ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Παρακολούθηση της διαδικασίας της ζύμωσης κατά την αποθήκευση των τροφίμων με χρήση αισθητήρων και τεχνολογίας Internet of Things

Διπλωματική Εργασία

Ακρίβος Κωνσταντίνος

Κολιγλιάτης Αριστοτέλης

Επιβλέπων: Αντωνόπουλος Χρήστος

Φεβρουάριος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Παρακολούθηση της διαδικασίας της ζύμωσης κατά την αποθήκευση των τροφίμων με χρήση αισθητήρων και τεχνολογίας Internet of Things

Διπλωματική Εργασία

Ακρίβος Κωνσταντίνος

Κολιγλιάτης Αριστοτέλης

Επιβλέπων: Αντωνόπουλος Χρήστος

Φεβρουάριος 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Monitoring the fermentation process during food storage using sensors and IoT technology

Diploma Thesis

Akrivos Konstantinos

Koligliatis Aristotelis

Supervisor: Antonopoulos Christos

February 2022

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων	**Χρήστος Αντωνόπουλος** Αναπληρωτής Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Σπύρος Λάλης** Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας
Μέλος	**Νικόλαος Μπέλλας** Καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελούν αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλουν οποιασδήποτε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχουν έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Δηλώνω επίσης ότι τα αποτελέσματα της εργασίας δεν έχουν χρησιμοποιηθεί για την απόκτηση άλλου πτυχίου. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής.

Ο Δηλών	Ο Δηλών
Ακρίβος Κωνσταντίνος	Κολιγλιάτης Αριστοτέλης

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

The Declarant	The Declarant
Akrivos Konstantinos	Koligliatis Aristotelis

Ευχαριστίες

Στο σημείο αυτό, θα θέλαμε να εκφράσουμε τις ειλικρινείς μας ευχαριστίες στον επιβλέποντα καθηγητή κύριο Αντωνόπουλο Χρήστο για την πολύτιμη συμβολή του σε όλη τη διάρκεια εκπόνησης της παρούσας διπλωματικής εργασίας. Η καθοδήγησή του, μέσω των γνώσεων και των χρήσιμων συμβουλών και παρατηρήσεων που μας παρείχε σε όλα τα στάδια της εργασίας, ήταν πολύ σημαντική. Επίσης θα θέλαμε να ευχαριστήσουμε τον κύριο Σπύρο Λάλη και κύριο Νικόλαο Μπέλλα για την συμμετοχή τους στην επιτροπή. Τέλος, ένα μεγάλο ευχαριστώ στις οικογένειες μας και στους φίλους μας για τη σημαντική συναισθηματική στήριξη και κατανόηση που μας παρείχαν σε όλο το διάστημα εκπόνησης της διπλωματικής μας εργασίας, καθώς και καθόλη τη διάρκεια των σπουδών μας.

Διπλωματική Εργασία

Παρακολούθηση της διαδικασίας της ζύμωσης κατά την αποθήκευση των τροφίμων με χρήση αισθητήρων και τεχνολογίας Internet of Things

Ακρίβος Κωνσταντίνος

Κολιγλιάτης Αριστοτέλης

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται την υλοποίηση μίας ολοκληρωμένης τεχνολογικής λύσης για την παρακολούθηση της διαδικασίας της ζύμωσης κατά την παραγωγή και την αποθήκευση των ζυμούμενων τροφίμων. Ως στόχος ορίζεται η βελτιστοποίηση του ποιοτικού ελέγχου των τροφίμων στις βιομηχανίες με την ταυτόχρονη μείωση των απωλειών, δημιουργώντας ένα εργαλείο, το οποίο συνδυάζει τεχνολογίες Internet of Things.Η ολοκληρωμένη λύση που περιγράφεται στην παρούσα εργασία βασίζεται σε μία συσκευή ανίχνευσης, η οποία συλλέγει τις μετρήσεις από το περιβάλλον αποθήκευσης των τροφίμων και τις προωθεί σε μια Internet of Things πλατφόρμα. Η τελευταία προσφέρει απομακρυσμένη παρακολούθηση και ειδοποιεί για προβλήματα που εμφανίζονται κατά την παραγωγή. Η αποθήκευση και επεξεργασία των δεδομένων γίνεται από μία υποδομή cloud και η πληροφορία για την οργάνωση και την διαχείριση της παραγωγής είναι διαθέσιμη στον τελικό χρήση από μία web based εφαρμογή. Η ανάπτυξή της έγινε σε συνεργασία από τους Ακρίβο Κωνσταντίνο και Κολιγλιάτη Αριστοτέλη για τον λόγο ότι η τεχνολογική λύση που παρουσιάζεται δεν θα μπορούσε να χωριστεί σε δύο διαφορετικές εργασίες για λόγους συνάφειας.

Λέξεις-κλειδιά:

Internet of Things, αισθητήρες, απομακρυσμένη παρακολούθηση, ειδοποίηση, ζυμούμενα τρόφιμα, διαδικασία ζύμωσης, IoT Core, Google Cloud Platform, Django Framework, Raspberry Pi.

Diploma Thesis

Monitoring the fermentation process during food storage using sensors and IoT technology

Akrivos Konstantinos

Koligliatis Aristotelis

Abstract

This Thesis deals with the implementation of an integrated technological solution for the monitoring of the fermentation process during the production and storage of fermented foods. The goal is to optimize food quality control in industries and reduce losses by creating a tool that combines Internet of Things technologies. The all-in-one solution described in the following work consists of a sensing device that collects data from the food storage environment and forwards it to an Internet of Things platform that offers remote monitoring and notification for issues that will occur during production. The data is stored and processed in a cloud infrastructure and is available for the end users via a web-based application that offers production management and monitoring. This Thesis was developed by the collaboration of Akrivos Konstantinos and Koligliatis Aristotelis, because the technological solution presented could not be divided into two different Theses each with a clear objective.

Keywords:

Internet of Things, sensors, remote monitoring, notifications, fermented foods, fermentation process, IoT Core, Google Cloud Platform, Django Framework, Raspberry Pi.

Πίνακας περιεχομένων

Κατάλογος εικόνων

Συντομογραφίες

βλπ	βλέπε
κ.λπ.	και λοιπά
κ.ο.κ	και ούτω καθεξής
π.χ.	παραδείγματος χάριν
ΒΙ.ΠΕ.	Βιομηχανική Περιοχή
IoT	Internet of Things
SoC	System on chip
API	Application Programming Interface

Κεφάλαιο 1 Εισαγωγή

1.1 Περιγραφή του προβλήματος

Ο παγκόσμιος οργανισμός τροφίμων και γεωργίας παρουσίασε ότι περίπου το 1/3 του παγκόσμιου φαγητού χάνεται ή σπαταλάται κάθε χρόνο [1] . Ως απώλεια τροφίμων ορίζεται η μείωση της ποσότητας ή της ποιότητας των τροφίμων που προκύπτει από αποφάσεις και ενέργειες των προμηθευτών τροφίμων σε όλη την τροφική αλυσίδα, εξαιρουμένων των λιανοπωλητών, των παροχών υπηρεσιών τροφίμων και των καταναλωτών. Ειδικότερα, η απώλεια τροφίμων εντοπίζεται πριν τα τρόφιμα φτάσουν στον καταναλωτή και οφείλεται σε ανεπάρκειες στο στάδιο παραγωγής και επεξεργασίας των τροφίμων, ενώ η σπατάλη τροφίμων εντοπίζεται αφού τα τρόφιμα φτάσουν στον καταναλωτή και αποδίδεται σε ανεπάρκειες στο στάδιο της κατανάλωσης [2]. Όπως προκύπτει, το πρόβλημα της απώλειας σημειώνεται κατά το πρώτο και το δεύτερο επίπεδο επεξεργασίας τροφίμων. Η παρούσα διπλωματική εργασία εστιάζει στο δεύτερο επίπεδο επεξεργασίας τροφίμων, κατά το οποίο δημιουργούνται τρόφιμα από υλικά που είναι έτοιμα προς χρήση, και συγκεκριμένα στην κατηγορία των ζυμούμενων τροφίμων. Σε αυτή την κατηγορία ανήκουν τα γαλακτοκομικά προϊόντα, οι ελιές, το κρασί, η μπύρα και άλλα [3] . Η ιδιαιτερότητα των συγκεκριμένων τροφίμων είναι ότι κατά διάρκεια της αποθήκευσής τους, υπόκεινται σε μια χημική διαδικασία, ώστε να ωριμάσουν και για να γίνουν βρώσιμα. Η διαδικασία αυτή ονομάζεται ωρίμανση και σε κάποια προϊόντα ζύμωση και τελικά αυτή καθορίζει την ποιότητα του προϊόντος. Έχοντας ως στόχο την βέλτιστη ποιότητα, απαιτούνται ιδανικές συνθήκες σε συγκεκριμένο χρόνο, ενώ αν αυτές αποκλίνουν, οδηγούν σε απώλειες του προϊόντος. Στις περισσότερες περιπτώσεις η διαδικασία είναι αναερόβια με αποτέλεσμα οι δεξαμενές αποθήκευσης να μοιάζουν με μαύρα κουτιά για τους παραγωγούς ή τους υπεύθυνους ποιότητας της βιομηχανίας. Αυτό σημαίνει πως αν δεν ολοκληρωθεί η διαδικασία, δεν γνωρίζουν τι ακριβώς γίνεται στο εσωτερικό της δεξαμενής και αναγκάζονται να εργάζονται εμπειρικά. Η αδυναμία ακριβούς χειρισμού και ελέγχου των συνθηκών κατά την επεξεργασία της συγκεκριμένης κατηγορίας τροφίμων τα καθιστά ως τα τρόφιμα με το μεγαλύτερο ρίσκο αλλά και απώλειας κατά την παραγωγή που μπορεί να φτάσουν έως και το 20% [1].Οι αποφάσεις

σχετικά με τον ποιοτικό έλεγχο λαμβάνονται από μια διαδικασία δειγματοληψίας με αμφισβητούμενη ακρίβεια που σπαταλά πολλές εργατοώρες και απαιτεί μεγάλη οικονομική επένδυση σε εργαστηριακά όργανα ανάλυσης. Βασική αιτία για την διενέργεια της παραπάνω διαδικασίας, αποτελεί η μη διαθεσιμότητα της τρέχουσας τεχνολογίας στις βιομηχανίες. Κατά τον παρόντα χρόνο, οι βιομηχανίες επενδύουν σε εργαστηριακό εξοπλισμό και εξειδικευμένο προσωπικό για την δειγματοληψία και την ανάλυση. Ωστόσο η τελευταία δεν πραγματοποιείται σε πραγματικό χρόνο και δεν μπορεί να προβλέψει τους κινδύνους που έπονται στην παραγωγή. Επιπρόσθετα στις βιομηχανίες μεσαίου μεγέθους, η επένδυση σε εργαστηριακό εξοπλισμό και ανθρώπινο δυναμικό για τη χρήση του, δεν αποτελεί βιώσιμη επιλογή. Ως αποτέλεσμα, αναγκάζονται να εργάζονται με απαρχαιωμένες μεθόδους όπως περιοδικούς ελέγχους και καθορισμένη διάρκεια για κάθε στάδιο παραγωγής.

1.2 Σκοπός της παρούσας διπλωματικής

Σκοπός της παρούσας διπλωματικής εργασίας είναι η πρόταση μιας τεχνολογικής λύσης που θα κάνει τον ποιοτικό έλεγχο των ζυμούμενων τροφίμων μια απλή και με ακρίβεια διαδικασία, χωρίς απώλειες στην ποιότητα και στην παραγωγή. Με την κατάργηση της διαδικασίας της δειγματοληψίας επιτυγχάνεται ο περιορισμός των λαθών εξαιτίας του ανθρώπινου παράγοντα αλλά και η αυτοματοποίηση μιας διαδικασίας που χρειάζεται πάρα πολλές εργατοώρες αλλά και εξειδικευμένα όργανα μέτρησης. Η τεχνολογική λύση που παρουσιάζεται είναι μια συσκευή αισθητήρων, πλήρως ασύρματη και plug-n-play, η οποία είναι προσαρμοσμένη στις δεξαμενές αποθήκευσης, συλλέγοντας δεδομένα για την κατάσταση του προϊόντος. Τα δεδομένα ανεβαίνουν σε πραγματικό χρόνο στην υποδομή cloud που δημιουργήθηκε και μέσα από τις διεργασίες που έχουν σχεδιαστεί καταλήγουν στην βάση δεδομένων για αποθήκευση. Όλα αυτά τα δεδομένα είναι διαθέσιμα στο χρήστη μέσω μιας διαδικτυακής εφαρμογής που μπορεί να χρησιμοποιηθεί από οποιαδήποτε συσκευή, με μόνη απαίτηση την σύνδεση στο διαδίκτυο. Μέσω αυτής, ο χρήστης έχει την δυνατότητα παρακολούθησης και οργάνωσης της παραγωγής απομακρυσμένα. Επίσης στο συγκεκριμένο τμήμα της επεξεργασίας τροφίμων, δεδομένα από τα διαφορετικά στάδια της παραγωγής αποθηκεύονται, παρέχοντας τη δυνατότητα

για αξιολόγηση και βελτίωση όλων των σταδίων. Η τεχνολογική λύση που παρουσιάζεται, δοκιμάστηκε σε μορφή πιλοτικού προγράμματος σε μία βιομηχανία τυποποίησης επιτραπέζιας ελιάς και μία βιομηχανία γαλακτοκομικών προϊόντων. Κατά το διάστημα της δοκιμής, μας δόθηκε η ευκαιρία να επικοινωνήσουμε με σημαντικούς ανθρώπους της αγοράς των ζυμούμενων τροφίμων και να συνειδητοποιήσουμε πως η αγορά είναι έτοιμη να υιοθετήσει νέες τεχνολογικές λύσεις για τη βελτιστοποίηση και την αυτοματοποίηση της γραμμής παραγωγής. Η αναγνώριση και κατανόηση της αναγκαιότητας της τεχνολογίας είναι μεγαλύτερη από τα προηγούμενα χρόνια και οι επιχειρήσεις έχουν ήδη εκτεθεί αρκετά σε τεχνολογικές λύσεις υψηλού επιπέδου σε άλλους τομείς της επιχείρησής τους, χωρίς ωστόσο να έχουν ακόμη πραγματοποιηθεί τα αντίστοιχα βήματα για την βελτιστοποίηση του ποιοτικού ελέγχου μέσω της χρήσης προηγμένης τεχνολογίας.

1.3 Οργάνωση του τόμου

Τα υπόλοιπα κεφάλαια της διπλωματικής εργασίας είναι οργανωμένα ως εξής:

- Κεφάλαιο 2: Το συγκεκριμένο κεφάλαιο αποτελεί το θεωρητικό υπόβαθρο της διπλωματικής εργασίας και περιέχει τις απαραίτητες γνώσεις που θα πρέπει να έχει κανείς για να κατανοήσει το περιεχόμενο.
- Κεφάλαιο 3: Το συγκεκριμένο κεφάλαιο αφορά την περιγραφή της υλοποίησης που έγινε σε κάθε στάδιο ξεχωριστά για την δημιουργία του βιομηχανικού πρωτοτύπου.
- Κεφάλαιο 4: Στο συγκεκριμένο κεφάλαιο περιγράφονται οι πιλοτικές δοκιμές που πραγματοποιήθηκαν σε βιομηχανικές συνθήκες, οι δυσκολίες που προέκυψαν κατά τη διαδικασία και οι στόχοι που τέθηκαν για την κάθε δοκιμή.
- Κεφάλαιο 5: Στο συγκεκριμένο κεφάλαιο περιλαμβάνονται τα συμπεράσματα της διπλωματικής εργασίας όπως αυτά προέκυψαν μετά την ανάπτυξη και την δοκιμή του συστήματος στις βιομηχανίες τροφίμων καθώς και οι προτάσεις για μελλοντικές επεκτάσεις της τεχνολογικής μας λύσης.

Κεφάλαιο 2 Υπόβαθρο

2.1 Διαδικασία ζύμωσης

2.1.1 Εισαγωγή

Στα ζυμούμενα τρόφιμα κατά την διάρκεια του σταδίου της αποθήκευσης και της ωρίμανσης λαμβάνει τόπο μια χημική διεργασία που ονομάζεται ζύμωση. Αποτελεί μια από τις παλαιότερες μεθόδους, οι οποίες συμβάλουν στην παραγωγή και επεξεργασία βασικών προϊόντων απαραίτητων για την διατροφή του ανθρώπου. Ειδικότερα, η ζύμωση ορίζεται ως η διαδικασία μετατροπής υδατανθράκων σε αλκοόλ ή οργανικά οξέα χρησιμοποιώντας μικροοργανισμούς - ζύμες ή βακτήρια - υπό αναερόβιες συνθήκες. Στη ζύμωση καθοριστικό ρόλο διαδραματίζει η ελεγχόμενη δράση μικροοργανισμών [4]. Η τεχνολογική λύση που αναπτύχθηκε στην παρούσα διπλωματική εργασία αναφέρεται συγκεκριμένα στην οξυγαλακτική ζύμωση. Η τελευταία αποτελεί μια πολύ ειδική διεργασία, στην οποία για να προκύψουν τα βέλτιστα δυνατά αποτελέσματα, απαιτούνται οι κατάλληλες διορθωτικές ενέργειες, ώστε να διατηρηθούν σταθερές οι παράμετροι που την επηρεάζουν στο εσωτερικό της δεξαμενής.

2.1.2 Παράμετροι που επηρεάζουν την ζύμωση

Η ζύμωση κατά την αποθήκευση και την ωρίμανση του τρόφιμου αποτελεί παραδοσιακά μια αναερόβια διαδικασία. Αυτό σημαίνει πως είναι απαραίτητη η διατήρηση αναερόβιων συνθηκών (απουσία οξυγόνου) εντός των δεξαμενών που τοποθετούνται τα τρόφιμα, με στόχο την αποφυγή ανάπτυξης επιφανειακών μυκήτων που θα επηρεάσουν την υφή και γεύση του προϊόντος. Επιπρόσθετα, η αποστείρωση είναι ένας σημαντικός παράγοντας που πρέπει να ληφθεί υπόψη κατά τη ζύμωση των τροφίμων. Η αποτυχία να αφαιρεθούν εντελώς τυχόν μικρόβια από τον εξοπλισμό και τις δεξαμενές αποθήκευσης, μπορεί να οδηγήσει στον πολλαπλασιασμό επιβλαβών μικροοργανισμών κατά τη ζύμωση, επηρεάζοντας την ποιότητα του τελικού προϊόντος και αυξάνοντας δυνητικά τον κίνδυνο τροφικών ασθενειών [3]. Οι πιο σημαντικές παράμετροι για την επιτυχή ολοκλήρωση της διαδικασίας της ζύμωσης, διακρίνονται σε ενδογενείς και εξωγενείς [5]. Στην πρώτη κατηγορία κατατάσσεται μεταξύ άλλων το pH, ενώ στη δεύτερη κατηγορία, η

θερμοκρασία ζύμωσης και η συγκέντρωση NaCl. Τόσο η θερμοκρασία και το pH, όσο και η περιεκτικότητα άλατος στο εσωτερικό της δεξαμενής αποτελούν τις βασικότερες παραμέτρους, οι οποίες είναι μετρήσιμες με αισθητήρα και στον έλεγχο των οποίων επικεντρώθηκε η παρούσα έρευνα.

2.1.3 Συστήματα ανάλυσης

Τα όργανα ανάλυσης που χρησιμοποιούνται κατά κόρων στην δειγματοληπτική ανάλυση των ζυμούμενων τροφίμων είναι αναλογικά και ψηφιακά φορητά πεχάμετρα. Για να πραγματοποιηθεί η μέτρηση απαιτείται ο υπεύθυνος να λάβει δείγμα από όλες τις δεξαμενές και να πραγματοποιηθεί η ανάλυση μέσω του οργάνου. Η πλειοψηφία των βιομηχανιών ζυμούμενων τροφίμων διαθέτουν ένα πλήρως εξοπλισμένο εργαστήριο με όργανα ανάλυσης όπως πεχάμετρα, θερμόμετρα και άλλα. Εικόνα 2.1. Όλα σχεδόν τα ψηφιακά όργανα ανάλυσης βγάζουν την τιμή της μέτρησης σε μία οθόνη τοπικά. Στη συνέχεια, απαιτείται η καταγραφή αυτής σε κάποιο αρχείο χειροκίνητα. Τα αναλογικά όργανα δεν έχουν μεγάλη ακρίβεια στην τιμή μέτρησης, δεδομένο που πιστοποιείται από τις αποκλίσεις που προκύπτουν ανάμεσα σε τιμές αναλογικών και ψηφιακών οργάνων σε δείγματα που έχουν ληφθεί από την ίδια δεξαμενή. Όπως προκύπτει, ολόκληρη η διαδικασία της αποθήκευσης και ζύμωσης των προϊόντων έχει σημαντικό ρίσκο και σε μεγάλο βαθμό εξαρτάται από την τιμή των οργάνων ανάλυσης κατά την δειγματοληψία.

Εικόνα 2.1: Φορητό & Εργαστηριακό ψηφιακό πεχάμετρο [6] [7]

2.2 Επισκόπηση της πλατφόρμας Raspberry

2.2.1 Εισαγωγή

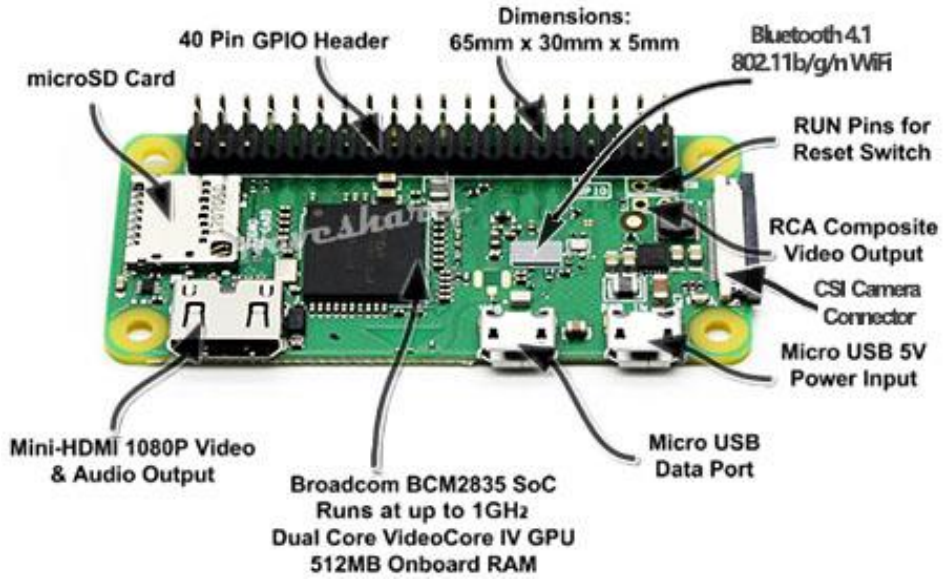
Για την ανάπτυξη του βιομηχανικού πρωτοτύπου, που μελετήθηκε στην παρούσα διπλωματική εργασία, στο κομμάτι του υλικού και της επεξεργαστικής μονάδας χρησιμοποιήθηκε ο μικροελεγκτής Raspberry Pi Zero W (Εικόνα 2.2). Η επιλογή του συγκεκριμένου μικροελεγκτή έγινε καθώς έχει πολλά modules ήδη εγκατεστημένα όπως το WiFi, SD card slot για την διαχείριση της μνήμης και άλλα. Επίσης δίνει τη δυνατότητα εγκατάστασης λειτουργικού συστήματος που συμβάλει στην άμεση ανάπτυξη του πρωτοτύπου αλλά και στην υποστήριξη διαφόρων πρωτοκόλλων επικοινωνίας με πλήθος αισθητήρων που υπάρχουν διαθέσιμοι στο εμπόριο [8].

Εικόνα 2.2: Raspberry Pi [8]

2.2.2 Τεχνικά χαρακτηριστικά του μικροελεγκτή Raspberry Pi Zero W.

Το Raspberry Pi Zero W (Εικόνα 2.3) αποτελείται από το SoC Broadcom BCM2835 και ο επεξεργαστής του που είναι βασισμένος σε αρχιτεκτονική ARM είναι ο ARM1176JZF-S, σε συχνότητα επεξεργαστή 1 GHz. Το βασικότερο τεχνολογικό χαρακτηριστικό για το Raspberry Pi Zero W είναι το 2.4 GHz 802.11n WiFi μέσω του chip Broadcom BCM43438 που δίνει τη δυνατότητα σύνδεσης στο διαδίκτυο και Bluetooth 4.1. Η μνήμη RAM του

Raspberry Pi Zero W είναι 512MB ενώ για την αποθήκευση των δεδομένων, γίνεται χρήση SD card. Τα λειτουργικά συστήματα που χρησιμοποιούνται στο Raspberry Pi έχουν ως βάση το Linux Kernel και ειδικότερα το πιο διαδεδομένο είναι το Raspbian OS. Ένα ακόμη τεχνικό χαρακτηριστικό είναι το 40-pin GPIO που μέσα από κώδικα δίνεται η δυνατότητα για χειρισμό τους ως ακίδες εισόδου ή εξόδου με πάρα πολλές εφαρμογές [9].



Εικόνα 2.3: Τεχνικά χαρακτηριστικά του Raspberry Pi [10]

2.2.3 Ανάπτυξη εφαρμογών στον μικροελεγκτή Raspberry Pi Zero W.

Η ανάπτυξη εφαρμογών στον μικροελεγκτή Raspberry Pi Zero W, ενδείκνυται για την δημιουργία του πρωτότυπου συστήματος σε γρήγορο χρόνο αλλά όχι για μαζική παραγωγή. Αυτό γίνεται γιατί είναι ένα από τα πιο διαδεδομένα boards και υπάρχουν πάρα πολλές πηγές αλλά και άρθρα στο internet που βοηθάνε στην άμεση ανάπτυξη και το οικοσύστημα που έχει δημιουργηθεί γύρω από αυτό είναι τεράστιο και βρίσκεις βοήθεια σε ότι πρόβλημα παρουσιαστεί. Επίσης η παρουσία λειτουργικού, με ή χωρίς το γραφικό περιβάλλον, αλλά και ότι διαθέτει πολλά πρωτόκολλα επικοινωνίας όπως I2C, UART κλπ. κάνει πολλούς αισθητήρες της αγοράς να προσαρμόζονται άμεσα σε αυτό. Υπάρχουν πολλά διαθέσιμα πακέτα μέσω του Advance Package Tool (apt) που δίνουν την δυνατότητα να εγκαταστήσεις και να χρησιμοποιήσεις διάφορα εργαλεία ανάπτυξης. Λόγω του λειτουργικού συστήματος που υποστηρίζει μπορεί να γίνει ανάπτυξη εφαρμογών σε οποιαδήποτε γλώσσα προγραμματισμού, ο σκοπός της χρήσης του μικροελεγκτή στην συγκεκριμένη εργασία είναι η δημιουργία των services που αποτελούν

τα δύο κύρια κομμάτια της τεχνολογικής λύσης μας, το Sensing Unit Control και το IoT Unit Control, τα οποία είναι υπεύθυνα για την συλλογή δεδομένων από τους αισθητήρες και την προώθηση των δεδομένων στο cloud αντίστοιχα.

2.3 Υποδομή Cloud

2.3.1 Εισαγωγή

Η υποδομή του cloud, για την ανάπτυξη του βιομηχανικού πρωτοτύπου, βασίστηκε στο Google Cloud Platform (Εικόνα 2.4) [11]. Η συγκεκριμένη υποδομή δίνεται από την Google και είναι μια σουίτα υπηρεσιών cloud computing που εκτελείται στην ίδια υποδομή που χρησιμοποιεί η εταιρία για τα εσωτερικά προϊόντα της, όπως είναι το Google Search, Gmail κτλ.

Εικόνα 2.4: Google Cloud Platform [12]

2.3.2 Τεχνικά χαρακτηριστικά του Google Cloud Platform

Η υποδομή του Google Cloud Platform προσφέρει ένα σύνολο από υπηρεσίες που βοηθάνε στην ανάπτυξη και στην δοκιμή των χαρακτηριστικών που περιλαμβάνει το κομμάτι της cloud υποδομής που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας. Όλη η διαχείριση των εφαρμογών που τρέχουν σε αυτό γίνεται από το κύριο application panel. Μέσα από αυτό μπορείς να παρακολουθήσεις απομακρυσμένα την κατάσταση των services αλλά και τα αρχεία καταγραφής για τυχόν σφάλματα που προκύπτουν κατά την λειτουργία [11]. Επίσης είναι ένα εργαλείο που προσφέρει την δυνατότητα να κάνεις scale up στο προϊόν που έχεις αναπτύξει γιατί οι υπηρεσίες του είναι ενιαίες και δεν υπάρχουν

διαχωρισμοί με βάση την πολυπλοκότητα του project. Μερικά από τα εργαλεία που διαθέτει είναι:

- **Compute Engine**
  Virtual Machines με προκαθορισμένο ή προσαρμοσμένο μέγεθος ανάλογα τις απαιτήσεις που έχει το project.

- **Cloud Storage**
  Αποθήκευση αντικειμένων που είναι ασφαλής, ανθεκτική και επεκτάσιμη.

- **Cloud SDK**
  Εργαλεία και βιβλιοθήκες για αλληλεπίδραση με προϊόντα και υπηρεσίες Google Cloud.

- **Pub/Sub**
  Υπηρεσία ανταλλαγής μηνυμάτων που βασίζεται στην λογική του event-driven.

- **IoT Core**
  Πλήρως διαχειριζόμενη υπηρεσία για την ασφαλή σύνδεση και την διαχείριση IoT συσκευών.

2.3.3 Ανάπτυξη εφαρμογών στο Google Cloud Platform

Η ανάπτυξη εφαρμογών στον Google Cloud Platform μπορεί να γίνει με διάφορους τρόπους ανάλογα με την προτίμηση και την εξοικείωση του καθενός. Τα περισσότερα εργαλεία υποστηρίζουν στην πλειοψηφία τους τις περισσότερες από τις διαδεδομένες γλώσσες προγραμματισμού και μέσα από το documentation που παρέχει μπορείς άμεσα να προσαρμόσεις τα κομμάτια κώδικα που απαιτεί το καθένα για να τα χρησιμοποιήσεις. Επίσης για την αλληλεπίδραση με την πλατφόρμα του Gcloud μπορεί να γίνει με δύο τρόπους. Αρχικά μέσα από το UI που περιλαμβάνει η web εφαρμογή τους αλλά μπορείς να εγκαταστήσεις στο μηχάνημα που εργάζεσαι το Google SDK και να τρέξεις στο τερματικό τις αντίστοιχες εντολές. Όσον αφορά την συντήρηση της εφαρμογής που έχεις δημιουργήσει σε αυτό, μέσα από operation εργαλεία που διαθέτει μπορείς να έχεις απομακρυσμένη παρακολούθηση των instances που τρέχουν σε αυτό αλλά και να λαμβάνεις ειδοποιήσεις σε περίπτωση που κάτι πάει στραβά.

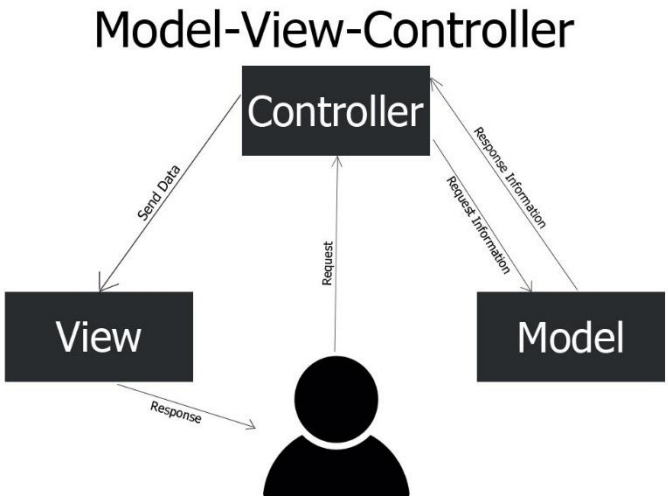
2.4 Επισκόπηση της Web εφαρμογής

2.4.1 Εισαγωγή

Για την υλοποίηση της διαδικτυακής εφαρμογής χρησιμοποιήθηκε το Django, ένα framework ανοιχτού-κώδικα για την ανάπτυξη διαδικτυακών εφαρμογών, το οποίο είναι βασισμένο στη γλώσσα προγραμματισμού Python [13]. Το framework αποτελεί ένα δομημένο σύνολο κανόνων, πρακτικών και κριτηρίων, το οποίο προορίζεται να παρέχει υποστήριξη και καθοδήγηση για την διαχείριση σύνθετων προβλημάτων, παρέχοντας τη δυνατότητα επαναχρησιμοποίησης κώδικα. Το Django Web Framework δημιουργήθηκε το φθινόπωρο του 2003, όταν οι προγραμματιστές στην εφημερίδα Lawrence Journal-World, Adrian Holovaty και Simon Willison, άρχισαν να χρησιμοποιούν την Python για την κατασκευή εφαρμογών [14]. Έχει ως κύριο γνώμονα την παροχή ευελιξίας στο χρήστη για τον σχεδιασμό υψηλής απόδοσης διαδικτυακών εφαρμογών σε σύντομο χρονικό διάστημα σε ένα σταθερό και ασφαλές περιβάλλον.

2.4.2 Τεχνικά χαρακτηριστικά του Django Web Framework

Το Django Framework βασίζεται στην αρχιτεκτονική MVC (Model, View, Controller). Εικόνα 2.5. Ειδικότερα, το «Model», περιλαμβάνει όλα όσα σχετίζονται με τη βάση δεδομένων και πιο συγκεκριμένα τον τρόπο απόκτησης πρόσβασης στα δεδομένα, τη συμπεριφορά αυτών και τις σχέσεις μεταξύ τους. Το «View» περιλαμβάνει την επιλογή των δεδομένων από το «Model» και τον τρόπο απεικόνισής τους, μέσω του «Template». Τέλος, ο «Controller» καλεί την κατάλληλη συνάρτηση Python για τη δεδομένη διεύθυνση URL, ανάλογα με το αίτημα του χρήστη, αποτελώντας τη γέφυρα μεταξύ των «Model» και «View».



Εικόνα 2.5: MVC concept [15]

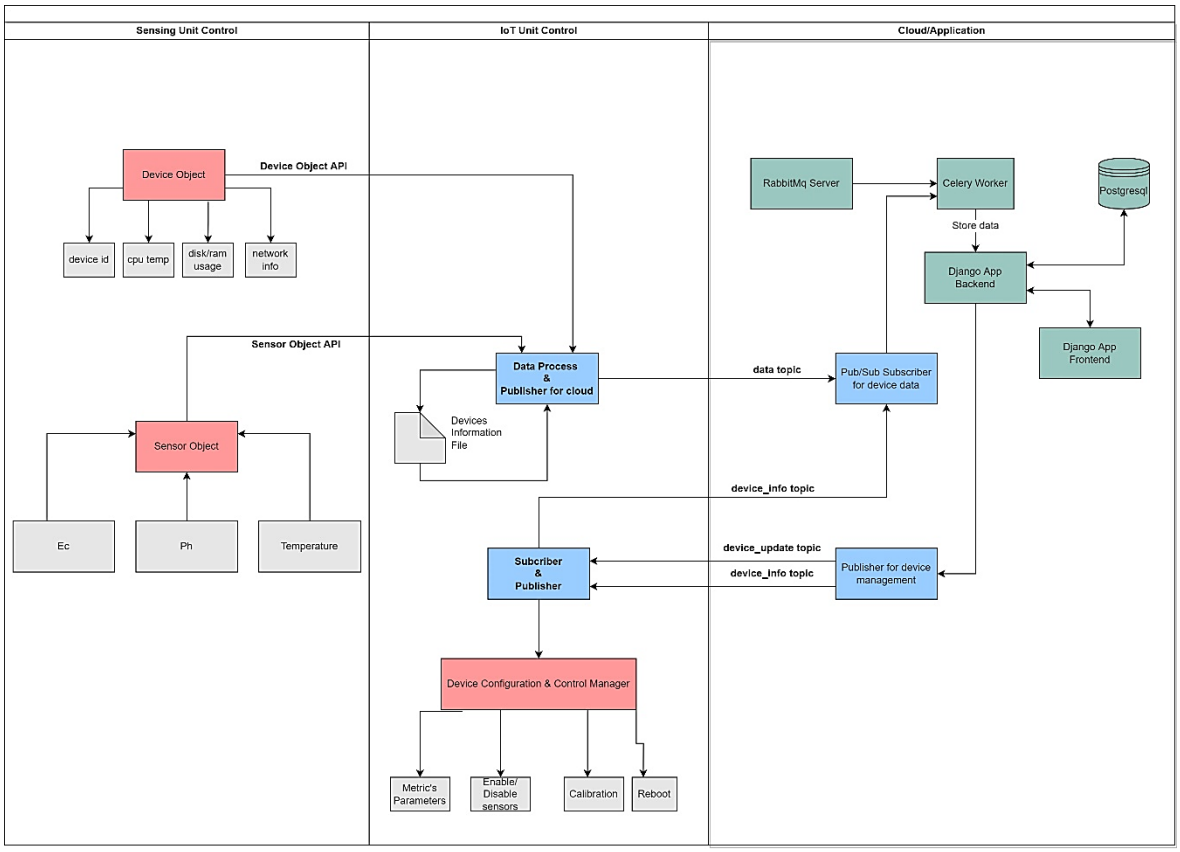
Συνολικά, το MVC είναι ένας τρόπος ανάπτυξης λογισμικού έτσι ώστε ο κώδικας για τον καθορισμό και την πρόσβαση σε δεδομένα (Model) να είναι ξεχωριστός από τη λογική δρομολόγησης αιτήματος (Controller), ο οποίος με τη σειρά του είναι ξεχωριστός από τη διεπαφή χρήστη (View). Αυτό συνεπάγεται ότι ο προγραμματιστής μπορεί να αλλάξει τη διεύθυνση URL για ένα δεδομένο τμήμα της εφαρμογής χωρίς να επηρεάσει την υποκείμενη υλοποίηση. Αντίστοιχα, ένας σχεδιαστής μπορεί να αλλάξει την HTML μιας σελίδας χωρίς να χρειάζεται να επεξεργαστεί τον κώδικα Python που την αποδίδει (render).

Κεφάλαιο 3 Υλοποίηση του βιομηχανικού πρωτοτύπου

3.1 Εισαγωγή

Το πρωτότυπο που αναπτύχθηκε στην παρούσα διπλωματική εργασία αποτελείται από ένα φυσικό σύστημα και την υποδομή του cloud. Πιο αναλυτικά, στο φυσικό σύστημα που αποτελείται από ένα μικροελεγκτή Raspberry Pi, φιλοξενούνται οι δύο βασικές διεργασίες. Το “Sensing Unit Control”, το οποίο περιλαμβάνει το κομμάτι της συλλογής των μετρήσεων από τους αισθητήρες με τα κατάλληλα πρωτόκολλα επικοινωνίας καθώς και το κομμάτι της διαχείρισης της κατάστασης της συσκευής σε ότι αφορά μετρήσεις όπως η θερμοκρασία επεξεργαστή, οι πληροφορίες συνδεσιμότητας, η χρήση RAM και δίσκου και άλλες παράμετροι. Επιπλέον το “IoT Unit Control”, το οποίο περιλαμβάνει το κομμάτι της ασφαλούς επικοινωνίας με την cloud υποδομή και την απομακρυσμένη διαχείριση της συσκευής με αμφίδρομη επικοινωνία. Με τον τρόπο αυτό, πραγματοποιείται η παραμετροποίηση της συσκευής και η εκτέλεση βασικών λειτουργιών της, όπως είναι η επανεκκίνηση ή ο τερματισμός της.

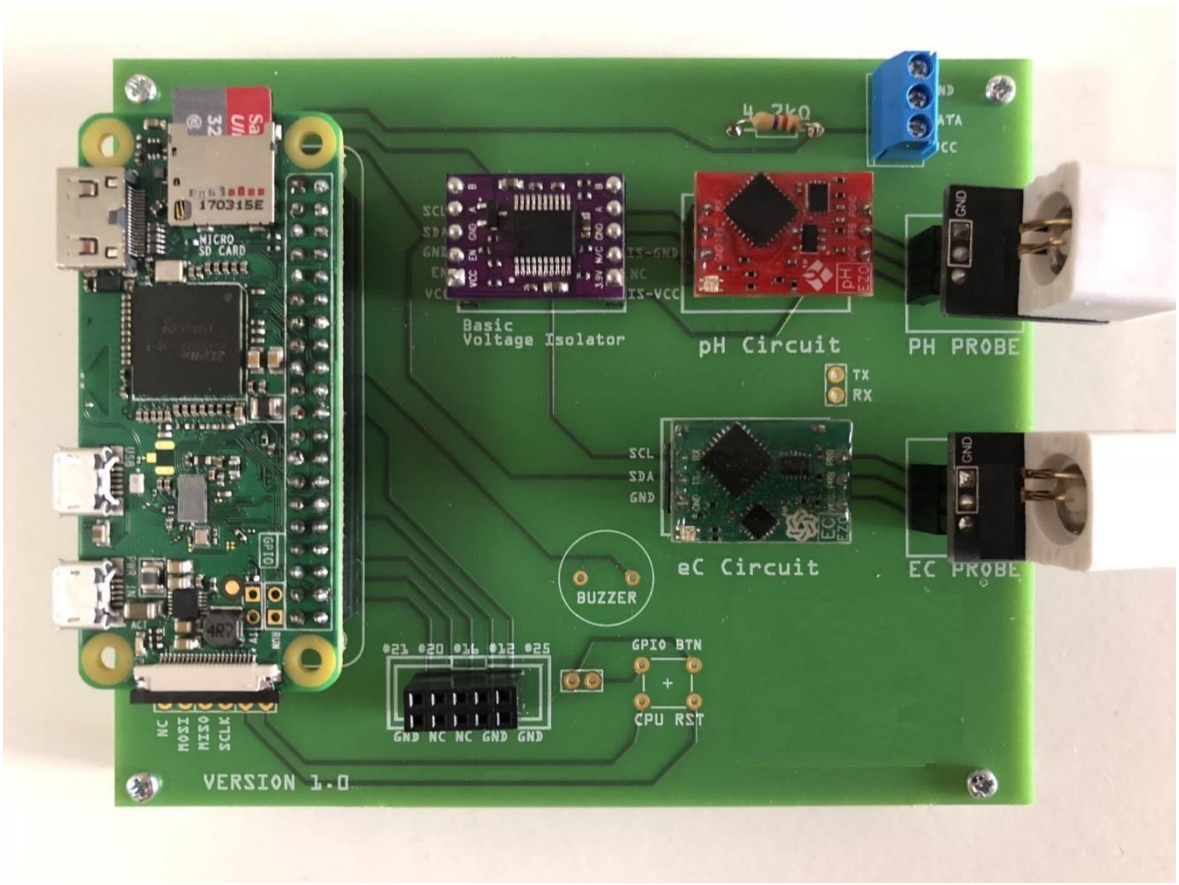
Όσον αφορά στην υποδομή του cloud, αναπτύχθηκαν διεργασίες που συμβάλλουν στη συλλογή και την αποθήκευση των δεδομένων στην SQL βάση, στη διαχείριση και την παραμετροποίηση των συσκευών και τέλος στη λειτουργία της web based εφαρμογής του τελικού χρήστη. Η web εφαρμογή δίνει την δυνατότητα άμεσης προβολής των δεδομένων που συλλέγονται με την μορφή διαγραμμάτων. Για την ανάπτυξη και την υλοποίηση του λογισμικού σε όλο το εύρος του βιομηχανικού πρωτοτύπου χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python σε συνδυασμό με πολλά βασικά πακέτα που προσφέρει. Τα πρωτόκολλα επικοινωνίας που χρησιμοποιήθηκαν σε όλα τα επίπεδα είναι το I2C, WiFi, MQTT, HTTP [16]. Επίσης για τον προγραμματισμό των διεργασιών που τρέχουν σε επίπεδο λειτουργικού χρησιμοποιήθηκε bash scripting [17].



Εικόνα 3.1: Component's & Functionality diagram

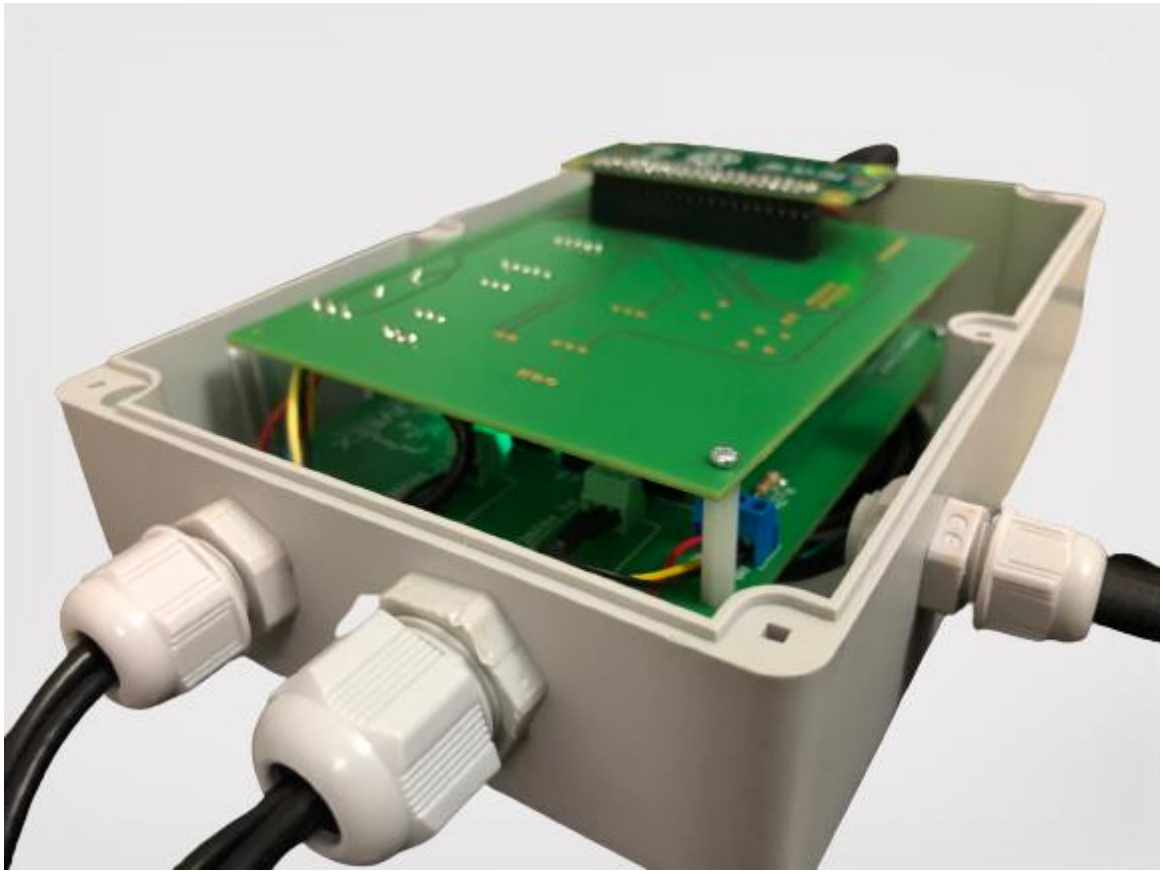
3.1.1 Περιγραφή της συσκευής ανίχνευσης

Το hardware της συσκευής ανίχνευσης αποτελείται από ένα Raspberry Pi Zero W και την πλακέτα, η οποία υλοποιεί το απαιτούμενο κύκλωμα για την σύνδεση του μικροελεγκτή με τους αισθητήρες που επιλέχθηκαν για τις ανάγκες του πρωτοτύπου Εικόνα 3.2. Το λογισμικό που αναλαμβάνει την λειτουργικότητα και την συνδεσιμότητα της συσκευής ανίχνευσης είναι Raspbian Lite, το οποίο είναι βασισμένο στο Unix. Η επιλογή του συγκεκριμένου μικροελεγκτή έγινε λόγω της δυνατότητας ενσωμάτωσης λειτουργικού που παρέχει. Αυτό εξυπηρετεί στη διαχείριση της συσκευής από το ενσωματωμένο τερματικό εργαλείο, στην άμεση δοκιμή της υλοποίησης, στην αλληλεπίδραση με τα περιφερειακά του Raspberry όπως το WiFi αλλά και στην ανάπτυξη και συντήρηση της υλοποίησης απομακρυσμένα. Για τη συνδεσιμότητα της συσκευής στο internet και την ανταλλαγή πακέτων με την cloud υποδομή απαιτείται η ύπαρξη WiFi δικτύου.



Εικόνα 3.2: Διάταξη μικροελεγκτή και αισθητήρων

Αναφορικά με την τροφοδοσία ρεύματος, η συσκευή ανίχνευσης απαιτεί την ύπαρξη τροφοδοτικού για το Raspberry μέσω micro-USB και την σύνδεση της στην πρίζα. Το κέλυφος που χρησιμοποιήθηκε για την συσκευή αισθητήρων, προσφέρει στεγανοποίηση, η οποία επιτρέπει την προσαρμογή της σε εσωτερικούς ή εξωτερικούς χώρους με υψηλή υγρασία, θερμοκρασία καθώς και άλλες περιβαλλοντικές συνθήκες. Εικόνα 3.3. Επίσης το κέλυφος διαθέτει τις κατάλληλες αναμονές για την στήριξή του στις δεξαμενές αποθήκευσης τροφίμων.

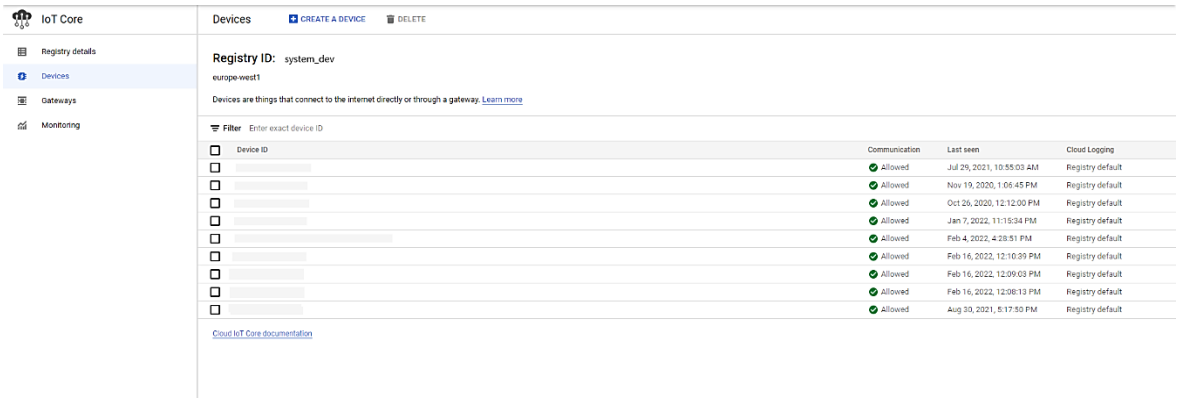


Εικόνα 3.3: Κέλυφος βιομηχανικού πρωτοτύπου

3.1.2 Περιγραφή της πλατφόρμας Internet of Things

Η πλατφόρμα Internet of Things που υλοποιήθηκε, έχει ως στόχο την προβολή των δεδομένων από τις συσκευές μετρήσεων στον τελικό χρήστη. Για να επιτευχθεί αυτό χρειάστηκε να δημιουργηθεί μία διεργασία η οποία θα επιτρέπει τη συλλογή των δεδομένων που παράγουν οι συσκευές ανίχνευσης. Προϋπόθεση για την δημιουργία της ήταν να είναι deployed σε κάποιο εικονικό μηχάνημα της cloud υποδομής και διαθέσιμη 24 ώρες/ημέρα, 7 ημέρες/βδομάδα. Επίσης για την αποθήκευση των δεδομένων και την προσπέλασή τους απομακρυσμένα, χρειάστηκε να δημιουργηθεί και να φιλοξενηθεί ένα εικονικό μηχάνημα στο cloud και μια PostgreSQL βάση δεδομένων [18]. Τα παραπάνω οδήγησαν στη δημιουργία μιας web based εφαρμογής με την χρήση του Django Framework, η οποία έγινε deploy με τη χρήση του Apache διακομιστή, έτσι ώστε να είναι προσβάσιμη από οποιαδήποτε υπολογιστή μέσω διαδικτύου [19].

Στο backend κομμάτι της IoT πλατφόρμας χρησιμοποιήθηκαν οι υπηρεσίες του Google Cloud. Συγκεκριμένα για το deploy των εικονικών μηχανών, οι οποίες φιλοξενούν τις διεργασίες για τη συλλογή δεδομένων, τη βάση δεδομένων και τον διακομιστή της web εφαρμογής χρησιμοποιήθηκε το εργαλείο «Compute Engine». Επιπλέον στο επίπεδο των τελικών συσκευών ανίχνευσης, για την παρακολούθηση, την παραμετροποίηση και την ασφαλή προώθηση των δεδομένων που συλλέγουν, χρησιμοποιήθηκε το εργαλείο «IoT Core» (Εικόνα 3.4). Το τελευταίο χρησιμοποιεί ως επέκταση και το «Pub/Sub» για την δημιουργία τον καναλιών επικοινωνίας και αποστολής δεδομένων.



Εικόνα 3.4: IoT Core main panel

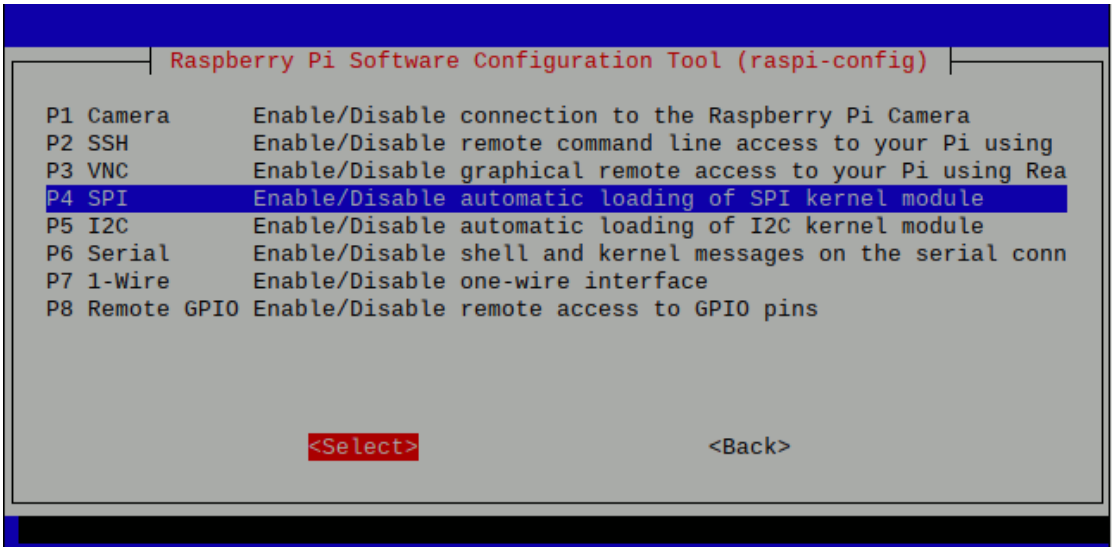
3.2 Συσκευή ανίχνευσης

Ένα από τα βασικά μέρη της παρούσας διπλωματικής αποτελεί η συσκευή ανίχνευσης, η οποία προσαρμόζεται στις υπάρχουσες δεξαμενές αποθήκευσης και ωρίμανσης τροφίμων και μέσω αυτής γίνεται σε πραγματικό χρόνο η δειγματοληψία που αποτελεί τον ποιοτικό έλεγχο για τις συγκεκριμένες βιομηχανίες. Η συσκευή είναι υπεύθυνη για τις μετρήσεις μέσω των αισθητήρων και την συλλογή των δεδομένων σχετικά με την ποιοτική κατάσταση του τροφίμου καθώς και την κατάσταση της ίδιας της συσκευής. Οι μετρήσεις που συλλέγει από τους περιφερειακούς αισθητήρες αφορούν τιμές θερμοκρασίας, pH και ηλεκτρικής αγωγιμότητας. Επίσης λόγω των περιβαλλοντικών συνθηκών που επικρατούν στις συγκεκριμένες βιομηχανίες αλλά και της σημαντικότητας του συστήματος στην παραγωγή, σημαντική είναι και παρακολούθηση τιμών από τους αισθητήρες που φέρει και ο ίδιος ο μικροελεγκτής όπως είναι η θερμοκρασία της CPU, η

διαθέσιμη μνήμη αποθήκευσης και η σύνδεση στο διαδίκτυο. Τέλος, πέραν της συλλογής των δεδομένων, μία άλλη σημαντική διεργασία της συγκεκριμένης συσκευής είναι η συνδεσιμότητα με την cloud υποδομή και τις λειτουργίες που αυτή προσφέρει.

3.2.1 Μονάδα διαχείρισης των αισθητήρων “Sensing Unit Control”

Η μονάδα διαχείρισης των αισθητήρων αποτελεί μια από τις δύο βασικές διεργασίες που φιλοξενείται στη συσκευή αισθητήρων. Για την ανάπτυξή της χρησιμοποιήθηκε εξολοκλήρου η γλώσσα προγραμματισμού Python και αποτελείται από δύο βασικά APIs. Το «Device Object» API αφορά τα δεδομένα που προέρχονται από το ίδιο το Raspberry και το «Sensor Object» API τα δεδομένα από τους περιφερειακούς αισθητήρες. Τα δύο αυτά APIs λειτουργούν ως δύο ανεξάρτητες διεργασίες. Για το «Sensor Object» που αφορά την αλληλεπίδραση με τους περιφερειακούς αισθητήρες χρειάστηκε να ενεργοποιηθούν τα απαραίτητα πρωτόκολλα επικοινωνίας από το μενού διαμόρφωσης του Raspberry με την χρήση της εντολής “raspi-config” (Εικόνα 3.5).



Εικόνα 3.5: Περιβάλλον διαμόρφωσης του Raspberry [20]

Στη διαδικασία της εγκατάστασης για την συσκευή ανίχνευσης χρειάστηκε να γίνει εγκατάσταση πακέτων που αφορούν τις λειτουργίες των πρωτοκόλλων I2C, SPI, με την χρήση του εργαλείου “apt” [21] .

Για τις δύο διεργασίες, που είναι βασισμένες στην γλώσσα προγραμματισμού Python, χρειάστηκε να δημιουργηθεί ένα εικονικό περιβάλλον ξεχωριστά για την κάθε μια. Αυτό έγινε για να έχει κάθε μια διεργασία το δικό της απομονωμένο περιβάλλον Python στο οποίο εγκαταστάθηκαν τα απαραίτητα πακέτα και εξαρτήσεις, για την σωστή λειτουργία

των διεργασιών, ανεξάρτητα από τον προεπιλεγμένο κατάλογο που είναι εγκατεστημένα τα πακέτα της Python του συστήματος [22].

3.2.1.1 Συλλογή δεδομένων από τους αισθητήρες

Η διεργασία για την συλλογή των δεδομένων από τους αισθητήρες αποτελείται από ένα ολοκληρωμένο API που υλοποιήθηκε για να μπορεί να επικοινωνήσει ο μικροελεγκτής με τους περιφερειακούς αισθητήρες και να μεταφράσει τις τιμές που του επιστρέφουν οι μετρήσεις όπως είναι το pH, η θερμοκρασία και η ηλεκτρική αγωγιμότητα. Ένα βασικό μέρος της υλοποίησης περιλαμβάνει τις διαδικασίες βαθμονόμησης των αισθητήρων, τον έλεγχο και δοκιμή των αισθητήρων καθώς και την διόρθωση των τιμών που συλλέχτηκαν για να αποκλειστούν οι ακραίες τιμές.

Για την επικοινωνία με τους αισθητήρες δημιουργήθηκε μία βιβλιοθήκη (python package) όπου σε αυτό υλοποιούνται οι ακόλουθες συναρτήσεις:

- **set_i2c_address()**: Τοποθετεί το I2C κανάλι επικοινωνίας σε μια συγκεκριμένη διεύθυνση bus.
- **read()**: Διαβάζει ένα συγκεκριμένο αριθμό bytes από το I2C, τα επεξεργάζεται και προβάλει τα αποτελέσματα.
- **write()**: Δημιουργεί μία συμβολοσειρά και την στέλνει χρησιμοποιώντας το I2C κανάλι.
- **close()**: Κλείνει τα αρχεία που χρησιμοποιεί η read() και η write().

Η συγκεκριμένη διεπαφή συμπεριλαμβάνεται ανεξάρτητα σε κάθε κλάση αισθητήρα που υλοποιείται. Οι αισθητήρες που το χρησιμοποιούν είναι αυτοί του pH και της ηλεκτρικής αγωγιμότητας που χρειάζονται το συγκεκριμένο πρωτόκολλο επικοινωνίας και ο καθένας εξυπηρετείται από διαφορετικό κανάλι I2C που στην προκειμένη περίπτωση πρόκειται για τις διευθύνσεις 99,100 . Η λειτουργικότητα της κλάσης των αισθητήρων αποτελείται από τις συναρτήσεις:

- **wakeUp()**: Αρχικοποιεί τον αισθητήρα και το θέτει σε κατάσταση λήψης τιμών.
- **getMetric()**: Κυρίως διεργασία η οποία κάθε φορά καλεί την wakeUp() στην αρχή, διαβάζει τιμές χρησιμοποιώντας την συνάρτηση read() του αισθητήρα. Αποθηκεύει τις δέκα τιμές που συλλέγει σε ένα πίνακα και κάνει στατιστική διόρθωση για να επιστρέψει μια τελική τιμή. Τέλος στέλνει μια εντολή “SLEEP” για να βάλει σε λειτουργία εξοικονόμησης ενέργειας τον αισθητήρα για να πετυχαίνουμε καλύτερη αποδοτικότητα ισχύος.
- **getStatus()**: Επιστρέφει την τρέχουσα κατάσταση του αισθητήρα. Αν είναι σε ενεργή κατάσταση ή σε εξοικονόμησης ενέργειας και αν υπάρχει κάποιος κωδικός σφάλματος κατά την λήψη των μετρήσεων.

Για τον αισθητήρα θερμοκρασίας χρησιμοποιήθηκε το πρωτόκολλο επικοινωνίας 1-Wire [23] . Η λειτουργικότητα της κλάσης του συγκριμένου αισθητήρα περιγράφεται από τις ακόλουθες συναρτήσεις:

- **init()**: Αρχικοποιεί τις απαραίτητες παραμέτρους που χρειάζεται το συγκεκριμένο πρωτόκολλο και το κανάλι ανάγνωσης για τις τιμές του αισθητήρα.
- **getTemperature()**: Κυρίως διεργασία που καλεί την init(), στην συνέχεια παίρνει δέκα τιμές από τον αισθητήρα τις αποθηκεύει σε ένα πίνακα, κάνει στατιστική διόρθωση τιμών και επιστρέφει την τρέχουσα θερμοκρασία.

Η συγκριμένη κλάση αποτελεί την τελευταία από τους αισθητήρες που χρησιμοποιεί το πακέτο που δημιουργήθηκε για την μονάδα διαχείρισης των αισθητήρων.

Οι υπόλοιπες κλάσεις που υλοποιήθηκαν για το συγκεκριμένο πακέτο είναι βοηθητικές που έχουν να κάνουν με διεργασίες όπως είναι η βαθμονόμηση και τα αρχεία ελέγχου και δοκιμής για το «Sensor Object» API. Η λειτουργικότητα αρχικά για την διεργασία της βαθμονόμησης αφορά μόνο τους αισθητήρες που χρησιμοποιούν το I2C πρωτόκολλο και περιγράφεται από μια συνάρτηση που καλεί την wakeUp() και στην συνέχεια συνεχόμενα καλείται η read() του αισθητήρα σε ένα loop με μια μικρή καθυστέρηση σε κάθε επανάληψη.

Επίσης αναπτύχθηκε μια διεπαφή που προσομοιώνει τις κλάσεις κάθε αισθητήρα και δημιουργεί εικονικά δεδομένα βασισμένα στο εύρος των πραγματικών τιμών. Αυτό έγινε και βοήθησε στην απομακρυσμένη ανάπτυξη και στην δοκιμή ολόκληρης της πλατφόρμας χωρίς την προϋπόθεση να υπήρχε το φυσικό κομμάτι υλικού του αισθητήρα.

3.2.1.2 Συλλογή δεδομένων συσκευής

Η δεύτερη σημαντική διεργασία της συσκευής ανίχνευσης αποτελείται από την διεργασία συλλογής δεδομένων από την συσκευή του Raspberry. Συγκεκριμένα οι τιμές που συλλέγονται από την συσκευή έχουν να κάνουν:

- Τροφοδοσία
- CPU
- Δίσκος (SD card)
- Μνήμη
- Network
- Μνήμη Swap

Για να καταφέρουμε να συλλέξουμε τις συγκεκριμένες παραμέτρους δημιουργήσαμε μια διεπαφή η οποία αποτελείται από ένα αντικείμενο που το ονομάζουμε “Device_Info” το οποίο περιέχει τις παραμέτρους σαν ξεχωριστές μεταβλητές και με τις κατάλληλες

συναρτήσεις γίνεται η αρχικοποίηση, παραμετροποίηση και η επιλογή κάθε μιας μεταβλητής ξεχωριστά.

Οι τιμές που αντλούνται από την ίδια την συσκευή υλοποιούνται κάθε μία ξεχωριστά από μία python συνάρτηση η οποία εκτελεί εντολές του συστήματος κάνοντας χρήση του πακέτου της python “subprocess” [24]. Κάθε κλήση για τις τιμές των αισθητήρων μέσα από την διεπαφή γίνεται ως μια υποδιεργασία της κεντρικής και το αποτέλεσμα της εντολής με την κατάλληλη επεξεργασία αποθηκεύει τις τιμές στις μεταβλητές της διεπαφής. Η πληροφορία από τους αισθητήρες συλλέγεται και τοποθετείται σε ένα αντικείμενο json από την κύρια διεργασία της συγκεκριμένης διεπαφής και προωθείται, μαζί με τα αναγνωριστικά της συσκευής για να μπορούν να αναγνωρίζονται τα δεδομένα, στην διεργασία που είναι υπεύθυνη για την συνδεσιμότητα με την cloud υπηρεσία που υλοποιήθηκε.

3.2.2 Μονάδα διαχείρισης συνδεσιμότητας με το Cloud “IoT Unit Control”

Στην συσκευή ανίχνευσης πέρα από την συλλογή των δεδομένων από τους αισθητήρες υλοποιήθηκε και το τμήμα που επεξεργάζεται την συνδεσιμότητα των συσκευών με την Cloud υποδομή μας και την προώθηση των δεδομένων σε αυτή. Επίσης μέσα σε αυτό συμπεριλαμβάνεται και η αμφίδρομη επικοινωνία για την παραμετροποίηση και τον έλεγχο της συσκευής. Για την διαχείριση της συνδεσιμότητας και της επικοινωνίας χρησιμοποιήθηκε ένας IoT Core client που προσφέρει μια αμφίδρομη επικοινωνία, την δυνατότητα απομακρυσμένης παραμετροποίησης αλλά και την ασφάλεια που προσφέρει η πλατφόρμα του Google Cloud για την μεταφορά των δεδομένων. Δημιουργήθηκε μια διεπαφή που περιλαμβάνει όλες αυτές τις λειτουργίες που η κάθε μια τρέχει ξεχωριστά σαν μία διεργασία από την γονική διεργασία. Η γονική διεργασία δημιουργεί τα νήματα για κάθε μια, συλλέγει τα αρχεία καταγραφής και επιτηρεί την κατάστασή τους.

3.2.2.1 Επεξεργασία και προώθηση δεδομένων

Για την συνδεσιμότητα της συσκευής και την προώθηση των δεδομένων στο cloud μέσω του IoT Core χρειάστηκε να χρησιμοποιήσουμε έναν MQTT client με την χρήση της βιβλιοθήκης Eclipse Paho MQTT Python client [25].

Δημιουργήσαμε μία διεπαφή η οποία δημιουργεί και αρχικοποιεί την κλάση “IoT Client” που περιέχει τον συγκεκριμένο client και στην συνέχεια κάθε μία διεργασία που ξεκινάει τον κληρονομεί από την κεντρική διεπαφή. Στην συγκεκριμένη κλάση ορίζονται και όλες οι συναρτήσεις που χρησιμοποιούνται για την αρχικοποίησή του αλλά και την ενημέρωσή του σε περίπτωση που προκύψει κάποιο σφάλμα ή λήξει το κλειδί ασφαλείας του.

Συγκεκριμένα οι συναρτήσεις του client που υλοποιούνται στην κλάση και χρησιμοποιούνται από τις υπόλοιπες διεργασίες είναι οι παρακάτω:

- **create_client(args)**: Αρχικοποίηση και δημιουργία του client. Εικόνα 3.6.

```
def create_client(self):

    self.jwt_iat = datetime.datetime.utcnow()
    self.client = get_client(
        self.project_id, self.cloud_region,
        self.registry_id, self.device_id, self.private_key_file,
        self.algorithm, self.ca_certs, self.mqtt_bridge_hostname,
        self.mqtt_bridge_port)
    self.client.loop_start()
```

Εικόνα 3.6: IoT Client-create

- **update_client(args)**: Αποσύνδεση και απενεργοποίηση του υφιστάμενου client και επαναδημιουργία του με καινούργια χαρακτηριστικά. Εικόνα 3.7.

```
def update_client(self):

    self.client.disconnect()
    self.client.loop_stop()
    self.jwt_iat = datetime.datetime.utcnow()
    self.client = get_client(
        self.project_id, self.cloud_region,
        self.registry_id, self.device_id, self.private_key_file,
        self.algorithm, self.ca_certs, self.mqtt_bridge_hostname,
        self.mqtt_bridge_port)
    self.client.loop_start()
```

Εικόνα 3.7: IoT Client-update

- **shutdown_client(args)**: Αποσύνδεση και τερματισμός του client. Εικόνα 3.8.

```
def shutdown_client(self):
    self.client.disconnect()
    self.client.loop_stop()
```

Εικόνα 3.8: IoT Client-close

Για την ταυτοποίηση και την επικύρωση του συγκεκριμένου client δημιουργήθηκε ένα αρχείο ρυθμίσεων που περιέχει τις απαραίτητες παραμέτρους. Εικόνα 3.9. Για τον λόγο αυτό δημιουργήθηκαν εσωτερικά στην κλάση οι κατάλληλες συναρτήσεις ώστε να μπορούμε να ορίσουμε, να προσπελάσουμε και να χρησιμοποιήσουμε τις παραμέτρους κατά την δημιουργία της σύνδεσης του client.

```
# Google Iot-Core parameters
# Must be setted for each device before running
with open("conf_file.json") as json_file:
    jsonfile = json.load(json_file)
    self.project_id = jsonfile['project_id']
    self.registry_id = jsonfile['registry_id']
    self.device_id = jsonfile['device_id']
    self.private_key_file = jsonfile['private_key_file']
    self.algorithm = jsonfile['algorithm']
    self.cloud_region = jsonfile['cloud_region']
    self.ca_certs = jsonfile['ca_certs']
    self.mqtt_bridge_hostname = jsonfile['mqtt_bridge_hostname']
    self.mqtt_bridge_port = jsonfile['mqtt_bridge_port']
    self.jwt_expires_minutes = jsonfile['jwt_expires_minutes']
```

Εικόνα 3.9: IoT-Core parameters

Αναλυτικά κάθε μια από τις παραπάνω μεταβλητές περιγράφεται ως:

- **“Project_id”:** Το αναγνωριστικό συμβολοσειράς του Cloud project που κατέχει το μητρώο και τη συσκευή.

- **“Registry_id”:** Το αναγνωριστικό συμβολοσειράς του μητρώου που ανήκει η συσκευή. Ένα μητρώο μπορεί να έχει πολλές συσκευές εγγεγραμμένες σε αυτό. Επίσης δίνεται η δυνατότητα δημιουργίας πολλών μητρώων για την καλύτερη οργάνωση των συσκευών στο cloud.

- **“Device_id”:** Το αναγνωριστικό συμβολοσειράς που ορίζεται από το χρήστη για τη συσκευή, για παράδειγμα, myRaspberry. Το αναγνωριστικό συσκευής πρέπει να είναι μοναδικό.

- **“Private_key_file”:** Αρχείο που περιέχει ένα ιδιωτικό κλειδί που δημιουργείται με μία από τις δύο κωδικοποιήσεις «RS256» ή «ES256».

- **“Algorithm”:** Το αναγνωριστικό συμβολοσειράς που ορίζει τον αλγόριθμο κωδικοποίησης για τα ιδιωτικά κλειδιά.

- **“Cloud Region”:** Το αναγνωριστικό συμβολοσειράς που ορίζει την περιοχή Google Cloud Platform του μητρώου συσκευών, για παράδειγμα, europe-west1

- **“Ca_certs”:** Αρχείο που περιέχει το πιστοποιητικό (CA) [26] το οποίο αποτελεί ένα πιστοποιητικό Secure Sockets Layer (SSL). Αυτά τα ψηφιακά πιστοποιητικά είναι αρχεία δεδομένων που χρησιμοποιούνται για την κρυπτογραφική σύνδεση μιας οντότητας με ένα δημόσιο κλειδί.

- **“Mqtt_bridge_hostname”:** Το αναγνωριστικό συμβολοσειράς που ορίζει την διεύθυνση του MQTT bridge του Google Cloud.

- **“Mqtt_bridge_port”:** Το αναγνωριστικό συμβολοσειράς που ορίζει την πόρτα της διεύθυνσης του MQTT bridge του Google Cloud. Οι τιμές που μπορεί να πάρει είναι μια από τις 8883 ή 443.

- **“Jwt_expires_minutes”:** Το αναγνωριστικό συμβολοσειράς που ορίζει την διάρκεια ζωής ενός JWT κλειδί.

Κατά την διάρκεια την σύνδεσης του client μας χρησιμοποιείται και ένα κωδικοποιημένο JWT (json web token) κλειδί [27], η διάρκεια του οποίου είναι 20 λεπτά

και στη συνέχεια ο client αποσυνδέεται και πρέπει να ξανασυνδεθεί με τον ίδιο τρόπο. Η δημιουργία της συγκεκριμένης λειτουργίας περιγράφεται από την συγκεκριμένη συνάρτηση:

- **create_jwt(args)**: Λαμβάνει σαν όρισμα το “project_id”, “private_key_file” και το “Algorithm”. Επιστρέφει ένα κωδικοποιημένο αλφαριθμητικό που χρησιμοποιείται ως κωδικός πρόσβασης κατά την δημιουργία του client.

Η συνάρτηση της κλάσης του “IoT Client” που αναλαμβάνει να συνδυάσει όλες τις παραπάνω λειτουργικότητες και να επιστρέψει έναν Paho Mqtt client ο οποίος έχει όλες τις απαραίτητες αρμοδιότητες, για να μπορέσει να χρησιμοποιεί το IoT Core εργαλείο του Google Cloud, υλοποιείται από την παρακάτω συνάρτηση:

- **get_client(args):** Με τα απαραίτητα ορίσματα δημιουργεί έναν Paho Mqtt client ο οποίος είναι κρυπτογραφημένος, συνδέεται στον “Mqtt_broker” του Google Cloud και ορίζει τα κανάλια για την προώθηση των δεδομένων, της κατάστασης και της παραμετροποίησης της συσκευής.

Αφού έχουμε δημιουργήσει τον client στην κύρια διεργασία, στην συνέχεια γίνεται αρχικοποίηση των νημάτων που τρέχουν παράλληλα τις διεργασίες με την επεξεργασία και την προώθηση των δεδομένων από τους αισθητήρες και την διεργασία που κρατάει ενημερωμένο τον client σε περίπτωση αποσύνδεσης.

Η διεργασία που αφορά την επεξεργασία των δεδομένων και την προώθηση αυτών στην cloud υποδομή μας υλοποιήθηκε από των συνδυασμό συναρτήσεων που έχουν να κάνουν με την συγκέντρωση των δεδομένων από την διεπαφή που διαχειρίζεται τους αισθητήρες, την επεξεργασία αυτών και την δημιουργία του επιθυμητού format για την προώθηση τους. Σε κάθε ένα από τα κομμάτια ξεχωριστά υλοποιείται και η διαδικασία για την διατήρηση ιστορικού της διεργασίας.

Η προώθηση των δεδομένων που συλλέγονται από την διεπαφή με τους αισθητήρες γίνεται σε δύο διαφορετικά κανάλια του Mqtt broker το ένα έχει να κάνει με το “events” και είναι αυτό στο οποίο προωθούνται τα δεδομένα από τους αισθητήρες που προσαρμόζονται στις δεξαμενές και το άλλο είναι το “state” στο οποίο προωθούνται τα δεδομένα που έχουν να κάνουν με τους αισθητήρες που είναι ενσωματωμένοι στο Raspberry και επιτηρούν την κατάσταση της ίδιας της συσκευής. Εικόνα 3.10.

```
#For event publish
sub_topic = 'events'
self.mqtt_topic = '/devices/{}/{}'.format(self.device_id, sub_topic)
#print ("EVENT TOPIC:" +mqtt_topic)

#For state publish
sub_topic_state = 'state'
self.mqtt_topic_state = '/devices/{}/{}'.format(self.device_id, sub_topic_state)
#print ("STATE TOPIC:"+mqtt_topic_state)
```

Εικόνα 3.10: IoT Client-topics

Οι συναρτήσεις για την συλλογή των δεδομένων και για την ομαδοποίηση τους υλοποιήθηκε από τις παρακάτω συναρτήσεις:

- **“get_data()”:** Χρησιμοποιεί την διεπαφή των αισθητήρων που συλλέγουν δεδομένα από τις δεξαμενές. Προσθέτει την κατάλληλη σήμανση στα δεδομένα και στην συνέχεια ομαδοποιεί τα δεδομένα και επιστρέφει ένα json αντικείμενο έτοιμο για προώθηση στο cloud. Εικόνα 3.11.

```
def get_data():
    # create Sensor object
    sensor = Sensor()
    # create device Status object
    rpi = Info()
    data={}
    data['type'] = 'data'
    data['user'] = rpi.get_username()
    data['pass'] = rpi.get_password()
    data['gatewayId'] = rpi.get_gateway_id()
    data['deviceId'] = rpi.get_device_id()
    date = str(sensor.get_date())
    date = date.split("+")
    data['created_at'] = date[0]
    data['tempin'] = sensor.get_inside_temperature()
    data['tempout'] = sensor.get_outside_temperature()
    data['ph_liquid'] = sensor.get_ph_liquid()
    data['ph_solid'] = sensor.get_ph_solid()
    ec = sensor.get_ec()
    data['conductivity'] = ec[0]
    data['tdsolids'] = ec[1]
    data['salinity'] = ec[2]
    data['sgravity'] = ec[3]

    return data
```

Εικόνα 3.11: Συνάρτηση συλλογής δεδομένων αισθητήρων

- **“get_state()”:** Χρησιμοποιεί την διεπαφή διαχείρισης της συσκευής και συλλέγει τα δεδομένα των αισθητήρων που αφορούν την ίδια την συσκευή. Ομαδοποιεί τα δεδομένα, προσθέτει την κατάλληλη σήμανση και επιστρέφει ένα json αντικείμενο για προώθηση. Εικόνα 3.12.

```
def get_state():
    # create Sensor object
    sensor = Sensor()
    # create device Status object
    rpi = Info()

    data={}
    data['type'] = "info"
    data['user'] = rpi.get_username()
    data['pass'] = rpi.get_password()
    data['gatewayId'] = rpi.get_gateway_id()
    data['deviceId'] = rpi.get_device_id()
    date = str(sensor.get_date())
    date = date.split("+")
    data['created_at'] = date[0]
    data['cpu_temp'] = rpi.get_cpu_temp()
    data['free_disk'] = rpi.get_available_space()
    data['battery_level'] = rpi.get_battery_level()

    return data
```

Εικόνα 3.12: Συνάρτηση συλλογής δεδομένων συσκευής

Τα json αντικείμενα, τα οποία παράγονται από τις παραπάνω συναρτήσεις, προωθούνται στην cloud υπηρεσία από την συνάρτηση “**send_data_iot**”. Αρχικά εξασφαλίζει ότι υπάρχει συνδεσιμότητα με το διαδίκτυο και στην συνέχεια χρησιμοποιεί τον “IoT Client” και την συνάρτηση “client.publish()” για να χρησιμοποιήσει την υποδομή που έχουμε αναπτύξει και να προωθήσει με ασφάλεια τα δεδομένα. Σε περίπτωση που η συσκευή δεν είναι συνδεδεμένη στο διαδίκτυο για αρκετό χρονικό διάστημα, τα δεδομένα αποθηκεύονται τοπικά και όταν η συσκευή συνδεθεί ξανά, δημιουργείται ένα νέο νήμα που είναι υπεύθυνο για την προώθηση τους. Επειδή κάθε ένα πακέτο δεδομένων που

δημιουργούν οι παραπάνω συναρτήσεις έχει σαν πεδίο την χρονική στιγμή που δημιουργήθηκε (timestamp), δεν αντιμετωπίζεται κάποιο θέμα κατά την αποθήκευση τους στην βάση δεδομένων μας.

- **“send_data_iot(args)” :** Δέχεται σαν ορίσματα τον “IoT Client” και το πακέτο με τα δεδομένα. Με την χρήση του client τελικά προωθεί τα δεδομένα στο κατάλληλο κανάλι.

3.2.2.2 Παραμετροποίηση και διαμόρφωση

Ένα βασικό χαρακτηριστικό που προσφέρει το εργαλείο IoT Core είναι ότι επιτρέπει την αμφίδρομη επικοινωνία μεταξύ την κεντρικής πλατφόρμας και τους MQTT clients που δημιουργούνται στις συσκευές που είναι εγγεγραμμένες σε κάποιο μητρώο του. Η κλάση του client που δημιουργήσαμε περιέχει την callback συνάρτηση που χρησιμοποιεί η πλατφόρμα του IoT Core για να στέλνει εντολές παραμετροποίησης και διαμόρφωσης. Τα μηνύματα που ανταλλάσσονται μεταξύ των δύο γίνονται σε ένα ξεχωριστό κανάλι και η callback συνάρτηση του client που είναι υπεύθυνη για την λήψη και μετέπειτα επεξεργασία των μηνυμάτων είναι η “**client.onmessage()**” . Εικόνα 3.13.

```
# This is the topic that the device will receive configuration updates on.
mqtt_config_topic = '/devices/{}/config'.format(device_id)
# Subscribe to the config topic.
client.subscribe(mqtt_config_topic, qos=0)
```

Εικόνα 3.13: IoT Client-configuration topic

Για καταφέρουμε να ορίσουμε ένα ενιαίο τρόπο παραμετροποίησης για όλες τις συσκευές δημιουργήσαμε μια πρότυπη μορφή (Εικόνα 3.14). Αυτό έγινε για να μπορέσουμε να εντάξουμε με τον ίδιο τρόπο πολλές λειτουργίες και να επιτύχουμε με μια διεργασία, την επεξεργασία των εντολών παραμετροποίησης και την υλοποίησή τους σε κάθε συσκευή ξεχωριστά.

```
{
  "config_type" : state        |  config          |  system

  "action"      : device_state |  set parameters  |  system actions (reboot etc)
}
```

Εικόνα 3.14: Πρότυπη μορφή εντολών παραμετροποίησης

Οι λειτουργίες που καταφέραμε να υλοποιήσουμε για την παραμετροποίηση και την διαμόρφωση της συσκευής έχουν να κάνουν:

- Κατάσταση διεργασιών που εκτελούνται.
- Αποστολή των αρχείων καταγραφής της συσκευής στην cloud υποδομή μας.
- Προβολή της τοπικής κατάστασης της διεπαφής δικτύου.
- Παραμετροποίηση της συχνότητας αποστολής δεδομένων μετρήσεων στο cloud.
- Εντολές απομακρυσμένης επανεκκίνησης της συσκευής.
- Εντολές απομακρυσμένης απενεργοποίησης της συσκευής.
- Εντολές απομακρυσμένης επανεκκίνησης διεργασιών της συσκευής.
- Δυνατότητα βαθμονόμησης των αισθητήρων της συσκευής.
- Δυνατότητα ενεργοποίησης και απενεργοποίησης της διεπαφής ενός ή περισσοτέρων αισθητήρων.

Η υλοποίηση της λειτουργικότητας των παραπάνω διεργασιών πραγματοποιήθηκε από μια συνάρτηση η οποία καλείται από την callback συνάρτηση του client για κάθε ένα μήνυμα που καλείται να εξυπηρετήσει. Για να μπορέσει η συγκεκριμένη λειτουργικότητα να διαχειριστεί τον όγκο σε περίπτωση πολλών μηνυμάτων, υλοποιήσαμε την διεργασία να γίνεται κάθε φορά σε ένα ξεχωριστό νήμα και με την δυνατότητα αμοιβαίου αποκλεισμού όπου χρειάζεται. Η λειτουργικότητα της συνάρτησης επεξεργασίας υλοποιήθηκε να λαμβάνει σαν όρισμα το μήνυμα που έρχεται από το κανάλι, να το διαπερνά και μετά την αποκωδικοποίησή του να κάνει αντιστοίχιση τα πεδία του

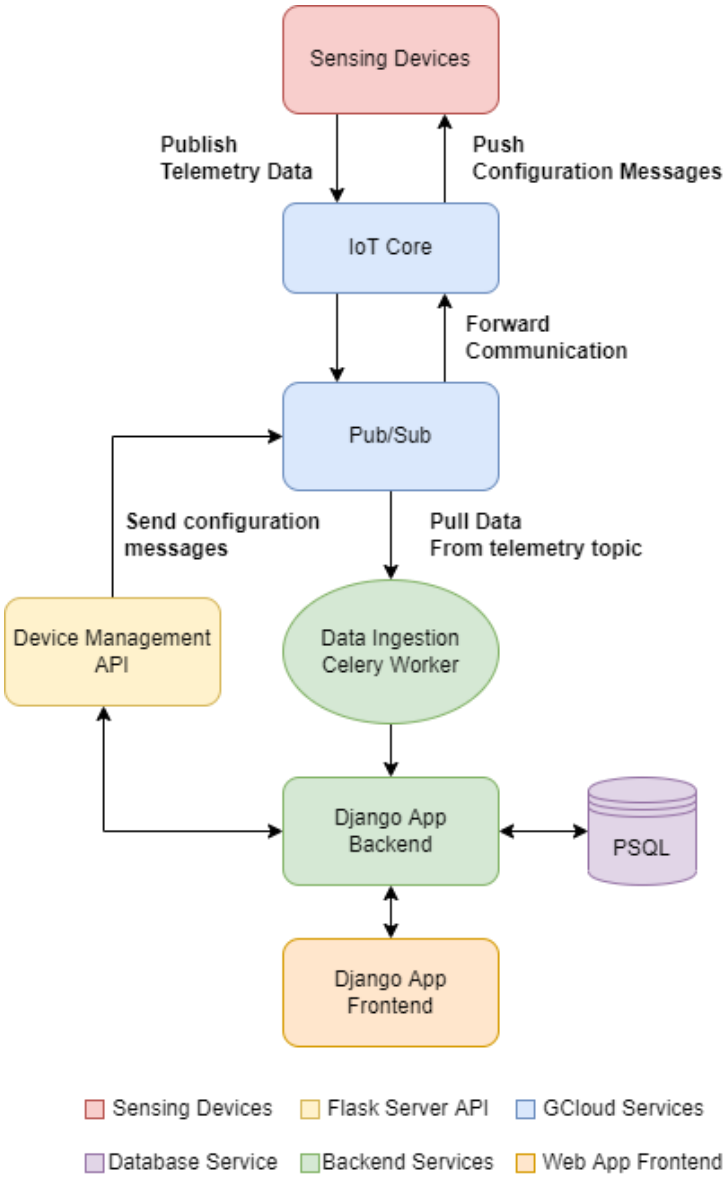
μηνύματος με τις συναρτήσεις που υλοποιούν την κάθε λειτουργία που περιεγράφηκε παραπάνω. Εικόνα 3.15.

```
elif conf["config_type"] == 'system':
    if conf["action"] == 'reboot':
        sys_reboot()
        zipfile_name = ''
        #print("System reboot..")
    elif conf["action"] == 'ifconfig':
        payload = json.dumps(ifconfig())
        zipfile_name = ''
        print('Publishing message: \'{}\''.format(payload))
        sys.stdout.flush()
        unused_client.publish(global_state_topic, payload, qos=0)
    elif conf["action"] == 'service':
        payload = json.dumps(read_status(conf["name"]))
        print('Publishing message: \'{}\''.format(payload))
        sys.stdout.flush()
        unused_client.publish(global_state_topic, payload, qos=0)
    elif conf["action"] == 'zip_and_send_log':
        [payload, zipfile_name] = logfile_handler()
        print('Publishing message: \'{}\''.format(payload))
        sys.stdout.flush()
        #unused_client.publish(global_state_topic, payload, qos=0)

    post_to_api(conf["action"], payload, zipfile_name)
```

Εικόνα 3.15: Λειτουργικότητα εντολών παραμετροποίησης και διαμόρφωσης

3.3 Επισκόπηση Cloud

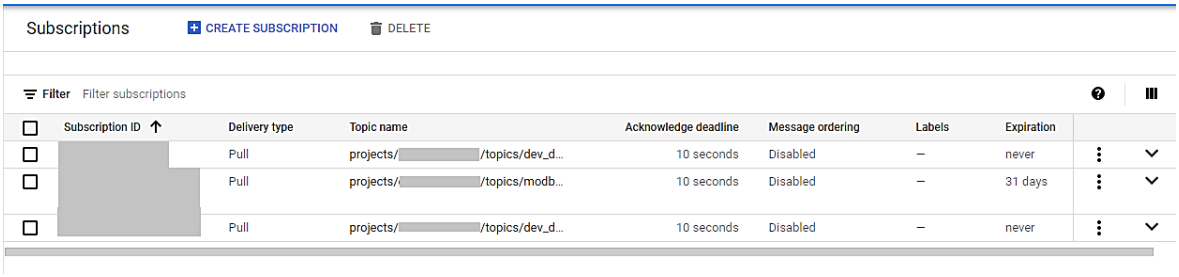
Με στόχο τη δημιουργία μίας σύγχρονης υποδομής, η οποία να μπορεί να φιλοξενεί τις συσκευές μας και να υποστηρίζει τη λειτουργικότητα τους συνεχώς (24/7), ήταν απαραίτητο να σχεδιάσουμε και να αναπτύξουμε το IoT Platform. Απαραίτητη προϋπόθεση για την υλοποίησή του αποτελεί η υποδομή cloud, η οποία φιλοξενεί διεργασίες που αφορούν τη διαχείριση των δεδομένων από τις συσκευές, τη σύνδεση και διαχείριση των συσκευών μέσω του IoT Core και τέλος την ανάπτυξη και την εγκατάσταση της διαδικτυακής εφαρμογής σε υπολογιστικά συστήματα που βρίσκονται στο cloud, για να είναι διαθέσιμη απομακρυσμένα και πάντα. Η υποδομή του cloud που δημιουργήσαμε φιλοξενείται εξ ’ολοκλήρου στις cloud υπηρεσίες το Google Cloud Platform. Εικόνα 3.16.



Εικόνα 3.16: Διάγραμμα υποδομής Cloud

3.3.1 Μονάδα διαχείρισης δεδομένων από τις συσκευές

Τα δεδομένα που παράγονται και προωθούνται στο cloud, όπως περιεγράφηκε στην ενότητα 3.2.2.1 Επεξεργασία και προώθηση δεδομένων, χρειάζεται να αποθηκευτούν για την μετέπειτα επεξεργασία τους. Αυτή η διαδικασία περιγράφεται στην υλοποίησή μας από την δημιουργία μίας διεπαφής που ονομάζεται “Data Ingestion”. Τα δεδομένα προωθούνται στο IoT Core από τις συσκευές μέσω ενός καναλιού που ορίζεται ως “telemetry topic”. Για να συλλέξουμε τα δεδομένα που εισέρχονται στο συγκεκριμένο κανάλι, το εργαλείο του IoT Core προσφέρει τη δυνατότητα δημιουργίας καναλιών Pub/Sub που επιτρέπουν την δημιουργία client καταναλωτών σε υψηλότερο επίπεδο. Εικόνα 3.17Εικόνα 3.17



Εικόνα 3.17: Πίνακας διαχείρισης Pub/Sub καναλιών εγγραφής

Η λειτουργικότητα της διεργασίας του “Data Ingestion” αποτελείται από μία συνάρτηση, η οποία δημιουργεί και αρχικοποιεί έναν Pub/Sub client, κάνοντας χρήση της python βιβλιοθήκης “google.cloud” [28] και συγκεκριμένα του πακέτου “pubsub_v1”. Για την αρχικοποίησή του, χρειάστηκε να ορίσουμε το “project_id” και το όνομα του καναλιού εγγραφής του Pub/Sub (Εικόνα 3.18). Η λήψη των μηνυμάτων που δημιουργεί ο client γίνεται στην callback συνάρτησή του, η οποία με τη σειρά της για κάθε ένα από τα μηνύματα που λαμβάνει, αναθέτει την δουλειά σε ένα νήμα για την ανάλυση και την αποθήκευση των δεδομένων.

```
# [START pubsub_quickstart_subscriber]
subscriber = pubsub_v1.SubscriberClient()
subscription_path = subscriber.subscription_path(
    '[redacted]', 'system_dev_sub')



def callback(message):
    print('Received message: {}'.format(message))
    #message = json.loads(message.data.decode('utf-8'))
    Thread(target=devices_data_handle.save_new_measurement,args=(message,mutex, )).start()
    message.ack()
```

Εικόνα 3.18: Pub/Sub client

Η επεξεργασία και η αποθήκευση των δεδομένων γίνεται μέσω της συνάρτησης “save_new_measurement”, η οποία είναι μέρος του πακέτου “devices_data_handle”. Η συνάρτηση αυτή λαμβάνει σαν όρισμα το μήνυμα με τα δεδομένα σε json μορφή, το αναλύει και στην συνέχεια με χρήση αμοιβαίου αποκλεισμού, αποθηκεύει τα δεδομένα στη βάση για κάθε μέτρηση και κάθε συσκευή. Η διαδικασία που ακολουθήθηκε για την ανάλυση του json αντικειμένου (Εικόνα 3.19) που περιέχει την επιθυμητή πληροφορία, είναι αρχικά η αποκωδικοποίησή του και στην συνέχεια τα πεδία που περιέχει γίνονται αντιστοίχιση στα πεδία του αντικειμένου της κλάσης που ορίζουμε εσωτερικά για να υπάρχουν ομαδοποιημένα τα δεδομένα του κάθε μηνύματος.

```
class datajson(object):

    def __init__(self):
        self.device_tag=""
        self.created_at=""
        self.ph = -100.0
        self.salinity = -100.0
        self.conductivity = -100.0
        self.total_dissolved_solids = -100.0
        self.specific_gravity = -100.0
        self.temperature = -100.0
        self.voltage = -100.0
        self.battery_level = -1
```

Εικόνα 3.19: Αντικείμενο που αναπαριστά τα δεδομένα των μηνυμάτων

Για την διαδικασία της ταυτοποίησης και αποθήκευσης των δεδομένων χρησιμοποιήθηκε η λειτουργικότητα που προσφέρει το Django Framework η οποία λέγεται ORM (object

relation model) και δίνει τη δυνατότητα να προσπελαστούν τα μοντέλα της βάσης δεδομένων μέσω του συστήματος κλάσεων και αντικειμένων της python [14]. Με τον τρόπο αυτό δημιουργήθηκε μια λογική που με βάση τα δεδομένα για αποθήκευση, τα μοντέλα της βάσης φιλτράρονται με τις παραμέτρους ξεχωριστά και στο τέλος δημιουργείται ένα νέο αντικείμενο τύπου measurement που αποθηκεύεται στην βάση. Παράλληλα με την αποθήκευση ξεκινάει και η διεργασία που εξετάζει αν η νέα τιμή που δημιουργήθηκε ξεπερνάει τα όρια που έχει ορίσει ο χρήστης και τον ειδοποιεί είτε με email είτε με γραπτό μήνυμα στο κινητό.

Για μπορέσει να υποστηριχθεί όλη αυτή η διαδικασία της επεξεργασίας και της αποθήκευσης των δεδομένων, που κάνει χρήση την λειτουργικότητα του Django Framework, υλοποιήθηκε με την βοήθεια του εργαλείου Django Celery [29], το οποίο προσφέρει την ασύγχρονη εκτέλεση εργασιών μέσα από μία ουρά μηνυμάτων. Η μονάδα διαχείρισης των δεδομένων της συσκευής εκτελείται σαν ένα “celery.task” στο background της εφαρμογής.

3.3.2 Μονάδα διαχείρισης συσκευών μέσω του IoT Core

Η μονάδα διαχείρισης των συσκευών δημιουργήθηκε για να χρησιμοποιήσουμε την αμφίδρομη επικοινωνία που προσφέρει το IoT core με σκοπό την παραμετροποίηση καθώς και την διαμόρφωσή τους. Η δυνατότητα αυτής της επικοινωνίας δίνεται μέσα από το γραφικό περιβάλλον που προσφέρει το Google Cloud στην web based εφαρμογή του, ωστόσο υλοποιήθηκε και σαν μέρος του δικού μας IoT Platform.

Για να επιτευχθεί αυτό αναπτύχθηκε ένα API βασισμένο σε έναν python Flask server [30] που φιλοξενείται εσωτερικά στην cloud υποδομή μας. Τα endpoint URLs που υπάρχουν διαθέσιμα από το API υλοποιήθηκαν με σκοπό κάθε ένα ξεχωριστά να υλοποιεί μια από τις λειτουργίες των συσκευών που περιγράφονται στην παράγραφο 3.2.2.2 Παραμετροποίηση και διαμόρφωση. Η λειτουργικότητα για κάθε ένα URL υλοποιείται από μία διαδικασία, η οποία παίρνει σαν όρισμα τα δεδομένα που υπάρχουν στο request αντικείμενο του HTTPS πρωτοκόλλου, στα οποία υπάρχουν το “device_id” και το “registry_id” της συσκευής που θα αποσταλεί το νέο μήνυμα και δημιουργεί έναν client τύπου “DeviceManagerClient” από την βιβλιοθήκη “google.cloud” [28] για να συνδεθεί στο κατάλληλο κανάλι του μητρώου που ανήκει η συσκευή και χρησιμοποιείται για παραμετροποίηση. Εικόνα 3.20.

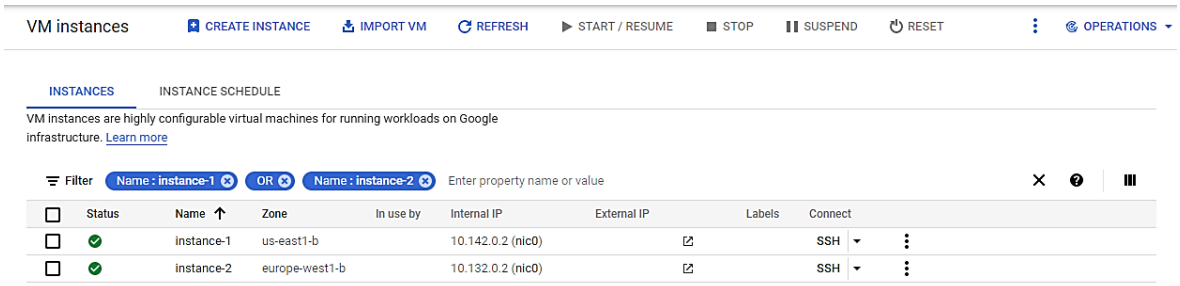
```
print("Sending [" + args.get("action") + "] command to device")
sys.stdout.flush()
client = iot_v1.DeviceManagerClient()
device_path = client.device_path(project_id, cloud_region, registry_id, device_id)
print(device_path)
command = json.dumps(command)
data = command.encode("utf-8")

return client.send_command_to_device(device_path, data)
```

Εικόνα 3.20: Διαδικασία αποστολής μηνυμάτων παραμετροποίησης

Το μήνυμα που στέλνεται από τον client στην συσκευή δημιουργείται και κωδικοποιείται ως ένα json αντικείμενο βασισμένο στην ενιαία μορφοποίηση που υποστηρίζουν οι συσκευές όπως περιεγράφηκε νωρίτερα και ορίζεται ξεχωριστά για κάθε ένα URL.

3.3.3 Ανάπτυξη(deployment,Παράταξη) και εγκατάσταση της εφαρμογής

Οι παραπάνω διεργασίες που περιγράφονται για να μπορέσουν να είναι διαθέσιμες στους χρήστες του IoT Platform μας, φιλοξενούνται στην cloud υποδομή μας και συγκριμένα σε εικονικά μηχανήματα. Ένα πολύ σημαντικό μέρος της υλοποίησης του βιομηχανικού πρωτοτύπου ήταν να μεταφέρουμε την λειτουργικότητα από τοπικό επίπεδο στο cloud. Για να καταφέρουμε να παρατάξουμε το IoT Platform μας στο cloud χρησιμοποιήθηκε το εργαλείο Compute Engine του Google Cloud,όπως φαίνεται στην Εικόνα 3.21, και συγκεκριμένα δημιουργήσαμε δύο εικονικές μηχανές που χρησιμοποιούσαν ως λειτουργικό σύστημα το Ubuntu 18.04 LTS.



Εικόνα 3.21: Compute Engine Instances

Το πρώτο εικονικό μηχάνημα δημιουργήθηκε για να φιλοξενεί σημαντικές λειτουργίες όπως είναι η PostgreSQL βάση δεδομένων μας και ο VPN server που χρησιμοποιούμε με τις συσκευές για απομακρυσμένη είσοδο και συντήρηση. Σε συνέχεια της δημιουργίας της εικονικής μηχανής για να γίνει η εγκατάσταση των απαραίτητων

πακέτων αλλά και εφαρμογών συνδεθήκαμε σε αυτό απομακρυσμένα μέσω SSH για να εκτελέσουμε μια σειρά εντολών. Ο VPN server που χρησιμοποιήθηκε έγινε εγκατάσταση μέσω του “apt” πακέτου και δημιουργήθηκε το επιθυμητό αρχείο παραμετροποίησης βασισμένο στις ανάγκες μας. Η PostgreSQL βάση δεδομένων εγκαταστάθηκε και αυτή με το ίδιο πακέτο και για να μπορέσουμε να συνδεθούμε και εξωτερικά από το δίκτυο που ορίζει το εικονικό μηχάνημα μορφοποιήσαμε το αρχείο παραμετροποίησης και ταυτοποίησης της διεργασίας.

Το δεύτερο εικονικό μηχάνημα δημιουργήθηκε για να φιλοξενεί την βασική διεργασία που αφορά τον διακομιστή που εξυπηρετεί την φιλοξενία της Web based εφαρμογής μας. Ο συγκεκριμένος διακομιστής πρόκειται για έναν Apache server με το κατάλληλο αρχείο παραμετροποίησης που ορίσαμε για να φιλοξενεί τόσο το backend server της εφαρμογής αλλά και την εξυπηρέτηση των καταλόγων των στατικών αρχείων (HTML,CSS) του Django Framework. Επίσης για να γίνει εγκατάσταση η διεργασία του “Data Ingestion” ως celery task που περιγράψαμε, χρειάστηκε να εγκαταστήσουμε τον RabbitMQ broker [31] για να μπορεί ο διακομιστής του backend της εφαρμογής να διαμοιράζει τα tasks στους workers που δημιουργεί το “Celery”. Τέλος για την εγκατάσταση πολλών microservices που αναπτύξαμε για βοηθητικό ρόλο, όπως για παράδειγμα μια διεργασία που επεξεργάζεται τα αρχεία καταγραφής των διεργασιών που φιλοξενούνται στο εικονικό μηχάνημα, δημιουργήσαμε bash scripting αρχεία και τα ενσωματώσαμε σαν διεργασίες του ίδιου του λειτουργικού.

3.4 Επισκόπηση Web Application

Οι περισσότερες βιομηχανίες αυτή την στιγμή παράγουν καθημερινά μεγάλο όγκο δεδομένων κατά την διάρκεια της παραγωγής. Όλα αυτά τα δεδομένα αποθηκεύονται και καταγράφονται σε υπολογιστικά φύλλα χωρίς καμία γραφική απεικόνιση αλλά και εκμετάλλευση προς όφελος της επιχείρησης. Το συγκριμένο πρόβλημα προσπαθήσαμε να το εξαλείψουμε με την δημιουργία της web εφαρμογή μας. Στόχος μας ήταν να οργανώσουμε τα δεδομένα που παράγονται από τις συσκευές μας και να τα οπτικοποιήσουμε μέσω γραφημάτων στον τελικό χρήστη δίνοντας επιπλέον και άλλες λειτουργικότητες. Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το εργαλείο Django Framework τόσο στο backend αλλά και στο τρόπο εξυπηρέτησης του frontend της εφαρμογής μας που ήταν βασισμένο σε στατικά αρχεία από HTML, JS, CSS.

3.4.1 Ανάπτυξη του backend βασισμένο σε Django Framework

Για την υλοποίηση μιας εφαρμογής με την χρήση του Django Framework χρειάστηκε να δημιουργήσουμε τις τρείς βασικές λειτουργικότητές του. Ορίσαμε και δημιουργήσαμε τα μοντέλα της βάσης δεδομένων μας (Models), τις συναρτήσεις για την λήψη και απεικόνιση των δεδομένων στο frontend της εφαρμογής (Views) και τέλος την δημιουργία του “Controller” για την εξυπηρέτηση των αιτημάτων.

Τα μοντέλα της βάσης δεδομένων που ορίσαμε για την υλοποίηση της εφαρμογής μας περιγράφουν τα παρακάτω:

- **User**
  Περιγράφει τον χρήστη της εφαρμογής επεκτείνοντας την κλάση του “AbstractUser” που ορίζει το Django από την λειτουργικότητα του. Τα πεδία του συγκεκριμένου μοντέλου που χρησιμοποιούνται αποτελούνται από:
  - **Username:** περιέχει το αλφαριθμητικό έως 150 χαρακτήρες, το οποίο αντιπροσωπεύει το username του χρήστη.
  - **Password:** αντιπροσωπεύει τον ακατέργαστο κωδικό πρόσβασης κατακερματισμένο και με μεταδεδομένα. Το Django δεν αποθηκεύει τον ακατέργαστο κωδικό. Το μέγεθος τους είναι αυθαίρετο και μπορούν να περιέχουν οποιοδήποτε χαρακτήρα.

- **Customer**

  Περιγράφει την οντότητα του ιδιοκτήτη της εγκατάστασης και περιέχει για αυτόν τις βασικές πληροφορίες του. Τα παρακάτω πεδία περιγράφουν την συγκεκριμένη οντότητα:
  - **Account:** αποτελεί το ξένο κλειδί προς την οντότητα User
  - **Type:** περιγράφει τον τύπο προϊόντος που επεξεργάζεται η βιομηχανία μέσα από τις προκαθορισμένες επιλογές που δίνονται (“Olives”, ”Dairy”).
  - **First Name:** αντιπροσωπεύει το αλφαριθμητικό για το όνομα του χρήστη μέχρι 30 χαρακτήρες.
  - **Last Name:** αντιπροσωπεύει το αλφαριθμητικό για το επίθετο του χρήστη μέχρι 30 χαρακτήρες.
  - **Email:** αντιπροσωπεύει το αλφαριθμητικό για το email του χρήστη.
  - **Telephone:** αντιπροσωπεύει το αλφαριθμητικό για τον αριθμό τηλεφώνου του χρήστη.

- **Gateway**

  Περιγράφει την οντότητα της συσκευής, η οποία έχει ως λειτουργικότητα την προώθηση των δεδομένων στην cloud υπηρεσία μας αλλά και την δημιουργία τοπικής σύνδεσης στο διαδίκτυο, όπου αυτή είναι απαραίτητη. Τα πεδία της οντότητας περιγράφονται ως:
  - **Customer:** αποτελεί το ξένο κλειδί προς την οντότητα Customer
  - **Created at:** αντιπροσωπεύει την χρονική στιγμή της δημιουργίας της κάθε οντότητας.
  - **Tag:** αντιπροσωπεύει το αλφαριθμητικό για την διεύθυνση MAC της συσκευής. Αποτελεί μοναδικό πεδίο για κάθε μια τέτοια οντότητα
  - **Status:** αντιπροσωπεύει το Boolean πεδίο για την κατάσταση της συσκευής.
  - **Battery level:** αντιπροσωπεύει το αλφαριθμητικό για το επίπεδο μπαταρίας.
  - **Location:** αντιπροσωπεύει το αλφαριθμητικό για το όνομα της τοποθεσίας που βρίσκεται η συσκευή.

- **Device**

  Περιγράφει την οντότητα της συσκευής που φέρει τους αισθητήρες και συλλέγει τα δεδομένα. Τα πεδία της οντότητας περιγράφονται ως:
  - **Gateway:** αποτελεί το ξένο κλειδί προς την οντότητα Gateway
  - **Created at:** αντιπροσωπεύει την χρονική στιγμή της δημιουργίας της κάθε οντότητας.
  - **Tag:** αντιπροσωπεύει το αλφαριθμητικό για την διεύθυνση MAC της συσκευής. Αποτελεί μοναδικό πεδίο για κάθε μια τέτοια οντότητα

- **Name:** αντιπροσωπεύει το αλφαριθμητικό για το όνομα της συσκευής που ορίζει ο χρήστης.
- **Status:** αντιπροσωπεύει το Boolean πεδίο για την κατάσταση της συσκευής (active ή inactive).
- **Calibration Status:** αντιπροσωπεύει το Boolean πεδίο για την κατάσταση των αισθητήρων της συσκευής. Δηλαδή αν χρειάζονται βαθμονόμηση ή όχι.
- **Notification:** αντιπροσωπεύει το Boolean πεδίο για την ενεργοποίηση ή απενεργοποίηση των ειδοποιήσεων για τις τιμές της συγκεκριμένης συσκευής.
- **Last Notification:** αντιπροσωπεύει την χρονική στιγμή της τελευταίας ειδοποίησης που έλαβε ο χρήστης για την συγκεκριμένη συσκευή.

- **Tank**

  Περιγράφει την οντότητα της δεξαμενής που είναι προσαρτημένες οι συσκευές και συλλέγουν δεδομένα από το εσωτερικό τους. Τα πεδία της συγκεκριμένης οντότητας περιγράφουν τα χαρακτηριστικά της δεξαμενής τα οποία είναι:
  - **Device:** αποτελεί το ξένο κλειδί προς την οντότητα Device
  - **Updated at:** αντιπροσωπεύει την χρονική στιγμή της δημιουργίας ή της ανανέωσης της κάθε οντότητας.
  - **Type:** αντιπροσωπεύει το αλφαριθμητικό για τον τύπο της δεξαμενής
  - **Variety:** περιγράφει την ποικιλία του προϊόντος που επεξεργάζεται η δεξαμενή μέσα από τις προκαθορισμένες επιλογές που δίνονται.
  - **Capacity:** αντιπροσωπεύει το αλφαριθμητικό για την χωρητικότητα της δεξαμενής.
  - **Product Weight:** αντιπροσωπεύει το αλφαριθμητικό για το βάρος που προϊόντος που περιέχει η δεξαμενή.

- **Measurement**

  Περιγράφει την οντότητα των μετρήσεων που προκύπτουν από τις τιμές που επιστρέφουν οι αισθητήρες που είναι προσαρμοσμένοι στις δεξαμενές. Τα πεδία της οντότητας περιγράφονται ως:
  - **Tank:** αποτελεί το ξένο κλειδί προς την οντότητα Tank
  - **Datetime:** αντιπροσωπεύει την χρονική στιγμή της δημιουργίας της κάθε οντότητας.
  - **pH:** αντιπροσωπεύει την Float τιμή της μέτρησης του pH
  - **Temperature:** αντιπροσωπεύει την Float τιμή της μέτρησης της θερμοκρασίας
  - **Conductivity:** αντιπροσωπεύει την Float τιμή της μέτρησης της ηλεκτρικής αγωγιμότητας.
  - **Salinity:** αντιπροσωπεύει την Float τιμή της μέτρησης της αλατότητας.

- **Specific gravity:** αντιπροσωπεύει την Float τιμή της μέτρησης της ειδικής βαρύτητας.

- **Critical Value**
  Περιγράφει την οντότητα των κρίσιμων τιμών που ορίζει ο χρήστης για κάθε δεξαμενή. Δηλαδή τα άνω και κάτω όρια με βάση τις μετρήσεις των αισθητήρων για τα οποία θα λάβει ειδοποίηση. Τα πεδία της οντότητας περιγράφονται ως:
  - **Tank:** αποτελεί το ξένο κλειδί προς την οντότητα Tank
  - **Created at:** αντιπροσωπεύει την χρονική στιγμή της δημιουργίας της κάθε οντότητας.
  - **Notification Checked:** αντιπροσωπεύει το Boolean πεδίο για το αν η ειδοποίηση για την συγκεκριμένη οντότητα προβλήθηκε στο χρήστη.
  - **pH upper:** αντιπροσωπεύει την Float τιμή του πάνω ορίου της μέτρησης του pH.
  - **pH lower:** αντιπροσωπεύει την Float τιμή του κάτω ορίου της μέτρησης του pH.
  - **Temperature upper:** αντιπροσωπεύει την Float τιμή του πάνω ορίου της μέτρησης της θερμοκρασίας
  - **Temperature lower:** αντιπροσωπεύει την Float τιμή του κάτω ορίου της μέτρησης της θερμοκρασίας
  - **Conductivity:** αντιπροσωπεύει την Float τιμή της μέτρησης της ηλεκτρικής αγωγιμότητας που δεν πρέπει να ξεπεραστεί.
  - **Salinity:** αντιπροσωπεύει την Float τιμή της μέτρησης της αλατότητας που δεν πρέπει να ξεπεραστεί.
  - **Specific gravity:** αντιπροσωπεύει την Float τιμή της μέτρησης της ειδικής βαρύτητας που δεν πρέπει να ξεπεραστεί.

Τα παραπάνω μοντέλα τα ορίσαμε στο αρχείο models.py της εφαρμογής μας και για να παραχθούν τα μοντέλα σε κώδικα SQL έγινε χρήση της εντολής του Django, που ορίζεται από το Django, “makemigrations”. Η συγκεκριμένη εντολή δημιουργεί ένα νέο αρχείο migration, στο φάκελο με τα migrations της κάθε εφαρμογής, που περιγράφει τις αλλαγές που προστέθηκαν στο αρχείο models.py σε εντολές SQL. Για να ενσωματώσουμε τα αρχεία με τα migrations τελικά στην βάση δεδομένων, έγινε χρήση της εντολής “migrate” η οποία εντοπίζει τα νέα αρχεία των migrations και μεταφέρει τις εντολές στην συνδεδεμένη βάση δεδομένων που ορίζεται από το αρχείο settings.py. Εικόνα 3.22.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': configuration.dev_db_name,
        'USER': configuration.dev_db_user,
        'PASSWORD': configuration.dev_db_password,
        'HOST': configuration.dev_db_host,
        'PORT': configuration.dev_db_port,
    }
}
```

Εικόνα 3.22: Configuration της βάσης δεδομένων

Η υλοποίηση των συναρτήσεων για την προβολή των δεδομένων που περιγράφονται από τα μοντέλα της βάσης έγινε στο αρχείο των views.py. Στο συγκεκριμένο αρχείο ορίσαμε τις συναρτήσεις για κάθε μια λειτουργία που προσφέρουμε στην εφαρμογή μας και χρειάζεται δεδομένα της βάσης δεδομένων. Η λογική στις περισσότερες συναρτήσεις ήταν ότι έφεραν ως decorator την εντολή “login_required” το οποίο εξασφαλίζει την ταυτοποίηση (authentication) του χρήστη που κάνει το αίτημα και ότι η εξυπηρέτηση του αιτήματος από το διακομιστή είναι εξουσιοδοτημένη. Σαν όρισμα στις συναρτήσεις δίνεται το αντικείμενο “request” που περιέχει την πληροφορία του αιτήματος που γίνεται προς το διακομιστή της εφαρμογής αλλά και τα μοναδικά αναγνωριστικά που είναι απαραίτητα από τα ερωτήματα (query) που ορίζονται στην συνάρτηση. Το αποτέλεσμα της κάθε συνάρτησης είναι ότι επιστρέφει ένα αντικείμενο, το οποίο περιέχει ένα στατικό αρχείο HTML και τα δεδομένα της βάσης που θα απεικονίζονται σε αυτό.

Τέλος, δημιουργήθηκε το αρχείο urls.py στο οποίο περιέχει τις διευθύνσεις με τα URLs τα οποία χρησιμοποιεί ο “Controller” του Django για να καταφέρει να συνδέσει τα αιτήματα που φτάνουν στον διακομιστή με τις python συναρτήσεις από τα “Views”. Για να είναι διαθέσιμα τα URLs στον διακομιστή της εφαρμογής έπρεπε να τα συμπεριλάβουμε στο κεντρικό αρχείο “URLs” που ορίζεται από το Django project. Εικόνα 3.23.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('app/', include('app.urls')),
    path('', RedirectView.as_view(url='app')),
    path('login/', auth_views.LoginView.as_view(template_name = 'app/login.html'), name='login'),
    path('logout/', auth_views.LogoutView.as_view( next_page = '/login'), name='logout')
]
```

Εικόνα 3.23: Απεικόνιση διευθύνσεων URL του Django project

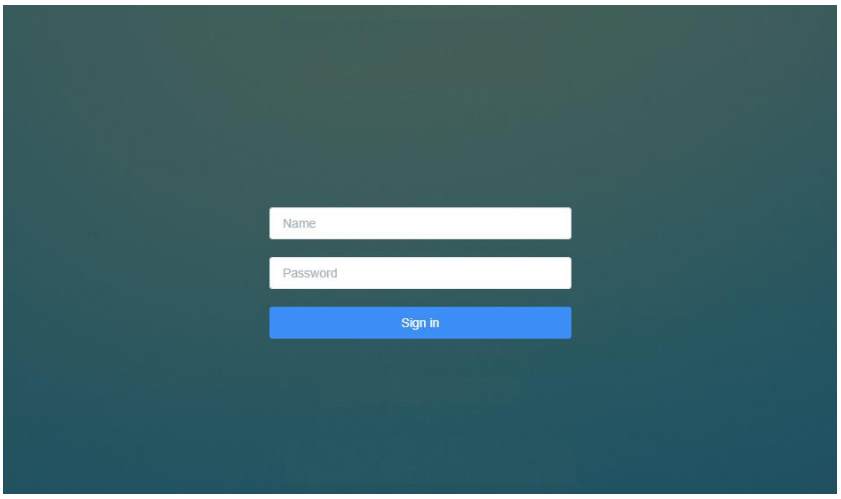
3.4.2 Ανάπτυξη και λειτουργικότητα του frontend της εφαρμογής

Για την απεικόνιση των δεδομένων που παράγουν οι συσκευές ανίχνευσης υλοποιήθηκε ένα σύνολο αρχείων με στατικό περιεχόμενο που αποτελείται από τον συνδυασμό HTML,CSS,JS. Στόχος ήταν να δημιουργηθεί ένα φιλικό προς το χρήστη γραφικό περιβάλλον στο οποίο θα προβάλλονται γραφήματα με τις τιμές που παράγει κάθε δεξαμενή ζύμωσης.

Αναλυτικότερα, για την δημιουργία της λειτουργικότητας του frontend της εφαρμογής αναπτύχθηκαν οι παρακάτω στατικές σελίδες:

- Login page

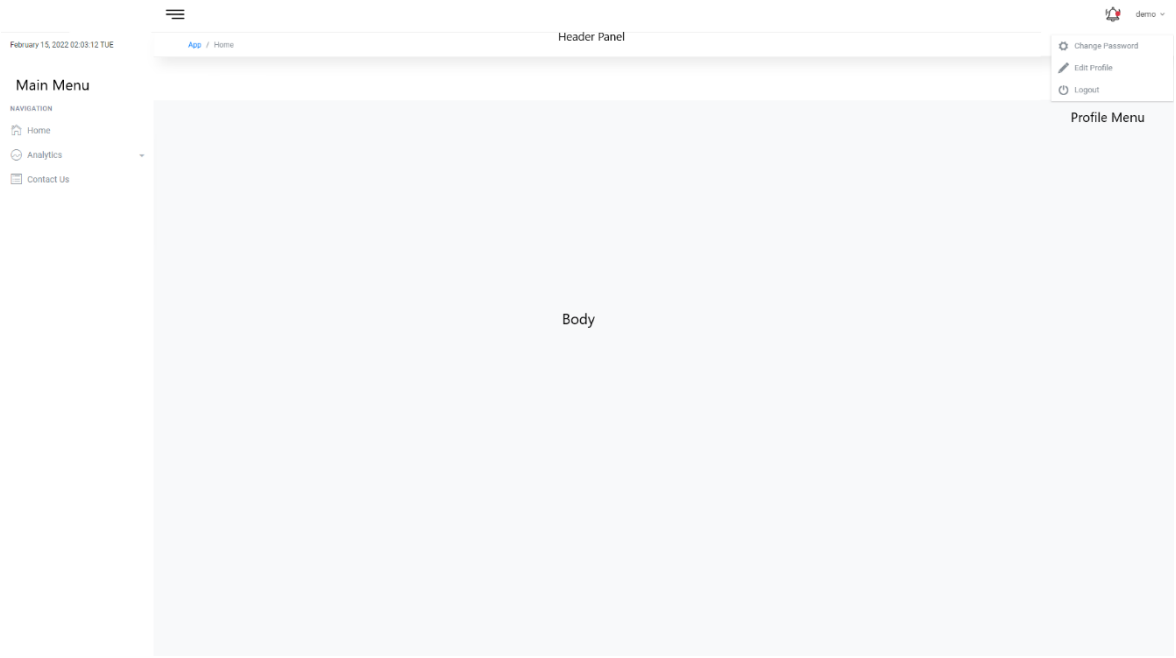
  Η συγκεκριμένη σελίδα δημιουργήθηκε για μπορεί ο χρήστης της πλατφόρμας να πληκτρολογεί τα στοιχεία του και να συνδεθεί στην εφαρμογή, αφού γίνει πρώτα η διαδικασία της ταυτοποίησή του. Εικόνα 3.24.



Εικόνα 3.24: Login Page

- Base Index

  Αποτελεί την βάση για κάθε σελίδα του frontend της εφαρμογής. Το στατικό κομμάτι που περιλαμβάνει χρησιμοποιείται σαν επέκταση σε όλα τα στατικά αρχεία της εφαρμογής. Αποτελείται από το βασικό μενού πλοήγησης στα αριστερά, το “Header Panel” που βρίσκεται στην κορυφή της σελίδας, το μενού προφίλ που εμφανίζεται όταν επιλεχθεί το dropdown μενού που βρίσκεται στο όνομα του χρήστη και “Body” το κομμάτι που είναι διαθέσιμο για να τοποθετηθεί το περιεχόμενο κάθε σελίδας ξεχωριστά. Εικόνα 3.25.



Εικόνα 3.25: Base Html

Στο βασικό μενού υπάρχουν τρείς επιλογές συνδέσμων που ανακατευθύνουν στις σελίδες “Home”, η οποία αφορά την βασική σελίδα της εφαρμογής που αναπαρίσταται από την σελίδα Main Index, στα “Analytics” που περιέχουν ιστορικά δεδομένα από τις μετρήσεις για κάθε μία δεξαμενή που αναπαρίσταται από την σελίδα Tank Analytics και στο “Contact us” που περιέχει την φόρμα επικοινωνίας για τεχνική υποστήριξη του χρήστη.

Το “Header Panel” που βρίσκεται στην κορυφή της σελίδας περιέχει το εικονίδιο με την ειδοποιήσεις και το όνομα του χρήστη που με την επιλογή του εμφανίζεται το μενού προφίλ.

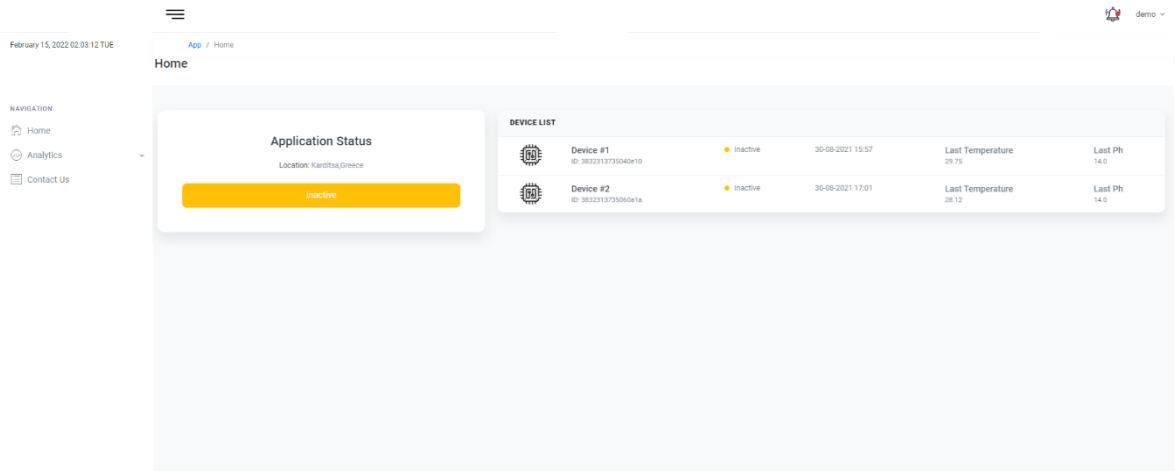
Το μενού προφίλ αποτελείται από τρείς συνδέσμους που με την επιλογή τους οδηγούν στις σελίδες “Edit Profile” για την παραμετροποίηση του προφίλ του χρήστη, “Change Password” για την αλλαγή κωδικού πρόσβασης και “Logout” για την αποσύνδεση του χρήστη από την εφαρμογή.

- Main Index

  Αποτελεί την βασική σελίδα της εφαρμογής που προβάλει δεδομένα σε δύο αντικείμενα όπως εμφανίζεται στην Εικόνα 3.26.

Αριστερά εμφανίζει δεδομένα που έχουν να κάνουν με την εγκατάσταση της βιομηχανίας όπως είναι η τοποθεσία της και το αν είναι ενεργή η κατάσταση της, δηλαδή αν υπάρχουν συσκευές που στέλνουν δεδομένα στο cloud.

Δεξιά στη σελίδα εμφανίζεται ένα πίνακας με τις συσκευές που είναι προσαρμοσμένες σε κάθε μια δεξαμενή στην βιομηχανία και για κάθε μια συσκευή προβάλλεται το όνομα της, η κατάσταση της αν είναι ενεργή ή ανενεργή, η ημερομηνία τελευταίας αποστολής δεδομένων και οι τελευταίες τιμές των αισθητήρων που διαθέτει.



Εικόνα 3.26: Main Index

- Device Details

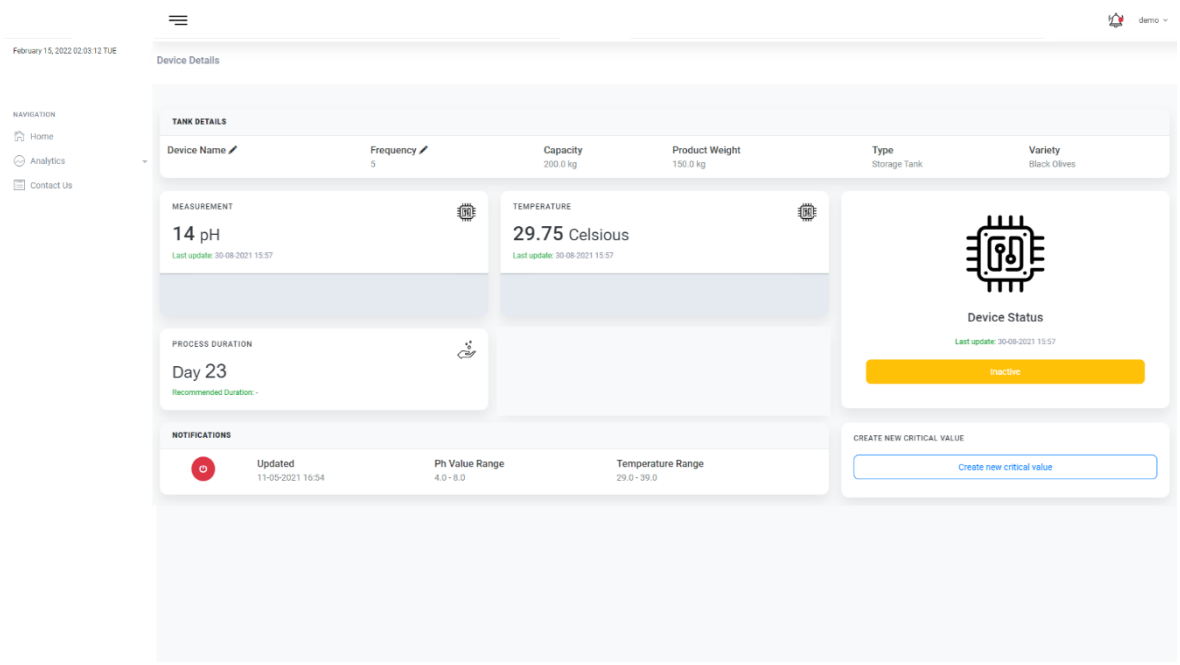
Αποτελεί την σελίδα frontend της εφαρμογής που απεικονίζει τα δεδομένα και τις πληροφορίες για κάθε μία συσκευή ξεχωριστά όπως εμφανίζεται στην Εικόνα 3.27.

Αρχικά εμφανίζει έναν πίνακα που περιέχει τις πληροφορίες και τα χαρακτηριστικά της δεξαμενής που είναι προσαρμοσμένη η συσκευή. Αναλυτικά περιέχει το όνομα της συσκευής, την συχνότητα που επιθυμεί ο χρήστης να έχουν οι μετρήσεις από την συγκεκριμένη δεξαμενή, την χωρητικότητα, το βάρους του προϊόντος που περιέχει, τον τύπο δεξαμενής και την ποικιλία του προϊόντος.

Κάτω από τον παραπάνω πίνακα προβάλλονται αριστερά οι κάρτες με τις τιμές των μετρήσεων από τους αισθητήρες και την διάρκεια της διαδικασίας που ακολουθεί κάθε δεξαμενή, δεξιά μία κάρτα που προβάλει την κατάσταση της συσκευής και την τιμή της ημερομηνίας της τελευταίας μέτρησης.

Στο τέλος της σελίδας προβάλλεται ο πίνακας με τα όρια από τις κρίσιμες τιμές για κάθε αισθητήρα που έχει η συσκευή και ένα κουμπί για ενεργοποίηση ή απενεργοποίηση των ειδοποιήσεων για τις συγκεκριμένες κρίσιμες τιμές. Επίσης

υπάρχει και μια κάρτα που περιέχει ένα κουμπί που οδηγεί στην σελίδα για την δημιουργία νέας κρίσιμης τιμής.



Εικόνα 3.27: Device details

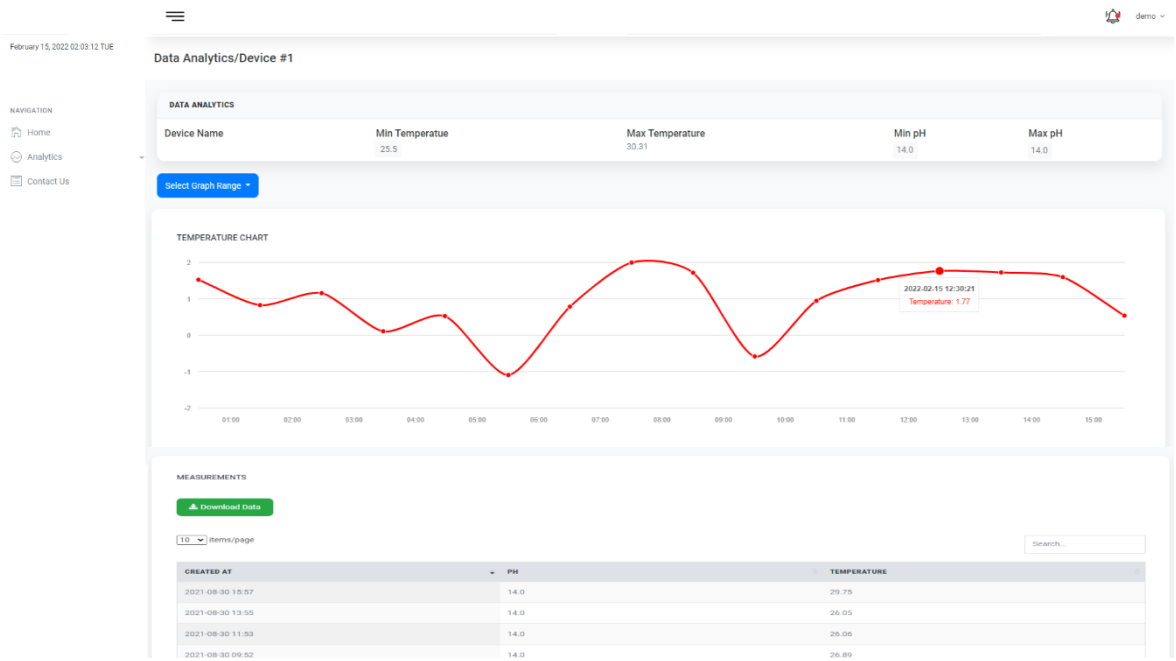
- Tank Analytics

Αποτελεί την σελίδα που δημιουργήθηκε για να μπορέσουμε να απεικονίσουμε, όπως φαίνεται στην Εικόνα 3.28, τα ιστορικά δεδομένα από κάθε δεξαμενή, να εντοπίσουμε στατιστικά δεδομένα και να κατασκευάσουμε γραφήματα ανάλογα με εύρος τιμών που ορίζονται από το χρήστη.

Αρχικά προβάλλεται ένας πίνακας με στατιστικά στοιχεία που αφορούν την χαμηλότερη και την υψηλότερη τιμή της μέτρησης του αισθητήρα μέσα στην μέρα. Ο πίνακας είναι δυναμικός στα δεδομένα που προβάλει ανάλογα με το πλήθος των αισθητήρων που συλλέγουν τιμές από την δεξαμενή .

Στην μεσαίο κομμάτι της σελίδας εμφανίζονται τα γραφήματα των μετρήσεων από τους αισθητήρες και ένα κουμπί πάνω από αυτά που επιτρέπει το χρήστη να διαλέξει το εύρος των τιμών της ημερομηνίας για την προβολή του γραφήματος.

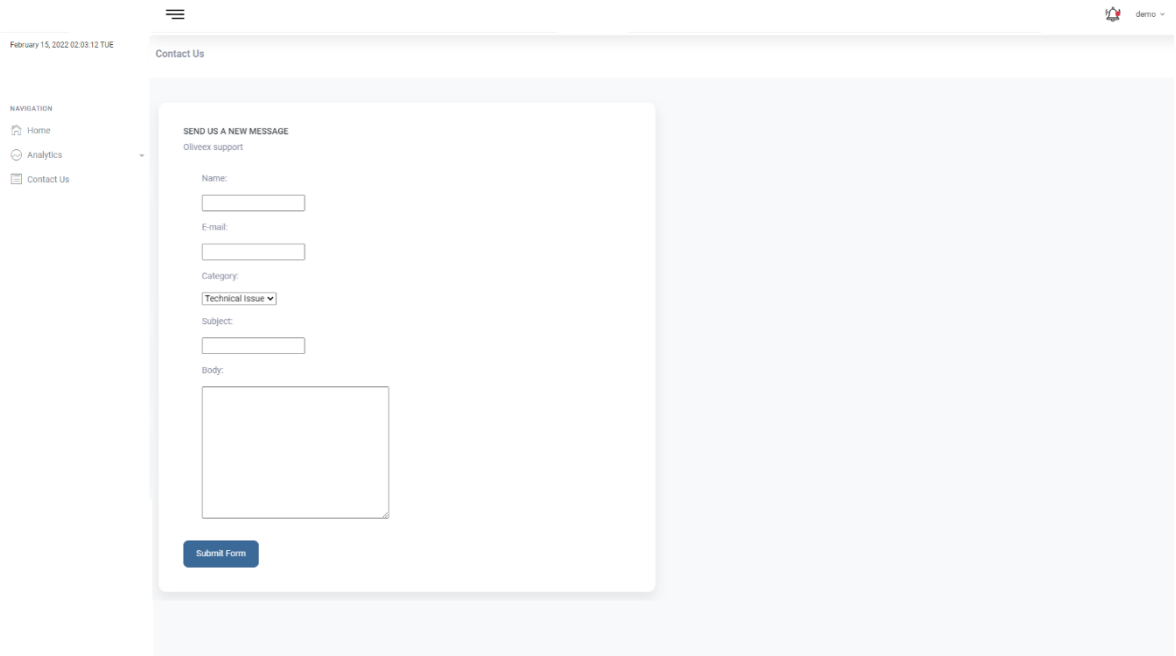
Τέλος, απεικονίζεται ένας πίνακας δεδομένων που περιέχει όλες τι τιμές που έχει συλλέξει ο αισθητήρας από την αρχή της λειτουργίας του για την συγκεκριμένη δεξαμενή. Δίνεται ακόμη η δυνατότητα στο χρήστη για αναζήτηση μέσα στο πίνακα αλλά και η επιλογή του πλήθους των αντικειμένων ανά σελίδα του πίνακα.



Εικόνα 3.28: Tank analytics

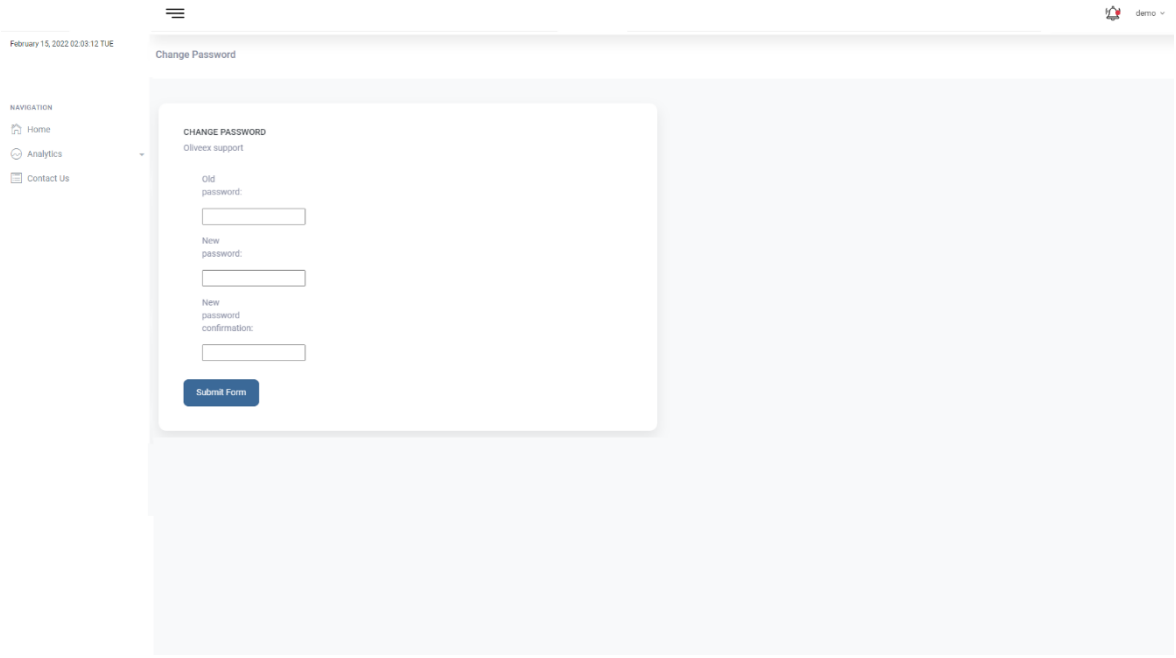
- Contact us

  Αποτελεί την σελίδα που υλοποιήθηκε για την επικοινωνία του χρήστη για τεχνική υποστήριξή του. Περιλαμβάνει μία φόρμα επικοινωνίας με απαραίτητα πεδία όπως εμφανίζεται στην Εικόνα 3.29.



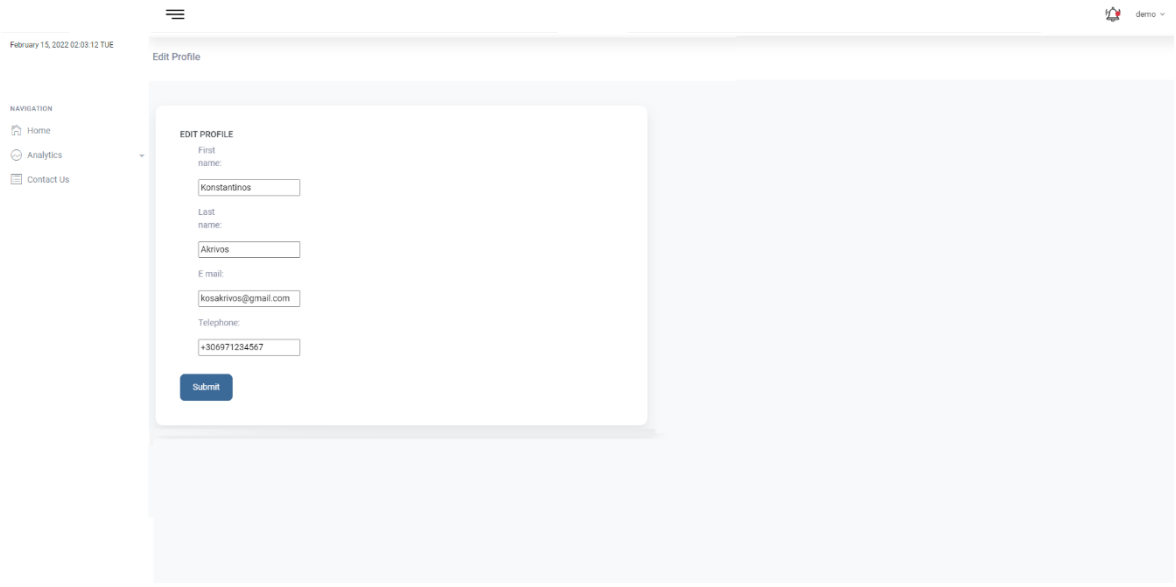
Εικόνα 3.29: Contact us form

- Change Password
  Αποτελεί την σελίδα που περιλαμβάνει μία φόρμα αλλαγής κωδικού πρόσβασης από το χρήστη με απαραίτητα πεδία όπως εμφανίζεται στην Εικόνα 3.30.



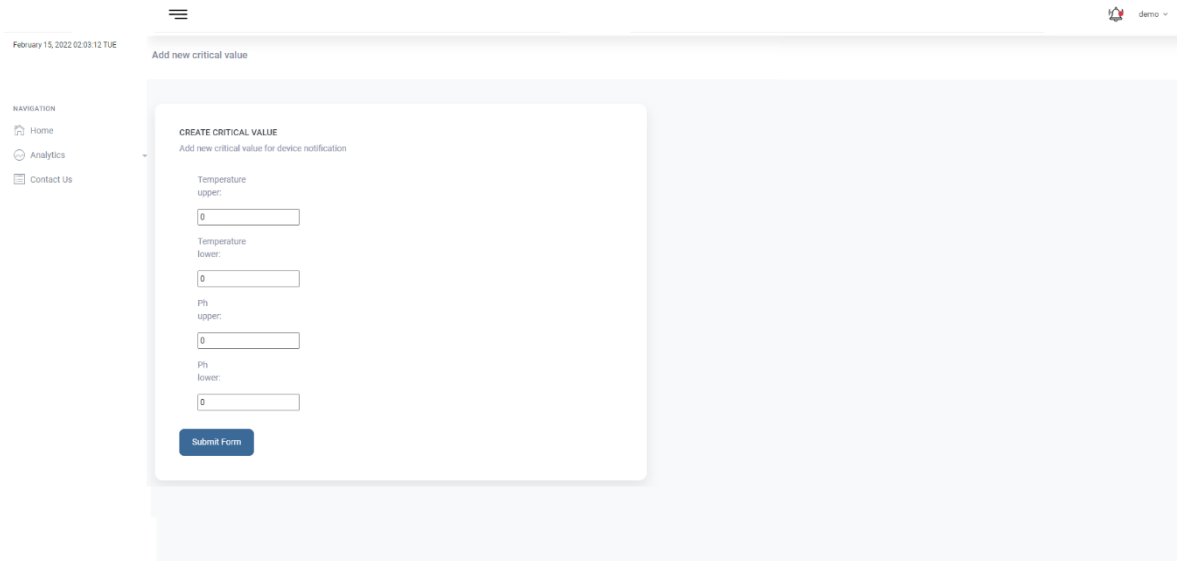
Εικόνα 3.30: Change password form

- Edit Profile
  Αποτελεί την σελίδα που απεικονίζει μία φόρμα με τα στοιχεία του χρήστη και την δυνατότητα επεξεργασίας τους. Τα πεδία που μπορεί να επεξεργαστεί ο χρήστης για το προφίλ του εμφανίζονται στην Εικόνα 3.31.



Εικόνα 3.31: Edit Profile form

- Create new critical value
  Αποτελεί την σελίδα που απεικονίζει μία φόρμα για την δημιουργία ορίων κρίσιμων τιμών για τις τιμές των αισθητήρων της συσκευής όπως φαίνεται στην Εικόνα 3.32.



Εικόνα 3.32: New critical value form

Συνδυάζοντας όλες τις παραπάνω στατικές σελίδες στόχος ήταν να δημιουργηθεί μια web εφαρμογή που θα έδινε την δυνατότητα στον τελικό χρήστη την απομακρυσμένη παρακολούθηση και την οργάνωση των δεδομένων που παράγονται κατά την παραγωγή καθώς και την έγκαιρη ενημέρωση του σε μεταβολές των τιμών σύμφωνα με τα όρια που έχει ο ίδιος ορίσει. Η ανάπτυξη έγινε με την ανατροφοδότηση που είχαμε από ανθρώπους που δουλεύουν σε βιομηχανίες ζυμούμενων τροφίμων και προσπαθήσαμε να περάσουμε στον frontend κομμάτι της εφαρμογής μας όσες περισσότερες λειτουργίες συζητήσαμε μαζί τους.

Κεφάλαιο 4 Δοκιμή συστήματος

Η συνολική λειτουργικότητα του βιομηχανικού πρωτοτύπου που αναπτύχθηκε στην παρούσα διπλωματική εργασία, πέρα από τις ελεγχόμενες συνθήκες κατά την υλοποίηση, δοκιμάστηκε και σε δύο πραγματικές βιομηχανικές συνθήκες. Στη δεύτερη περίπτωση μας δόθηκε η δυνατότητα να αντλήσουμε συμπεράσματα για την υλοποίησή μας από τον τρόπο λειτουργίας της σε ρεαλιστικές συνθήκες καθώς και να δοκιμαστεί από επαγγελματίες εξειδικευμένους στις διαδικασίες παραγωγής τροφίμων. Η δοκιμή του συστήματος έγινε σε δύο βιομηχανίες της ΒΙ.ΠΕ. Βόλου με διαφορετικές κατηγορίες ζυμούμενων τροφίμων.

4.1 Πιλοτική δοκιμή του συστήματος σε βρώσιμες ελιές

Η πρώτη πιλοτική δοκιμή του συστήματος που αναπτύχθηκε, πραγματοποιήθηκε σε μια βιομηχανία παραγωγής και τυποποίησης βρώσιμης ελιάς. Η εγκατάσταση του συστήματος με τους αισθητήρες έγινε σε μια δεξαμενή μικρότερης χωρητικότητας από τις κανονικές αλλά προσομοίωνε όλη τη διαδικασία της ζύμωσης (Εικόνα 4.1) . Η διάρκεια της πιλοτικής δοκιμής που επιχειρήσαμε στην συγκεκριμένη βιομηχανία ήταν δύο μήνες. Οι παράμετροι που επιχειρήσαμε να μετρήσουμε ήταν το pH, η ηλεκτρική αγωγιμότητα και η θερμοκρασία του υγρού της άλμης, στο οποίο οι βρώσιμες ελιές υφίστανται τη διαδικασία της ζύμωσης. Για τον έλεγχο των παραπάνω παραμέτρων, χρησιμοποιήθηκαν οι αντίστοιχοι αισθητήρες pH, ηλεκτρικής αγωγιμότητας και θερμοκρασίας, όπως περιεγράφηκαν αναλυτικά στην παράγραφο 3.2.1.1 Συλλογή δεδομένων από τους αισθητήρες. Για να καταφέρουμε να εγκαταστήσουμε τη συσκευή μας, τροποποιήσαμε το καπάκι της δεξαμενής, ώστε να έχουμε πρόσβαση στο εσωτερικό της, αλλά με τέτοιο τρόπο ώστε να εξασφαλίσουμε ότι η διαδικασία της ζύμωσης θα γίνει σε αναερόβιο περιβάλλον. Κατά τη χρονική διάρκεια της πιλοτικής δοκιμής, καταφέραμε να συλλέξουμε δεδομένα από δύο διαφορετικές παρτίδες παραγωγής. Ο έλεγχος των δεδομένων από τις μετρήσεις πραγματοποιούνταν σε καθημερινή βάση μέσω της εφαρμογής, ενώ βρισκόμασταν και σε συνεχή επικοινωνία με το υπεύθυνο προσωπικό της βιομηχανίας για την επίλυση τυχόν προβλημάτων.

Εικόνα 4.1: Πιλοτική δοκιμή του συστήματος σε ζύμωση βρώσιμων ελιών

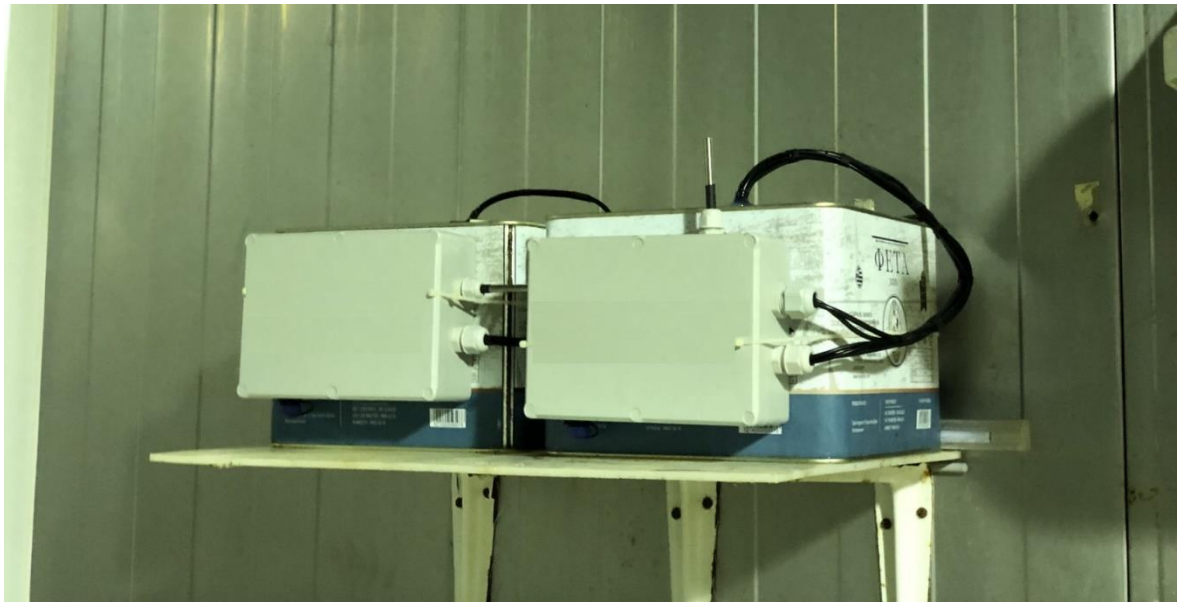
Στόχος ήταν να εξετάσουμε την ακρίβεια των μετρήσεων από τους αισθητήρες σε σύγκριση με τα εργαστηριακά εργαλεία που διαθέτει μια βιομηχανία, τη συνδεσιμότητα της συσκευής με την cloud υπηρεσία μας και την εξοικείωση των επαγγελματιών της βιομηχανίας με μία τεχνολογική λύση που απλοποιεί τον ποιοτικό έλεγχο κατά την παραγωγή των ζυμούμενων τροφίμων.

Η ακρίβεια των αισθητήρων στις τιμές των παραμέτρων που συλλέγουν, επαληθευτήκαν από τα εργαστηριακά μηχανήματα και πετύχαμε ακρίβεια μέχρι και το τρίτο δεκαδικό της μέτρησης. Ωστόσο, επειδή το τμήμα του αισθητήρα που έρχεται σε επαφή με το προϊόν είναι αναλώσιμο και χρειάζεται αλλαγή ανά συγκεκριμένο χρονικό διάστημα, δεν είναι γνωστό σε βάθος χρόνου πως θα συμπεριφερθούν οι μετρήσεις. Η συνδεσιμότητα της συσκευής στη συγκεκριμένη εγκατάσταση ήταν σταθερή καθώς συνδεθήκαμε στο δίκτυο WiFi της βιομηχανίας. Το μειονέκτημα με το συγκεκριμένο τρόπο είναι ότι σε μεγαλύτερη κλίμακα εγκατάστασης οι περισσότερες βιομηχανίες δεν επιτρέπουν την σύνδεση πολλών τρίτων συσκευών στο δίκτυό τους για λόγους ασφαλείας**.** Η ανατροφοδότηση που λάβαμε από τους ανθρώπους του ποιοτικού ελέγχου της βιομηχανίας ήταν ότι η συγκεκριμένη λύση θα μπορούσε να αποτελέσει εργαλείο για τον ποιοτικό έλεγχο που κάνουν καθημερινά καθώς για πρώτη φορά είχαν τα δεδομένα τους

οργανωμένα και μπορούσαν να παρακολουθήσουν σε πραγματικό χρόνο τις διαδικασίες της παραγωγής μέσα από γραφήματα και πίνακες δεδομένων, χωρίς να απαιτείται η διαδικασία της δειγματοληψίας.

4.2 Πιλοτική δοκιμή του συστήματος σε βιομηχανία γαλακτοκομικών

Η δεύτερη πιλοτική δοκιμή του συστήματος πραγματοποιήθηκε σε μια βιομηχανία παραγωγής γαλακτοκομικών προϊόντων και συγκεκριμένα η δοκιμή έγινε κατά τη διαδικασία ζύμωσης και ωρίμανσης της φέτας. Η διάρκεια της συγκεκριμένης πιλοτικής δοκιμής ήταν τρείς μήνες. Η εγκατάσταση έγινε σε δύο διαφορετικά δοχεία που περιείχαν φέτα όπως φαίνεται και στην Εικόνα 4.2. Ειδικότερα ακολουθήθηκε διαφορετική διαδικασία στο κάθε δοχείο με σκοπό την σύγκριση των δύο μεθόδων και την επίδρασή τους στο τελικό προϊόν. Οι παράμετροι που μετρήσαμε και στα δύο δοχεία ήταν το pH, η θερμοκρασία και η ηλεκτρική αγωγιμότητα στο υγρό της άλμης και επιπλέον στο ένα δοχείο πραγματοποιήθηκε μέτρηση και της τιμής του pH της φέτας. Για την εγκατάσταση των συσκευών στα δύο διαφορετικά δοχεία χρειάστηκε να γίνει μια προσαρμογή στο καπάκι του κάθε δοχείου, με τρόπο τέτοιο ώστε να εξασφαλίσουμε την αναερόβια κατάσταση. Όπως και στην προηγούμενη εγκατάσταση, έτσι και στη συγκεκριμένη, τα δεδομένα από τις μετρήσεις ελέγχονταν μέσω της εφαρμογής καθημερινά.



Εικόνα 4.2: Πιλοτική δοκιμή του συστήματος σε βιομηχανία γαλακτοκομικών προϊόντων

Καθώς το περιβάλλον της συγκεκριμένης εγκατάστασης παρουσίαζε σημαντικές διαφορές σε σχέση με αυτό της προηγούμενης, προέκυψαν δύο δυσκολίες οι οποίες έπρεπε να αντιμετωπιστούν στην υλοποίησή μας. Η πρώτη αφορούσε στο περιβάλλον, στο οποίο έγινε η εγκατάσταση των συστημάτων, όπου υπήρχε μεγάλη συγκέντρωση υγρασίας και έκθεσή αυτών στο νερό. Η δεύτερη δυσκολία σχετιζόταν με τη συνδεσιμότητα των συσκευών ως προς την τροφοδοσία τους και το διαδίκτυο. Η δυσκολία της συνδεσιμότητας προέκυψε εξαιτίας των χώρων ζύμωσης και ωρίμανσης, οι οποίοι ήταν μεγάλα δωμάτια από μεταλλικά πάνελ, στα οποία ήταν δύσκολο να υπάρξει πρίζα για τροφοδοσία και η ισχύς του σήματος του WiFi της βιομηχανίας δεν ήταν αρκετή ώστε να περάσει μέσα από αυτά. Η συγκεκριμένη βιομηχανία ήταν πολύ μικρότερη σε μέγεθος σε σχέση με την βιομηχανία του πρώτου πιλοτικού, γεγονός που την οδηγούσε να υλοποιεί τις διαδικασίες της περισσότερο με εμπειρικούς τρόπους παρά με πραγματικά δεδομένα. Τέλος, παρά τις δυσκολίες που αντιμετωπίσαμε στην συγκεκριμένη πιλοτική δοκιμή, τα σχόλια που πήραμε από τους ανθρώπους της βιομηχανίας ήταν ότι κατάφεραν να έχουν σε πλήρη έλεγχο την παραγωγή τους με πραγματικά δεδομένα που περιγράφουν την κατάσταση του προϊόντος απομακρυσμένα.

Κεφάλαιο 5 Συμπεράσματα και μελλοντικές προοπτικές

Ύστερα από τις πιλοτικές δοκιμές σε πραγματικές συνθήκες του συστήματος που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας και με τις δυσκολίες που προέκυψαν τόσο κατά τις δοκιμές, όσο και κατά την ανάπτυξη, καταλήξαμε στα παρακάτω συμπεράσματα αλλά και σε μελλοντικές επεκτάσεις που θα μπορούσαν να συμβάλλουν στο τελικό αποτέλεσμα.

5.1 Συμπεράσματα

Αναλύοντας τα αποτελέσματα που συλλέχθηκαν από τις δοκιμές που διενεργήθηκαν σε πραγματικές συνθήκες, προέκυψε το συμπέρασμα ότι μια τεχνολογική λύση που αποτελείται από ένα σύστημα αισθητήρων, το οποίο χρησιμοποιεί τεχνολογίες IoT, θα ήταν χρήσιμη για την παρακολούθηση της ζύμωσης κατά την παραγωγή των ζυμούμενων τροφίμων. Η κύρια ανατροφοδότηση που λάβαμε και από τις δύο βιομηχανίες μετά το πέρα των πιλοτικών δοκιμών, στέκεται στο γεγονός της έγκαιρης καθώς και της έγκυρης όπως αποδείχτηκε έπειτα από σύγκριση με τα δικά τους όργανα μέτρησης, ενημέρωσης που είχαν οι υπεύθυνοι της παραγωγής συνολικά από την τεχνολογική μας λύση. Με βάση την παραπάνω ανατροφοδότηση, η τεχνολογική μας πρόταση σε μία μαζική και βιομηχανική μορφή ξεπερνώντας τα προβλήματα που παρουσιάστηκαν, θα οδηγούσε σε μείωση των απωλειών της παραγωγής των ζυμούμενων τροφίμων λόγω της παρακολούθησης σε πραγματικό χρόνο που προσφέρει. Επιπλέον αυτό μας οδήγησε στο συμπέρασμα ότι με τις δυνατότητες που έχουν τα τεχνολογικά εργαλεία στις μέρες μας, μπορούν να αντικαταστήσουν διαδικασίες ρουτίνας που γίνονται στις βιομηχανίες, οι οποίες θέτουν ακόμη και σε κίνδυνο την ακεραιότητα των ανθρώπων που δουλεύουν σε αυτές.

Με την χρήση τεχνολογιών IoT, όπως αυτή που αναπτύχθηκε στην συγκεκριμένη διπλωματική, οι βιομηχανίες τροφίμων πλέον ανεξάρτητα από το μέγεθος παραγωγής τους, θα μπορούν να μετατρέψουν την παραγωγή τους από “experience-driven” σε “data-driven”, αξιοποιώντας τα δεδομένα που παράγουν. Με την ανατροφοδότηση που δόθηκε από τους ανθρώπους που χρησιμοποιήσαν την ολοκληρωμένη τεχνολογική μας λύση συμπεραίνουμε ότι πλέον οι βιομηχανίες στρέφονται όλο και περισσότερο σε

τεχνολογικές λύσεις που φέρουν νέες τεχνολογίες και ότι έχουμε φτάσει στο σημείο που το προσωπικό που δουλεύει σε αυτές αντιλαμβάνεται την αναγκαιότητα τέτοιων λύσεων.

Τα μειονεκτήματα της συγκεκριμένης υλοποίησης σχετίζονται με τον συγκεκριμένο μικροελεγκτή για την ανάπτυξη του συστήματος που περιέχει τους αισθητήρες και προωθεί τις τιμές. Ειδικότερα, η τροφοδοσία αυτού θα πρέπει να είναι σταθερή και μόνιμη. Αυτό σημαίνει ότι απαιτείται η σύνδεσή του σε μία πρίζα, πράγμα που προκαλεί δυσκολίες στην εγκατάσταση του. Η χρήση ωστόσο μπαταρίας για τον συγκεκριμένο μικροελεγκτή δεν αποτελεί βιώσιμη λύση γιατί η μεγάλη επεξεργαστική ισχύ που διαθέτει είναι ανάλογη των ενεργειακών απαιτήσεων. Επιπρόσθετα η συνδεσιμότητα στο διαδίκτυο που προσφέρει μέσω του WiFi απαιτεί την ύπαρξη τοπικής σύνδεσης. Η ισχύς του σήματός του WiFi σε πολλές βιομηχανίες, λόγω των εκτάσεων αλλά και των εμποδίων που πρέπει να καλύψει, δεν αρκεί ώστε να δημιουργηθεί μια σταθερή και αξιόπιστη σύνδεση. Τέλος, για λόγους ασφαλείας η κάθε βιομηχανία δεν προτιμάει την είσοδο συσκευών από τρίτους στο τοπικό της δίκτυο.

5.2 Μελλοντικές προοπτικές

Η τεχνολογική λύση που υλοποιήθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας, παρά τα σημαντικά πλεονεκτήματα που την χαρακτηρίζουν, έχει περιθώρια εξέλιξης. Ειδικότερα, μία μελλοντική πρόταση για την βελτίωσή της, θα ήταν η δημιουργία ενός μηχανισμού απομακρυσμένης ενημέρωσης και αναβάθμισης του λογισμικού της συσκευής ανίχνευσης. Η διαδικασία της απομακρυσμένης ενημέρωσης θα συνέβαλε σημαντικά τόσο στην άμεση και μαζική επίλυση τυχόν προβλημάτων που προκύπτουν, όσο και στην προσθήκη νέων χαρακτηριστικών που αφορούν στη διαχείριση των συσκευών και τις μετρήσεις των αισθητήρων. Επιπρόσθετα, οι περιορισμοί που προκύπτουν λόγω του αδύναμου σήματος του WiFi στις μεγάλες εκτάσεις των βιομηχανιών, θα μπορούσαν να ξεπεραστούν με την προσθήκη GSM module [32] στη συσκευή ανίχνευσης, το οποίο προσφέρει τη δυνατότητα 4G σύνδεσης. Τέλος, μία επιπλέον πρόταση για μελλοντική βελτίωση της συγκεκριμένης υλοποίησης αποτελεί η μελέτη, σχεδίαση και κατασκευή ενός κελύφους της συσκευής ανίχνευσης, το οποίο θα ακολουθεί τις προδιαγραφές που ορίζονται από τις βιομηχανίες τροφίμων.

Βιβλιογραφία

[1] ««Food Loss and Food Waste,» FAO,» [Ηλεκτρονικό]. Available: http://www.fao.org/food-loss-and-food-waste/en. [Πρόσβαση 10 02 2021].

[2] Ε. Συμβούλιο, «Μείωση των απωλειών και της σπατάλης τροφίμων,» 2021. [Ηλεκτρονικό]. Available: https://www.consilium.europa.eu/el/policies/food-losses-waste/. [Πρόσβαση 1 11 2021].

[3] Wikipedia.org, «Food Processing,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Food_processing. [Πρόσβαση 15 02 2021].

[4] M. Battcock και S. Azam-Ali, «FERMENTED FRUITS AND VEGETABLES. A GLOBAL PERSPECTIVE.,» [Ηλεκτρονικό]. Available: https://www.fao.org/3/x0560e/x0560e00.htm. [Πρόσβαση 10 11 2021].

[5] K. Spyropoulou, N. Chorianopoulos, P. Skandamis και G.-J. ENychas, «Survival of Escherichia coli O157:H7 during the fermentation of Spanish-style green table olives (conservolea variety) supplemented with different carbon sources,» [Ηλεκτρονικό]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0168160500005109. [Πρόσβαση 20 10 2021].

[6] «Adwa Waterproof EC/TDS/ Temp Pocket Tester,» [Ηλεκτρονικό]. Available: https://labtexbd.com/product/adwa-waterproof-ec-tds-temp-pocket-tester-high-range-ad-32/. [Πρόσβαση 21 02 2022].

[7] «HIAC 9703+,» [Ηλεκτρονικό]. Available: https://www.industrie-online.com/en/p/produit-2/controle-mesure-vision/analyse-et-procedes/laboratoire-equipement-et-procedes/beckman-coulter-9703. [Πρόσβαση 21 02 2022].

[8] R. Docs, «Raspberry Pi Foundation,» [Ηλεκτρονικό]. Available: https://www.raspberrypi.org/. [Πρόσβαση 15 02 2022].

[9] Wikipedia, «Raspberry Pi,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi. [Πρόσβαση 15 06 2021].

[10] «Raspberry Pi zero W Diagram,» 16 01 2017. [Ηλεκτρονικό]. Available: https://www.waveshare.com/raspberry-pi-zero-wh-package-d.htm. [Πρόσβαση 15 02 2022].

[11] «Google Cloud Platform,» [Ηλεκτρονικό]. Available: https://cloud.google.com/docs. [Πρόσβαση 20 02 2022].

[12] «Knowledge about Google Cloud Platform that you cannot omit,» [Ηλεκτρονικό]. Available: https://namtech.com.au/knowledge-about-google-cloud-platform-that-you-cannot-omit/. [Πρόσβαση 15 02 2022].

[13] «Django Project,» [Ηλεκτρονικό]. Available: https://www.djangoproject.com/start/overview/. [Πρόσβαση 10 01 2022].

[14] «Django Book,» [Ηλεκτρονικό]. Available: https://django-book.readthedocs.io. [Πρόσβαση 10 01 2022].

[15] «Model View Controller,» 19 12 2018. [Ηλεκτρονικό]. Available: https://medium.com/@joespinelli_6190/mvc-model-view-controller-ef878e2fd6f5. [Πρόσβαση 17 02 2022].

[16] «MQTT (Message Queuing Telemetry Transport) Protocol,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/MQTT. [Πρόσβαση 10 03 2021].

[17] Wikipedia, «Bash (Unix shell),» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Bash_(Unix_shell). [Πρόσβαση 15 01 2022].

[18] «Postgre SQL,» [Ηλεκτρονικό]. Available: https://www.postgresql.org/. [Πρόσβαση 19/02/2022].

[19] «Apache server,» [Ηλεκτρονικό]. Available: https://httpd.apache.org/. [Πρόσβαση 19 02 2022].

[20] «How to enable SPI master driver in Raspbian using raspi-config,» [Ηλεκτρονικό]. Available: https://www.techcoil.com/blog/how-to-enable-spi-master-driver-in-raspbian-using-raspi-config/. [Πρόσβαση 15 02 2022].

[21] A. wiki, «APT (software) wiki,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/APT_(software). [Πρόσβαση 09 08 2021].

[22] P. Docs, «Python Docs - venv,» [Ηλεκτρονικό]. Available: https://docs.python.org/3/library/venv.html. [Πρόσβαση 05 09 2021].

[23] Wikipedia, «Wikipedia, One-Wire,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/1-Wire. [Πρόσβαση 20 12 2021].

[24] P. Docs, «Python Docs Subprocess,» [Ηλεκτρονικό]. Available: https://docs.python.org/3/library/subprocess.html. [Πρόσβαση 27 11 2021].

[25] «Eclipse Paho MQTT,» [Ηλεκτρονικό]. Available: https://www.eclipse.org/paho/. [Πρόσβαση 08 02 2022].

[26] Wikipedia, «Wikipedia Certificate authority,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Certificate_authority. [Πρόσβαση 11 01 2022].

[27] Wikipedia, «JSON Web Token Wiki,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/JSON_Web_Token. [Πρόσβαση 25 01 2022].

[28] Pypi, «Google Cloud Python Client,» [Ηλεκτρονικό]. Available: https://pypi.org/project/google-cloud/. [Πρόσβαση 10 Φεβρουάριος 2022].

[29] C. Docs, «Celery Project Documentation,» [Ηλεκτρονικό]. Available: https://docs.celeryproject.org/en/stable/django/first-steps-with-django.html. [Πρόσβαση 20 01 2022].

[30] «Flask web framework,» [Ηλεκτρονικό]. Available: https://flask.palletsprojects.com/en/2.0.x/. [Πρόσβαση 19 02 2022].

[31] R. Docs, «RabbitMQ Documentantion,» [Ηλεκτρονικό]. Available: https://www.rabbitmq.com/documentation.html. [Πρόσβαση 10 01 2022].

[32] «GSM,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/GSM_modem. [Πρόσβαση 11 02 2022].

Παράρτημα

Επιμέρους συνεισφορά συγγραφέων

Η παρούσα διπλωματική εργασία αναπτύχθηκε σε συνεργασία από τους Ακρίβο Κωνσταντίνο και Κολιγλιάτη Αριστοτέλη, καθώς η τεχνολογική λύση που παρουσιάζεται δεν θα μπορούσε να χωριστεί σε δύο διαφορετικές εργασίες για λόγους πολύ στενής αλληλεπίδρασης των επιμέρους υποέργων. Το πρόβλημα, το οποίο διερευνάται απαιτεί την ύπαρξη όλων των επιμέρους αντικειμένων (συσκευή ανίχνευσης, υποδομή Cloud, Web εφαρμογή), των οποίων η υλοποίηση περιγράφεται στο Κεφάλαιο 3 , έτσι ώστε να μπορεί να γίνει κατανοητό και να παρουσιαστούν κατάλληλα τα συμπεράσματα.

Σε ότι αφορά στην υλοποίηση της συσκευής ανίχνευσης, της διασύνδεσής της με την Cloud υποδομή μας καθώς και της διαχείρισής της μέσω αυτής, αυτά ερευνήθηκαν και αναπτύχθηκαν από τον Κολιγλιάτη Αριστοτέλη. Η υλοποίηση της λειτουργικότητας που αφορά την εγκατάσταση και ανάπτυξη της cloud υποδομής, την λήψη και την διαχείριση των δεδομένων από τις συσκευές ανίχνευσης καθώς και την web εφαρμογή, ερευνήθηκε και αναπτύχθηκε από τον Ακρίβο Κωνσταντίνο. Τέλος οι πιλοτικές δοκιμές πραγματοποιήθηκαν με την συμβολή και των δύο.