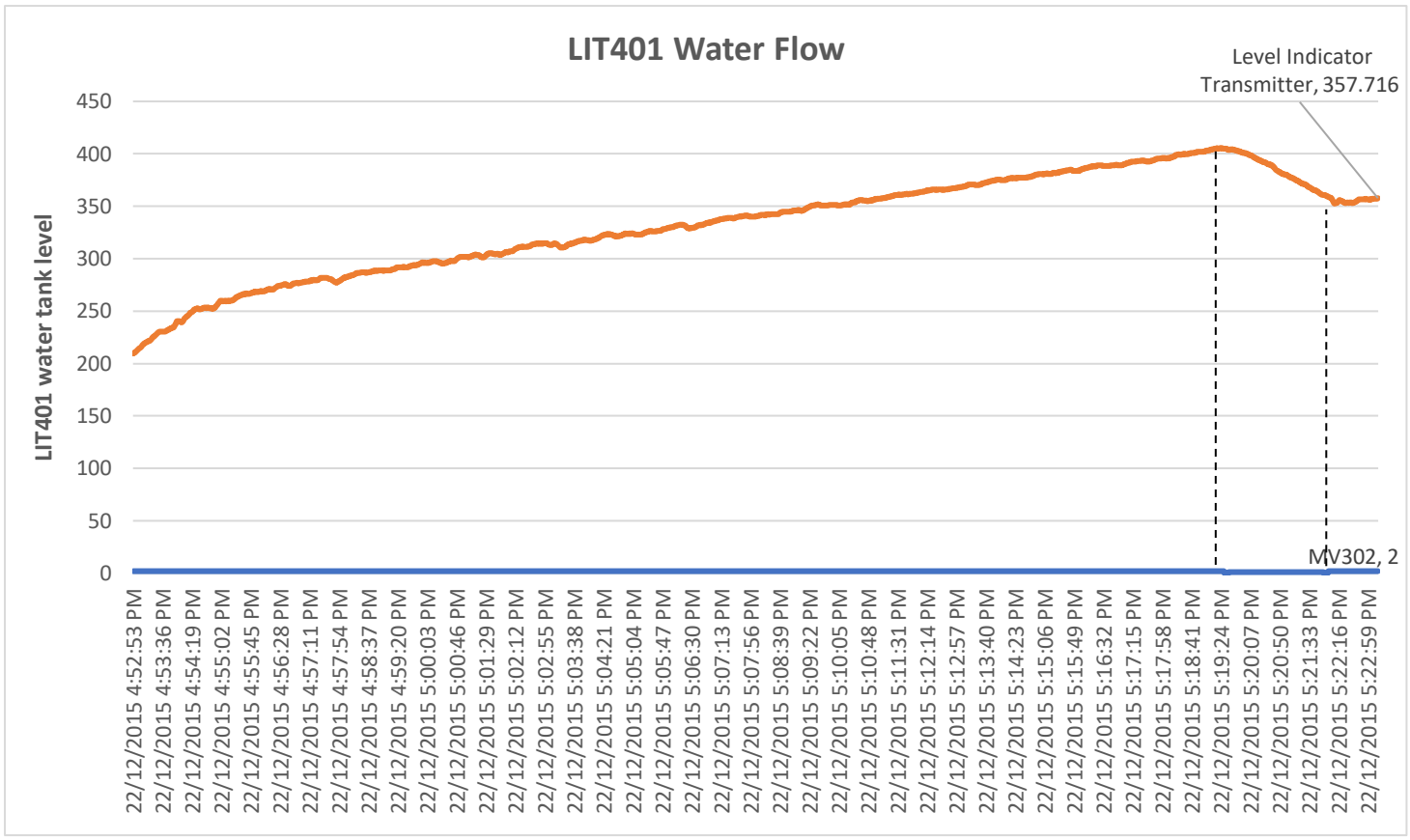
```
35
36 Current Time = 10:48:55 The water tank level status is very LOW --> 123.3714
37
38 Current Time = 10:48:55 The actuator P101 current status is OFF --> 1
39
40 Current Time = 10:48:55 The actuator MV101 current status is OFF --> 1
41
42 Current Time = 10:49:55 The pump's water flow is  very LOW (FIT101) --> 0.0
43
44 Current Time = 10:49:55 The water tank level status is very LOW --> 122.5864
45
46 Current Time = 10:49:55 The actuator P101 current status is OFF --> 1
47
48 Current Time = 10:49:55 The actuator MV101 current status is OFF --> 1
49
50 Current Time = 10:50:55 The pump's water flow is  very LOW (FIT101) --> 0.0
51
52 Current Time = 10:50:55 The water tank level status is very LOW --> 122.1938
53
54 Current Time = 10:50:55 The actuator P101 current status is OFF --> 1
55
56 Current Time = 10:50:55 The actuator MV101 current status is OFF --> 1
57
58 Current Time = 10:51:55 The pump's water flow is  very LOW (FIT101) --> 0.0
59
60 Current Time = 10:51:55 The water tank level status is very LOW --> 121.5658
61
62 Current Time = 10:51:55 The actuator P101 current status is OFF --> 1
63
64 Current Time = 10:51:55 The actuator MV101 current status is OFF --> 1
65
66 Current Time = 10:52:56 The actuator MV101 current status is OFF --> 1
67
68 Current Time = 10:52:56 The actuator P101 current status is OFF --> 1
69
70 Current Time = 10:52:56 The pump's water flow is  very LOW (FIT101) --> 0.0
71
72 Current Time = 10:52:56 The water tank level status is very LOW --> 121.8013
73 Current Time = 10:53:55 The status for MV101 has changed: The actuator MV101 is paused --> 0
74 Current Time = 10:53:56 The actuator MV101 current status is PAUSED --> 0
75
76 Current Time = 10:53:56 The actuator P101 current status is OFF --> 1
77
78 Current Time = 10:53:56 The water tank level status is very LOW --> 121.605
79
80 Current Time = 10:53:56 The pump's water flow is  very LOW (FIT101) --> 0.0
81 Current Time = 10:54:04 The status for MV101 has changed: The actuator MV101 is ON --> 2
```

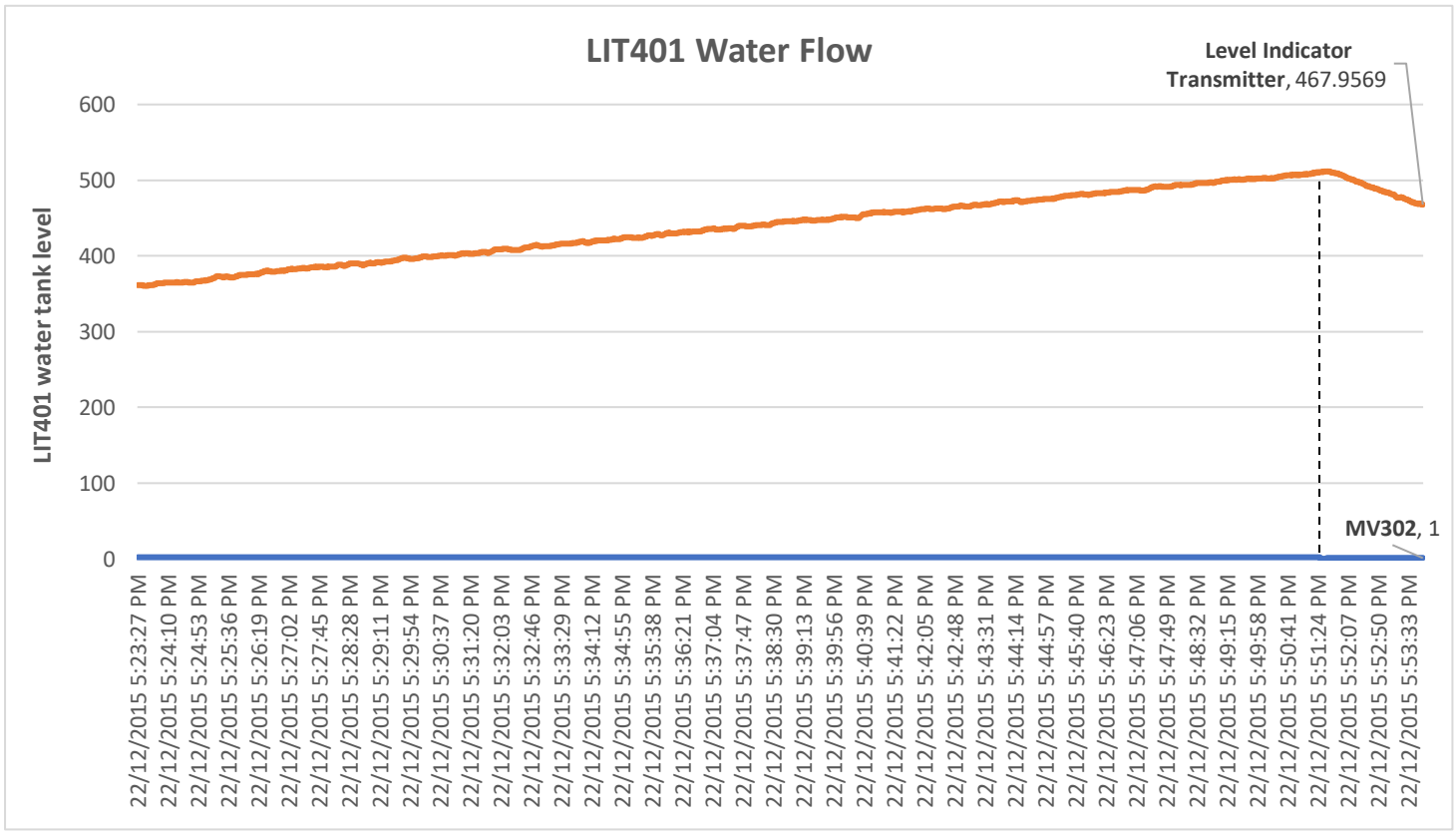
```
Current Time = 10:51:55 The actuator P101 current status is OFF --> 1

Current Time = 10:51:55 The actuator MV101 current status is OFF --> 1

Current Time = 10:52:56 The actuator MV101 current status is OFF --> 1

Current Time = 10:52:56 The actuator P101 current status is OFF --> 1

Current Time = 10:52:56 The pump's water flow is  very LOW (FIT101) --> 0.0

Current Time = 10:52:56 The water tank level status is very LOW --> 121.8013
Current Time = 10:53:55 The status for MV101 has changed: The actuator MV101 is paused --> 0
Current Time = 10:53:56 The actuator MV101 current status is PAUSED --> 0

Current Time = 10:53:56 The actuator P101 current status is OFF --> 1

Current Time = 10:53:56 The water tank level status is very LOW --> 121.605

Current Time = 10:53:56 The pump's water flow is  very LOW (FIT101) --> 0.0
Current Time = 10:54:04 The status for MV101 has changed: The actuator MV101 is ON --> 2
Current Time = 10:54:56 The actuator MV101 current status is ON --> 2

Current Time = 10:54:56 The actuator P101 current status is OFF --> 1
Current Time = 10:54:23 The status for LIT101 has changed: The water tank level is normal  --> 126.6294
Current Time = 10:54:56 The water tank level status is NORMAL --> 141.4669
Current Time = 10:54:00 The status for FIT101 has changed: The pump's water flow is normal  --> 0.10793330000000001
Current Time = 10:54:56 The pump's water flow is NORMAL (FIT101) --> 2.478621
Current Time = 10:54:04 The status for MV101 has changed: The actuator MV101 is ON --> 2
Current Time = 10:55:56 The actuator MV101 current status is ON --> 2

Current Time = 10:55:56 The actuator P101 current status is OFF --> 1
Current Time = 10:54:23 The status for LIT101 has changed: The water tank level is normal  --> 126.6294
Current Time = 10:55:56 The water tank level status is NORMAL --> 172.2411
Current Time = 10:54:00 The status for FIT101 has changed: The pump's water flow is normal  --> 0.10793330000000001
Current Time = 10:55:56 The pump's water flow is NORMAL (FIT101) --> 2.618903
Current Time = 10:54:00 The status for FIT101 has changed: The pump's water flow is normal  --> 0.10793330000000001
Current Time = 10:56:57 The pump's water flow is NORMAL (FIT101) --> 2.521538
Current Time = 10:54:04 The status for MV101 has changed: The actuator MV101 is ON --> 2
Current Time = 10:56:57 The actuator MV101 current status is ON --> 2
Current Time = 10:54:23 The status for LIT101 has changed: The water tank level is normal  --> 126.6294
Current Time = 10:56:57 The water tank level status is NORMAL --> 202.034

Current Time = 10:56:57 The actuator P101 current status is OFF --> 1
Current Time = 10:54:04 The status for MV101 has changed: The actuator MV101 is ON --> 2
Current Time = 10:57:57 The actuator MV101 current status is ON --> 2
Current Time = 10:54:23 The status for LIT101 has changed: The water tank level is normal  --> 126.6294
```

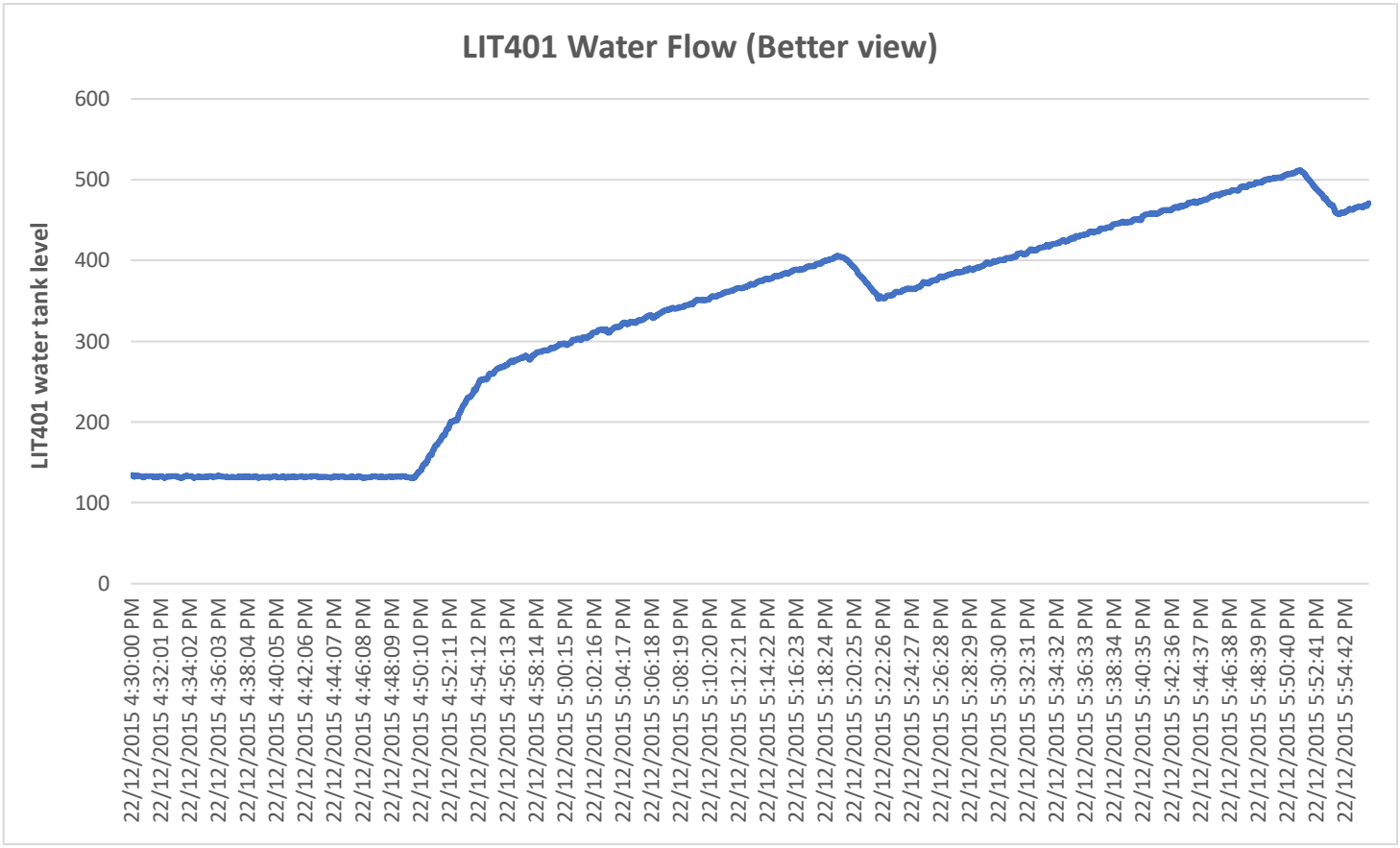
```
Current Time = 10:54:00 The status for FIT101 has changed: The pump's water flow is normal  --> 0.10793330000000001
Current Time = 10:57:57 The pump's water flow is NORMAL (FIT101) --> 2.432181
Current Time = 10:54:04 The status for MV101 has changed: The actuator MV101 is ON --> 2
Current Time = 10:58:57 The actuator MV101 current status is ON --> 2
Current Time = 10:58:29 The status for P101 has changed: The actuator P101 is ON --> 2
Current Time = 10:58:57 The actuator P101 current status is ON --> 2
Current Time = 10:54:23 The status for LIT101 has changed: The water tank level is normal  --> 126.6294
Current Time = 10:58:57 The water tank level status is NORMAL --> 253.8476
Current Time = 10:54:00 The status for FIT101 has changed: The pump's water flow is normal  --> 0.10793330000000001
Current Time = 10:58:57 The pump's water flow is NORMAL (FIT101) --> 2.661179
Current Time = 10:58:29 The status for P101 has changed: The actuator P101 is ON --> 2
Current Time = 10:59:57 The actuator P101 current status is ON --> 2
Current Time = 10:54:04 The status for MV101 has changed: The actuator MV101 is ON --> 2
Current Time = 10:59:57 The actuator MV101 current status is ON --> 2
Current Time = 10:54:23 The status for LIT101 has changed: The water tank level is normal  --> 126.6294
Current Time = 10:59:57 The water tank level status is NORMAL --> 252.0027
Current Time = 10:54:00 The status for FIT101 has changed: The pump's water flow is normal  --> 0.10793330000000001
Current Time = 10:59:57 The pump's water flow is NORMAL (FIT101) --> 2.559972
Current Time = 10:58:29 The status for P101 has changed: The actuator P101 is ON --> 2
Current Time = 11:00:57 The actuator P101 current status is ON --> 2
Current Time = 10:54:04 The status for MV101 has changed: The actuator MV101 is ON --> 2
Current Time = 11:00:57 The actuator MV101 current status is ON --> 2
Current Time = 10:54:23 The status for LIT101 has changed: The water tank level is normal  --> 126.6294
Current Time = 11:00:57 The water tank level status is NORMAL --> 256.0065
Current Time = 10:54:00 The status for FIT101 has changed: The pump's water flow is normal  --> 0.10793330000000001
Current Time = 11:00:57 The pump's water flow is NORMAL (FIT101) --> 2.448836
Current Time = 10:54:04 The status for MV101 has changed: The actuator MV101 is ON --> 2
Current Time = 11:01:58 The actuator MV101 current status is ON --> 2
Current Time = 10:58:29 The status for P101 has changed: The actuator P101 is ON --> 2
Current Time = 11:01:58 The actuator P101 current status is ON --> 2
Current Time = 10:54:23 The status for LIT101 has changed: The water tank level is normal  --> 126.6294
Current Time = 11:01:58 The water tank level status is NORMAL --> 253.0233
Current Time = 10:54:00 The status for FIT101 has changed: The pump's water flow is normal  --> 0.10793330000000001
Current Time = 11:01:58 The pump's water flow is NORMAL (FIT101) --> 2.646446
Current Time = 10:54:04 The status for MV101 has changed: The actuator MV101 is ON --> 2
Current Time = 11:02:58 The actuator MV101 current status is ON --> 2
Current Time = 10:54:23 The status for LIT101 has changed: The water tank level is normal  --> 126.6294
Current Time = 11:02:58 The water tank level status is NORMAL --> 258.7542
Current Time = 10:58:29 The status for P101 has changed: The actuator P101 is ON --> 2
Current Time = 11:02:58 The actuator P101 current status is ON --> 2
Current Time = 10:54:00 The status for FIT101 has changed: The pump's water flow is normal  --> 0.10793330000000001
Current Time = 11:02:58 The pump's water flow is NORMAL (FIT101) --> 2.582071
Current Time = 10:54:04 The status for MV101 has changed: The actuator MV101 is ON --> 2
Current Time = 11:03:58 The actuator MV101 current status is ON --> 2
Current Time = 10:54:23 The status for LIT101 has changed: The water tank level is normal  --> 126.6294
Current Time = 11:03:58 The water tank level status is NORMAL --> 258.6757
```

Τα πιο πάνω αποτελέσματα μας δείχνουν την κανονική ροή του πρώτου σταδίου του συστήματος μέσω του csv αρχείου. Δίπλα από κάθε κατάσταση δηλώνουμε και την ακριβή ώρα για να ξέρουμε πότε συνέβηκε μια αλλαγή κατάστασης και όπως θα παρατηρήσουμε τυπώνουμε πάντα την τελευταία αλλαγή κατάστασης του κάθε field device για να γνωρίζουμε πότε έγινε, πριν εμφανιστεί η νέα αλλαγή κατάστασης.

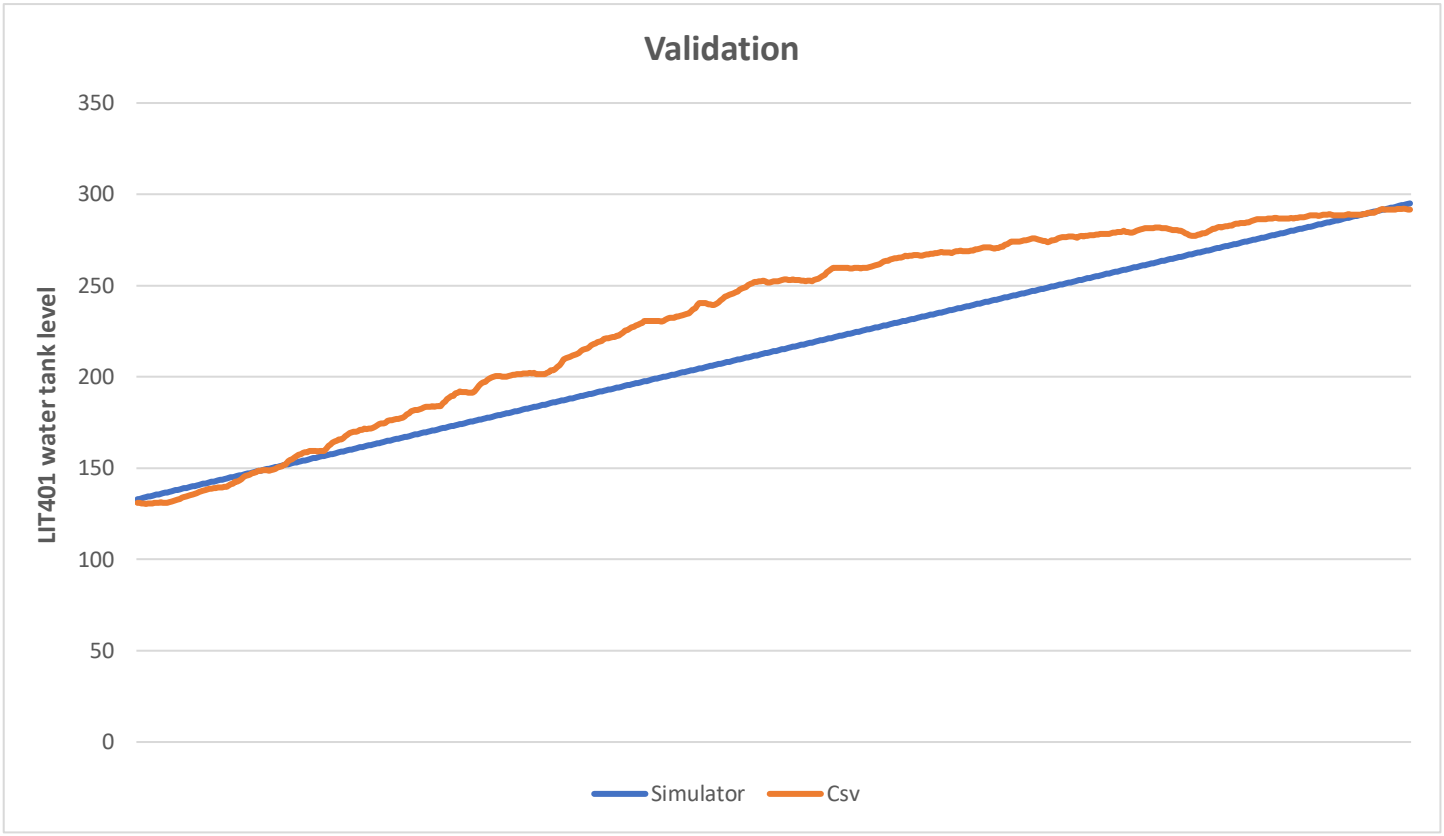


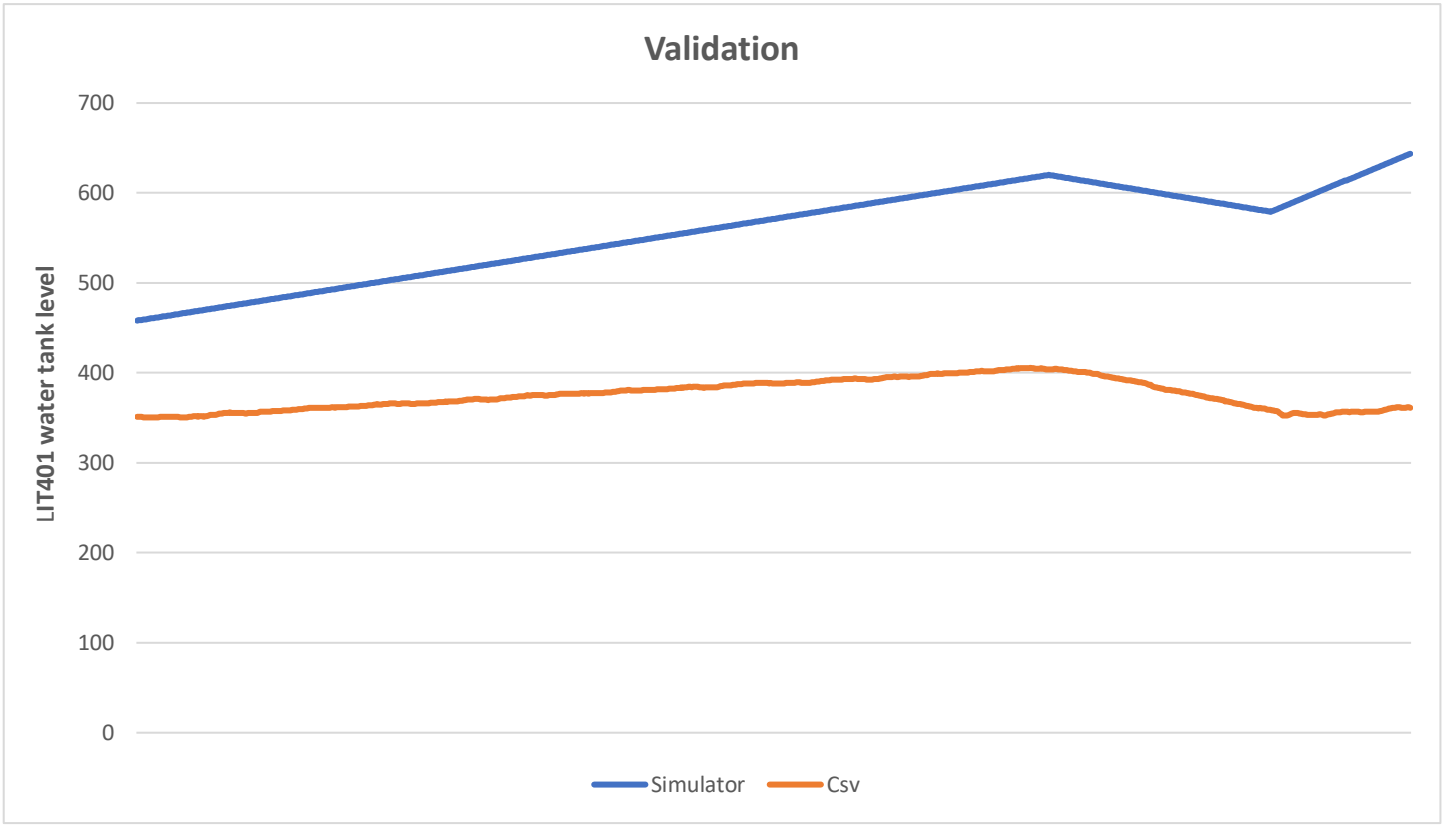

LIT401 Water Flow
Level Indicator Transmitter, 357.716
MV302, 2
LIT401 water tank level
450
400
350
300
250
200
150
100
50
0
22/12/2015 4:52:53 PM
22/12/2015 4:53:36 PM
22/12/2015 4:54:19 PM
22/12/2015 4:55:02 PM
22/12/2015 4:55:45 PM
22/12/2015 4:56:28 PM
22/12/2015 4:57:11 PM
22/12/2015 4:57:54 PM
22/12/2015 4:58:37 PM
22/12/2015 4:59:20 PM
22/12/2015 5:00:03 PM
22/12/2015 5:00:46 PM
22/12/2015 5:01:29 PM
22/12/2015 5:02:12 PM
22/12/2015 5:02:55 PM
22/12/2015 5:03:38 PM
22/12/2015 5:04:21 PM
22/12/2015 5:05:04 PM
22/12/2015 5:05:47 PM
22/12/2015 5:06:30 PM
22/12/2015 5:07:13 PM
22/12/2015 5:07:56 PM
22/12/2015 5:08:39 PM
22/12/2015 5:09:22 PM
22/12/2015 5:10:05 PM
22/12/2015 5:10:48 PM
22/12/2015 5:11:31 PM
22/12/2015 5:12:14 PM
22/12/2015 5:12:57 PM
22/12/2015 5:13:40 PM
22/12/2015 5:14:23 PM
22/12/2015 5:15:06 PM
22/12/2015 5:15:49 PM
22/12/2015 5:16:32 PM
22/12/2015 5:17:15 PM
22/12/2015 5:17:58 PM
22/12/2015 5:18:41 PM
22/12/2015 5:19:24 PM
22/12/2015 5:20:07 PM
22/12/2015 5:20:50 PM
22/12/2015 5:21:33 PM
22/12/2015 5:22:16 PM
22/12/2015 5:22:59 PM


LIT401 Water Flow
Level Indicator Transmitter, 467.9569
MV302, 1
LIT401 water tank level
600
500
400
300
200
100
0
22/12/2015 5:23:27 PM
22/12/2015 5:24:10 PM
22/12/2015 5:24:53 PM
22/12/2015 5:25:36 PM
22/12/2015 5:26:19 PM
22/12/2015 5:27:02 PM
22/12/2015 5:27:45 PM
22/12/2015 5:28:28 PM
22/12/2015 5:29:11 PM
22/12/2015 5:29:54 PM
22/12/2015 5:30:37 PM
22/12/2015 5:31:20 PM
22/12/2015 5:32:03 PM
22/12/2015 5:32:46 PM
22/12/2015 5:33:29 PM
22/12/2015 5:34:12 PM
22/12/2015 5:34:55 PM
22/12/2015 5:35:38 PM
22/12/2015 5:36:21 PM
22/12/2015 5:37:04 PM
22/12/2015 5:37:47 PM
22/12/2015 5:38:30 PM
22/12/2015 5:39:13 PM
22/12/2015 5:39:56 PM
22/12/2015 5:40:39 PM
22/12/2015 5:41:22 PM
22/12/2015 5:42:05 PM
22/12/2015 5:42:48 PM
22/12/2015 5:43:31 PM
22/12/2015 5:44:14 PM
22/12/2015 5:44:57 PM
22/12/2015 5:45:40 PM
22/12/2015 5:46:23 PM
22/12/2015 5:47:06 PM
22/12/2015 5:47:49 PM
22/12/2015 5:48:32 PM
22/12/2015 5:49:15 PM
22/12/2015 5:49:58 PM
22/12/2015 5:50:41 PM
22/12/2015 5:51:24 PM
22/12/2015 5:52:07 PM
22/12/2015 5:52:50 PM
22/12/2015 5:53:33 PM



Στις πιο πάνω γραφικές παραστάσεις βλέπουμε τη ροή λειτουργίας του sensor Level Indicator Transmitter του τέταρτου σταδίου (LIT401) σε συνδυασμό με την κατάσταση της βαλβίδας του τρίτου σταδίου (MV302), με βάσει τα αποτελέσματα του csv αρχείου. Οι πρώτες τρείς γραφικές παραστάσεις μας δείχνουν τις αλλαγές κατάστασης που μπορεί να συμβεί στον LIT401 όταν η βαλβίδα MV302 είναι είτε κλειστή είτε ανοιχτή. Στην αρχή όταν είναι κλειστή, η ροή νερού του της δεξαμενής του LIT401 βρίσκεται σε χαμηλά επίπεδα αλλά όταν η βαλβίδα γίνεται ανοιχτή τότε παρατηρούμε την αύξηση ροής του νερού σε κάποια αντίστοιχα χρονικά διαστήματα. Όταν φτάσει στο σημείο η βαλβίδα να ξανακλείσει, τότε σιγά σιγά αρχίζει και μειώνεται το επίπεδο ροής νερού της δεξαμενής μέχρι σ' ένα σημείο με αποτέλεσμα αυτή εν αλλαγή καταστάσεων να συνεχίζεται για αρκετό χρονικό διάστημα. Η τελευταία γραφική παράσταση παρουσιάζεται ούτως ώστε να δείξουμε μια καλύτερη γενική εικόνα ως προς τη συμπεριφορά ροής του LIT401.


Validation
LIT401 water tank level
350
300
250
200
150
100
50
0
Simulator
Csv


Validation
LIT401 water tank level
700
600
500
400
300
200
100
0
Simulator
Csv

Στις δύο πιο πάνω γραφικές παραστάσεις βλέπουμε τη ροή λειτουργίας του sensor Level Indicator Transmitter του τέταρτου σταδίου (LIT401) σε συνδυασμό με την κατάσταση της βαλβίδας του τρίτου σταδίου (MV302) συγκρίνοντας τα αποτελέσματα του προσομοιωτή SCADASim μ' αυτά του csv αρχείου. Παρατηρούμε ότι ο προσομοιωτής έβγαλε αρκετά καλά αποτελέσματα, καθώς η ροή του νερού του LIT401 τόσο τη χρονική στιγμή όπου η βαλβίδα ανοίγει (πρώτη γραφική παράσταση) όσο και στη χρονική στιγμή που η βαλβίδα κλείνει (δεύτερη γραφική παράσταση) είναι σχεδόν παρόμοια μ' αυτή του csv αρχείου.

Κεφάλαιο 6

Συμπεράσματα

6.1 Τελικά Συμπεράσματα

Στην παρούσα διπλωματική εργασία σκοπός μας ήταν να δείξουμε μια ομαλή λειτουργία του συστήματος με τη χρήση του προσομοιωτή SCADASim. Έχουμε υλοποιήσει το πρώτο μέρος της υλοποίησης το οποίο ήταν να παράξουμε κάποια αποτελέσματα δείχνοντας την πραγματική ροή του συστήματος μέσω του csv αρχείου. Αυτό το πετύχαμε με τη χρήση των δύο registers, τον holding και τον coil οι οποιοί μπορούσαν και να διαβάσουν αλλά και να γράψουν δεδομένα, για τους sensors και actuators αντίστοιχα έτσι ώστε να ξέρουν την κάθε αλλαγή κατάστασης που συμβαίνει. Στη συνέχεια, όσον αφορά το δεύτερο μέρος της υλοποίησης που βασικά αυτός ήταν και ο κύριος σκοπός, να δείξουμε την αποτελεσματικότητα του προσομοιωτή SCADASim ούτως ώστε τα αποτελέσματα τα οποία θα βγάλει να είναι όσο πιο κοντά στα ρεαλιστικά αποτελέσματα του συστήματος.

Από τα πειραματικά μας αποτελέσματα, μπορούμε να διαπιστώσουμε αυτή την αποτελεσματικότητα καθώς τόσο ο προσομοιωτής όσο και το csv αρχείο δημιούργησαν παρόμοια συμπεριφορά ως προς τη ροή νερού του sensor LIT401. Οπότε, αυτό μας οδηγεί στο συμπέρασμα ότι ο προσομοιωτής λειτουργεί σωστά και είναι σε θέση να δημιουργήσει ενθαρρυντικά αποτελέσματα και για τα υπόλοιπα στάδια του συστήματος.

6.2 Μελλοντική Δουλειά

Μια επιπρόσθετη δουλειά που θα μπορούσε να πραγματοποιηθεί σ' αυτό το κομμάτι είναι η ολοκλήρωση του συστήματος όσον αφορά το validation, έτσι ώστε να έχουμε μια πιο γενική εικόνα ως προς τον τρόπο λειτουργίας του συστήματος με την χρήση του προσομοιωτή SCADASim. Αυτό θα έχει ως αποτέλεσμα, τη δημιουργία περισσότερων αποτελεσμάτων αλλά και γραφικών παραστάσεων κατανοώντας καλύτερα τη ροή του κάθε σταδίου του συστήματος.

Επίσης, η ένταξη μερικών επιθέσεων στο σύστημα θα ήταν πολύ χρήσιμο ούτως ώστε ο προσομοιωτής να προσπαθήσει ν' ανιχνεύσει αυτές τις επιθέσεις για να είμαστε προετοιμασμένοι σ' έτσι είδους συμβάντα. Αυτό θ' αποτελεί πλεονέκτημα καθώς όπως προαναφέραμε και προηγουμένως, ο προσομοιωτής SCADASim υποστηρίζει μερικά είδη επιθέσεων.

Βιβλιογραφία

[1] Industrial Internet of things [Online]. Available: https://www.ge.com/digital/blog/what-industrial-internet-things-iiot#:~:text=The%20Industrial%20Internet%20of%20Things%20(IIoT)%2C%20also%20known%20as,analytics%2C%20and%20modern%20industrial%20workers

[2] 3PILLAR GLOBAL, https://www.3pillarglobal.com/insights/industrial-iot-challenges-risks-pitfalls/

[3] IMPERO, https://www.imperosoftware.com/blog/industrial-internet-of-things-complete-guide/

[4] Wireless Sensor Network [Online]. Available: https://en.wikipedia.org/wiki/Wireless_sensor_network

[5] Song Han, Miao Xie, Hsiao-Hwa Chen, and Yun Ling. 2014. Intrusion detection in cyber-physical systems: Techniques and challenges. IEEE systems journal 8, 4 (2014)

[6] Robert Mitchell and Ing-Ray Chen. 2014. A survey of intrusion detection techniques for cyber-physical systems. ACM Computing Surveys (CSUR) 46, 4 (2014)

[7] Fabio Pasqualetti, Florian Dörfler, and Francesco Bullo. 2011. Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design. In Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on. IEEE

[8] André Teixeira, Daniel Pérez, Henrik Sandberg, and Karl Henrik Johansson. 2012. Attack models and scenarios for networked control systems. In Proceedings of the 1st international conference on High Confidence Networked Systems. ACM

[9] DPS TELECOM, https://www.dpstele.com/blog/major-scada-hacks.php

[10] EEO, https://electricenergyonline.com/energy/magazine/181/article/SCADA-System-Vulnerabilities-to-Cyber-Attack.htm

[11] J. Slay and M. Miller, “Lessons learned from the maroochy water breach,” in Critical Infrastructure Protection, ser. IFIP International Federation for Information Processing, E. Goetz and S. Shenoi, Eds. New York: Springer, 2007

[12] K. Poulsen, “Slammer worm crashed Ohio nuke plant network,” 2009 [Online]. Available: http://www.securityfocus.com/news/6767

[13] N. Falliere, L. O. Murchu, and E. Chien, W32.Stuxnet Dossier, Symantec Tech. Rep. 1.4, 2011.

[14] Compass IT Compliance, https://www.compassitc.com/blog/protecting-scada-systems-from-cyber-attacks

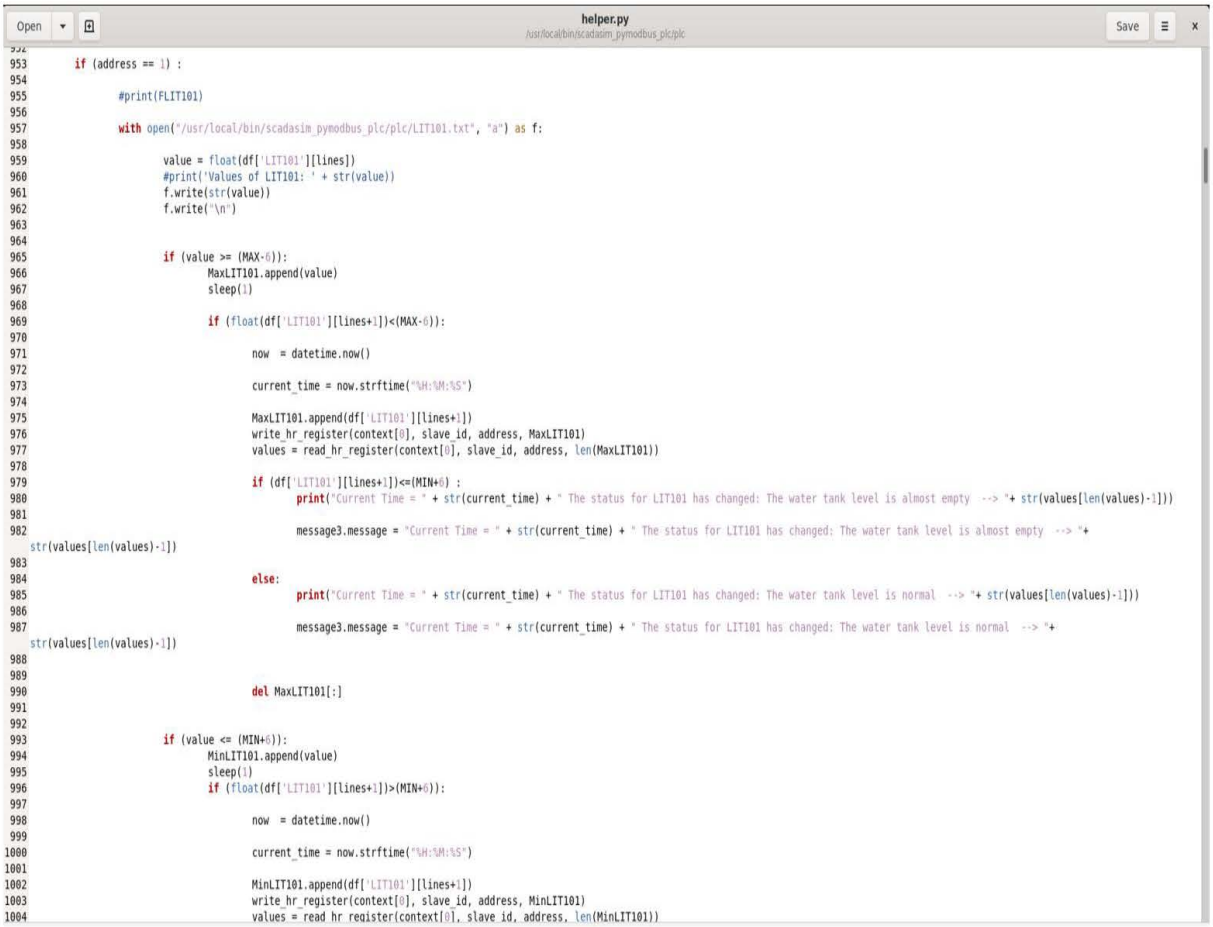
[15] Ahmad Almadhor, University of Denver, A Survey on Generic Scada Simulators

[16] Carlos Queiroz, Abdun Mahmood and Zahir Tari, SCADASim—A Framework for Building SCADA Simulations, IEEE TRANSACTIONS ON SMART GRID, VOL. 2, NO. 4, DECEMBER 2011

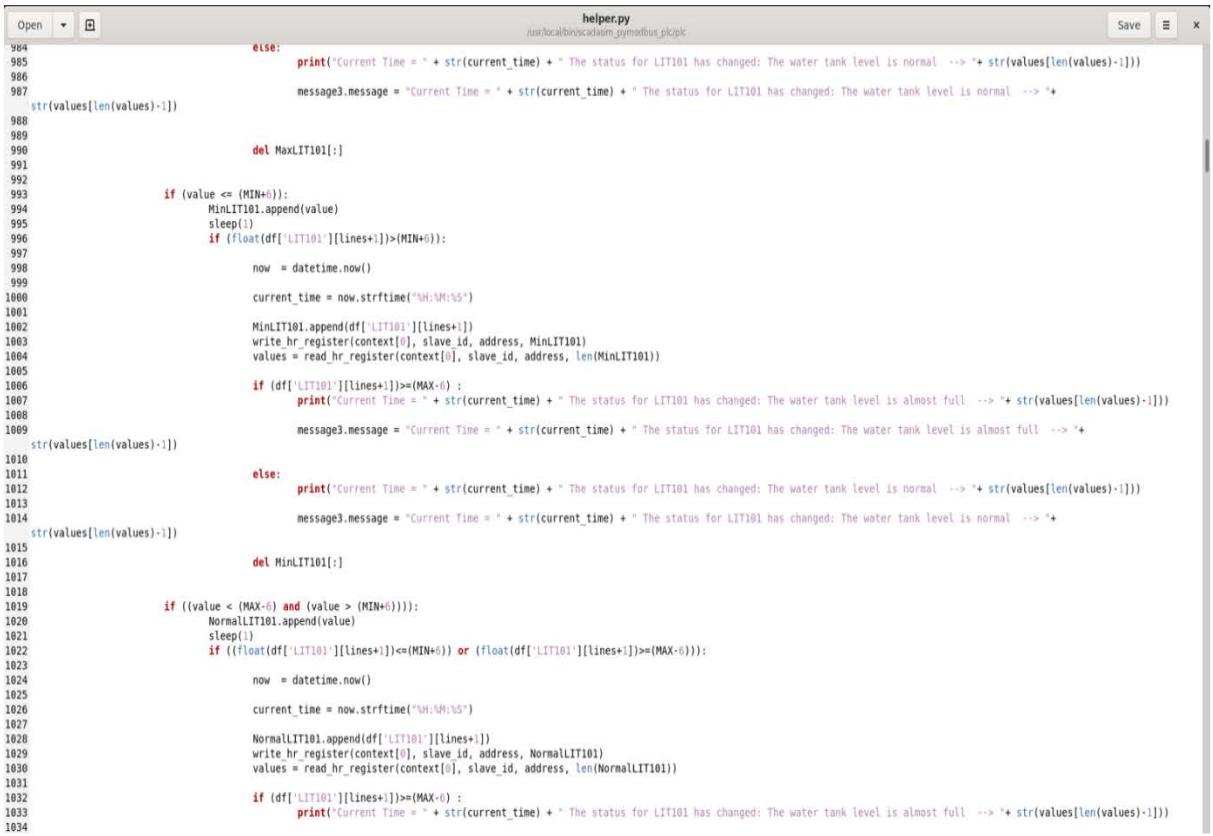
[17] iTrust Centre of Research in Cyber Security, https://itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat/

Παράρτημα Α

Σε αυτό το παράρτημα παρατίθονται κομμάτια του κώδικα υλοποίησης του πρώτου σταδίου του συστήματος για τη δημιουργία αποτελεσμάτων χρησιμοποιώντας τους δύο καταχωρητές Holding_register και Coil_register για διάβασμα και αποθήκευση νέων δεδομένων μέσω του csv αρχείου.




```
    if (address == 1) :

            #print(FLIT101)

            with open("/usr/local/bin/scadasim_pymodbus_plc/plc/LIT101.txt", "a") as f:

                    value = float(df['LIT101'][lines])
                    #print('Values of LIT101: ' + str(value))
                    f.write(str(value))
                    f.write('\n')


                    if (value >= (MAX-6)):
                            MaxLIT101.append(value)
                            sleep(1)

                            if (float(df['LIT101'][lines+1])<(MAX-6)):

                                    now  = datetime.now()

                                    current_time = now.strftime("%H:%M:%S")

                                    MaxLIT101.append(df['LIT101'][lines+1])
                                    write_hr_register(context[0], slave_id, address, MaxLIT101)
                                    values = read_hr_register(context[0], slave_id, address, len(MaxLIT101))

                                    if (df['LIT101'][lines+1])<=(MIN+6) :
                                            print("Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is almost empty  --> "+ str(values[len(values)-1]))

                                            message3.message = "Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is almost empty  --> "+
str(values[len(values)-1])

                                    else:
                                            print("Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is normal  --> "+ str(values[len(values)-1]))

                                            message3.message = "Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is normal  --> "+
str(values[len(values)-1])


                                    del MaxLIT101[:]


                    if (value <= (MIN+6)):
                            MinLIT101.append(value)
                            sleep(1)
                            if (float(df['LIT101'][lines+1])>(MIN+6)):

                                    now  = datetime.now()

                                    current_time = now.strftime("%H:%M:%S")

                                    MinLIT101.append(df['LIT101'][lines+1])
                                    write_hr_register(context[0], slave_id, address, MinLIT101)
                                    values = read_hr_register(context[0], slave_id, address, len(MinLIT101))
```



helper.py
/usr/local/bin/scadasim_pymodbus_plc/plc

```
                                        else:
                                                print("Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is normal  --> "+ str(values[len(values)-1]))

                                                message3.message = "Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is normal  --> "+
    str(values[len(values)-1])


                                        del MaxLIT101[:]


                        if (value <= (MIN+6)):
                                MinLIT101.append(value)
                                sleep(1)
                                if (float(df['LIT101'][lines+1])>(MIN+6)):

                                        now  = datetime.now()

                                        current_time = now.strftime("%H:%M:%S")

                                        MinLIT101.append(df['LIT101'][lines+1])
                                        write_hr_register(context[0], slave_id, address, MinLIT101)
                                        values = read_hr_register(context[0], slave_id, address, len(MinLIT101))

                                        if (df['LIT101'][lines+1])>=(MAX-6) :
                                                print("Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is almost full  --> "+ str(values[len(values)-1]))

                                                message3.message = "Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is almost full  --> "+
    str(values[len(values)-1])

                                        else:
                                                print("Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is normal  --> "+ str(values[len(values)-1]))

                                                message3.message = "Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is normal  --> "+
    str(values[len(values)-1])

                                        del MinLIT101[:]


                        if ((value < (MAX-6) and (value > (MIN+6)))):
                                NormalLIT101.append(value)
                                sleep(1)
                                if ((float(df['LIT101'][lines+1])<=(MIN+6)) or (float(df['LIT101'][lines+1])>=(MAX-6))):

                                        now  = datetime.now()

                                        current_time = now.strftime("%H:%M:%S")

                                        NormalLIT101.append(df['LIT101'][lines+1])
                                        write_hr_register(context[0], slave_id, address, NormalLIT101)
                                        values = read_hr_register(context[0], slave_id, address, len(NormalLIT101))

                                        if (df['LIT101'][lines+1])>=(MAX-6) :
                                                print("Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is almost full  --> "+ str(values[len(values)-1]))
```

helper.py
/usr/local/bin/scadasim_pymodbus_plc/plc

```
1010
1011                                    else:
1012                                            print("Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is normal  --> "+ str(values[len(values)-1]))
1013
1014                                            message3.message = "Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is normal  --> "+
    str(values[len(values)-1])
1015
1016                                    del MinLIT101[:]
1017
1018
1019                        if ((value < (MAX-6) and (value > (MIN+6)))):
1020                                NormalLIT101.append(value)
1021                                sleep(1)
1022                                if ((float(df['LIT101'][lines+1])<=(MIN+6)) or (float(df['LIT101'][lines+1])>=(MAX-6))):
1023
1024                                        now  = datetime.now()
1025
1026                                        current_time = now.strftime("%H:%M:%S")
1027
1028                                        NormalLIT101.append(df['LIT101'][lines+1])
1029                                        write_hr_register(context[0], slave_id, address, NormalLIT101)
1030                                        values = read_hr_register(context[0], slave_id, address, len(NormalLIT101))
1031
1032                                        if (df['LIT101'][lines+1])>=(MAX-6) :
1033                                                print("Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is almost full  --> "+ str(values[len(values)-1]))
1034
1035                                                message3.message = "Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is almost full  --> "+
    str(values[len(values)-1])
1036
1037                                        else:
1038                                                print("Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is almost empty  --> "+ str(values[len(values)-1]))
1039
1040                                                message3.message = "Current Time = " + str(current_time) + " The status for LIT101 has changed: The water tank level is almost empty  --> "+
    str(values[len(values)-1])
1041
1042                                        del NormalLIT101[:]
1043
1044
1045                f.close()
```

Open helper.py /usr/local/bin/scadasim_pymodbus_plc/plc Save

```
840
841
842     if (address == 2) :
843
844         with open("/usr/local/bin/scadasim_pymodbus_plc/plc/MV101.txt", "a") as f:
845
846
847             value = (df['MV101'][lines])
848             #print('Values of MV101: ' + str(value))
849             f.write(str(value))
850             f.write("\n")
851
852
853             if (value == 2):
854
855                 ON_MV101.append(value)
856                 sleep(1)
857                 if (((df['MV101'][lines+1]) != 2)):
858
859
860                     now  = datetime.now()
861
862                     current_time = now.strftime("%H:%M:%S")
863
864                     ON_MV101.append(df['MV101'][lines+1])
865                     write_co_register(context[0], slave_id, address, ON_MV101)
866                     values = read_co_register(context[0], slave_id, address, len(ON_MV101))
867                     if (values[len(values)-1]==0):
868
869                         print("Current Time = " + str(current_time) + ' The status for MV101 has changed: The actuator MV101 is paused --> ' + str(values[len(values)-1]))
870
871                         message2.message = "Current Time = " + str(current_time) + ' The status for MV101 has changed: The actuator MV101 is paused --> ' + str(values[len(values)-1])
872
873                     else:
874                         print("Current Time = " + str(current_time) + ' The status for MV101 has changed: The actuator MV101 is OFF --> ' + str(values[len(values)-1]))
875
876
877                         message2.message = "Current Time = " + str(current_time) + ' The status for MV101 has changed: The actuator MV101 is OFF --> ' + str(values[len(values)-1])
878
879
880                     del ON_MV101[:]
881                     #sleep(2)
882
883                     #f.write('The actuator MV101 was turned ON --> ')
884                     #f.write(str(values[len(values)-1]))
885                     #f.write('\n')
886
887
```

Open helper.py /usr/local/bin/scadasim_pymodbus_plc/plc Save

```
889
890
891
892
893             if (value == 1):
894                 OFF_MV101.append(value)
895                 sleep(1)
896                 if ((df['MV101'][lines+1]) != 1):
897
898                     now  = datetime.now()
899
900                     current_time = now.strftime("%H:%M:%S")
901
902                     OFF_MV101.append(df['MV101'][lines+1])
903                     write_co_register(context[0], slave_id, address, OFF_MV101)
904                     values = read_co_register(context[0], slave_id, address, len(OFF_MV101))
905                     if (values[len(values)-1]==0):
906                         print("Current Time = " + str(current_time) + ' The status for MV101 has changed: The actuator MV101 is paused --> ' + str(values[len(values)-1]))
907
908                         message2.message = "Current Time = " + str(current_time) + ' The status for MV101 has changed: The actuator MV101 is paused --> ' + str(values[len(values)-1])
909
910                     else:
911                         print("Current Time = " + str(current_time) + ' The status for MV101 has changed: The actuator MV101 is ON --> ' + str(values[len(values)-1]))
912
913                         message2.message = "Current Time = " + str(current_time) + ' The status for MV101 has changed: The actuator MV101 is ON --> ' + str(values[len(values)-1])
914
915
916                     del OFF_MV101[:]
917
918
919
920
921             if (value == 0):
922                 P_MV101.append(value)
923                 sleep(1)
924                 if ((df['MV101'][lines+1]) != 0 ):
925
926                     now  = datetime.now()
927
928                     current_time = now.strftime("%H:%M:%S")
929
930                     P_MV101.append(df['MV101'][lines+1])
931                     write_co_register(context[0], slave_id, address, P_MV101)
932                     values = read_co_register(context[0], slave_id, address, len(P_MV101))
933                     if (values[len(values)-1]==2):
934                                                                                                                 print("Current Time = " + str(current_time) + ' The
    status for MV101 has changed: The actuator MV101 is ON --> ' + str(values[len(values)-1]))
935                                                                                                                 message2.message = "Current Time = " + str(current_time)
    + ' The status for MV101 has changed: The actuator MV101 is ON --> ' + str(values[len(values)-1])
936
937
938                     else:
939                         print("Current Time = " + str(current_time) + ' The status for MV101 has changed: The actuator MV101 is OFF --> ' + str(values[len(values)-1]))
940
```

Open ▾ | helper.py — /usr/local/bin/scadasim_pymodbus_plc/plc | Save

```
911                                     print("Current Time = " + str(current_time) + ' The status for MV101 has changed: The actuator MV101 is ON --> ' + str(values[len(values)-1]))
912
913                                     message2.message = "Current Time = " + str(current_time) + ' The status for MV101 has changed: The actuator MV101 is ON --> ' + str(values[len(values)-1])
914
915
916                                 del OFF_MV101[:]
917
918
919
920
921                         if (value == 0):
922                                 P_MV101.append(value)
923                                 sleep(1)
924                                 if ((df['MV101'][lines+1]) != 0 ):
925
926                                         now  = datetime.now()
927
928                                         current_time = now.strftime("%H:%M:%S")
929
930                                         P_MV101.append(df['MV101'][lines+1])
931                                         write_co_register(context[0], slave_id, address, P_MV101)
932                                         values = read_co_register(context[0], slave_id, address, len(P_MV101))
933                                         if (values[len(values)-1]==2):
934                                                                                                                                                 print("Current Time = " + str(current_time) + ' The
    status for MV101 has changed: The actuator MV101 is ON --> ' + str(values[len(values)-1]))
935                                                                                                                                                 message2.message = "Current Time = " + str(current_time)
    + ' The status for MV101 has changed: The actuator MV101 is ON --> ' + str(values[len(values)-1])
936
937
938                                         else:
939                                                 print("Current Time = " + str(current_time) + ' The status for MV101 has changed: The actuator MV101 is OFF --> ' + str(values[len(values)-1]))
940
941                                                 message2.message = "Current Time = " + str(current_time) + ' The status for MV101 has changed: The actuator MV101 is OFF --> ' + str(values[len(values)-1])
942
943
944                                         del P_MV101[:]
945
946
947
948
949                 f.close()
950
951
```

Open ▾ | helper.py — /usr/local/bin/scadasim_pymodbus_plc/plc | Save

```
370
371
372
373
374 def func(context, slave_id, address, count, MAX, MIN, FIT101, MinFIT101, NormalFIT101, MaxLIT101, MinLIT101, NormalLIT101, ON_P101, OFF_P101, ON_MV101, OFF_MV101, P_MV101, FlagFIT101, FLIT101, FlagP101,
    message1, message2, message3, message4):
375
376
377         global FlagMV101
378
379
380         now  = datetime.now()
381
382         current_time = now.strftime("%H:%M:%S")
383
384         with open("/usr/local/bin/scadasim_pymodbus_plc/plc/Schedule.txt", "a") as f:
385
386
387                 if (address == 0) :
388
389
390
391                         f.write(message1.message)
392                         f.write('\n')
393
394                         write_hr_register(context[0], slave_id, address, FIT101)
395
396                         values = read_hr_register(context[0], slave_id, address, len(FIT101))
397
398
399                         write_hr_register(context[0], slave_id, address, MinFIT101)
400
401                         valuesMin = read_hr_register(context[0], slave_id, address, len(MinFIT101))
402
403                         write_hr_register(context[0], slave_id, address, NormalFIT101)
404
405                         valuesNormal = read_hr_register(context[0], slave_id, address, len(NormalFIT101))
406
407
408                         if (len(values) > 0) :
409
410                                 if (values[len(values)-1] >= (MAX-0.05)):
411
412                                         print("Current Time = " + str(current_time) + " The pump's water flow is almost FULL (FIT101) --> " + str(values[len(values)-1]))
413                                         f.write("Current Time = " + str(current_time) + " The pump's water flow is almost FULL (FIT101) --> ")
414                                         f.write(str(values[len(values)-1]))
415                                         f.write('\n')
416
417
418                         if (len(valuesMin)  > 0 ) :
419
420                                 if (valuesMin[len(valuesMin)-1] <= (MIN+0.05)):
421
422                                         print("Current Time = " + str(current_time) + " The pump's water flow is  very LOW (FIT101) --> " + str(valuesMin[len(valuesMin)-1]))
```

Open | helper.py /usr/local/bin/scadasim_pymodbus_plc/plc | Save

```
391                     f.write(message1.message)
392                     f.write('\n')
393
394                     write_hr_register(context[0], slave_id, address, FIT101)
395
396                     values = read_hr_register(context[0], slave_id, address, len(FIT101))
397
398
399                     write_hr_register(context[0], slave_id, address, MinFIT101)
400
401                     valuesMin = read_hr_register(context[0], slave_id, address, len(MinFIT101))
402
403                     write_hr_register(context[0], slave_id, address, NormalFIT101)
404
405                     valuesNormal = read_hr_register(context[0], slave_id, address, len(NormalFIT101))
406
407
408                     if (len(values) > 0) :
409
410                             if (values[len(values)-1] >= (MAX-0.05)):
411
412                                     print("Current Time = " + str(current_time) + " The pump's water flow is almost FULL (FIT101) --> " + str(values[len(values)-1]))
413                                     f.write("Current Time = " + str(current_time) + " The pump's water flow is almost FULL (FIT101) --> ")
414                                     f.write(str(values[len(values)-1]))
415                                     f.write('\n')
416
417
418                     if (len(valuesMin)  > 0 ) :
419
420                             if (valuesMin[len(valuesMin)-1] <= (MIN+0.05)):
421
422                                     print("Current Time = " + str(current_time) + " The pump's water flow is  very LOW (FIT101) --> " + str(valuesMin[len(valuesMin)-1]))
423                                     f.write("Current Time = " + str(current_time) + " The pump's water flow is  very LOW (FIT101) --> ")
424                                     f.write(str(valuesMin[len(valuesMin)-1]))
425                                     f.write('\n')
426
427
428                     if (len(valuesNormal)  > 0 ) :
429
430                             if ((valuesNormal[len(valuesNormal)-1] < (MAX-0.05)) and (valuesNormal[len(valuesNormal)-1] > (MIN+0.05))):
431
432                                     print("Current Time = " + str(current_time) + " The pump's water flow is NORMAL (FIT101) --> " + str(valuesNormal[len(valuesNormal)-1]))
433                                     f.write("Current Time = " + str(current_time) + " The pump's water flow is NORMAL (FIT101) --> ")
434                                     f.write(str(valuesNormal[len(valuesNormal)-1]))
435                                     f.write('\n')
436
437
438
439             if (address == 2) :
440
441
```

Open | helper.py /usr/local/bin/scadasim_pymodbus_plc/plc | Save

```
430                             if ((valuesNormal[len(valuesNormal)-1] < (MAX-0.05)) and (valuesNormal[len(valuesNormal)-1] > (MIN+0.05))):
431
432                                     print("Current Time = " + str(current_time) + " The pump's water flow is NORMAL (FIT101) --> " + str(valuesNormal[len(valuesNormal)-1]))
433                                     f.write("Current Time = " + str(current_time) + " The pump's water flow is NORMAL (FIT101) --> ")
434                                     f.write(str(valuesNormal[len(valuesNormal)-1]))
435                                     f.write('\n')
436
437
438
439             if (address == 2) :
440
441
442
443                     f.write(message2.message)
444                     f.write('\n')
445
446
447                     write_co_register(context[0], slave_id, address, ON_MV101)
448
449                     valuesON = read_co_register(context[0], slave_id, address, len(ON_MV101))
450
451                     write_co_register(context[0], slave_id, address, OFF_MV101)
452
453                     valuesOFF = read_co_register(context[0], slave_id, address, len(OFF_MV101))
454
455                     write_co_register(context[0], slave_id, address, P_MV101)
456
457                     valuesP = read_co_register(context[0], slave_id, address, len(P_MV101))
458
459
460
461                     if (len(valuesON)  > 0 ) :
462
463
464                             if (valuesON[len(valuesON)-1] == 2):
465
466                                     print("Current Time = " + str(current_time) + " The actuator MV101 current status is ON --> " + str(valuesON[len(valuesON)-1]))
467                                     f.write("Current Time = " + str(current_time) + " The actuator MV101 current status is ON --> ")
468                                     f.write(str(valuesON[len(valuesON)-1]))
469                                     f.write('\n')
470
471
472                     if (len(valuesOFF)  > 0 ) :
473
474
475                             if (valuesOFF[len(valuesOFF)-1] == 1):
476
477                                     print("Current Time = " + str(current_time) + " The actuator MV101 current status is OFF --> " + str(valuesOFF[len(valuesOFF)-1]))
478                                     f.write("Current Time = " + str(current_time) + " The actuator MV101 current status is OFF --> ")
479                                     f.write(str(valuesOFF[len(valuesOFF)-1]))
480                                     f.write('\n')
```

Open | helper.py | /usr/local/bin/scadasim_pymodbus_plc/plc | Save

```
451                     write_co_register(context[0], slave_id, address, OFF_MV101)
452
453                     valuesOFF = read_co_register(context[0], slave_id, address, len(OFF_MV101))
454
455                     write_co_register(context[0], slave_id, address, P_MV101)
456
457                     valuesP = read_co_register(context[0], slave_id, address, len(P_MV101))
458
459
460
461                     if (len(valuesON)  > 0 ) :
462
463
464                             if (valuesON[len(valuesON)-1] == 2):
465
466                                     print("Current Time = " + str(current_time) + " The actuator MV101 current status is ON --> " + str(valuesON[len(valuesON)-1]))
467                                     f.write("Current Time = " + str(current_time) + " The actuator MV101 current status is ON --> ")
468                                     f.write(str(valuesON[len(valuesON)-1]))
469                                     f.write('\n')
470
471
472                     if (len(valuesOFF)  > 0 ) :
473
474
475                             if (valuesOFF[len(valuesOFF)-1] == 1):
476
477                                     print("Current Time = " + str(current_time) + " The actuator MV101 current status is OFF --> " + str(valuesOFF[len(valuesOFF)-1]))
478                                     f.write("Current Time = " + str(current_time) + " The actuator MV101 current status is OFF --> ")
479                                     f.write(str(valuesOFF[len(valuesOFF)-1]))
480                                     f.write('\n')
481
482
483                     if (len(valuesP)  > 0 ) :
484
485
486                             if (valuesP[len(valuesP)-1] == 0):
487
488                                     print("Current Time = " + str(current_time) + " The actuator MV101 current status is PAUSED --> " + str(valuesP[len(valuesP)-1]))
489                                     f.write("Current Time = " + str(current_time) + " The actuator MV101 current status is PAUSED --> ")
490                                     f.write(str(valuesP[len(valuesP)-1]))
491                                     f.write('\n')
492
```

Open | helper.py | /usr/local/bin/scadasim_pymodbus_plc/plc | Save

```
495
496             if (address == 1) :
497
498
499                     f.write(message3.message)
500                     f.write('\n')
501
502
503                     write_hr_register(context[0], slave_id, address, MaxLIT101)
504
505                     valuesMAX = read_hr_register(context[0], slave_id, address, len(MaxLIT101))
506
507
508                     write_hr_register(context[0], slave_id, address, MinLIT101)
509
510                     valuesMIN = read_hr_register(context[0], slave_id, address, len(MinLIT101))
511
512
513                     write_hr_register(context[0], slave_id, address, NormalLIT101)
514
515                     valuesNOR = read_hr_register(context[0], slave_id, address, len(NormalLIT101))
516
517
518                     if (len(valuesMAX)  > 0 ) :
519
520
521                             if (valuesMAX[len(valuesMAX)-1] >= (MAX-5)):
522
523
524                                     print("Current Time = " + str(current_time) + " The water tank level status is almost FULL --> " + str(valuesMAX[len(valuesMAX)-1]))
525                                     f.write("Current Time = " + str(current_time) + " The water tank level status is almost FULL --> ")
526                                     f.write(str(valuesMAX[len(valuesMAX)-1]))
527                                     f.write('\n')
528
529
530                     if (len(valuesMIN)  > 0 ) :
531
532
533                             if (valuesMIN[len(valuesMIN)-1] <= (MIN+5)):
534
535                                     print("Current Time = " + str(current_time) + " The water tank level current status is very LOW --> " + str(valuesMIN[len(valuesMIN)-1]))
536                                     f.write("Current Time = " + str(current_time) + " The water tank level status is very LOW --> ")
537                                     f.write(str(valuesMIN[len(valuesMIN)-1]))
538                                     f.write('\n')
539
540
541                     if (len(valuesNOR)  > 0 ) :
542
543                             if ((valuesNOR[len(valuesNOR)-1] < (MAX-5) and (valuesNOR[len(valuesNOR)-1] > (MIN+5)))):
544
545                                     print("Current Time = " + str(current_time) + " The water tank level current status is NORMAL --> " + str(valuesNOR[len(valuesNOR)-1]))
546                                     f.write("Current Time = " + str(current_time) + " The water tank level status is NORMAL --> ")
547                                     f.write(str(valuesNOR[len(valuesNOR)-1]))
```

Open | helper.py | /usr/local/bin/scadasim_pymodbus_plc/plc | Save

```
                                print("Current Time = " + str(current_time) + " The water tank level current status is NORMAL --> " + str(valuesNOR[len(valuesNOR)-1]))
                                f.write("Current Time = " + str(current_time) + " The water tank level status is NORMAL --> ")
                                f.write(str(valuesNOR[len(valuesNOR)-1]))
                                f.write('\n')



        if (address == 3) :


                f.write(message4.message)
                f.write('\n')


                write_co_register(context[0], slave_id, address, ON_P101)

                valuesOnP = read_co_register(context[0], slave_id, address, len(ON_P101))


                write_co_register(context[0], slave_id, address, OFF_P101)

                valuesOffP = read_co_register(context[0], slave_id, address, len(OFF_P101))


                if (len(valuesOnP)  > 0 ) :

                        if (valuesOnP[len(valuesOnP)-1] == 2):

                                print("Current Time = " + str(current_time) + " The actuator P101 current status is ON --> " + str(valuesOnP[len(valuesOnP)-1]))
                                f.write("Current Time = " + str(current_time) + " The actuator P101 current status is ON --> ")
                                f.write(str(valuesOnP[len(valuesOnP)-1]))
                                f.write('\n')

                if (len(valuesOffP)  > 0 ) :

                        if (valuesOffP[len(valuesOffP)-1] == 1):


                                print("Current Time = " + str(current_time) + " The actuator P101 current status is OFF --> " + str(valuesOffP[len(valuesOffP)-1]))
                                f.write("Current Time = " + str(current_time) + " The actuator P101 current status is OFF --> ")
                                f.write(str(valuesOffP[len(valuesOffP)-1]))
                                f.write('\n')

f.close()
```