ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Προσδιορισμός & Εκτέλεση IoT Σεναρίων για Smart Plugs

(IoT Scenario Modelling and Execution for Smart Plugs)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σταύρος Βραχνής

Επιβλέπων : Κυριάκος Κρητικός

Μέλη εξεταστικής επιτροπής: Χαράλαμπος Σκιάνης, Νίκος Παπαδάκης

Σάμος, Ιούλιος 2021

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πρόλογος και ευχαριστίες

Θα ήθελα να εκφράσω τις βαθύτατες ευχαριστίες μου στον επιβλέποντα καθηγητή μου κύριο Κυριάκο Κρητικό, που μου προσέφερε την ευκαιρία να προτείνω θέματα που μου κεντρίζουν το ενδιαφέρον και κατόπιν συζήτησης να καταλήξουμε σε ένα από αυτά. Η βοήθεια του ήταν πολύτιμη για την σχεδίαση και την επιλογή των καταλληλότερων τεχνολογιών αποτελώντας αρωγό τόσο στην εκπόνηση αυτής της εργασίας, όσο και στη μετέπειτα εξέλιξη μου. Επιπρόσθετα, θα ήθελα να ευχαριστήσω κάθε άτομο που στάθηκε δίπλα μου καθ' όλη τη διάρκεια των σπουδών μου και ιδιαίτερα την οικογένεια μου, χωρίς τον μόχθο των οποίων δεν θα ήταν δυνατή η ολοκλήρωση των σπουδών μου.

Σταύρος Βραχνής

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

Λίστα Σχημάτων

Λίστα Πινάκων

Ακρωνύμια

IoT	Internet Of Things
WSN	Wireless Sensor Network
RDBMS	Relational Databases
API	Application Programming Interface
PHP	Hypertext PreProcessor
HTML	Hypertext Markup Language
DB	Database
SQL	Structured Query Language
ACID	Atomicity, Consistency, Isolation & Durability
CRUD	Create, Read, Update & Delete
WiFi	Wireless Fidelity
LAN	Local Area Network
IEEE	Institute of Electrical and Electronics Engineers
P2P	Peer 2 Peer
EEPROM	Electrically Erasable Programmable Read-Only Memory
QSPI	Queued Serial Peripheral Interface
CPU	Central Processing Unit
UI	User Interface
IDE	Integrated Development Environment
SDK	Software Development Kit
OS	Operation System
URL	Uniform Resource Locator
DOM	Document Object Model

Περίληψη

Τα τελευταία χρόνια ένα ξεχωριστό επιστημονικό πεδίο προσπαθεί να εισβάλει ολοένα και πιο ενεργά στην καθημερινή μας ζωή, γνωστό ως το διαδίκτυο των πραγμάτων (Internet Of Things). Ο παραπάνω όρος δεν είναι καινούριος, άρχισε να γίνεται ιδιαίτερα γνωστός το καλοκαίρι του 2010 από την Google, με το project Street View. Έχουν ήδη κυκλοφορήσει πολλές εφαρμογές IoT στην αγορά για την υποστήριξή του, παρόλα αυτά όμως, όπως δείχνει η αργή πορεία εξέλιξης του, υπάρχει ανάγκη για βελτιστοποίηση απόδοσης καθώς και η εύρεση ενός πιο αφηρημένου τρόπου διαχείρισης συσκευών IoT από τον χρήστη. Σκοπός της παρούσας εργασίας είναι μια πλήρης σχεδίαση και υλοποίηση ενός ευέλικτου, ασφαλούς συστήματος διαχείρισης συσκευών Internet of Things με χρήση σεναρίων (scenarios). Μέσω του συστήματος αυτού, κάθε χρήστης θα είναι σε θέση να εκμεταλλεύεται οποιοδήποτε είδος πηγής πληροφορίας, είτε εσωτερική (αισθητήρες) είτε εξωτερική (δεδομένα διαδικτύου μέσω διεπαφών), για τον προσδιορισμό και την εκτέλεση των σεναρίων. Τα δεδομένα διαδικτύου διευρύνουν κατά πολύ τις επιλογές του χρήστη στην συγγραφή σεναρίων ενώ παράλληλα αποτελούν εναλλακτικές πηγές δεδομένων που ο χρήστης δεν αναγκάζεται να πληρώσει. Επίσης, το σύστημα αυτό προσφέρει την δυνατότητα επεξεργασίας των δεδομένων αισθητήρων μέσω στατιστικών συναρτήσεων, προσφέροντας πιο αξιόλογες και ολοκληρωμένες παρατηρήσεις για την διεξαγωγή ορθών συμπερασμάτων. Τέλος, εκτός από την ανάλυση του προτεινόμενου συστήματος, η διπλωματική αυτή εργασία αναφέρεται και στον τρόπο κατασκευής και ενοποίησης ηλεκτρονικών συσκευών (Things) με το σύστημα αυτό. Οι συσκευές αυτές μπορεί να είναι είτε αισθητήρες για την παραγωγή δεδομένων ή διακόπτες για την εφαρμογή δράσεων.

Λέξεις Κλειδιά: *Internet Of Things, παρακολούθηση δεδομένων, σενάρια, διαχείριση συσκευών, αντικείμενα, χρονοσειρές, στατιστική επεξεργασία δεδομένων IoT.*

Abstract

The Internet of Things is now being embedded into our life, in a more active way. The term is not new, it started to become popular back to 2010, when Google was running the Street View project. Already, there a lot of IoT applications in the market, but as its slow evolution shows, there are a lot of points that need improvement. The most critical points are the security and the abstract management of these devices. The scope of this thesis is to design and implement a flexible, secure platform for managing IoT devices using scenarios. Through these scenarios, the user will be able to totally manage all kind of data sources, internal (sensor's data) or external (Internet data), in order to command his/her things, thus covering both simple and more complex IoT use cases. Further, by allowing the use of Internet data, the user is no longer required to pay for various sensor's devices in order to create corresponding IoT scenario conditions. The proposed system is also able to manipulate sensor metrics and data using statistical functions, in order to create more generic and suitable conditions covering the enactment of appropriate actions in the context of IoT scenarios. Finally, in addition to the analysis of the proposed system, this dissertation also refers to the way electronic devices can be constructed and integrated into this system. These devices can be either sensors for generating data or switches for performing corresponding actions.

Keywords: Internet Of Things, monitoring, scenarios, devices management, things, time series, IoT data statistical processing.

1 *Εισαγωγή*

1.1 Το διαδίκτυο των πραγμάτων

Το Διαδίκτυο των πραγμάτων (IoT) περιγράφει το δίκτυο των φυσικών αντικειμένων - γνωστά ως "αντικείμενα" - που είναι ενσωματωμένα με αισθητήρες, λογισμικό και άλλες τεχνολογίες με σκοπό τη σύνδεση και την ανταλλαγή δεδομένων με άλλες συσκευές και συστήματα μέσω του Διαδικτύου [1][2]. Τέτοια αντικείμενα μπορούν να αποτελούν οικιακές συσκευές, οχήματα, δρόμοι, βιομηχανικά ή ιατρικά εργαλεία και γενικά μπορούν να πάρουν την μορφή οποιασδήποτε ηλεκτρονικής συσκευής. Η ενσωμάτωση αυτών των αντικειμένων στο ανθρώπινο περιβάλλον έρχεται να δώσει μια σειρά από λύσεις σε καθημερινά αλλά και πιο περίπλοκα προβλήματα. Έξυπνα σπίτια, πόλεις, εργοστάσια και γενικά έξυπνα συστήματα έχουν ήδη αναπτυχθεί αξιοποιώντας στο έπακρο τις διασυνδεδεμένες τεχνολογίες. Σύμφωνα με τους οραματιστές και αναλυτές του IoT, το τελευταίο δεν αποτελεί μια αυτοτελή τεχνολογία αλλά περισσότερο μια ιδέα ή έναν επιστημονικό κλάδο που συνδυάζει ένα πλήθος από τεχνολογίες.

1.2 Η ανάπτυξη του IoT

Ο παραπάνω όρος δεν είναι καινούριος, επινοήθηκε στα τέλη της δεκαετίας του 90΄ και πιο συγκεκριμένα το 1999 από τον επιχειρηματία Kevin Ashton, έναν από τους ιδρυτές του auto-id center στο MIT. Ο Kevin Ashton εργαζόταν τότε στο Procter & Gamble, στο τμήμα βελτιστοποίησης της αλυσίδας εφοδιασμού. Σε μια από τις παρουσιάσεις του, ήθελε να προσελκύσει την προσοχή των ανώτερων διευθυντών με μια νέα συναρπαστική τεχνολογία που ονομάζεται RFID.. Κατά την παρουσίαση αυτή αναφέρθηκε για πρώτη φορά ο όρος the Internet of Things [3] και από τότε καθιερώθηκε η χρήση του. Άρχισε όμως να γίνεται ιδιαίτερα γνωστός το καλοκαίρι του 2010 από την Google. Σύμφωνα με διαρροές η εταιρία παράλληλα με τον έργο

Street Views, έκανε τεράστια συλλογή από δεδομένα των τοπικών δικτύων χρηστών, ευρετηριάζοντας έτσι τον φυσικό κόσμο [3]. Την ίδια χρονιά η κυβέρνηση της Κίνας ανακοίνωσε ότι θα έκανε το διαδίκτυο των πραγμάτων στρατηγική προτεραιότητα στο πενταετές σχέδιο τους [3]. Το 2012, μεγάλα επιστημονικά περιοδικά ανάμεσα τους το Forbes, η Fast Company, και το Wired άρχισαν να εισάγουν τον όρο IoT στα δημοσιεύματά τους ως ένα έκτακτο τεχνολογικό φαινόμενο [3]. Τα επόμενα χρόνια ακολούθησαν και άλλοι οργανισμοί, γνωστοποιώντας το ενδιαφέρον τους για τον παραπάνω επιστημονικό κλάδο. Σύμφωνα με τις τότε έρευνες, η αγορά του IoT θα άγγιζε τα 8.9 τρισεκατομμύρια δολάρια μέσα στο έτος 2020 [3]. Σήμερα, η παγκόσμια αγορά του κλάδου δεν ξεπερνά τα 800 δισεκατομμύρια δολάρια [4], πολύ κάτω δηλαδή από την παραπάνω πρόβλεψη, γεννώντας ερωτηματικά στους αναλυτές του είδους (δες την επόμενη αμέσως ενότητα για τα ζητήματα που εμποδίζουν την περαιτέρω ανάπτυξη του κλάδου).

1.3 Αντικείμενο διπλωματικής

Το IoT θεωρείται πλέον ξεχωριστός επιστημονικός κλάδος και όχι απλή εφαρμογή τεχνολογίας. Τα τελευταία 20 χρόνια, η ανάπτυξη του κλάδου αυτού φαίνεται να πραγματοποιείται με αργούς και όχι τόσο σταθερούς ρυθμούς. Στο πεδίο της έρευνας, γίνονται συνεχόμενα καινούριες προσπάθειες για την εύρεση νέων και πιο αποτελεσματικών μεθόδων διαχείρισης των αντικειμένων. Όσον αφορά την αγορά, οι εταιρίες έχουν ήδη δημιουργήσει τα πρώτα εμπορικά προϊόντα, τα οποία έχουν δοκιμαστεί από εκατομμύρια χρήστες. Παρόλα αυτά όμως έχουν επικριθεί για πλεονασμό υλικού, έλλειψη ασφάλειας και ελάχιστη αξία χρήσης [5]. Οι εταιρίες παράγουν περισσότερα είδη προϊόντων από ότι χρειάζεται. Πολλές φορές είναι απαραίτητη η ύπαρξη επιπλέον υλικού, επιβαρύνοντας οικονομικά τον χρήστη, προκειμένου τα αντικείμενα να μπορέσουν να λειτουργήσουν. Πολλά από αυτά θα μπορούσαν να συμπτυχθούν με άλλα ή να χρησιμοποιηθούν πηγές δεδομένων του διαδικτύου αντί αυτών. Καθώς, το λογισμικό των συσκευών βρίσκεται πολύ κοντά στο υλικό, δημιουργείται ένα τεράστιο πρόβλημα στο κομμάτι της ασφάλειας [6][7]. Τέλος, επειδή αυτά τα προϊόντα δυσκολεύονται να εκμεταλλευτούν αποτελεσματικά τις διαθέσιμες πηγές δεδομένων, η αξία χρήσης ελαχιστοποιείται, κάνοντας τους χρήστες να αδιαφορήσουν για αυτά.

Στόχος της εργασίας είναι ο σχεδιασμός και υλοποίηση ενός ευέλικτου, ασφαλούς συστήματος διαχείρισης συσκευών IoT με τη χρήση σεναρίων (scenarios). Μέσω των τελευταίων, κάθε χρήστης θα είναι σε θέση να εκμεταλλεύεται οποιοδήποτε είδος πηγής πληροφορίας, τόσο εσωτερικές πηγές (αισθητήρες) όσο και εξωτερικές (δεδομένα διαδικτύου μέσω διεπαφών), για τον προσδιορισμό και την εκτέλεση των σεναρίων. Επιπλέον, δίνεται έμφαση στο πως αυτά τα δεδομένα σε συνδυασμό με πλήθος δράσεων μπορούν να καλύψουν ακόμη περισσότερες περιπτώσεις χρήσης IoT. Σημαντική επίσης, είναι και η διαδικασία εύρεσης αξιόπιστων και ενεργειακά αποδοτικών τρόπων αποθήκευσης και ανάλυσης των δεδομένων που θα παράγουν ταυτόχρονα δισεκατομμύρια συσκευές, οι οποίες θα ομαδοποιούνται σε έξυπνους τόπους. Με τον όρο “έξυπνο τόπο” εννοείται η ομαδοποίηση αντικειμένων που βρίσκονται στην ίδια γεωγραφική περιοχή. Ο παραπάνω όρος αναπαριστά ένα σύνολο που περιλαμβάνει τους όρους “έξυπνο σπίτι”, “έξυπνη πόλη” και γενικά οποιοδήποτε “έξυπνο” υποσύνολο από αντικείμενα IoT.

1.4 Δομή της διπλωματικής

Αρχικά, στο Κεφάλαιο 2, πραγματοποιείται έρευνα και ανάλυση εφαρμογών IoT που υπάρχουν ήδη στην αγορά επισημαίνοντας αδυναμίες και κρίσιμα ζητήματα. Στην συνέχεια, στο Κεφάλαιο 3 περιγράφονται οι απαιτήσεις που θα πρέπει να ικανοποιεί η λύση που αναπτύσσεται. Δίνεται επίσης η πλήρη σχεδίαση, αρχιτεκτονική, τα σχήματα δεδομένων, η υλοποίηση και ο τρόπος ικανοποίησης αυτών των απαιτήσεων. Στο Κεφάλαιο 4, γίνεται αναφορά στην οικογένεια μικροελεγκτών ESP8266. Επίσης παρουσιάζονται όλα τα εξαρτήματα που επιλέχτηκαν για την κατασκευή των έξυπνων αντικειμένων, τα οποία και χρησιμοποιούνται από το πρωτότυπο της προτεινόμενης λύσης για την παραγωγή μετρήσεων και την διενέργεια σχετικών δράσεων. Στο Κεφάλαιο 5, αναλύονται όλα τα συστατικά μέρη της ψευδογλώσσας που χρησιμοποιούνται για την συγγραφή σεναρίων IoT. Στο Κεφάλαιο 6 παρουσιάζονται αναλυτικά, βήμα προς βήμα, οι δυνατότητες που προσφέρει το σύστημα στον χρήστη ενώ ταυτόχρονα γίνεται και επίδειξη σεναρίων, τόσο απλών όσο και περίπλοκων. Τέλος, στο Κεφάλαιο 7 πραγματοποιείται σύνοψη της εργασίας και αναφέρονται σημεία που θα πρέπει να επεκταθούν, προκειμένου η προτεινόμενη εφαρμογή να είναι πραγματικά βιώσιμη στην αγορά.

2 *Εφαρμογές IoT Στην Σημερινή Εποχή*

2.1 Εισαγωγή

Στο κεφάλαιο αυτό γίνεται μια έρευνα και παρουσίαση λύσεων που υπάρχουν στην σημερινή εποχή. Κύριος στόχος της έρευνας αυτής είναι η επισήμανση των αδυναμιών που παρουσιάζουν αυτές οι εφαρμογές με σκοπό την δημιουργία αποδοτικών λύσεων και καινοτομιών. Γίνεται λοιπόν εστίαση στα προβλήματα που έρχεται να λύσει αυτή η εργασία. Τα πιο σημαντικά από αυτά τα προϊόντα αναλύονται στις επόμενες υπό-ενότητες.

2.2 Εφαρμογές

Το Internet of Things αποτελεί κάτι περισσότερο από μια διευκόλυνση για τους καταναλωτές, δεδομένου ότι δημιουργεί νέες πηγές πληροφοριών, νέα επιχειρηματικά μοντέλα, νέες υπηρεσίες και νέα καινοτόμα προϊόντα σε πολλούς κλάδους.

2.2.1 IKEA Smart Home

Η IKEA πρωτοεμφάνισε το έργο Smart Home το 2012 [8], προσδοκώντας να αλλάξει για πάντα την ζωή των χρηστών της, στο σπίτι. Η κύρια ιδέα ήταν η διαχείριση των ηλεκτρικών συσκευών της, όπως λάμπες, ηχεία, αφυγραντήρες και κουρτίνες, μέσω εφαρμογής κινητού, την οποία και ονόμασαν “TRADFRI App”. Το 2019, η εφαρμογή μετονομάστηκε σε “IKEA Smart Home App” και η εταιρία αποφάσισε να επενδύσει ακόμη περισσότερο στο έργο [9]. Μέχρι σήμερα ακολούθησαν πολλές τροποποιήσεις του έργου για την εξασφάλιση της ασφάλειας, ευκολίας χρήσης αλλά και την ευελιξία του συστήματος.

Σύμφωνα με την τελευταία ενημέρωση, οι χρήστες προκειμένου να διαχειριστούν τις συσκευές τους, θα πρέπει απαραίτητα να εξοπλιστούν με μια επιπλέον ηλεκτρονική συσκευή, γνωστή ως πύλη (gateway) που φέρει την ονομασία “TRADFRI Gateway” και να την συνδέσουν στο τοπικό τους δίκτυο (WiFi) μέσω καλωδίου Ethernet. Στην συνέχεια θα πρέπει να συνδέσουν κάθε συσκευή με την παραπάνω πύλη κρατώντας τα προηγούμενα σε κοντινή απόσταση για μερικά δευτερόλεπτα. Η εγκατάσταση φαίνεται να είναι αρκετά περίπλοκη για τον μέσο χρήστη, για αυτό και η εταιρία παρέχει αναλυτικούς οδηγούς. Αφού ολοκληρώσουν την παραπάνω διαδικασία, οι χρήστες μπορούν να διαχειριστούν και να χρονοπρογραμματίσουν τις συσκευές τους από το κινητό, το οποίο θα πρέπει αναγκαστικά να είναι συνδεδεμένο με το ίδιο οικιακό δίκτυο. Επίσης, μια ενδιαφέρουσα επιλογή είναι η ομαδοποίηση ρυθμίσεων πολλαπλών συσκευών σε σκηνές (scenes), όπως αυτές αποκαλούνται. Έτσι ο χρήστης μπορεί να ρυθμίσει (ένταση φωτισμού, ήχου) μαζικά πλήθος συσκευών με το άγγιγμα ενός κουμπιού ανάλογα την διάθεση του.

2.2.2 Kasa Smart

Το Kasa Smart είναι ένα έργο της εταιρίας TP-LINK, η οποία είναι γνωστή στην αγορά για τα προσφερόμενα προϊόντα δικτύου της. Ιδρύθηκε το 2015 [10], με αφορμή τις έξυπνες πρίζες που είχαν κεντρίσει κατά πολύ το ενδιαφέρον της τότε αγοράς. Σήμερα προσφέρει πλήθος προϊόντων όπως έξυπνες κουρτίνες, κάμερες, πρίζες και πολλά άλλα. Στόχος της εταιρίας, όπως η ίδια επισημαίνει, δεν είναι απλά η ύπαρξη πλούσιου υλικού (hardware) αλλά και κατάλληλου λογισμικού (software) προκειμένου να σχεδιαστούν πραγματικά έξυπνα σπίτια.

Παρόμοια με το έργο της ΙΚΕΑ, η TP-LINK προσφέρει την εφαρμογή Kasa Smart για την διαχείριση των συσκευών της, αλλά αυτή την φορά από οπουδήποτε στο κόσμο, αρκεί να υπάρχει σύνδεση στο διαδίκτυο. Η εγκατάσταση των συσκευών είναι αρκετά απλή, δεν απαιτείται καμία πύλη ή ειδικό υλικό. Ο χρήστης χρησιμοποιεί την εφαρμογή για να συνδέσει την συσκευή στο οικιακό του δίκτυο, εισάγοντας τα απαραίτητα διαπιστευτήρια. Η εφαρμογή, προσφέρει άμεση διαχείριση των συσκευών, χρονοπρογραματισμούς, απλές συνθήκες if-then-else χρησιμοποιώντας ως πηγές δεδομένων αποκλειστικά τις συσκευές, και τέλος είναι συμβατή με την εικονική βοηθό Alexa της Amazon. Η παραπάνω λύση φαίνεται να ανταποκρίνεται στις απαιτήσεις τις εποχής, όμως υπάρχει μεγάλος περιορισμός στα διαθέσιμα δεδομένα, αφού ως πηγές αντικειμένων προσφέρονται μόνο κάμερες για την ανίχνευση κίνησης. Ουσιαστικά ο χρήστης μπορεί μόνο να ορίσει κανόνες με βάση δεδομένα χρόνου ή ανίχνευσης κίνησης.

2.3 Κρίσιμα ζητήματα εφαρμογών IoT

Αναλύοντας την αργή εξέλιξη του IoT καταλαβαίνει κανείς ότι δεν υπάρχει μια και μόνο ορθολογικά σωστή επιλογή για την ανάπτυξη ενός πλήρους συστήματος διαχείρισης συσκευών IoT. Εξετάζοντας τις παραπάνω λύσεις σε βάθος, μπορούμε να εντοπίσουμε τις αδυναμίες τους, να σκεφτούμε εναλλακτικές και να δημιουργήσουμε καινοτομίες.

Κρίσιμα ζητήματα εφαρμογών IoT αποτελούν η ευκολία εγκατάστασης των συσκευών και χρήσης του πληροφοριακού συστήματος. Σύμφωνα με τον ορισμό του IoT, οι συσκευές θα πρέπει να αλληλεπιδρούν μεταξύ τους. Θα πρέπει, λοιπόν, να υπάρχουν προκαθορισμένες οδηγίες, σενάρια (scenarios) που θα χρησιμοποιούν πλήθος πηγών δεδομένων, θα τις επεξεργάζονται, και θα διαχειρίζονται συσκευές ή θα πραγματοποιούν άλλες δράσεις. Το πιο κρίσιμο ζήτημα των εφαρμογών αυτού του τομέα είναι η ανάγκη για επεκτασιμότητα, ευελιξία και επεξεργασία των δεδομένων [11]. Τα σενάρια θα πρέπει να μπορούν να αντλήσουν δεδομένα από οποιαδήποτε πηγή του διαδικτύου και όχι μόνο από αισθητήρες που ο χρήστης αναγκάζεται να κατέχει κατόπιν οικονομικής επιβάρυνσης. Οι συσκευές θα πρέπει να είναι προσβάσιμες από οπουδήποτε στον κόσμο, ώστε να υπάρχει ο πλήρης έλεγχος ανά πάσα στιγμή από τον χρήστη. Τέλος, θα πρέπει να υπάρχει η δυνατότητα στατιστικής επεξεργασίας των δεδομένων για την λήψη αποφάσεων σύμφωνα με ένα πιο εκτεταμένο σύνολο παρατηρήσεων. Για παράδειγμα, σε περιπτώσεις δεδομένων αισθητήρων είναι απαραίτητο η δράση να πραγματοποιείται με βάση μια στατιστική συνάρτηση πάνω από ένα σύνολο από δεδομένα και όχι λαμβάνοντας υπόψη μόνο την τελευταία μέτρηση. Η διαχείριση των συσκευών είναι αρκετά απλουστευμένη και μη ικανή να καλύψει ένα μεγάλο ποσοστό από σενάρια χρήσης. Η έλλειψη ικανοποιητικού επιπέδου πολυπλοκότητας και εκφραστικότητας σεναρίων μπορεί να οδηγήσει εν τέλει σε μια ασταθή και προβληματική διαχείριση των συσκευών IoT. Στο παρακάτω πίνακα γίνεται μια σύνοψη των αδυναμιών που εντοπίστηκαν στις εφαρμογές που αναλύθηκαν στις προηγούμενες δύο ενότητες.

	IKEA Smart Home	**Kasa Smart**
Ευκολία εγκατάστασης	✘ (δύσκολο για τον μέσο χρήστη)	✓ (εγκατάσταση μέσω λογισμικού)
Απευθείας σύνδεση με router	✘ (απαιτείται η ύπαρξη επιπλέον υλικού - πύλης)	✓ (δεν απαιτείται η ύπαρξη επιπλέον υλικού)
Ευκολία χρήσης	✓ (αρκετά εύκολη)	✓ (αρκετά εύκολη)
Άμεση διαχείριση συσκευών	✓ (μέσω εφαρμογής κινητού)	✓ (μέσω εφαρμογής κινητού)
Περιγραφή σύνθετων σεναρίων	✘ (δεν υποστηρίζεται)	~ (απλοϊκά σενάρια τύπου if-then-else)
Εξωτερικές πηγές δεδομένων	✘ (μόνο δεδομένα συσκευών που κατέχει ο χρήστης)	✘ (μόνο δεδομένα συσκευών που κατέχει ο χρήστης)
Επεξεργασία δεδομένων	✘ (χρήση μόνο της τελευταίας μέτρησης)	✘ (χρήση μόνο της τελευταίας μέτρησης)
Προσβασιμότητα από το διαδίκτυο	✘ (μόνο εντός οικιακού δικτύου)	✓ (από οπουδήποτε στον κόσμο)
Συμβατότητα με προϊόντα / υπηρεσίες τρίτων	✘ (καμία)	✓ (Virtual Alexa)

Πίνακας 1: Σύγκριση χαρακτηριστικών & δυνατοτήτων εφαρμογών IoT

3 *Σχεδίαση Εφαρμογής*

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο καθορίζονται οι λειτουργικές και μη λειτουργικές απαιτήσεις του πληροφοριακού συστήματος που πρόκειται να υλοποιηθεί. Γίνεται αναφορά σε σύγχρονες τεχνολογίες βάσεων δεδομένων, και εργαλεία ανοιχτού κώδικα που μπορούν να συνδράμουν στην λύση των προβλημάτων που εμφανίζουν οι εφαρμογές IoT στην σημερινή εποχή. Αναλύεται πλήθος πιθανών λύσεων, πραγματοποιείται σύγκριση και η επιλογή ορισμένων από αυτές καθώς και έπειτα παρέχεται η πλήρης αρχιτεκτονική, σχεδίαση και υλοποίηση της επιλεχθείσας προσέγγισης επισημαίνοντας τα πλεονεκτήματα της και τον βαθμό ικανοποίησης των απαιτήσεων που καθορίστηκαν.

3.2 Απαιτήσεις

Στο προηγούμενο κεφάλαιο αναφέρθηκαν τα κρίσιμα ζητήματα εφαρμογών IoT που υπάρχουν στην σημερινή εποχή. Στο κεφάλαιο αυτό θα μετατρέψουμε αυτά τα ζητήματα σε απαιτήσεις του καινούριου συστήματος. Γενικά, οι απαιτήσεις εκφράζουν το τι θα πρέπει να κάνει το πληροφοριακό σύστημα ως σύνολο από την σκοπιά της επιχείρησης, χωρίς να αναφέρονται σε τεχνικά χαρακτηριστικά. Συχνά αποτελούν μια μορφή συμβολαίου ανάμεσα στον πελάτη και τον κατασκευαστή. Συνήθως κατά την ανάλυση και το σχεδιασμό οι απαιτήσεις αναφέρονται σε δυο ξεχωριστές υποκατηγορίες, τις λειτουργικές και μη λειτουργικές.

3.2.1 Λειτουργικές Απαιτήσεις

Οι λειτουργικές απαιτήσεις (functional requirements) δηλώνουν το τι θα πρέπει να κάνει το σύστημα, δηλαδή εκφράζουν τις δυνατότητες που έχει ο χρήστης. Οι τελευταίες μπορούν να παρομοιαστούν και ως συναρτήσεις που λαμβάνουν είσοδο και δίδουν έξοδο. Ο χρήστης θα πρέπει να έχει την δυνατότητα:

- να δημιουργεί λογαριασμό για την αποκλειστική διαχείριση των αγαθών του. Για να δοθεί πρόσβαση χρήσης από το σύστημα, θα προηγείται ταυτοποίηση μέσω ηλεκτρονικής διεύθυνσης και συνθηματικού (password).
- να μπορεί να δημιουργεί και να διαχειρίζεται έξυπνους τόπους (smart places).
- να διαχειρίζεται συσκευές που του ανήκουν ή κατέχει την σχετική εξουσιοδότηση από τον κάτοχο. Η διαχείριση θα πρέπει να περιλαμβάνει την εισαγωγή των συσκευών σε έξυπνους τόπους, την επεξεργασία των συσκευών και την διαγραφή τους.
- να έχει πρόσβαση και να οπτικοποιεί τα δεδομένα των συσκευών του.
- να μπορεί να εφαρμόζει στατιστικές συναρτήσεις σε δεδομένα μετρήσεων με σκοπό την διεξαγωγή συμπερασμάτων.
- να διαχειρίζεται σενάρια (εισαγωγή, επεξεργασία, ενεργοποίηση/απενεργοποίηση και διαγραφή).
- πρόσβασης σε δεδομένα που εκθέτουν καταστάσεις πυροδότησης αλλαγών στα αγαθά του χρήστη (traceability).
- αναζήτησης εξωτερικών διεπαφών που έχουν οριστεί από άλλους χρήστες καθώς και εισαγωγής των δικών του διεπαφών.
- να χρησιμοποιεί ορισμένες τρίτες υπηρεσίες ως πηγές δεδομένων (πχ Alexa για φωνητικές εντολές) ή για την εφαρμογή δράσεων (πχ ειδοποιήσεις).

3.2.2 Μη Λειτουργικές Απαιτήσεις

Οι μη λειτουργικές απαιτήσεις περιγράφουν το πως ή το πόσο καλά το σύστημα θα υποστηρίξει τις λειτουργικές απαιτήσεις. Συνήθως εκφράζονται βάση χαρακτηριστικών όπως είναι η απόδοση (performance), χρηστικότητα (usabillity), ασφάλεια (security), νομιμότητα (legislative) και ιδιωτικότητα (privacy). Οι ακόλουθες μη λειτουργικές απαιτήσεις τέθηκαν για το σύστημα που αναπτύχτηκε στα πλαίσια της διπλωματικής αυτής εργασίας:

- Το σύστημα θα πρέπει να διαχειρίζεται τεράστιους όγκους δεδομένων σε ελάχιστο χρόνο. Θα πρέπει, λοιπόν να δοθεί ιδιαίτερη προσοχή στο πως τα δεδομένα αποθηκεύονται, επεξεργάζονται και ανακτώνται εξασφαλίζοντας υψηλά επίπεδα απόδοσης.
- Το σύστημα θα πρέπει να εκτελεί τα σενάρια σε κατάλληλους χρόνους, ώστε να εξασφαλίζεται η ορθή λειτουργία των συσκευών και ταυτόχρονα να αποφεύγεται η

υπερφόρτωση του συστήματος. Έτσι τα αντικείμενα είναι άμεσα διαθέσιμα για εκτέλεση οδηγιών ανά πάσα στιγμή χωρίς καθυστερήσεις.

- Ο χρήστης θα πρέπει να έχει πλήθος επιλογών κατά την έκφραση των σεναρίων. Αυτό περιλαμβάνει πληθώρα πηγών δεδομένων και συναρτήσεων δράσεων καθώς και την δυνατότητα έκφρασης περίπλοκων συνθηκών. Ως πηγές δεδομένων δεν θα πρέπει να εννοούνται μόνο οι μετρήσεις που παράγονται από τις συσκευές του αλλά και δεδομένα του διαδικτύου που είναι διαθέσιμα μέσω διεπαφών. Έτσι κάθε χρήστης θα είναι σε θέση να δημιουργεί σενάρια IoT ακόμη και αν δεν μπορεί να καλύψει το κόστος αισθητήρων διευρύνοντας έτσι το πελατολόγιο της εφαρμογής (target group). Οι δράσεις δεν θα πρέπει να περιορίζονται μόνο σε αλλαγές ιδιοτήτων των αντικειμένων αλλά και σε συναρτήσεις που επιτρέπουν την επικοινωνία με τρίτες υπηρεσίες (π.χ Discord, Telegram ή οτιδήποτε). Τέλος, οι συνθήκες θα πρέπει να επιτρέπουν στον χρήστη να συνδυάσει πολλαπλές πηγές δεδομένων και να ξεφεύγουν από τις απλοϊκές συνθήκες του τύπου if-then-else που υπάρχουν μέχρι σήμερα.
- Το σύστημα θα πρέπει να ελέγχει την ορθότητα των σεναρίων κατά την υποβολή τους, αποτρέποντας την καταχώριση εσφαλμένου κώδικα. Επίσης, θα πρέπει να καθοδηγεί τον χρήστη με κατατοπιστικά και επεξηγηματικά σχόλια για τον εντοπισμό και διόρθωση των συντακτικών σφαλμάτων.
- Ο χρήστης θα πρέπει να μπορεί με ευκολία να εγκαταστήσει συσκευές στους έξυπνους τόπους του αλλά και να τους διαχειριστεί.
- Ο χρήστης θα πρέπει να έχει πρόσβαση στις συσκευές IoT από οπουδήποτε στον κόσμο, αρκεί να υπάρχει πρόσβαση στο διαδίκτυο.
- Το σύστημα θα πρέπει να αποτρέπει κάθε είδους μη εξουσιοδοτημένης πρόσβασης.
- Το σύστημα θα πρέπει να τηρεί αυστηρά τους γενικούς κανόνες προστασίας προσωπικών δεδομένων.

3.3 Εντοπισμός Τεχνολογιών

Στη προηγούμενη ενότητα έγινε καθορισμός των απαιτήσεων, αντιμετωπίζοντας έτσι τα κρίσιμα ζητήματα που αδυνατούν να λύσουν οι σημερινές εφαρμογές IoT. Σε αυτή την ενότητα παρουσιάζονται τεχνολογίες που μπορούν να καλύψουν τις παραπάνω ανάγκες. Καθώς η μεγαλύτερη πρόκληση που διαπραγματεύεται μια εφαρμογή IoT είναι η αποτελεσματική διαχείριση τεράστιων όγκων δεδομένων, στην συνέχεια γίνεται αναφορά σε τεχνολογίες βάσεων δεδομένων.

3.3.1 Σχεσιακές Βάσεις Δεδομένων

Για την αποθήκευση δεδομένων του χρήστη, όπως των στοιχείων λογαριασμού, των έξυπνων τόπων, των συσκευών, των σεναρίων και γενικά οποιονδήποτε πληροφοριών που απαιτούν συσχέτιση, θα πρέπει να γίνει επιλογή κατάλληλης τεχνολογίας. Οι σχεσιακές βάσεις δεδομένων

είναι μια τέτοια τεχνολογία που επιτρέπει την αποθήκευση δεδομένων διατηρώντας μια προκαθορισμένη δομή πίνακα. Κάθε εγγραφή στον πίνακα, γνωστή και ως πλειάδα, περιλαμβάνει απαραίτητα ένα μοναδικό κλειδί (unique id), με σκοπό την δυνατότητα αναφοράς σε αυτή. Οι βάσεις αυτές ακολουθούν το σχεσιακό μοντέλο, έτσι ώστε κάθε πλειάδα να μπορεί να σχετίζεται με μια ή περισσότερες πλειάδες διαφορετικού πίνακα [12]. Θα πρέπει επίσης να αναφερθεί, ότι τηρούν το σύνολο των ιδιοτήτων ACID (Atomicity, Consistency, Isolation, Durability) εξασφαλίζοντας την ορθή συναλλαγή των δεδομένων. Με τον όρο ατομικότητα (Atomicity) η βάση εξασφαλίζει ότι είτε μια συναλλαγή θα είναι εντελώς ολοκληρωμένη ή αποτυχημένη. Στην περίπτωση που μια δήλωση περιέχει κάποιο σφάλμα δεν θα πραγματοποιηθεί καμιά αλλαγή. Κατά την δεύτερη ιδιότητα της συνέπειας, τα δεδομένα που εισάγονται θα πρέπει να τηρούν προκαθορισμένους κανόνες συνοχής, ώστε η βάση δεδομένων να βρίσκεται πάντοτε σε μια συνεπή κατάσταση. Για παράδειγμα, ένας κανόνας μπορεί να καθορίσει ότι δεν επιτρέπονται διπλές πλειάδες σε έναν πίνακα προκειμένου να εξαλειφθεί η πιθανότητα εσφαλμένων πληροφοριών που εισέρχονται στη βάση δεδομένων. Με τον τρίτο όρο, την απομόνωση (isolation), εξασφαλίζεται ότι μια συναλλαγή δεν μπορεί να επηρεαστεί από άλλες αν αυτές δεν έχουν ολοκληρωθεί. Τέλος, με τον τελευταίο όρο, ανοχή (durability), η βάση επιβεβαιώνει οτι μετά την ολοκλήρωση μια συναλλαγής (commit), η πληροφορία αποθηκεύεται στο σύστημα, ακόμη και αν ακολουθήσει κάποιο κρίσιμο σφάλμα συστήματος.

Το απλό αλλά ισχυρό και ώριμο σχεσιακό μοντέλο χρησιμοποιείται από οργανισμούς όλων των τύπων και μεγεθών για την κάλυψη μιας ευρείας γκάμας αναγκών. Οι παραπάνω βάσεις χρησιμοποιούνται για την καταγραφή αποθεμάτων, τη διεκπεραίωση συναλλαγών ηλεκτρονικού εμπορίου, τη διαχείριση κρίσιμης πληροφορίας πελατών και πολλά άλλα. Μια σχεσιακή βάση δεδομένων μπορεί να εξεταστεί για οποιαδήποτε ανάγκη πληροφοριών στην οποία τα σημεία δεδομένων σχετίζονται μεταξύ τους και πρέπει να διαχειρίζεται με ασφαλή, βασισμένο σε κανόνες, συνεπή τρόπο. Οι σχεσιακές βάσεις δεδομένων υπάρχουν από τη δεκαετία του 1970 [13], καθιερώνοντας την ως μια ώριμη τεχνολογία που έχει αναπτυχθεί αρκετά. Σήμερα, τα πλεονεκτήματα του σχεσιακού μοντέλου συνεχίζουν να το καθιστούν το πιο ευρέως αποδεκτό μοντέλο για βάσεις δεδομένων.

Από την άλλη μεριά, οι σχεσιακές βάσεις δεδομένων αδυνατούν να αποδώσουν στην διαχείριση μεγάλου όγκου δεδομένων, όπως απαιτείται σε έργα IoT. Καθώς γίνεται εισαγωγή περισσότερων πλειάδων και γνωρισμάτων, η ανάγκη για φυσική μνήμη κλιμακώνεται σημαντικά. Επίσης, όπως αναφέρθηκε προηγουμένως, τα δεδομένα ακολουθούν αυστηρά μια προκαθορισμένη δομή αποθήκευσης, πράγμα που δεν επιτρέπει την εύκολη εισαγωγή νέων χαρακτηριστικών μετρήσεων. Ακόμη και αν πραγματοποιούταν αλλαγές στο σχήμα της βάσης χειροκίνητα, μετά από κάποιο σημείο, οι σχέσεις μεταξύ των πλειάδων θα ήταν τρομερά περίπλοκες.

3.3.1.1 MySQL

Η MySQL είναι μια από τις πιο δημοφιλής, ανοιχτού κώδικα σχεσιακή βάση δεδομένων, που μετρά πάνω από 11 εκατομμύρια εγκαταστάσεις παγκοσμίως. Ιδρύθηκε το 1995 από τον Michael Widenius. Το εκτελέσιμο της στήνει έναν εξυπηρετητή (server) παρέχοντας πρόσβαση πολλών

χρηστών σε ένα σύνολο βάσεων. Χρησιμοποιεί κυρίως την μηχανή βάσης (storage engine) InnoDB, υποκείμενο στοιχείο λογισμικού δηλαδή, που χρησιμοποιείται για τη δημιουργία, ανάγνωση, ενημέρωση και διαγραφή δεδομένων. Οι χρήστες μπορούν να διαχειρίζονται τα δεδομένα της βάσης υποβάλλοντας δηλώσεις SQL (Structured Query Language). Προσφέρει πλήθος βιβλιοθηκών πελάτη καλύπτοντας μια ευρεία γκάμα γλωσσών προγραμματισμού. Αποτελεί συστατικό στοιχείο της στοίβας διαδικτυακού λογισμικού LAMP (Linux, Apache, MySQL, PHP/Python/Perl). Επίσης χρησιμοποιείται από τις δημοφιλέστερες πλατφόρμες διαχείρισης περιεχομένου ιστού (CMS – Content Management System), όπως Wordpress, Drupal, Joomla και phpBB, ενώ έχει υιοθετηθεί από γνωστούς ιστοτόπους, ανάμεσα στους οποίους ξεχωρίζουν το Facebook, Twitter και το YouTube. Τέλος, εφαρμόζει το σύνολο των ιδιοτήτων ACID, εξασφαλίζοντας αξιόπιστες συναλλαγές δεδομένων. [14]

3.3.1.2 MariaDB

Η MariaDB, αποτελεί επίσης μια σχεσιακή βάση δεδομένων, που ο πυρήνας της βασίζεται στην MySQL. Προορίζεται να παραμείνει δωρεάν και ανοιχτού λογισμικού βάσει της άδειας GNU (General Public License). Ιδρύθηκε το 2009 και η ανάπτυξη της διευθύνεται από μερικούς από τους αρχικούς ιδρυτές της MySQL. Οι βασικές νέες προσθήκες είναι ότι υποστηρίζει και άλλες μηχανές αποθήκευσης (storage engines) όπως την Aria, ColumnStore και MyRocks. Οι προσθήκες αυτές επιτρέπουν στον χρήστη να επιλέξει από ένα ακόμα μεγαλύτερο σύνολο αυτή που ταιριάζει καλύτερα με το έργο του. Επίσης, έχουν πραγματοποιηθεί μεγάλες βελτιώσεις στους χρόνους απόκρισης, καθιστώντας την MariaDB συγκριτικά πιο γρήγορη από την μητρική της. Τέλος, μπορεί να δεχθεί πάνω από 200.000 ταυτόχρονες συναλλαγές τηρώντας πιστά το σύνολο ιδιοτήτων ACID, αυξάνοντας την απόδοση των εφαρμογών που την χρησιμοποιούν. [15]

3.3.1.3 PostgreSQL

Η PostgreSQL ή αλλιώς Postgres είναι και αυτή μια σχεσιακή βάση δεδομένων ανοιχτού κώδικα που δίνει έμφαση στην επεκτασιμότητα και την συμβατότητα με ερωτήματα SQL. Ιδρύθηκε το 1989 στο πανεπιστήμιο της Καλιφόρνιας, Berkeley. Έχει σχεδιαστεί για να χειρίζεται μια σειρά φορτίων εργασίας, από μεμονωμένα μηχανήματα έως αποθήκες δεδομένων ή υπηρεσίες ιστού (web services) με πολλούς ταυτόχρονους χρήστες. Σύμφωνα με τους ιδρυτές της προσφέρει υψηλή απόδοση, επεκτασιμότητα, γρήγορους χρόνους απόκρισης και ενεργή κοινότητα. Προσφέρει πλήθος βιβλιοθηκών πελάτη καλύπτοντας μια ευρεία γκάμα γλωσσών προγραμματισμού. Τέλος, αποτελεί την προκαθορισμένη βάση δεδομένων του MacOS, ενώ είναι διαθέσιμη και για άλλα λειτουργικά συστήματα όπως Linux, FreeBSD, Windows και OpenBSD. [16]

3.3.1.4 Σύγκριση Βάσεων

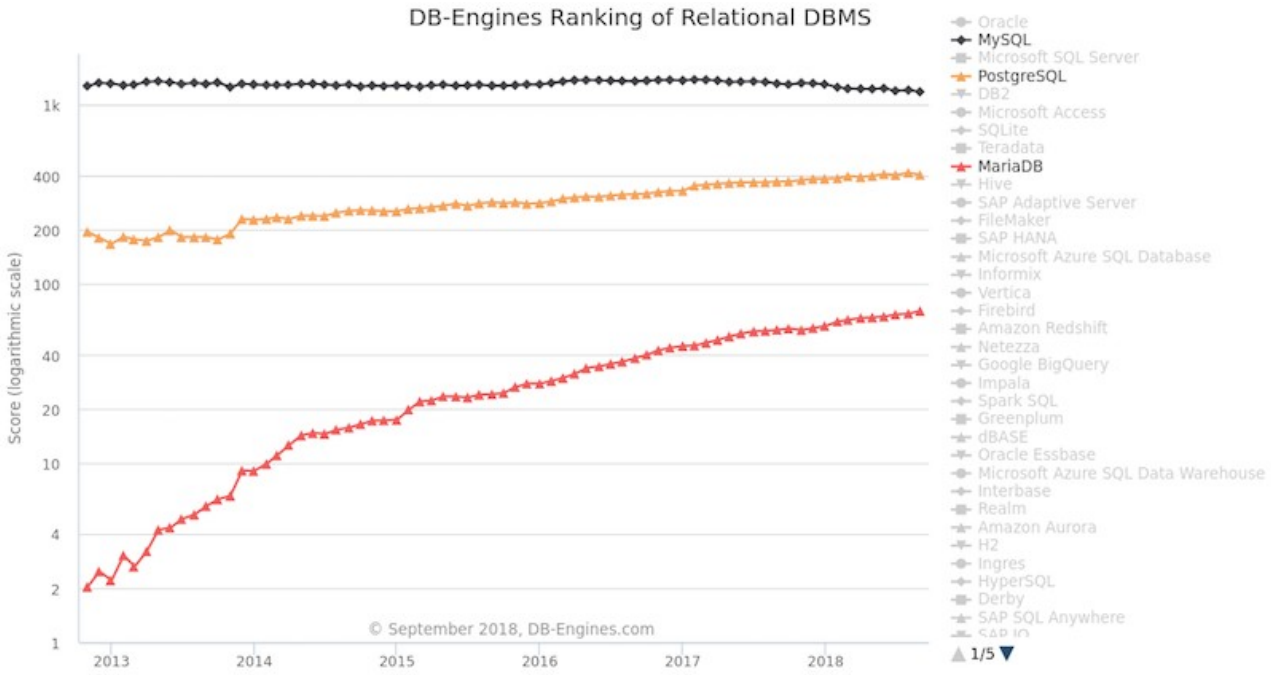
Οι παραπάνω βάσεις έχουν πολλά κοινά χαρακτηριστικά και είναι αρκετά γνωστές στον χώρο. Προκειμένου να γίνει η κατάλληλη επιλογή λήφθηκαν σοβαρά υπόψη οι αξιολογήσεις της DB-Engines. Η τελευταία αποτελεί μια υπηρεσία αξιολόγησης βάσεων δεδομένων αναλύοντας ανά σύντομα χρονικά διαστήματα στατιστικές μηχανών αναζήτησης, κοινωνικών δικτύων, ζήτηση εργασιών και συζητήσεις τεχνικού περιεχομένου. Στην συνέχεια ακολουθεί ένας πίνακας σύγκρισης χαρακτηριστικών [17][18].

	MySQL	**PostgreSQL**	**MariaDB**
Engine Ranking Score	1228.38	577.15	97.98
Overall Rank	2	4	12
Storage Engines	8	1	12
Threading Pool Size	200.000	100	200.000+
Dynamic Columns	Ναι	Ναι	Όχι

Πίνακας 2: Σύγκριση σχεσιακών βάσεων δεδομένων

Τα δυο πρώτα χαρακτηριστικά αποτελούν αξιολογήσεις του οργανισμού DB-engines. Το πρώτο αφορά την βαθμολογία της βάσης δεδομένων σχετικά με τον μηχανισμό βάσης που χρησιμοποιεί (υψηλότερη βαθμολογία σημαίνει μεγαλύτερη κατάταξη) και το δεύτερο αποτελεί μια συνολική κατάταξη. Το επόμενο χαρακτηριστικό εκφράζει το πλήθος των μηχανών βάσης που χρησιμοποιείται κάθε φορά. Στην συνέχεια, το “Threading pool size” αποτελεί το επιτρεπτό πλήθος συνδέσεων πελατών ανά νήμα. Τέλος, το τελευταίο είναι ένα χαρακτηριστικό που επιτρέπει την δυναμική εισαγωγή διαφορετικών γνωρισμάτων σε πλειάδες του ίδιου πίνακα.

Στην παρακάτω εικόνα φαίνεται η γενική βαθμολογία (DB-Engines) των σχεσιακών βάσεων δεδομένων (πρώτο από τα 5 κριτήρια) που αναφέρθηκαν πιο πάνω σε συνάρτηση με τον χρόνο. Η MySQL και η PostgreSQL, ως ώριμες βάσεις, βρίσκονται ψηλά σε όλο το μήκος του γραφήματος, ενώ παρατηρείται επίσης μια αργή αλλά σταθερή αύξηση της βαθμολογίας της δεύτερης (PostgreSQL). Η MariaDB, ως νεοεισαχθείσα στο χώρο, φαίνεται να ανεβαίνει με μεγαλύτερη κλίση.

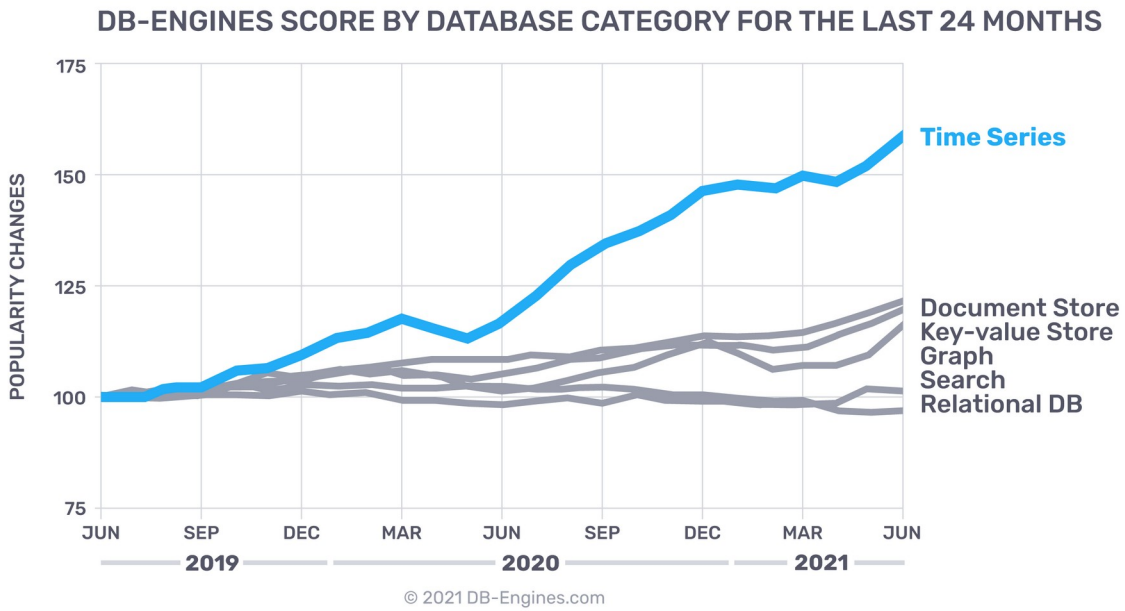


Σχήμα 1 - Αξιολόγηση σχεσιακών βάσεων δεδομένων από την DB-Engines

Τέλος, δεν θα μπορούσαμε να ξεχωρίσουμε απόλυτα κάποια βάση δεδομένων από τις παραπάνω, αφού συνήθως η επιλογή της πραγματοποιείται με βάση τις ανάγκες του έργου και τα επιθυμητά χαρακτηριστικά. Πράγματι, αν κοιτάξουμε το πρώτο και τρίτο κριτήριο, θα παρατηρήσουμε πως η MySQL θεωρείται καλύτερη ως προς το πρώτο κριτήριο (σε σχέση με τις υπόλοιπες) ενώ η MariaDB καλύτερη ως προς το τρίτο.

3.3.2 Βάσεις Δεδομένων Χρονοσειρών

Οι βάσεις δεδομένων Χρονοσειρών (Time Series Databases - TSDB) είναι ειδικά σχεδιασμένες βάσεις για την αποθήκευση ζευγαριών χρόνων (timestamps) και τιμών (data). Συνήθως, πρόκειται για μετρήσεις ή συμβάντα που παρακολουθούνται και συγκεντρώνονται με την πάροδο του χρόνου. Αρχικά, αναπτύχθηκαν για την αποδοτική διαχείριση μεγάλων όγκων δεδομένων αισθητήρων (data historians), αλλά σήμερα υποστηρίζουν μια ευρεία περιοχή εφαρμογών. Μερικά παραδείγματα αποτελούν τα δεδομένα δικτύου, χρηματιστηρίου, απόδοσης εφαρμογών και γενικά στοιχείων που απαιτούν ανάλυση. Τα παραπάνω δεδομένα απαιτούν μια διαφορετική προσέγγιση λόγω του κύκλου ζωής και του τεράστιου όγκου τους που παράγονται από τις πηγές. Οι βάσεις αυτές αποθηκεύουν τις πληροφορίες χωρίς να ακολουθούν μια προκαθορισμένη δομή που επιτρέπει συσχετίσεις, για αυτό και ονομάζονται ως NoSQL βάσεις. Έτσι, επιτρέπεται η διαχείριση ακατάστατων και απρόβλεπτων δεδομένων. Τις περισσότερες φορές, εφαρμόζονται περίπλοκοι αλγόριθμοι συμπίεσης με σκοπό την αποδοτική διαχείριση των δεδομένων. Σύμφωνα με στατιστικές της DB-Engines, οι βάσεις δεδομένων χρονοσειρών γίνονται ολοένα και περισσότερο δημοφιλής και αποδεκτές τα τελευταία χρόνια. [19][20]



Σχήμα 2 - Προτιμήσεις βάσεων δεδομένων από έρευνα της DB-Engines

3.3.2.1 InfluxDB

Η InfluxDB, βρίσκεται στην κορυφή της λίστας βάσεων δεδομένων χρονοσειρών του οργανισμού DB-Engine. Διατίθεται εντελώς δωρεάν και αποτελεί έργο ανοιχτού κώδικα. Γράφτηκε στην γλώσσα Go τον Σεπτέμβρη του 2013 [21] με σκοπό την βέλτιστη ταχύτητα ανάκτησης, αποθήκευσης και επεξεργασίας χρονοσειρών. Χρησιμοποιείται σε ευρεία γκάμα εφαρμογών και όπως αναφέρεται στον επίσημο ιστότοπο της InfluxData, σήμερα βρίσκονται σε λειτουργία πάνω από 450.000 διακομιστές που τρέχουν το παραπάνω λογισμικό [22]. Για την εγγραφή δεδομένων ο χρήστης μπορεί να χρησιμοποιήσει μια σειρά από μεθόδους. Για χρήστες με λιγοστή η μηδαμινή εμπειρία προγραμματισμού ο κατάλληλος τρόπος είναι το “Telegraf” ή “Scrapers” όπως αποκαλούνται από τους ιδρυτές. Στη ουσία επιτρέπουν την συλλογή δεδομένων ανά τακτά χρονικά διαστήματα μέσω γραφικής διεπαφής από προκαθορισμένες υπηρεσίες ιστού (web services) μέσω του πρωτοκόλλου http. Για τους προγραμματιστές υπάρχει πληθώρα επιλογών, όπως η διεπαφή γραμμής εντολών Influx CLI, το InfluxDB API και οι βιβλιοθήκες πελάτη (client libraries). Τα ερωτήματα προς την βάση γίνονται μέσω ερωτημάτων InfluxQL ή Flux που είναι παρόμοιες με την γλώσσα SQL.

Θα πρέπει να αναφερθεί οτι η InfluxDB δεν είναι απλά μια βάση δεδομένων, αφού ταυτόχρονα προσφέρει πλήθος διεπαφών και εργαλείων. Προσφέρει βιβλιοθήκες πελάτη (client libraries) με πλούσια εγχειρίδια σε αρκετές γλώσσες προγραμματισμού, όπως PHP, Java, Go, C#, Kotlin, Javascript/NodeJs, Python, Ruby, Scala, Swift και Arduino. Παρέχει την δυνατότητα δημιουργίας εργασιών (tasks) για την εκτέλεση ερωτημάτων ανά προκαθορισμένα χρονικά διαστήματα. Τέλος, δίνει την δυνατότητα στον χρήστη να δημιουργεί ειδοποιήσεις (alerts) όταν τα δεδομένα που τον ενδιαφέρουν ικανοποιούν ανάλογες συνθήκες.

3.3.2.2 Prometheus

Το Prometheus αποτελεί και αυτό μια βάση δεδομένων χρονοσειρών. Ιδρύθηκε το 2012 από την SoundCloud και υιοθετήθηκε γρήγορα από πολλές εταιρίες και οργανισμούς δημιουργώντας έτσι μια ενεργή κοινότητα [23]. Το μοντέλο δεδομένων που χρησιμοποιεί είναι παρόμοιο με της InfluxDB, δηλαδή αποθηκεύει πληροφορίες ως ζευγάρια κλειδιών και τιμών (key / values pairs). Η βάση χρησιμοποιεί επίσης scrapers, για την εισαγωγή μετρήσεων από υπηρεσίες ιστού. Επίσης παρέχει πλήθος βιβλιοθηκών πελάτη για την εύκολη ενσωμάτωση του έργου από προγραμματιστές άλλων έργων. Τα ερωτήματα προς την βάση γίνονται μέσω της PromQL, μια γλώσσα παρόμοια με την SQL που δημιουργήθηκε από τον ίδιο τον οργανισμό. Επίσης, η βάση παρέχει πλήθος επιλογών απεικόνισης και οπτικοποίησης των δεδομένων. Τέλος, δίνεται η δυνατότητα στον χρήστη να δημιουργεί ειδοποιήσεις (alerts) όταν τα δεδομένα που τον ενδιαφέρουν ικανοποιούν τις ανάλογες συνθήκες. [24]

3.3.2.3 Σύγκριση

Στον παρακάτω πίνακα δίνεται επίσης και η βαθμολογία των βάσεων σύμφωνα με την DB-Engines. [25][26]

	InfluxDB	**Prometheus**
DB-Engine Rank	29.24 (1η θέση)	6.02 (3η θέση)
Συλλογή δεδομένων από υπηρεσίες ιστού (Web Services)	Ναι	Ναι
Ειδοποιήσεις (Alerts)	Ναι	Ναι
Εργασίες (Tasks)	Ναι	Όχι
Χρησιμοποιείται κυρίως για	Αποθήκευση δεδομένων χρόνου	Παρακολούθηση δεδομένων (monitoring)
Μέγεθος χρονοσειρών	nanoseconds	milliseconds
Τύποι δεδομένων	float64, int64, bool & string	Float64 και περιορισμένη υποστήριξη για αλφαριθμητικά

Πίνακας 3: Σύγκριση βάσεων δεδομένων χρονοσειρών

Παρόμοια και με τις σχεσιακές βάσεις δεδομένων, βασικό κριτήριο αποτελεί η βαθμολογία του οργανισμού DB-Engines. Το δεύτερο κριτήριο, εκφράζει αν η βάση επιτρέπει την συλλογή δεδομένων από υπηρεσίες ιστού σε τακτά χρονικά διαστήματα. Ουσιαστικά, ο χρήστης μπορεί να προκαθορίσει APIs, από τα οποία το σύστημα μπορεί να ανακτά δεδομένα συνεχόμενα κατά την πάροδο του χρόνου. Το τρίτο χαρακτηριστικό, δηλώνει αν η βάση προσφέρει ειδοποιήσεις για την ικανοποίηση προκαθορισμένων συνθηκών από τον χρήστη. Οι εργασίες όπως αναφέρθηκαν και

σε προηγούμενη ενότητα είναι εντολές οι οποίες εκτελούνται σε περιοδικούς χρόνους. Στην συνέχεια επισημαίνεται ο κύριος λόγος ανάπτυξης των βάσεων δεδομένων, όπως αναφέρουν οι ιδρυτές τους. Τα επόμενα χαρακτηριστικά, αφορούν το χρονικό μέγεθος και γενικά τους τύπους δεδομένων που υποστηρίζει η κάθε βάση.

Τέλος, πάλι δεν θα μπορούσαμε να ξεχωρίσουμε κάποια βάση δεδομένων, αφού η επιλογή σχετίζεται άμεσα από τις ανάγκες του έργου. Συνήθως αποτελεί και θέμα προτίμησης των σχεδιαστών λογισμικού.

3.4 Προτεινόμενες λύσεις

Κατά τα στάδια της σχεδίασης της εφαρμογής, εξετάστηκαν μια σειρά από λύσεις όσον αφορά την διαχείριση των μετρήσεων και την εκτέλεση των σεναρίων. Για την διεπαφή του χρήστη σε όλες τις πιθανές λύσεις, επιλέχτηκαν οι τεχνολογίες και τα πακέτα html5, bootstrap, php, javascript και jquery, επειδή είναι δοκιμασμένες και ευρέως χρησιμοποιούμενες τεχνολογίες στις οποίες υπήρχε κατάλληλη γνώση και εξειδίκευση από τον προγραμματιστή του συστήματος / εφαρμογής. Θα μπορούσε να πραγματοποιηθεί μια τεχνική ανάλυση διαφορετικών τεχνολογιών αλλά το ζήτημα αυτό δεν ήταν κρίσιμο ως προς τους κύριους στόχους και απαιτήσεις της εφαρμογής.

Στις υπό ενότητες που ακολουθούν παρουσιάζεται αναλυτικά η κάθε λύση. Η σειρά παρουσίασης πραγματοποιείται από την χειρότερη προς την καλύτερη λύση, όπου η τρέχουσα είναι πάντοτε μια πιο βέλτιστη έκδοση της προηγούμενης.

3.4.1 MySQL / PHP & εκτέλεση σεναρίων με cronjobs

Σε αυτή την προσέγγιση για την διαχείριση όλων των δεδομένων προτάθηκε η σχεσιακή βάση δεδομένων MySQL. Η επιλογή της τελευταίας από τις υπόλοιπες σχεσιακές βάσεις που αναφέρονται στις παραπάνω ενότητες, πραγματοποιήθηκε λαμβάνοντας σοβαρά υπόψη το μέγεθος των ταυτόχρονων συνδέσεων που υποστηρίζει καθώς και την υψηλή κατάταξή στο πίνακα γενικής βαθμολογίας της DB-Engines (κριτήρια 1 & 4 στον Πίνακα 4).

Με βάση αυτή την λύση, τα σενάρια θα πρέπει να γραφθούν στην διαδικτυακή γλώσσα προγραμματισμού php και να αποθηκεύονται ως αλφαριθμητικά στο κατάλληλο γνώρισμα της βάσης. Για την εκτέλεση των σεναρίων το σύστημα θα πρέπει να ανακτήσει τους κώδικες από την βάση και να τους εκτελέσει, χρησιμοποιώντας την ειδική συνάρτηση eval() [27]. Η συνάρτηση δέχεται ως όρισμα ένα αλφαριθμητικό που αναπαριστά τον κώδικα php, τον οποίο και εκτελεί. Για την εκκίνηση της παραπάνω ρουτίνας θα μπορούσαν να χρησιμοποιηθούν cronjobs, περιοδικές εργασίες δηλαδή που τρέχουν σε προκαθορισμένο χρόνο λόγω της ανάγκης περιοδικότητας επεξεργασίας δεδομένων για ένα μεγάλο ποσοστό από πιθανά σενάρια. Οι εργασίες αυτές αποτελούν συστατικό μέρος των λειτουργικών συστημάτων, επομένως δεν υπήρχε ανάγκη για εγκατάσταση περαιτέρω εργαλείων.

Στο πλαίσιο της λύσης αυτής, άμεσα προέκυψαν κρίσιμα ζητήματα απόδοσης αλλά και ασφάλειας. Αρχικά, υπήρχε αμφιβολία για την διαχείριση των τεράστιων όγκων δεδομένων που

θα παρήγαγαν οι συσκευές. Συμπεραίνοντας λοιπόν ότι κάθε μέτρηση αντικειμένου είναι αναγκαίο να φθάνει σε χρόνους της κλίμακας δευτερολέπτων, έγινε κατανοητό οτι μια τέτοια βάση δεδομένων δεν θα μπορούσε να ανταπεξέλθει στις απαιτήσεις που είχαν τεθεί. Συγκεκριμένα το πρόβλημα δεν ήταν μόνο το μέγεθος των παραγόμενων δεδομένων, αφού θα μπορούσε εύκολα να λυθεί με μια κάθετη κλιμάκωση (vertical scaling) του διακομιστή εφόσον υπάρχει η αντίστοιχη φυσική χωρητικότητα. Δηλαδή επέκταση των χαρακτηριστικών του μηχανήματος όπως του σκληρού δίσκου, της μνήμης ram, των πυρήνων του επεξεργαστή κ.α, χωρίς να πρέπει να τροποποιηθεί ο κώδικας. Επιπλέον, θα πρέπει να αναφερθεί ότι κατά την διάρκεια την κάθετης κλιμάκωσης, η διαθεσιμότητα της εφαρμογής θα ήταν περιορισμένη ή μηδαμινή. Κρίσιμο ζήτημα αποτελούσε επίσης η ανάγκη για αποδοτική επεξεργασία αυτών των δεδομένων. Για παράδειγμα, η εύρεση μεγίστου, ελαχίστου ή ο υπολογισμός του μέσου όρου από ένα μεγάλο σύνολο μετρήσεων κατά μήκος πολλαπλών χρηστών. Φυσικά αυτές οι στατιστικές συναρτήσεις είναι υπαρκτές και στις σχεσιακές βάσεις δεδομένων, όμως η δομή αποθήκευσης τους δεν επιτρέπει την γρήγορη διεκπεραίωση τέτοιων συναρτήσεων.

Τα αρνητικά στοιχεία, όμως, δεν ήταν μόνο αυτά. Στις εργασίες τύπου cronjobs, ο τρόπος υποβολής τους είναι εξαρτημένος άμεσα από το εκάστοτε λειτουργικό σύστημα του διακομιστή. Επομένως, δημιουργήθηκε άμεσα ένα πρόβλημα συμβατότητας, το οποίο θα μπορούσε να λυθεί είτε αναπτύσσοντας διαφορετικές εκδόσεις της εφαρμογής ανά λειτουργικό σύστημα, είτε εστιάζοντας σε συγκεκριμένα λειτουργικά συστήματα (OS-Locking). Ακόμη και αν προσπερνούσαμε το παραπάνω πρόβλημα, ήταν από την αρχή γνωστό οτι η διαχείριση αυτών των εργασιών ήταν αρκετά περίπλοκη. Η ιδέα της δημιουργίας μιας διεπαφής για την διαχείριση των cronjobs έκρυβε πολλά ζητήματα ασφάλειας, αφού μια εσφαλμένη εισαγωγή μιας τέτοιας εργασίας θα μπορούσε να προκαλέσει ανεπάντεχο σφάλμα ολόκληρου του υπολογιστικού κόμβου φιλοξενίας.

3.4.2 MySQL / PHP / Prometheus & εκτέλεση σεναρίων με cronjobs

Μια άλλη εναλλακτική λύση, επέκταση της προηγούμενης, αφορούσε την καλύτερη και αποδοτικότερη διαχείριση των δεδομένων. Ειδικότερα, η λύση αυτή σχετίζεται με την εισαγωγή τεχνολογίας βάσεων χρονοσειρών για την κάλυψη των αναγκών αποθήκευσης και επεξεργασίας μεγάλου όγκου δεδομένων. Όπως έχει αναφερθεί και στην ενότητα 3.3.2.1, η InfluxDB και η Prometheus αποτελούν βάσεις δεδομένων χρονοσειρών, συνεπώς είναι κατάλληλες για έργα IoT. Σε αυτή την πρόταση υλοποίησης, επιλέχτηκε η Prometheus επειδή σύμφωνα με τους ιδρυτές της εστίαζε συγκεκριμένα σε έργα παρακολούθησης δεδομένων, χαρακτηριστικό που ταίριαζε απόλυτα με το πρόβλημα μας. Αντίθετα η InfluxDB υπόσχεται λύσεις σε μια σειρά από πιο γενικά ζητήματα, παρέχοντας περισσότερα εργαλεία και εγχειρίδια, κρίνοντας την έτσι ως πλεονάζουσα. Η κεντρική ιδέα ήταν η απομόνωση των μετρήσεων στην βάση χρονοσειρών, βελτιώνοντας έτσι την ταχύτητα διεκπεραίωσης στατιστικών συναρτήσεων. Ενώ τα σενάρια, οι πληροφορίες λογαριασμών και συσκευών θα παρέμεναν στην MySQL σχεσιακή βάση δεδομένων. Η εκτέλεση των σεναρίων θα πραγματοποιούνταν με βάση την προηγούμενη εναλλακτική, δηλαδή της ενότητας 3.4.1.

Ενώ η βάση χρονοσειρών έλυνε αποτελεσματικά το πρόβλημα διαχείρισης των δεδομένων, η εκτέλεση των σεναρίων γεννούσε και πάλι πολλές αμφιβολίες. Όπως αναφέρθηκε και στην παραπάνω ενότητα, τα προβλήματα συμβατότητας και ασφάλειας είναι εμφανή και χαρακτηρίζουν και την τρέχουσα λύση, η οποία είναι πανομοιότυπη στο μέρος που αφορά την εκτέλεση των σεναρίων.

3.4.3 MYSQL / PHP / InfluxDB & εκτέλεση σεναρίων με tasks v1

Σύμφωνα με τις προηγούμενες λύσεις που είχαν προταθεί, ήταν πλέον γνωστό οτι θα έπρεπε να γίνει μια διαφορετική προσέγγιση στον τρόπο εκτέλεσης των σεναρίων. Την λύση στο παραπάνω πρόβλημα έρχεται να δώσει το εργαλείο “Tasks” της InfluxDB. Τα “tasks” είναι εργασίες που εκτελούνται περιοδικά ανά προκαθορισμένο χρόνο, παρόμοια με τα cronjobs που αναφέρθηκαν σε προηγούμενη ενότητα. Αυτή την φορά όμως δεν υπάρχει ανάγκη για ανάπτυξη διεπαφής διαχείρισης, αφού προσφέρεται απλόχερα από την InfluxDB. Η εισαγωγή, επεξεργασία και διαγραφή (CRUD) εργασιών μπορεί να επιτευχτεί πολύ εύκολα από υπηρεσίες ιστού (web services). Παράλληλα το εργαλείο προσφέρει δικλείδες ασφαλείας, ελέγχοντας το περιεχόμενο της εισερχόμενης εργασίας για λάθη και αποτρέποντας την υπερφόρτωση του συστήματος. Επίσης με την χρήση της InfluxDB, αυτόματα το σύστημα μπορεί να λειτουργήσει κατανεμημένα, αφού η ίδια προσφέρει διαχείριση δεδομένων από πλήθος κόμβων. Τέλος, τα σενάρια, οι πληροφορίες λογαριασμών και συσκευών θα αποθηκευόταν στην MySQL σχεσιακή βάση δεδομένων, παρόμοια με την προηγούμενη λύση.

Με τον παραπάνω σχεδιασμό όμως, σύντομα εμφανίστηκαν ζητήματα περιορισμού έκφρασης των σεναρίων. Το πρόβλημα δεν απειλούσε αυτή την φορά την λειτουργία της εφαρμογής αλλά τις μη λειτουργικές απαιτήσεις. Ο χρήστης σύμφωνα με τις απαιτήσεις συστήματος που τέθηκαν, θα πρέπει να είναι σε θέση να δημιουργεί ευέλικτα σενάρια, να μην περιορίζεται δηλαδή σε μια προκαθορισμένη και περιοριστική δομή οδηγιών. Οι εργασίες που προσφέρει η InfluxDB μπορούν να εκτελέσουν αποκλειστικά κώδικα Flux, γλώσσα η οποία περιόριζε και περιέπλεκε κατά πολύ την εκφραστικότητα των σεναρίων. Αρχικά η συγγραφή και σύνταξη του κώδικα είναι τρομερά δύσκολη για τον μέσο χρήστη, αφού δεν ακολουθεί μια λογική δομή του τύπου if-then-else και δεν προσφέρει επεξηγηματικά μηνύματα σφάλματος. Αντίθετα χρησιμοποιεί την δομή αλυσίδας, κατά την οποία τα δεδομένα μεταφέρονται από συνάρτηση σε συνάρτηση, καταλήγοντας σε δράσεις. Η Flux, υποστηρίζει την εκτέλεση αιτημάτων http μόνο για την αποστολή δεδομένων και όχι για την ανάγνωση και επεξεργασία των δεδομένων που αυτά επιστρέφουν, επομένως δεν θα μπορούσε να καλυφθεί η λειτουργική απαίτηση για χρήση δεδομένων από εξωτερικές πηγές. Τέλος, η εκφραστικότητα των σεναρίων θα περιοριζόταν αποκλειστικά στις δυνατότητες που παρέχει η InfluxDB, δημιουργώντας έτσι μια εξάρτηση που δεν θα επέτρεπε την περαιτέρω ανάπτυξη και εύρεση νέων καινοτομιών της εφαρμογής.

3.4.4 MYSQL / PHP / InfluxDB & εκτέλεση σεναρίων με tasks v2

Αφού είχαν λυθεί τα ζητήματα απόδοσης και ασφάλειας, δεν έμενε πάρα μόνο να λυθεί και το πρόβλημα χρηστικότητας που αφορούσε την προηγούμενη λύση. Εστιάζοντας στο καινούριο ζήτημα αλλά παράλληλα αποφεύγοντας να αναιρεθεί η κατάλληλη αντιμετώπιση προηγούμενων προβλημάτων, αναπτύχθηκε μια εναλλακτική διατηρώντας όλα τα συστατικά μέρη της προηγούμενης λύσης. Η ιδέα ήταν να χρησιμοποιηθεί κώδικας Flux που θα έκανε αίτηση σε μια προσαρμοσμένη υπηρεσία ιστού γραμμένη σε php, μέρος της προτεινόμενης εφαρμογής, με σκοπό την ανάκτηση και εκτέλεση του σεναρίου εφόσον ικανοποιούνται οι σχετικές συνθήκες, μέρος των οποίων αποτελεί η αποστελλόμενη αίτηση. Η παραπάνω λύση, υλοποιεί το μοντέλο της επανάκλησης (callback), δηλαδή γίνεται κλήση ρουτίνας όταν λαμβάνει χώρα ένα συμβάν. Καθήκον λοιπόν της προσαρμοσμένης υπηρεσίας ιστού είναι να ανακτήσει όλα τα δεδομένα που αφορούν το συγκεκριμένο σενάριο και μόνο αν όλα είναι διαθέσιμα και ικανοποιούνται οι σχετικές συνθήκες, να τρέξει τον κώδικα δράσης του χρήστη. Με άλλα λόγια, κομμάτια / υπό-συνθήκες που αφορούν το κεφάλι του κανόνα του κάθε σεναρίου αντιμετωπίζονται μέσω της χρήσης της InfluxDB και όταν τα κομμάτια αυτά ικανοποιούνται, ελέγχεται όλη η συνθήκη (από άλλο μέρος της προτεινόμενης εφαρμογής) και εφόσον ικανοποιείται, εκτελείται το σώμα του κανόνα του σεναρίου που αφορά τις αντίστοιχες δράσεις. Αυτή η λύση αντιμετωπίζει αποτελεσματικά τις απαιτήσεις που τέθηκαν σε προηγούμενη ενότητα χωρίς καμία αμφιβολία, ειδικότερα διότι προσφέρει ευελιξία ως προς την μοντελοποίηση των κανόνων και αίρει τους περιορισμούς χρηστικότητας.

3.4.5 Σύγκριση λύσεων

Σε αυτή την ενότητα δίνεται ένας τελικός πίνακας σύγκρισης των παραπάνω πιθανών λύσεων. Τα κριτήρια σύγκρισης που αναλύθηκαν στις προηγούμενες ενότητες είναι η χρηστικότητα, απόδοση / επεξεργασία μεγάλων όγκων δεδομένων, ανεξαρτησία λειτουργικού συστήματος, επεκτασιμότητα και ασφάλεια. Όπως φαίνεται από τον πίνακα αυτό, η τελευταία εναλλακτική, που αποτελεί και την προτεινόμενη λύση, υπερτερεί των άλλων εναλλακτικών κατά μήκος όλων των κριτηρίων.

	Λύση 3.4.1	**Λύση 3.4.2**	**Λύση 3.4.3**	**Λύση 3.4.4**
Χρηστικότητα	✓ (πλούσια)	✓ (πλούσια)	~ (περιορισμός σε ότι προσφέρεται από την InfluxDB)	✓ (πλούσια)
Απόδοση / επεξεργασία μεγάλων όγκων δεδομένων	✘ (MySQL χαμηλή)	✓ (Prometheus υψηλή)	✓ (InfluxDB υψηλή)	✓ (InfluxDB υψηλή
Ανεξαρτησία λειτουργικού	✘ (Τα Cronjobs,	✘ (Τα Cronjobs, ορίζονται	✓ (Χρήση Tasks	✓ (Χρήση Tasks

συστήματος	ορίζονται διαφορετικά σε κάθε ΛΣ)	διαφορετικά σε κάθε ΛΣ)	InfluxDB – Λύση Cross Platform)	InfluxDB – Λύση Cross Platform)
Επεκτασιμότητα	✓ (νέες δράσεις είναι δυνατό να γραφτούν μέσω της php)	✓ (νέες δράσεις είναι δυνατό να γραφτούν μέσω της php)	✘ (δεν μπορούν να αναπτυχθούν άλλες δράσεις, εξάρτηση από InfluxDB)	✓ (νέες δράσεις είναι δυνατό να γραφτούν μέσω της php)
Ασφάλεια	✘ (πρόβλημα διαχείρισης εργασιών – κίνδυνος υπερφόρτωσης)	✘ (πρόβλημα διαχείρισης εργασιών – κίνδυνος υπερφόρτωσης)	✓ (ασφαλής – δοκιμασμένη λύση από την InfluxDB)	✓ (ασφαλής – δοκιμασμένη λύση από την InfluxDB)

3.5 Αρχιτεκτονική εφαρμογής

Σε αυτή την ενότητα δίνεται μια αναλυτική περιγραφή της αρχιτεκτονικής που ακολουθεί η εφαρμογή, σύμφωνα με την επιλεγμένη λύση. Στις παρακάτω υπό-ενότητες αναφέρονται τα κύρια συστατικά της αρχιτεκτονικής, οι αλληλεπιδράσεις των μερών της, τα πλεονεκτήματα της και ο βαθμός ικανοποίησης της ως προς τις απαιτήσεις συστήματος, όπως αυτές τέθηκαν στο Ενότητα 3.2.

3.5.1 Κύρια συστατικά μέρη

Τα σύνθετα έργα πληροφορικής απαιτούν την διάσπαση τους σε μικρότερα, απλούστερα υπό-έργα, προκειμένου να μπορούν να επιλυθούν αποτελεσματικά, δίνοντας την καλύτερη δυνατή λύση. Σε αυτή την ενότητα παρουσιάζονται τα συστατικά μέρη της εφαρμογής που υλοποιήθηκε στα πλαίσια της διπλωματικής αυτής εργασίας.

Αρχικά, για την αλληλεπίδραση του συστήματος με τον χρήστη παρέχεται εφαρμογή ιστού (web app), η οποία με την σειρά της διακλαδώνεται σε μια ιστοσελίδα πληροφόρησης και την περιοχή χρήστη. Το παραπάνω μέρος χαρακτηρίζεται ως το frontend της εφαρμογής, το μέσο δηλαδή με το οποίο ο χρήστης υποβάλει ερωτήματα στο σύστημα και λαμβάνει απαντήσεις. Η σελίδα πληροφόρησης είναι προσβάσιμη από όλους τους επισκέπτες. Σκοπός της τελευταίας είναι να γνωστοποιήσει το αντικείμενο της εφαρμογής καθώς και την διάθεση της λειτουργίας

δημιουργίας λογαριασμού. Στην περιοχή χρήστη, μετά την απαραίτητη αυθεντικοποίηση, δίνεται πρόσβαση σε υπηρεσίες και αγαθά για την διαχείριση συσκευών IoT. Μέσω της τελευταίας ο χρήστης μπορεί να επεξεργαστεί τους έξυπνους τόπους, τις συσκευές, τα σενάρια, τις εγγραφές ιχνηλασιμότητας, τις διεπαφές ιστού και τα μέλη / χρήστες του έξυπνου τόπου, ενώ παράλληλα έχει πρόσβαση σε γραφικές απεικονίσεις των μετρήσεων. Η εφαρμογή ιστού είναι γραμμένη σε HTML5, PHP, Javascript και JQuery. Η πρώτες δυο χρησιμοποιούνται για τον σχηματισμό του ανάλογου εγγράφου από τον διακομιστή κατόπιν αίτησης από τον χρήστη. Οι άλλες δυο, για άμεση επεξεργασία των στοιχείων DOM κατά την πραγματοποίηση συμβάντων καθώς και την ενημέρωση της σελίδας σε ασύγχρονους χρόνους με καινούρια δεδομένα. Επίσης χρησιμοποιείται το δημοφιλές πακέτο Bootstrap επιτρέποντας προσαρμοσμένες εμφανίσεις ανάλογα με την συσκευή του χρήστη.

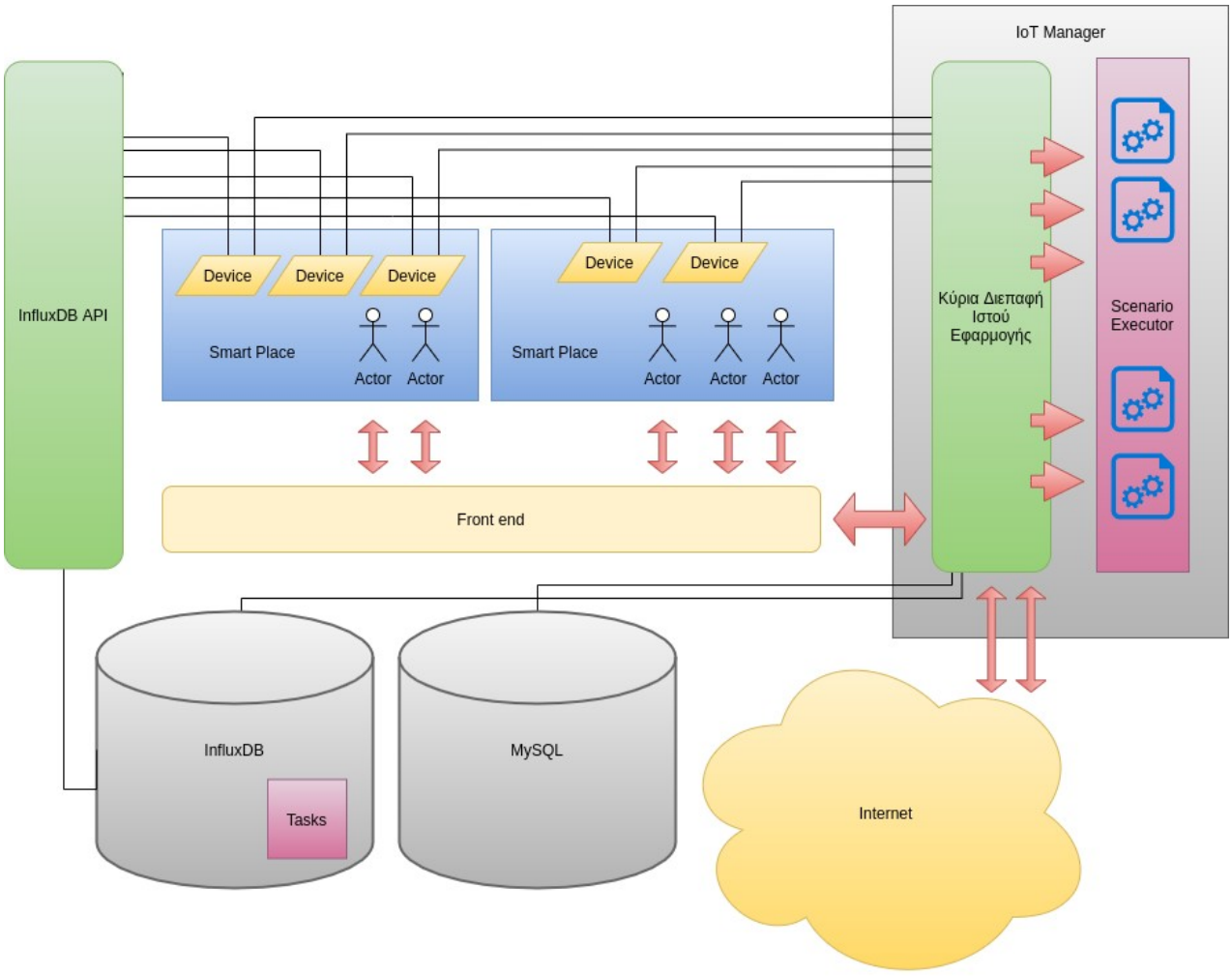
Για την ανταλλαγή δεδομένων μεταξύ των συστατικών μερών της εφαρμογής (ειδικότερα μεταξύ του frontend & του backend καθώς και μεταξύ των συσκευών και του backend) και την εκτέλεση των σεναρίων αναπτύχθηκε ένα σύνθετο συστατικό, ο IoT Manager. Το τελευταίο αποτελείται από την κύρια διεπαφή ιστού της εφαρμογής, γραμμένη σε php, που λειτουργεί ως συνδετικός κρίκος και τον εκτελεστή των σεναρίων, με την ονομασία “ScenarioExecutor”. Η διεπαφή ιστού περιλαμβάνει υπηρεσίες για την εισαγωγή, επεξεργασία και διαγραφή όλων των οντοτήτων, όπως τους έξυπνους τόπους, τις συσκευές, τα σενάρια, τις μετρήσεις και τις εγγραφές ιχνηλασιμότητας. Οι παραπάνω ρουτίνες μπορούν να κληθούν αποκλειστικά από αυθεντικοποιημένους χρήστες που έχουν την κατάλληλη εξουσιοδότηση. Από εκεί επίσης, προσφέρονται υπηρεσίες εγγραφής, αυθεντικοποίησης και αποσύνδεσης του χρήστη. Επίσης, η κύρια διεπαφή είναι υπεύθυνη για την εκτέλεση των σεναρίων κατά την λήψη κατάλληλου ερεθίσματος από τον διακομιστή που φιλοξενεί την InfluxDB. Το ερέθισμα αυτό θα πρέπει αποκλειστικά να προέρχεται από τον κόμβο ή τους κόμβους που φιλοξενούν την InfluxDB. Αυτό επιτυγχάνεται επιτρέποντας την πρόσβαση μόνο σε διακομιστές που η διεύθυνση τους περιλαμβάνεται σε σχετική λίστα (whitelist). Ο ορισμός αυτών των διευθύνσεων πραγματοποιείται από τον διαχειριστή του συστήματος τροποποιώντας το ανάλογο αρχείο παραμετροποίησης.

Όπως αναφέρθηκε στην παραπάνω παράγραφο η κύρια διεπαφή είναι υπεύθυνη για την πυροδότηση της εκτέλεσης των σεναρίων. Όταν όλες οι τιμές που απαιτούνται στο σενάριο είναι έτοιμες, έχουν δηλαδή υπολογιστεί, η διεπαφή δημιουργεί ένα νήμα εκτέλεσης “Scenario Executor”, όπως αυτό αποκαλείται και το εκτελεί στο παρασκήνιο. Το νήμα παράγει τον τελικό κώδικα και τον εκτελεί χρησιμοποιώντας την ενσωματωμένη συνάρτηση της php, την eval(). Η τελευταία δέχεται ως όρισμα ένα αλφαριθμητικό που αναπαριστά τον κώδικα php, τον οποίο και εκτελεί.

Βασικό συστατικό μέρος της εφαρμογής, αποτελεί ο τρόπος διαχείρισης των δεδομένων, αφού η απόδοση αποτελεί χαρακτηριστικό υψίστης σημασίας για έργα IoT. Για την ανάπτυξη του πληροφοριακού συστήματος επιλέχθηκαν δυο βάσεις δεδομένων, οι οποίες και συγκροτούν μαζί με τον IoT Manager το backend. Με τον τελευταίο όρο εννοούνται οι τεχνολογίες που δρουν στο παρασκήνιο, προκειμένου να παρέχουν πληροφορίες κατ’ απαίτηση και να υλοποιούν τις κύριες λειτουργίες μιας εφαρμογής. Για την αποθήκευση δεδομένων του χρήστη, όπως τα σενάρια, ρυθμίσεις λογαριασμού και συσκευών, επιλέχθηκε η σχεσιακή βάση MySQL ενώ για τις μετρήσεις, την επεξεργασία των δεδομένων, την παραγωγή και εξαγωγή των γεγονότων που

αφορούν τα σενάρια η InfluxDB. Η τελευταία, όπως έχει αναφερθεί και σε προηγούμενη ενότητα, δεν αποτελεί μόνο μια βάση δεδομένων, αφού προσφέρει μια σειρά από εργαλεία. Η εφαρμογή υιοθετεί από την InfluxDB το εργαλείο “Tasks” και το InfluxDB API. Το πρώτο για την υποβολή εργασιών, προκειμένου να ανακτούνται μετρήσεις, να εφαρμόζονται στατιστικές διαδικασίες και να στέλνονται πίσω τα σχετικά συμβάντα στο κύριο API της εφαρμογής, προκειμένου να πυροδοτηθεί η εκτέλεση των σεναρίων. Το δεύτερο για την επικοινωνία των αντικειμένων και της κύριας διεπαφής με την βάση χρονοσειρών InfluxDB. Οι συσκευές IoT μέσω της διεπαφής InfluxDB API μπορούν να στέλνουν μετρήσεις ενώ η κύρια διεπαφή μπορεί να λαμβάνει μετρήσεις και να διαχειρίζεται τις εργασίες.

Οι συσκευές και τα εξωτερικά APIs με την σειρά τους, αποτελούν και αυτές βασικό μέρος της εφαρμογής, αφού λειτουργούν ως είσοδοι δεδομένων. Οι πρώτες μπορούν να λειτουργήσουν και ως έξοδοι, αφού είναι επιτρεπτή η αλλαγή της κατάστασής τους, για παράδειγμα η ενεργοποίηση ή απενεργοποίηση μια πρίζας. Οι αισθητήρες λαμβάνουν μετρήσεις από το εξωτερικό περιβάλλον και τις αποστέλλουν στην εφαρμογή, ενώ οι πρίζες χρησιμοποιούνται για την διαχείριση οικιακών συσκευών. Τα εξωτερικά APIs, προσπελάζονται κατά την εκτέλεση των σεναρίων, για την ανάκτηση και διάθεση δεδομένων διαδικτύου στα σενάρια του χρήστη.



Σχήμα 3 - Συστατικά μέρη της εφαρμογής

3.5.2 Αλληλεπιδράσεις μερών

Τα παραπάνω συστατικά μέρη έχουν την ανάγκη για αλληλεπίδραση προκειμένου να συγκροτήσουν ένα ολοκληρωμένο, λειτουργικό πληροφοριακό σύστημα. Για την ανταλλαγή πληροφοριών μεταξύ όλων των παραπάνω μερών, όπως αναφέρθηκε στην προηγούμενη ενότητα, χρησιμοποιούνται δυο διεπαφές ιστού. Η κύρια διεπαφή της εφαρμογής και η InfluxDB API, οι οποίες λειτουργούν με βάση το πρωτόκολλο http.

Η γραφική διεπαφή χρήστη δημιουργεί ασύγχρονα ερωτήματα στην κύρια διεπαφή της εφαρμογής για την ανάκτηση, επεξεργασία και αποθήκευση δεδομένων λογαριασμού, έξυπνων τόπων, συσκευών, σεναρίων, μελών / χρηστών του έξυπνου τόπου και εξωτερικών διεπαφών ιστού. Η υποβολή ερωτημάτων σε ασύγχρονους χρόνους, πραγματοποιείται μέσω της βιβλιοθήκης JQuery, πετυχαίνοντας ελαχιστοποίηση του χρόνου φόρτωσης των πληροφοριών και των γραφικών τμημάτων. Η παραπάνω τεχνική επιτρέπει την συγγραφή ιστοτόπων δυναμικού περιεχομένου, που εξασφαλίζει από την μεριά των προγραμματιστών καλύτερη οργάνωση και συντήρηση του κώδικα, ενώ από την μεριά του χρήστη καλύτερη εμπειρία χρήσης.

Όπως αναφέρθηκε και στην ενότητα 3.4.4, η εφαρμογή θα χρησιμοποιεί δυο βάσεις δεδομένων, την σχεσιακή βάση MySQL και την βάση χρονοσειρών InfluxDB. Για την προσπέλαση δεδομένων στην πρώτη, χρησιμοποιείτε η κύρια διεπαφή της εφαρμογής καλύπτοντας όλες τις λειτουργίες CRUD. Στην δεύτερη βάση, την InfluxDB, η προσπέλαση των δεδομένων αλλά και η διαχείριση των εργασιών πραγματοποιείται μέσω της διεπαφής ιστού InfluxDB API. Η τελευταία διεπαφή επιτρέπει την πρόσβαση στις υπηρεσίες της, μόνο αν παρέχεται το σωστό κλειδί (token).

Η κύρια διεπαφή, όπως ήδη αναφέρθηκε πυροδοτεί επίσης, την εκτέλεση των σεναρίων, έπειτα από ερεθίσματα που λαμβάνει από την InfluxDB. Όταν διατίθενται όλες οι τιμές που χρησιμοποιεί το σενάριο, η κύρια διεπαφή δημιουργεί ένα νήμα “Scenario Executor” και το εκτελεί στο παρασκήνιο.

Οι συσκευές ανά περιοδικούς χρόνους, που καθορίζονται από τον χρήστη, επικοινωνούν με την InfluxDB μέσω της διεπαφής InfluxDB API για την αποστολή μετρήσεων που παράγονται από το εξωτερικό περιβάλλον. Επίσης, οι συσκευές επικοινωνούν και με την κύρια διεπαφή της εφαρμογής που αναφέρθηκε πιο πάνω με σκοπό την ενημέρωση των ιδιοτήτων τους. Για παράδειγμα, οι έξυπνες πρίζες αιτούνται ανά τακτά χρονικά διαστήματα εγγραφές από την σχεσιακή βάση που τις αντιπροσωπεύουν, με σκοπό τον συγχρονισμό τους με τις ανάλογες τιμές. Αυτές οι ιδιότητες μπορεί να είναι είτε η κατάσταση είτε η συχνότητα ενημέρωσης της συσκευής, οι οποίες μπορούν να τροποποιηθούν από κάποιο άλλο συστατικό μέρος της εφαρμογής, όπως την γραφική διεπαφή (front end) ή από την εκτέλεση σεναρίων.

3.5.3 Πλεονεκτήματα αρχιτεκτονικής

Σε αυτό το σημείο είναι απαραίτητο να αποσαφηνιστούν τα πλεονεκτήματα της επιλεγμένης αρχιτεκτονικής, απομακρύνοντας τυχόν αμφιβολίες. Λίγο πολύ τα πλεονεκτήματα των επιλογών έχουν ήδη αναφερθεί παραπάνω, επομένως σε αυτή την ενότητα γίνεται μια σύνοψη αυτών.

Ο διαχωρισμός των πληροφοριών σε δυο βάσεις δεδομένων ανάλογα με την φύση τους,

μεγιστοποιεί την απόδοση και ευελιξία του συστήματος. Η απόδοση μπορεί να μετρηθεί αξιολογώντας την ταχύτητα απόκρισης, το μέγεθος των δεδομένων καθώς και το πλήθος των ταυτόχρονων συνδέσεων που μπορούν να πραγματοποιηθούν. Η ευελιξία χαρακτηρίζει το κατά πόσο το σύστημα είναι ανεκτικό σε αναβαθμίσεις και επεκτάσεις λειτουργικότητας. Καθώς τα έργα IoT έχουν κεντρίσει το ενδιαφέρον των επιστημόνων τα τελευταία χρόνια, η ανακάλυψη νέων τεχνικών και τεχνολογιών είναι αναμενόμενη. Λαμβάνοντας υπόψη τα παραπάνω, είναι αναγκαία προϋπόθεση οι εφαρμογές IoT, να αναπτύσσονται με τέτοιο τρόπο ώστε να επιτρέπουν εύκολα την συντήρηση και προσθήκη νέων λειτουργιών. Η σχεσιακή βάση επιτρέπει την εύκολη προσθήκη νέων χαρακτηριστικών και συσχετίσεων δημιουργώντας καινούριους πίνακες αλλά και την τροποποίηση των υπαρχόντων. Από την άλλη η βάση χρονοσειρών μπορεί να διαχειριστεί αυτόματα νέου είδους μετρήσεων, χωρίς να απαιτείται καμία απολύτως ενέργεια.

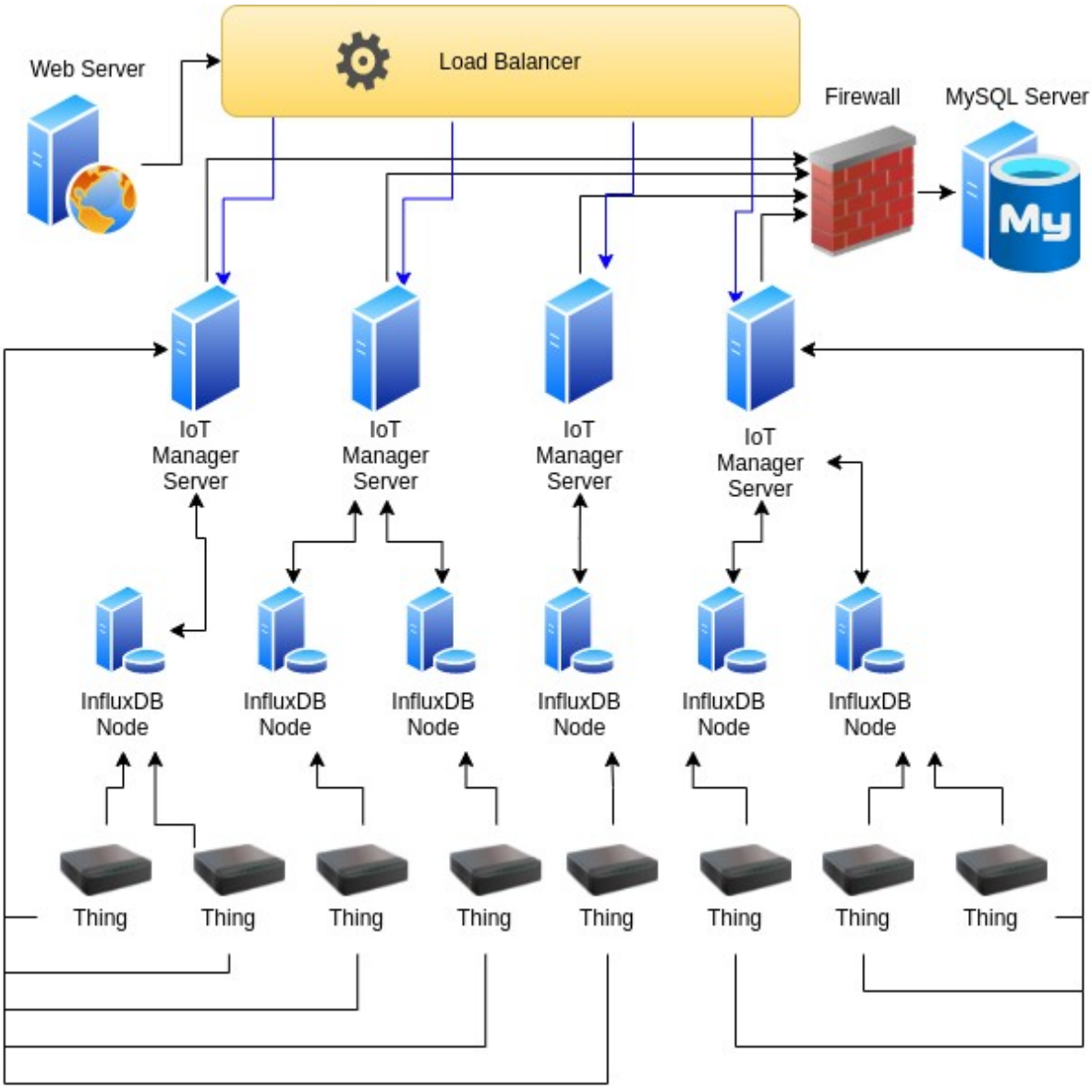
Πλεονέκτημα της αρχιτεκτονικής αποτελεί επίσης η σωστή επιλογή των βάσεων δεδομένων με βάση τις απαιτούμενες λειτουργίες συστήματος και τις αντίστοιχες μη λειτουργικές που τέθηκαν. Όπως αναφέρθηκε και πιο πάνω, τα επιλεγμένα εργαλεία δεν είναι τα μοναδικά του είδους. Έπειτα από σύγκριση χαρακτηριστικών, η MySQL και InfluxDB είναι οι καταλληλότερες για το έργο που τέθηκε. Αρχικά, η παραπάνω σχεσιακή βάση δεδομένων είναι αρκετά δημοφιλής στον χώρο, είναι αρκετά δοκιμασμένη και κατέχει την υψηλότερη βαθμολογία στον οργανισμό DB-Engines. Η επιλογή της InfluxDB, οφείλεται κατά κύριο λόγο στην παροχή του εργαλείου “Tasks”, ενώ παράλληλα υποστηρίζει άψογα δεδομένα χρονοσειρών. Σημαντικός παράγοντας επίσης, αποτέλεσε η ενεργή κοινότητα και τα πλούσια εγχειρίδια που έχουν αναπτυχθεί τα τελευταία χρόνια. Ως επιβεβαίωση των παραπάνω, η DB-Engines την κατατάσσει στην κορυφή της λίστας βάσεων δεδομένων χρονοσειρών.

Ο εμπλουτισμός κανόνων με δεδομένα διαδικτύου από υπηρεσίες ιστού (web services) αποτελεί μια πρωτοπορία στις εφαρμογές IoT, σημειώνοντας έτσι ένα από τα βασικότερα πλεονεκτήματα της αρχιτεκτονικής που επιλέχθηκε. Έχοντας πρόσβαση σε δεδομένα του διαδικτύου για την διαχείριση έξυπνων συσκευών, διευρύνεται σε τεράστιο βαθμό η εκφραστικότητα των σεναρίων. Ταυτόχρονα, επιτρέπεται η σχεδίαση έργων IoT χαμηλού κόστους, αφού ο χρήστης δεν είναι πλέον αναγκασμένος να πληρώσει για την παραγωγή δεδομένων μέσω της αγοράς σχετικών συσκευών IoT.

Η έκφραση των σεναρίων με ψευδοκώδικα βασισμένο στην διαδικτυακή γλώσσα προγραμματισμού php, επιτρέπει την πλούσια περιγραφή δράσεων για την διαχείριση των συσκευών IoT. Συγκεκριμένα, οι μεταβλητές, οι ροές ελέγχου, οι στατιστικές συναρτήσεις και οι τελεστές του ψευδοκώδικα επιτρέπουν την δημιουργία περίπλοκων σεναρίων, καλύπτοντας έτσι ένα ακόμη μεγαλύτερο σύνολο περιπτώσεων χρήσης IoT.

Τα σενάρια του χρήστη εκτελούνται στον κεντρικό κόμβο έπειτα από κατάλληλο ερέθισμα που δημιουργείτε από τα “Tasks”. Με την παραπάνω αρχιτεκτονική είναι εφικτή η διαμέριση του συστήματος σε πολλαπλούς κόμβους ακόμη και διαφορετικών δικτύων / υποδομών. Ουσιαστικά, είναι εφικτό να έχουμε έναν διακομιστή ιστού αποκλειστικά για την αλληλεπίδραση του χρήστη με το σύστημα. Έναν διακομιστή που θα φιλοξενεί την σχεσιακή βάση δεδομένων MySQL, απομονώνοντας τα ευαίσθητα δεδομένα σε ασφαλές μέρος, χρησιμοποιώντας κατάλληλες πολιτικές ασφαλείας. Πολλαπλούς διακομιστές που θα φιλοξενούν την κύρια διεπαφή της εφαρμογής για την εκτέλεση σεναρίων αλλά και την ανταλλαγή πληροφοριών με τις συσκευές

IoT. Τέλος, πολλαπλούς κόμβους που θα φιλοξενούν το λογισμικό της InfluxDB για την αποθήκευση και επεξεργασία των μετρήσεων. Έτσι, το πληροφοριακό σύστημα μπορεί εύκολα να λειτουργήσει κατανεμημένα, εξυπηρετώντας εκατομμύρια χρήστες. Στο παρακάτω σχήμα φαίνεται το προαναφερόμενο παράδειγμα σχεδίασης.



Σχήμα 4 - Κατανεμημένη σχεδίαση εφαρμογής

3.5.4 Ικανοποίηση απαιτήσεων

Σε αυτή την ενότητα αναφέρονται οι τρόποι με τους οποίους ικανοποιούνται οι απαιτήσεις που τέθηκαν, προκειμένου να λυθούν αποτελεσματικά τα προβλήματα των εφαρμογών IoT. Όπως παρουσιάστηκε σε προηγούμενες ενότητες, οι απαιτήσεις διαχωρίζονται σε λειτουργικές και μη λειτουργικές. Οι πρώτες αφορούν το τι πρέπει να κάνει το σύστημα, ενώ οι δεύτερες εκφράζουν το κατά πόσο καλά θα πρέπει να υλοποιηθούν οι λειτουργικές απαιτήσεις.

Η διεπαφή χρήστη αποτελεί το μέσο με το οποίο ο χρήστης μπορεί να απαιτήσει λειτουργίες από το σύστημα. Ο τελευταίος μπορεί να δημιουργήσει λογαριασμό μέσω κατάλληλης φόρμας εγγραφής. Μετά την εγγραφή και σύνδεση στο σύστημα μπορεί να δη να παρακολουθεί μετρήσεις μέσω γραφημάτων. Το σύστημα λειτουργεί με βάση το μοντέλο δικαιωμάτων (privileges model), σύμφωνα με το οποίο οι χρήστες έχουν πρόσβαση σε αγαθά, κατόπιν εξουσιοδότησης. Οι χρήστες μπορούν να παραχωρήσουν πρόσβαση σε άλλα μέλη μέσω κατάλληλης φόρμας, για την από κοινού επεξεργασία αγαθών έξυπνων τόπων. Επίσης, μπορούν να ενημερώνονται για αλλαγές ιδιοτήτων των συσκευών IoT τους μέσω κατάλληλων εγγραφών ιχνηλασιμότητας που καταχωρούνται αυτόματα από το σύστημα. Χρησιμοποιώντας τον κειμενογράφο, οι χρήστες μπορούν να υποβάλουν με ευκολία σενάρια που λαμβάνουν είσοδο από πηγές δεδομένων και παράγουν έξοδο με την μορφή δράσεων. Τέλος, για την γενίκευση των πηγών δεδομένων, το σύστημα μπορεί να δεχτεί οποιαδήποτε υπηρεσία ιστού, μέσω κατάλληλης φόρμας. Τα δεδομένα αυτά μπορούν να χρησιμοποιηθούν για την λήψη αποφάσεων εντός των σεναρίων. Οι υπηρεσίες αυτές συγκροτούν ένα ευρετήριο, που είναι προσβάσιμο από όλους τους χρήστες της εφαρμογής.

Όπως έχει αναφερθεί σε προηγούμενες ενότητες, το σύστημα αποθηκεύει τα δεδομένα σε δυο βάσεις δεδομένων. Ο τεράστιος όγκος μετρήσεων που φτάνει στην εφαρμογή διαχειρίζεται αποδοτικά από την βάση χρονοσειρών InfluxDB. Επίσης, με την τελευταία βάση ικανοποιείται η απαίτηση για αποδοτική και πλούσια επεξεργασία των δεδομένων με στατιστικές συναρτήσεις.

Η εκτέλεση των σεναρίων πυροδοτείται από την κύρια διεπαφή της εφαρμογής μετά από κατάλληλα ερεθίσματα που λαμβάνονται από εργασίες της InfluxDB. Όταν όλες οι τιμές του σεναρίου είναι έτοιμες, τότε το σενάριο εκτελείται χωρίς να χάνεται πολύτιμος χρόνος αλλά και ούτε να πραγματοποιείται υπερφόρτωση των πόρων του συστήματος.

Τα σενάρια μπορούν να αντλήσουν πληροφορίες από διεπαφές του διαδικτύου, με την κλήση της συνάρτησης fetch_endpoint(). Η συνάρτηση επιστρέφει τα δεδομένα σε μορφή json, επιτρέποντας στον χρήστη να εξάγει την πληροφορία που χρειάζεται κάθε φορά. Η νέα αυτή πηγή δεδομένων επιτρέπει την σχεδίαση χαμηλού κόστους έργων IoT, αφού ο χρήστης δεν είναι πλέον αναγκασμένος να πληρώσει για αισθητήρες, χωρίς όμως να δημιουργείται ιδιαίτερος περιορισμός στις περιπτώσεις χρήσης.

Η έκφραση των σεναρίων σε ψευδοκώδικα, που βασίζεται στην διαδικτυακή γλώσσα προγραμματισμού php, κλιμακώνει αυξητικά τις περιπτώσεις χρήσης του IoT επίσης, ταυτόχρονα προσφέρει συντακτικό έλεγχο του κώδικα του χρήστη, για την ορθή εισαγωγή των σεναρίων. Στην περίπτωση λανθασμένου κώδικα, χρησιμοποιούνται απευθείας τα σχόλια της php για τον εντοπισμό και την αποσφαλμάτωση του.

Οι συσκευές IoT μπορούν εγκατασταθούν στην εφαρμογή μέσω λογισμικού. Ο χρήστης δεν έχει πάρα μόνο να εισάγει τα διαπιστευτήρια του τοπικού του δικτύου στην κατάλληλη φόρμα εγκατάστασης. Επίσης, θα πρέπει να σημειωθεί ότι δεν απαιτείται η ύπαρξη κάποιας επιπλέον συσκευής, όπως σε άλλες λύσεις της αγοράς (π.χ IKEA gateway). Για την διαγραφή τους, ο χρήστης μπορεί να χρησιμοποιήσει την διεπαφή χρήστη ή να πατήσει το κουμπί επαναφοράς που βρίσκεται στο κάτω μέρος των αντικειμένων.

Η διαχείριση των συσκευών IoT του χρήστη μπορεί να πραγματοποιηθεί από οπουδήποτε στον κόσμο αρκεί να υπάρχει πρόσβαση στο διαδίκτυο. Επίσης, οι τεχνολογίες ανάπτυξης της εφαρμογής που επιλέχτηκαν επιτρέπουν την προσαρμογή της γραφικής διεπαφής στην ανάλογη οθόνη που διαθέτει η συσκευή του χρήστη.

Τέλος, το σύστημα αποτρέπει κάθε είδους μη εξουσιοδοτημένης πρόσβασης χρησιμοποιώντας κατάλληλα κλειδιά (hashes). Έτσι μόνο τα συστατικά μέρη της εφαρμογής μπορούν να προσπελάσουν τους αντίστοιχους πόρους του συστήματος. Στο επίπεδο της γραφικής διεπαφής ακολουθούνται βασικές τεχνικές ασφάλειας [28] για την αποτροπή επιθέσεων τύπου SQL Injection, Cross Site Scripting (XSS) και Cross site request forgery (CSRF).

Στον παρακάτω πίνακα εμφανίζεται ο βαθμός ικανοποίησης των απαιτήσεων του συστήματος που είχαν τεθεί στην Ενότητα 3.2.

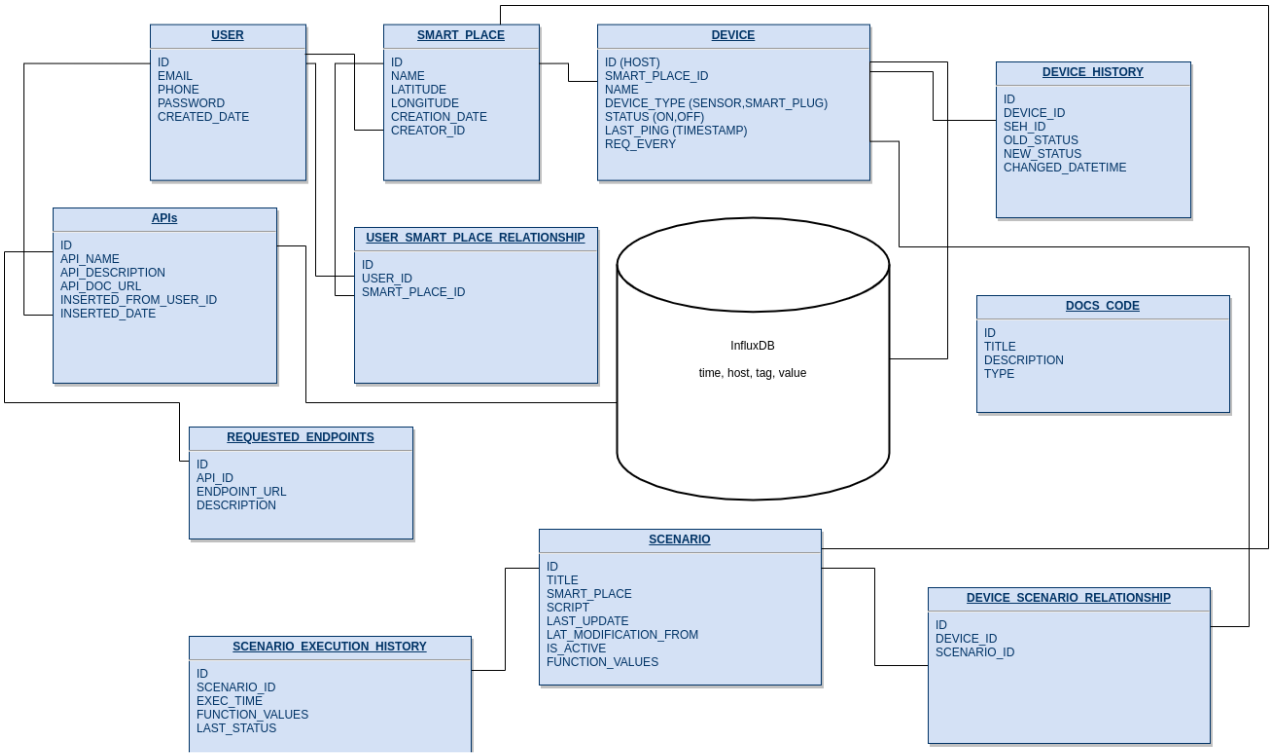
	Βαθμός ικανοποίησης	**Σχόλιο**
Διαχείριση τεράστιων όγκων δεδομένων	***	Ικανοποίηση μέσω της βάσης χρονοσειρών InfluxDB
Πλήθος στατιστικών συναρτήσεων	***	Υποστηρίζονται σχεδόν όλες οι στατιστικές συναρτήσεις της InfluxDB
Εκτέλεση σεναρίων σε κατάλληλους χρόνους	***	Διαμοιράζοντας τις συναρτήσεις ανάκτησης σε εργασίες της InfluxDB
Ανάκτηση δεδομένων από εξωτερικές πηγές	***	Παρέχοντας την συνάρτηση fetch_endpoint()
Ιχνηλάτηση γεγονότων	**	Παρέχονται δεδομένα αλλαγής κατάστασης των συσκευών IoT μόνο από σενάρια. Θα μπορούσαμε να έχουμε ιχνηλάτηση ακόμα και σε αλλαγές που πραγματοποιεί ο χρήστης μέσω της γραφικής διεπαφής. Επίσης θα ήταν εφικτό να έχουμε τον τίτλο του σεναρίου, τις τιμές και τον κώδικα που προκάλεσαν τις αλλαγές.
Εκφραστικότητα σεναρίων	***	Πλούσια με την χρήση ψευδογλώσσας που βασίζεται στην php.
Ευκολία εγκατάστασης συσκευών IoT	***	Μέσω λογισμικού, δεν απαιτείται καμία τεχνική γνώση ούτε επιπλέον συσκευές και περίπλοκες συνδεσιμότητες.
Προσβασιμότητα	***	Από οπουδήποτε στον κόσμο
Συμβατότητα εφαρμογής με την συσκευή του πελάτη	**	Από οποιαδήποτε συσκευή που παρέχει φυλλομετρητή (browser). Για την μέγιστη ικανοποίηση της απαίτησης θα έπρεπε να αναπτυχθεί εφαρμογή κινητού (native app)
Ασφάλεια της εφαρμογής	**	Ικανοποιούνται οι βασικές πολιτικές ασφαλείας. Για την

		μέγιστη ικανοποίηση της απαίτησης θα μπορούσαμε να χρησιμοποιήσουμε κάποιο δοκιμασμένο πακέτο λογισμικού (π.χ Laravel, Symfony)
Ευκολία εισαγωγής σεναρίων	*	Μέσω ψευδοκώδικα. Για την μέγιστη ικανοποίηση της απαίτησης θα έπρεπε να χρησιμοποιηθεί γραφική διεπαφή τύπου Drag & Drop

Πίνακας 4: Βαθμός ικανοποίησης των απαιτήσεων

3.6 Σχήματα Δεδομένων

Το σύστημα εκτός από την ανάγκη για αποδοτική αποθήκευση μετρήσεων, θα πρέπει επίσης να διαχειρίζεται κατάλληλα και τα δεδομένα του χρήστη. Τα τελευταία αποθηκεύονται σύμφωνα με το σχεσιακό μοντέλο, που επιτρέπει την συσχέτιση εγγραφών μεταξύ διαφορετικών πινάκων. Στην παρακάτω εικόνα δίνεται το τελικό σχήμα των βάσεων δεδομένων.



Σχήμα 6 - Σχήμα δεδομένων

Αρχικά, στον πίνακα “user”, αποθηκεύονται οι πληροφορίες λογαριασμού του χρήστη. Κατά την εγγραφή ενός χρήστη στο σύστημα, δίνονται και αποθηκεύονται, η ηλεκτρονική διεύθυνση (email), το τηλέφωνο (phone) και το επιθυμητό συνθηματικό του έπειτα από κρυπτογράφηση (password). Ο πίνακας κατά την εισαγωγή νέων εγγραφών εκχωρεί αυτόματα ένα μοναδικό αναγνωριστικό (id) και την τρέχουσα ημερομηνία (created_date).

Στον πίνακα “smart_place” αποθηκεύονται πληροφορίες του έξυπνου τόπου. Τα χαρακτηριστικά του τελευταίου είναι το μοναδικό αναγνωριστικό (id), το όνομα (name), οι γεωγραφικές συντεταγμένες (latitude & longitude), η ημερομηνία δημιουργίας (created_date) και το μοναδικό αναγνωριστικό του ιδρυτή (creator_id), δηλαδή του χρήστη που δημιούργησε τον έξυπνο τόπο.

Ο πίνακας “device” συγκροτεί χαρακτηριστικά των συσκευών, όπως το όνομα (name), το είδος (device_type), την κατάσταση (status) και την συχνότητα ενημέρωσης της συσκευής (req_every). Επίσης, αυτόματα εισάγεται η χρονοσφραγίδα τελευταίας επικοινωνίας (last_ping) και το μοναδικό αναγνωριστικό της συσκευής (id).

Στον πίνακα “APIs” διατηρούνται δεδομένα διεπαφών που εισάγονται από τον χρήστη. Για κάθε διεπαφή αποθηκεύεται ο τίτλος (api_name), η περιγραφή (api_description), ο σύνδεσμος εγχειριδίου (api_doc), το μοναδικό αναγνωριστικό του χρήστη που την εισήγαγε (inserted_from_user_id) και τέλος η ημερομηνία εισαγωγής (inserted_date). Επίσης αποτελεί πίνακα γονέα για τον πίνακα “requested endpoints”. Ένα API αποτελείται από πολλά endpoints. Ο πίνακας “requested_endpoints” συγκρατεί τον σύνδεσμο (endpoint_url) και μια περιγραφή (description) των δεδομένων που προσφέρει η υπηρεσία ιστού.

Ο πίνακας “docs_code” διατηρεί βοηθητικό περιεχόμενο για την ενημέρωση του χρήστη σχετικά με την χρήση των διαθέσιμων συναρτήσεων. Αποτελείται από το μοναδικό αναγνωριστικό (id), την επικεφαλίδα (title), περιγραφή (description) και το τύπο της συνάρτησης (type).

Στον πίνακα “scenario” αποθηκεύονται οι πληροφορίες σεναρίου, τα βασικά χαρακτηριστικά του οποίου είναι ο τίτλος (title), ο κώδικας (script), πληροφορίες σχετικά με την τελευταία τροποποίηση (last_update & last_modification_from), η τρέχουσα κατάσταση του (is_active) και το μοναδικό αναγνωριστικό του έξυπνου τόπου όπου ανήκει (smart_place). Στο γνώρισμα “function values” διατηρούνται οι διαθέσιμες τιμές των πηγών δεδομένων σε μορφή json. Κατά την λήψη μια τιμής από την InfluxDB, ενημερώνεται το παραπάνω γνώρισμα, ενώ αν υπάρχουν τιμές για όλες τις συναρτήσεις, το σενάριο εκτελείται.

Στον πίνακα “device_history” αποθηκεύονται εγγραφές ιχνηλασιμότητας για τον εντοπισμό αλλαγών ιδιοτήτων των συσκευών που πραγματοποιούνται από τα σενάρια. Τα χαρακτηριστικά του πίνακα είναι το μοναδικό αναγνωριστικό (id), το αναγνωριστικό της συσκευής (device_id), πληροφορίες του χαρακτηριστικού και της τιμής που άλλαξε (old_status & new_status) καθώς και η ώρα που πραγματοποιήθηκε το συμβάν (changed_datetime).

Στον πίνακα “scenario_execution_history” αποθηκεύονται εγγραφές εκτέλεσης των σεναρίων. Τα χαρακτηριστικά του πίνακα είναι το μοναδικό αναγνωριστικό (id) της εκτέλεσης, το μοναδικό αναγνωριστικό του σεναρίου που εκτελέστηκε (scenario_id), η χρονοσφραγίδα

εκτέλεσης (exec_time), οι τότε διαθέσιμες τιμές των συναρτήσεων ανάκτησης (function_values) και τέλος ένας κωδικός που δηλώνει αν εκτελέστηκε σωστά το σενάριο (last_status).

Στην InfluxDB αποθηκεύονται οι μετρήσεις που παράγονται από τα αντικείμενα. Τα δεδομένα που συγκρατούνται για κάθε μέτρηση είναι ο χρόνος εισαγωγής (time), το μοναδικό αναγνωριστικό της συσκευής (host), το είδος της μέτρησης (tag) και η τιμή (value).

Οι υπόλοιποι πίνακες, εκφράζουν τις σχέσεις μεταξύ εγγραφών διαφορετικών πινάκων. Οι συσχετίσεις πραγματοποιούνται με κατάλληλες εγγραφές, χρησιμοποιώντας τα μοναδικά κλειδιά των πλειάδων.

3.7 Συμπεράσματα

Σε αυτό το κεφάλαιο, παρουσιάστηκε αναλυτικά η προτεινόμενη λύση που έρχεται να αντιμετωπίσει τα σύγχρονα ζητήματα εφαρμογών IoT από τα οποία προέκυψαν οι απαιτήσεις της λύσης αυτής, οι οποίες ικανοποιούνται όλες σε ικανοποιητικό βαθμό. Ο παρακάτω πίνακας είναι μια επέκταση του πίνακα που απεικονίζεται στην Ενότητα 2.3, συμπεριλαμβάνοντας αυτή την φορά την προτεινόμενη λύση ώστε να συγκριθεί με αυτές της αγοράς.

	IKEA Smart Home	**Kasa Smart**	**Προτεινόμενη λύση**
Ευκολία εγκατάστασης	✘ (δύσκολο για το μέσο χρήστη)	✓ (εγκατάσταση μέσω λογισμικού)	✓ (εγκατάσταση μέσω λογισμικού)
Απευθείας σύνδεση με router	✘ (απαιτείται η ύπαρξη επιπλέον υλικού - πύλης)	✓ (δεν απαιτείται η ύπαρξη επιπλέον υλικού)	✓ (δεν απαιτείται η ύπαρξη επιπλέον υλικού)
Ευκολία χρήσης	✓ (αρκετά εύκολη)	✓ (αρκετά εύκολη)	✓ (αρκετά εύκολη)
Άμεση διαχείριση συσκευών	✓ (μέσω εφαρμογής κινητού)	✓ (μέσω εφαρμογής κινητού)	✓ (μέσω εφαρμογής ιστού)
Περιγραφή σεναρίων	✘ (δεν υποστηρίζεται)	✓ (απλή if-then-else)	✓ (πλούσια και εκτεταμένη λίστα στατιστικών συναρτήσεων με χρήση ψευδοκώδικα)
Εξωτερικές πηγές δεδομένων	✘ (μόνο δεδομένα συσκευών που κατέχει ο χρήστης)	✘ (μόνο δεδομένα συσκευών που κατέχει ο χρήστης)	✓ (υποστήριξη δεδομένων από υπηρεσίες ιστού)
Επεξεργασία δεδομένων	✘ (χρήση μόνο της τελευταίας μέτρησης)	✘ (χρήση μόνο της τελευταίας μέτρησης)	✓ (υποστήριξη πλήθους στατιστικών συναρτήσεων)
Προσβασιμότητα από το διαδίκτυο	✘ (μόνο εντός οικιακού δικτύου)	✓ (από οπουδήποτε στον κόσμο)	✓ (από οπουδήποτε στον κόσμο)
Συμβατότητα με προϊόντα / υπηρεσίες τρίτων	✘ (καμία)	✓ (Virtual Alexa)	✓ (Telegram, Discord)

Πίνακας 5: Σύγκριση χαρακτηριστικών & δυνατοτήτων εφαρμογών IoT 2

Συγκρίνοντας τις λύσεις της αγοράς με την προτεινόμενη, με την βοήθεια του παραπάνω πίνακα, μπορούμε με ευκολία να συμπεράνουμε ότι η τελευταία καλύπτει με επιτυχία τα σημεία στα οποία οι άλλες αποτυγχάνουν όπως την ευκολία εγκατάστασης, την δυνατότητα απευθείας σύνδεσης με τον δρομολογητή, την πλούσια περιγραφή σεναρίων, την χρήση εξωτερικών πηγών δεδομένων, την στατιστική επεξεργασία δεδομένων, την προσβασιμότητα και την συμβατότητα με προϊόντα / υπηρεσίες τρίτων.

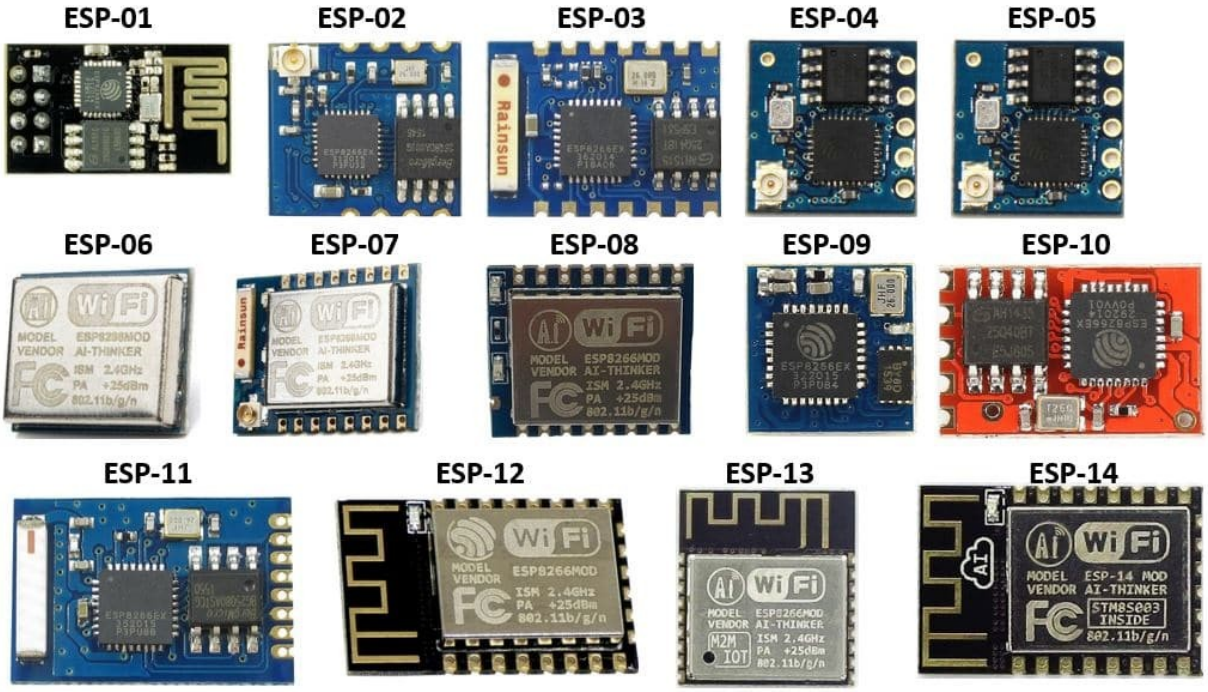
4 *Σχεδιασμός & υλοποίηση συσκευών*

4.1 Εισαγωγή

Στο κεφάλαιο αυτό, παρουσιάζονται εν συντομία οι μικροελεγκτές της οικογένειας ESP8266, ενώ γίνεται μια πιο αναλυτική αναφορά στα χαρακτηριστικά του ολοκληρωμένου κυκλώματος ESP-01, το οποίο και θα χρησιμοποιηθεί για την κατασκευή των συσκευών. Στα πλαίσια της εργασίας, παρέχεται η πλήρη κατασκευή μόνο δυο συσκευών IoT καθώς υπήρχαν οικονομικοί και χρονικοί περιορισμοί. Τα αντικείμενα που θα κατασκευαστούν είναι ένας αισθητήρας θερμοκρασίας - υγρασίας και δυο έξυπνες πρίζες. Για τον σχεδιασμό και την υλοποίηση των παραπάνω δίνεται η πλήρη λίστα των υλικών, τα ηλεκτρονικά κυκλώματα, ο αλγόριθμος και εικόνες της κατασκευής.

4.2 Η οικογένεια ESP8266

Για την κατασκευή των συσκευών IoT έγινε χρήση μικροελεγκτών (microchips) της οικογένειας ESP8266, επειδή είναι ευρέος γνωστοί στον χώρο και προσφέρουν τεράστιες δυνατότητες. Πρόκειται για χαμηλού κόστους μικροελεγκτές που επιτρέπουν ασύρματη σύνδεση στον διαδίκτυο μέσω τοπικών, οικιακών δικτύων (2.4 GHz WiFi) με προδιαγραφές IEEE 802.11. Οι παραπάνω παράγονται από πληθώρα κατασκευαστικών εταιριών, με τις πιο δημοφιλής, την Espressif Systems, Ai-Thinker, WeMos, Adafruit και Olimex. Σήμερα υποστηρίζονται μια σειρά από μικροελεγκτές του ίδιου είδους, με διαφορετικά χαρακτηριστικά μνήμης, ταχύτητας, φυσικού μεγέθους, πλήθος ακροδεκτών και κατανάλωσης ενέργειας προσφέροντας στους χρήστες πλήθος επιλογών ανάλογα με το έργο και τις προτιμήσεις τους. Τα παραπάνω ολοκληρωμένα κυκλώματα ταυτίζονται άμεσα με τον όρο Internet Of Things αφού αποτελούν την πρώτη επιλογή των κατασκευαστών τέτοιων έργων. [29][30]



Σχήμα 7 - Μικροελεγκτές της οικογένειας ESP8266

Οι βασικές λειτουργίες των παραπάνω μικροελεγκτών είναι η δυνατότητα σύνδεσης σε δρομολογητές και η μεταφορά δεδομένων, η είσοδος – έξοδος αναλογικών ή ψηφιακών σημάτων, η άμεση σύνδεση από κόμβο σε κόμβο (P2P – Peer to Peer) και η δημιουργία διακομιστών ιστοτόπου (Web Servers) [31].

Σήμερα μια μεγάλη σειρά από εμπορικές αλλά και ανοιχτού κώδικα (open source) βιβλιοθήκες ανάπτυξης λογισμικού (SDK) είναι διαθέσιμες, διευκολύνοντας έτσι κατά πολύ τον προγραμματισμό των ολοκληρωμένων κυκλωμάτων. Κάποιες από αυτές είναι η ESP Arduino Core (C++ based firmware), η οποία και χρησιμοποιήθηκε στην παρούσα εργασία, η ESP8266 BASIC (Open-source BASIC-like environment for IoT), η ESP-Open-RTOS (FreeRTOS open-source framework for ESP8266), η ESP-Open-SDK (Open integrated SDK for ESP8266) και άλλες [31].

Για το σκοπό της εργασίας, θα χρησιμοποιηθεί ο ESP-01 καθώς είναι ο καταλληλότερος μικροελεγκτής από άποψη χαρακτηριστικών κατανάλωσης ενέργειας, κόστους, πλήθος ακροδεκτών (pins) και φυσικού μεγέθους.

4.3 Ο μικροελεγκτής ESP-01

Πρόκειται για έναν μικροελεκτή εξαιρετικά χαμηλού κόστους. Στην Ελλάδα το κόστος κυμαίνεται από τέσσερα έως πέντε ευρώ και είναι πλέον διαθέσιμο σε αρκετά καταστήματα. Διαθέτει

- 8Mbit εξωτερική QSPI flash memory μεγέθους 1MByte

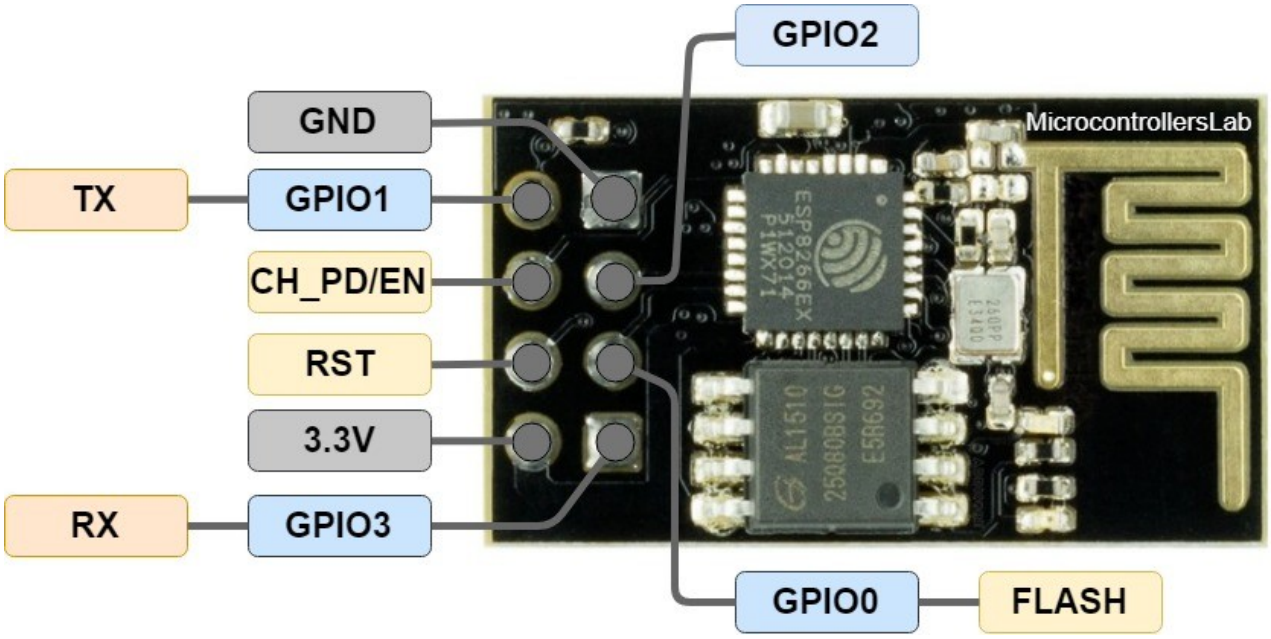
- 32-bit Tensilica Xtensa LX106 CPU με συχνότητα 80MHz.
- 3.3V supply
- 8 ακροδέκτες
- το μέγεθος του είναι 14.3 x 24.8mm.
- Βάρος 1.5 γραμμάρια

Παρακάτω γίνεται αναφορά των ακροδεκτών.

Ακροδέκτης	**Περιγραφή**
GND	Γείωση 0V
GPIO2	Γενικής χρήσης είσοδος / έξοδος
GPIO0	Γενικής χρήσης είσοδος / έξοδος
RX & GPIO3	Σειριακή είσοδο και γενικής χρήσης είσοδος / έξοδος
VCC	Παροχή ενέργειας 3.3V (μέγιστο 3.6V)
RST	Επαναφορά
CH_PD	Εκκίνηση
TX	Σειριακή έξοδο

Πίνακας 6: Ακροδέκτες μικροελεγκτή ESP-01

Το GND και το VCC χρησιμοποιούνται για την τροφοδότηση του ολοκληρωμένου κυκλώματος. Οι ακροδέκτες GPIO0, GPIO2 και GPIO3 μπορούν να χρησιμοποιηθούν από το λογισμικό προκειμένου να διαβάσουν ή να γράψουν αναλογικά ή ψηφιακά σήματα. Οι TX και RX είναι ακροδέκτες που υποστηρίζουν σειριακοί μετάδοση δεδομένων, οι οποίοι χρησιμοποιούνται κυρίως για τον προγραμματισμό του μικροελεγκτή. Το RST, επιτρέπει την επαναφορά του ολοκληρωμένου κυκλώματος δίνοντάς του ένα μικρό ερέθισμα. Το CH_PD χρησιμοποιείται από το ολοκληρωμένο κύκλωμα για τις λειτουργίες εκκίνησης. Επιπρόσθετα, ο μικροελεγκτής έχει ενσωματωμένους δυο φωτοδιόδους (leds). Ο πρώτος λειτουργεί ως ένδειξη λειτουργίας ενώ ο δεύτερος αφήνεται να χρησιμοποιηθεί από τον κώδικα. [32]



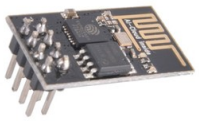
Σχήμα 8 - Ο μικροελεγκτής ESP-01

4.4 Αισθητήρας Θερμοκρασίας / Υγρασίας

Σε αυτή την ενότητα, γίνεται ο σχεδιασμός του αισθητήρα θερμοκρασίας – υγρασίας. Η συσκευή αυτή αποτελεί μια πηγή δεδομένων περιβάλλοντος και θα χρησιμοποιηθεί αργότερα από την εφαρμογή, για την δημιουργία σεναρίων IoT. [33]

4.4.1 Εξαρτήματα Κατασκευής

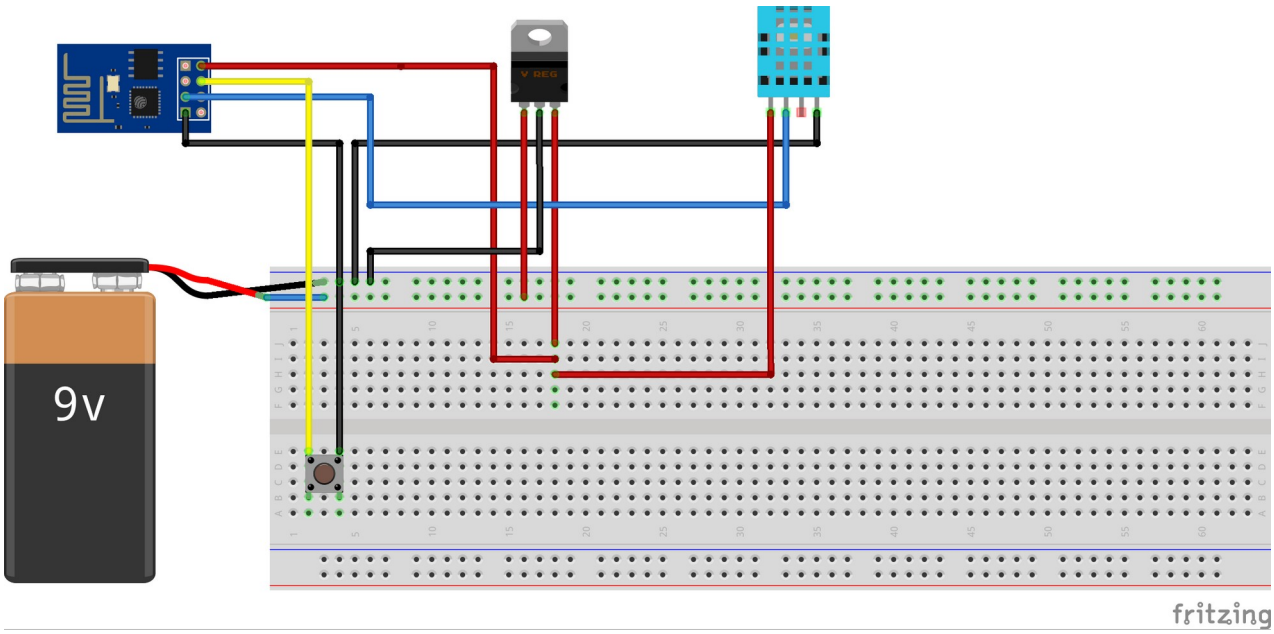
Για την κατασκευή της συσκευής γίνεται χρήση του ολοκληρωμένου κυκλώματος ESP-01, ο αισθητήρας θερμοκρασίας – υγρασίας DHT11, ο ρυθμιστής διαφοράς δυναμικού (voltage regulator) LF33ABV και ένα κουμπί (push button) για την επαναφορά της συσκευής.

Εξάρτημα	Περιγραφή
	ESP-01, ο μικροελεγκτής που αναφέρθηκε στην προηγούμενη ενότητα. Τροφοδοσία με 3.3V
	DHT11, αισθητήρας θερμοκρασίας και υγρασίας περιβάλλοντος. Τροφοδοσία με 3V - 5V
	LF33ABV, ρυθμιστής διαφοράς δυναμικού (voltage regulator) Είσοδος 2.5V – 16V Έξοδος 3.3V
	Κουμπί (push button) για την επαναφορά του ολοκληρωμένου κυκλώματος ESP-01

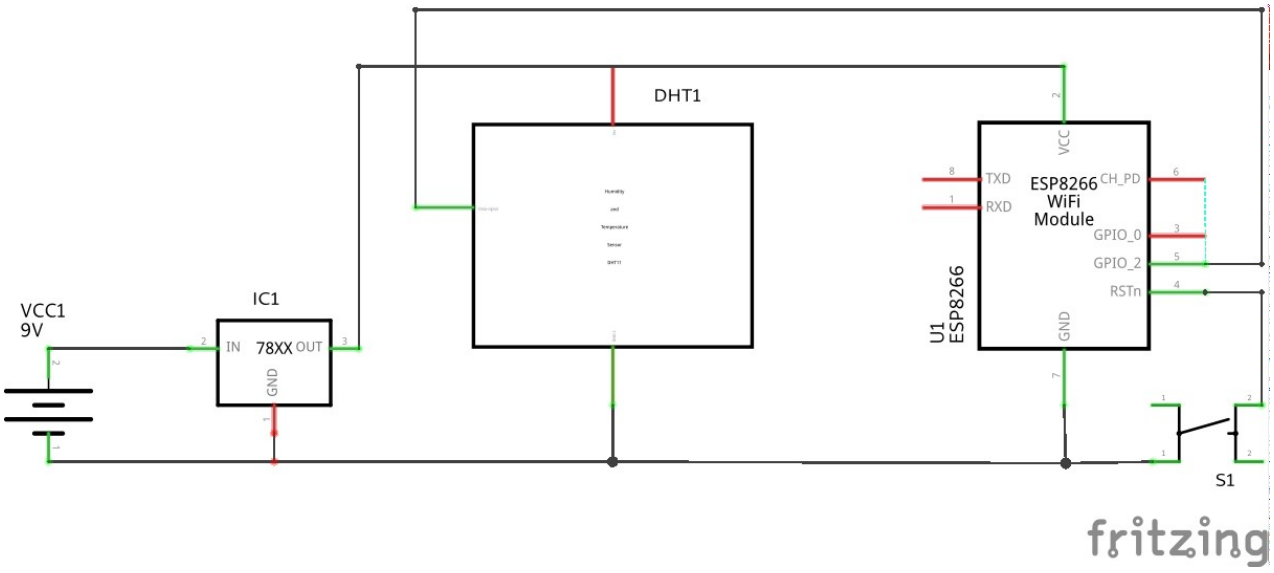
Πίνακας 7: Εξαρτήματα κατασκευής αισθητήρα

4.4.2 Ηλεκτρονικό Κύκλωμα

Στις παρακάτω εικόνες δίνεται το ηλεκτρονικό σχήμα και κύκλωμα της συσκευής. Αρχικά χρησιμοποιούμε τον ρυθμιστή διαφοράς δυναμικού LF33ABV για να ρίξουμε σταδιακά τα 9V της πηγής στα 3.3V, που απαιτούνται αυστηρά από τον μικροελεγκτή ESP-01 και τον αισθητήρα DHT11. Στην συνέχεια συνδέουμε τον ακροδέκτη γενικής χρήσης GPIO2 του ολοκληρωμένου στον ακροδέκτη δεδομένων του αισθητήρα θερμοκρασίας – υγρασίας DHT11, ώστε να μπορούμε να διαβάσουμε τις σχετικές τιμές από το περιβάλλον. Τέλος, συνδέουμε το κουμπί (pushbutton) επαναφοράς στον ακροδέκτη RST.



Σχήμα 9 - Σχήμα συσκευής αισθητήρα



Σχήμα 10 - Κύκλωμα συσκευής αισθητήρα

4.4.3 Αλγόριθμος Συσκευής

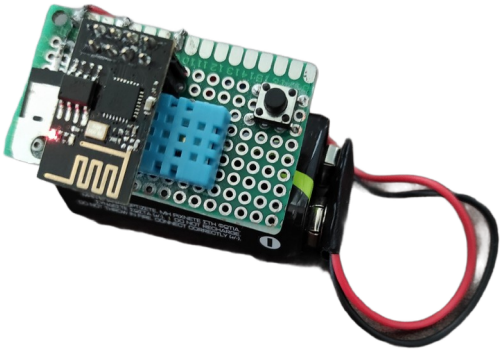
Στην παρακάτω εικόνα φαίνεται ο αλγόριθμος του αισθητήρα θερμοκρασίας – υγρασίας. Αρχικά γίνεται έλεγχος στην μνήμη EEPROM για τον εντοπισμό διαπιστευτηρίων οικιακού δικτύου. Σε περίπτωση που η μνήμη είναι άδεια, η συσκευή δημιουργεί ένα ανοιχτό τοπικό δίκτυο με την ονομασία “IoT Sensor” ενώ παράλληλα στήνει έναν web server με σκοπό την αλληλεπίδραση με την εφαρμογή του χρήστη. Στην συνέχεια ζητείται από τον τελευταίο να εισάγει τα διαπιστευτήρια του οικιακού του δικτύου και τον μοναδικό κωδικό του έξυπνου τόπου που παράγεται από την εφαρμογή. Μετά την επιτυχή σύνδεση της συσκευής με το διαδίκτυο, δεδομένα θερμοκρασίας και υγρασίας στέλνονται ανά τακτά χρονικά διαστήματα στην πλατφόρμα που αναπτύχθηκε στην διπλωματική. Στην περίπτωση που η μνήμη έχει ήδη τα παραπάνω διαπιστευτήρια, η συσκευή μεταβαίνει απευθείας στον ατέρμονα βρόγχο για την αποστολή μετρήσεων. Σε περίπτωση που ο χρήστης υποβάλει λανθασμένα διαπιστευτήρια δικτύου ή το δίκτυο δεν είναι πλέον διαθέσιμο, γίνεται μετάβαση στην αρχή του κώδικα καλώντας την συνάρτηση restartScript().

```
if not WiFiCredentialsSavedToEEPROM:
    createAccessPoint();
    createWebServer();
    if WiFiCredentialsGiven():
        if connectToWiFi():
            storeWiFiCredentialsToEEPROM();
            while(forever):
                pushToInfluxDB(humidity, temperature);
        else restartScript();
    else restartScript();
else
    retrieveWiFiCredentialsFromEEPROM();
    if connectToWiFi():
        while(forever):
            pushToInfluxDB(humidity, temperature);
    else:
        eraseEEPROM();
        restartScript();
```

Σχήμα 11 - Αλγόριθμος συσκευής αισθητήρα

4.4.4 Κατασκευή

Στις παρακάτω εικόνες απεικονίζεται η συσκευή, το μέγεθος της οποίας δεν ξεπερνά τις διαστάσεις 48mm μήκος x 25mm πλάτος x 50mm ύψος.



Σχήμα 13 - Συσκευή αισθητήρα 1

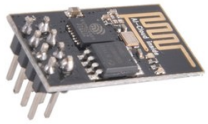
Σχήμα 12 - Συσκευή αισθητήρα 2

4.5 Έξυπνη πρίζα

Σε αυτή την ενότητα πραγματοποιείται ο σχεδιασμός της έξυπνης πρίζας. Η κατάσταση αυτής συσκευής μπορεί να τροποποιηθεί από σενάρια IoT ή χειροκίνητα από τον χρήστη μέσω της εφαρμογής. [34]

4.5.1 Εξαρτήματα Κατασκευής

Για την κατασκευή της έξυπνης πρίζας γίνεται χρήση του ολοκληρωμένου ESP-01, ο ρυθμιστής διαφοράς δυναμικού (voltage regulator) HLK-PM03 για μετατροπή από 220V στα 3.3V, ένα Relay στα 5V, μια ασφάλεια και ένα κουμπί (pushbutton) για την επαναφορά της συσκευής.

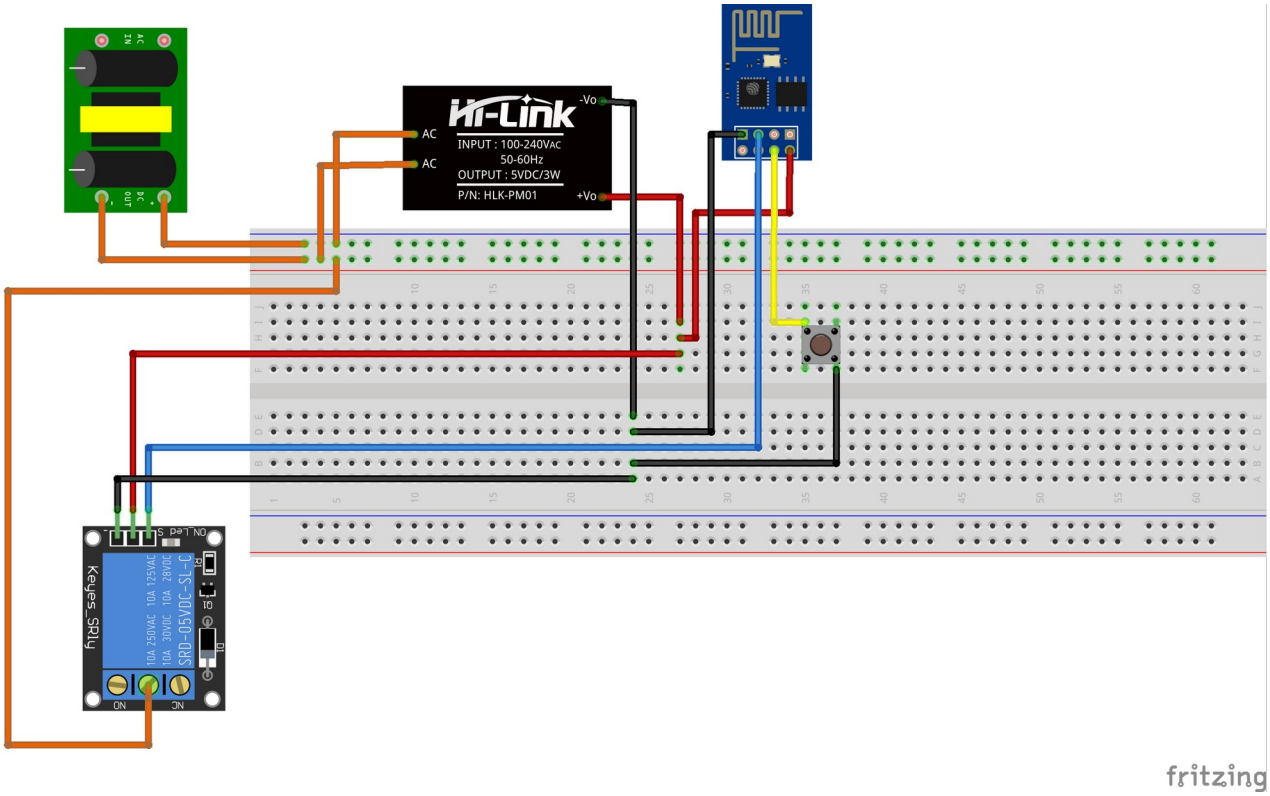
	ESP-01, ο μικροελεγκτής που αναφέρθηκε στην Ενότητα 4.3 Τροφοδοσία με 3.3V
	HLK-PM03, ρυθμιστής διαφοράς δυναμικού (voltage regulator) για μετατροπή από 220V (πρίζες με ευρωπαϊκά πρότυπα) στα 3.3V
	Relay 5V, διακόπτης που διεγείρεται μέσω χαμηλής έντασης ρεύματος
	Κουμπί (push button) για την επαναφορά το ολοκληρωμένου ESP-01
	Ασφάλεια, για την προστασία του ηλεκτρονικού κυκλώματος και της συσκευής που τοποθετείται στην πρίζα.

Πίνακας 8: Εξαρτήματα κατασκευής πρίζας

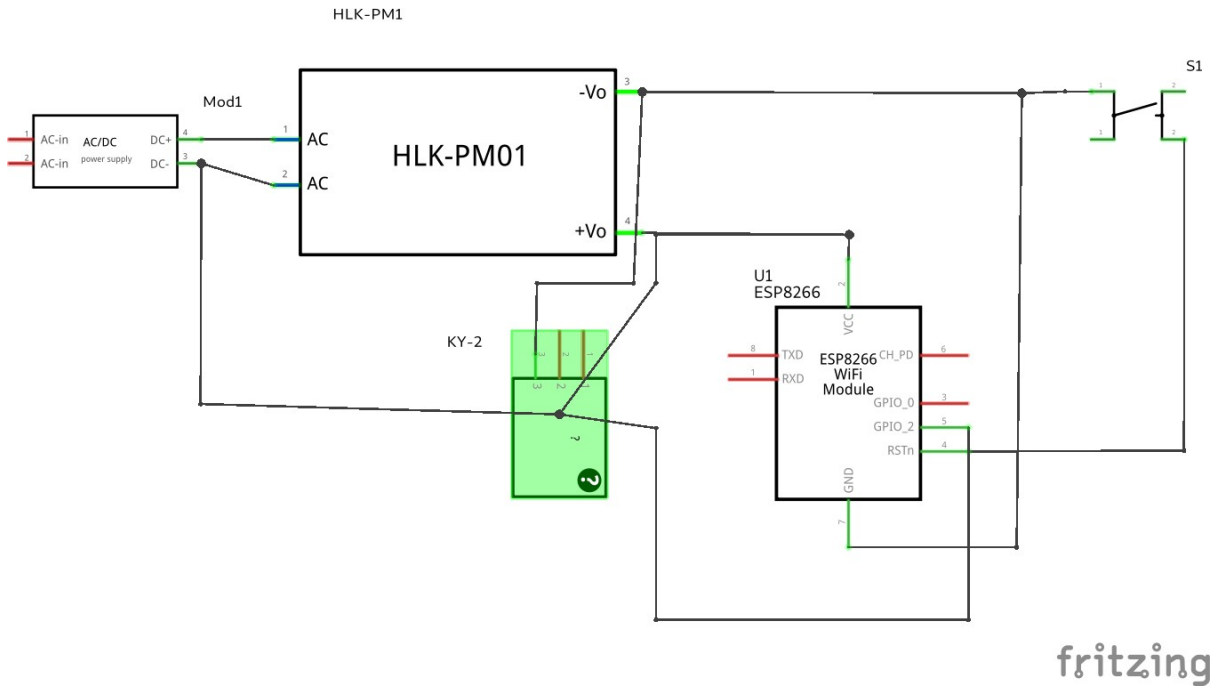
4.5.2 Ηλεκτρονικό Κύκλωμα

Στις παρακάτω εικόνες δίνεται το ηλεκτρονικό σχήμα και κύκλωμα της συσκευής. Αρχικά μετατρέπουμε τα 220V της πρίζας (βάση ευρωπαϊκών προτύπων) στα 3.3V που απαιτούνται αυστηρά από τον μικροελεγκτή ESP-01 χρησιμοποιώντας το ολοκληρωμένο HLK-PM03. Στην

συνέχεια συνδέουμε τον ακροδέκτη γενικής χρήσης GPIO2 στο Relay. Στέλνοντας ένα ψηφιακό σήμα μικρής ισχύος, μπορούμε να διεγείρουμε τον διακόπτη. Επίσης συνδέουμε το κεντρικό ρεύμα στο Relay στην υποδοχή Common και έπειτα το χρησιμοποιούμε στην πρίζα είτε μέσω της υποδοχής CN (current on) είτε της CF (current off). Στην πρώτη περίπτωση το κύκλωμα είναι ανοιχτό που σημαίνει οτι το διαπερνά ρεύμα μόνο αν σταλθεί σήμα διέγερσης. Στην δεύτερη περίπτωση το κύκλωμα είναι κλειστό που σημαίνει οτι δεν το διαπερνά ρεύμα μόνο αν σταλθει σήμα διέγερσης. Τέλος, όπως και στην συσκευή του αισθητήρα, γίνεται σύνδεση ενός κουμπιού (pushbutton) με τον ακροδέκτη RST για την επαναφορά του μικροελεγκτή στις εργοστασιακές ρυθμίσεις.



Σχήμα 14 - Σχήμα συσκευής πρίζας



Σχήμα 15 - Κύκλωμα συσκευής πρίζας

4.5.3 Αλγόριθμος Συσκευής

Στην παρακάτω εικόνα φαίνεται ο αλγόριθμος της έξυπνης πρίζας. Αρχικά, παρόμοια με την συσκευή του αισθητήρα, γίνεται έλεγχος στην ενσωματωμένη μνήμη για τον εντοπισμό πληροφοριών του τοπικού δικτύου. Σε περίπτωση που η μνήμη είναι άδεια, η συσκευή δημιουργεί ένα ανοιχτό τοπικό δίκτυο με την ονομασία “IoT Plug” και παράλληλα τρέχει έναν διακομιστή ιστοτόπου με στόχο την αλληλεπίδραση με τον χρήστη. Στην συνέχεια ζητείται από τον τελευταίο να εισάγει τα διαπιστευτήρια του οικιακού του δικτύου και τον μοναδικό κωδικό του έξυπνου τόπου που παράγεται από την εφαρμογή. Μετά την επιτυχή σύνδεση της συσκευής με το διαδίκτυο, η τελευταία μπορεί να ενημερώνεται για αλλαγές κατάστασης ή της συχνότητας ενημέρωσης. Στην περίπτωση που η μνήμη έχει ήδη τα παραπάνω διαπιστευτήρια, η συσκευή εκτελεί απευθείας το κομμάτι του κώδικα που είναι υπεύθυνο για την λήψη και εφαρμογή των ενημερώσεων. Σε περίπτωση που ο χρήστης υποβάλει λανθασμένα διαπιστευτήρια δικτύου ή το δίκτυο δεν είναι πλέον διαθέσιμο, γίνεται μετάβαση στην αρχή του κώδικα καλώντας την συνάρτηση restartScript().

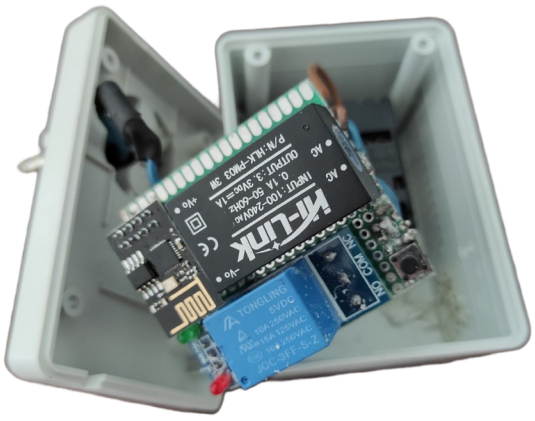
```
if not WiFiCredentialsSavedToEEPROM:
    createAccessPoint();
    createWebServer();
    if WiFiCredentialsGiven():
        if connectToWiFi():
            storeWiFiCredentialsToEEPROM();
            while(forever):
                device = pullDeviceInfo();
                deviceStatus(device.status);
        else restartScript();
    else restartScript();
else
    retrieveWiFiCredentialsFromEEPROM();
    if connectToWiFi():
        while(forever):
            device = pullDeviceInfo();
            deviceStatus(device.status);
    else:
        eraseEEPROM();
        restartScript();
```

Σχήμα 16 - Αλγόριθμος συσκευής πρίζας

4.5.4 Κατασκευή

Στις παρακάτω εικόνες απεικονίζεται η συσκευή.

Σχήμα 18 - Συσκευή πρίζας 1



Σχήμα 19 - Συσκευή πρίζας 2

Σχήμα 17 - Συσκευή πρίζας 3

5 *Μοντελοποίηση Σεναρίων*

5.1 Εισαγωγή

Όπως αναφέρθηκε στην Ενότητα 2.3, ένα από τα κρίσιμα ζητήματα των εφαρμογών IoT είναι η δυσκολία του χρήστη να προκαθορίσει οδηγίες, που θα εκτελούν τα αντικείμενα ανάλογα με ερεθίσματα που αντλούν από το εξωτερικό περιβάλλον ή από οποιαδήποτε πηγή δεδομένων. Στην προταθείσα αρχιτεκτονική, οι οδηγίες αυτές περιγράφονται με την μορφή σεναρίων (scenarios). Η γενική ιδέα είναι ότι ο χρήστης θα εισάγει ψευδοκώδικα σε έναν επεξεργαστή κειμένου μέσω της εφαρμογής. Ο τελευταίος θα πραγματοποιεί συντακτική ανάλυση και έλεγχο των εισαχθέντων κανόνων, εξασφαλίζοντας την σωστή λειτουργία των σεναρίων. Στο κεφάλαιο αυτό αναφέρονται όλα τα συστατικά μέρη του ψευδοκώδικα (μεταβλητές, τελεστές, ροές έλεγχου και συναρτήσεις).

5.2 Χαρακτηριστικά ψευδοκώδικα

Για την δημιουργία των σεναρίων ο χρήστης καλείτε να γράψει ψευδοκώδικα βασισμένο στην γλώσσα προγραμματισμού php. Ο ψευδοκώδικας είναι αρκετά απλοποιημένος, αφού έχει γίνει απομάκρυνση των ενσωματωμένων συναρτήσεων, βρόγχων και κλάσεων της γλώσσας τόσο για θέμα ευκολίας χρήσης όσο και ασφάλειας. Έτσι ο χρήστης μπορεί να χρησιμοποιήσει μόνο μεταβλητές, ροές ελέγχου, συναρτήσεις συστήματος και όλους τους τελεστές. Η παραπάνω επιλογή επιτρέπει την χρήση του συντακτικού αναλυτή της γλώσσας για την αποφυγή εισαγωγής εσφαλμένου κώδικα. Επίσης τα εύστοχα μηνύματα σφάλματος της γλώσσας βοηθούν τον χρήστη να εντοπίσει πιθανά λάθη σύνταξης.

5.3 Μεταβλητές

Οι μεταβλητές αποτελούν έναν φιλικό τρόπο αναφοράς σε δεδομένα, χρησιμοποιώντας προσιτές ως προς τον χρήστη ονομασίες. Επιτρέπεται η αποθήκευση οποιουδήποτε είδους δεδομένων (πχ ακέραιοι, δεκαδικοί, αλφαριθμητικά κλπ). Στον ψευδοκώδικα της εφαρμογής, οι μεταβλητές ονοματίζονται ακριβώς όπως στην php.

- θα πρέπει να ξεκινούν με το σύμβολο του δολαρίου “$”
- το όνομα θα πρέπει να αρχίζει με γράμμα ή την κάτω παύλα
- στην συνέχεια θα πρέπει να περιέχουν μόνο αλφαριθμητικούς χαρακτήρες ή την κάτω παύλα
- υπάρχει διάκριση μεταξύ πεζών και κεφαλαίων

Αποδεκτά ονόματα: $wind, $Temperature, $_humidity1

Μη αποδεκτά ονόματα: $0movement, isFire, $earth-humidity

5.4 Τελεστές

Οι τελεστές χρησιμοποιούνται για την εφαρμογή λειτουργιών ανάμεσα σε μεταβλητές και δεδομένα. Στην ψευδογλώσσα επιτρέπονται όλοι οι τελεστές που υποστηρίζονται από την php. Η τελευταία τους ομαδοποιεί στις ακόλουθες κατηγορίες.

Αριθμητικούς τελεστές (Arithmetic operators), για την εφαρμογή μαθηματικών πράξεων.

Τελεστής	**Περιγραφή**
+	Πρόσθεση.
-	Αφαίρεση.
*	Πολλαπλασιασμός.
/	Διαίρεση.
%	Υπόλοιπο ακέραιας διαίρεσης.
++	Αύξηση ακεραίου κατά ένα.
--	Μείωση ακεραίου κατά ένα.

Πίνακας 9: Αριθμητικοί τελεστές

Τελεστές ανάθεσης (Assignment operators), για την ανάθεση τιμών σε μεταβλητές.

Τελεστής	**Περιγραφή**
=	Ανάθεση του δεξιού τελεστέου στον αριστερό.
+=	Αύξηση του αριστερού τελεστέου με τον δεξιό.
-=	Μείωση του αριστερού τελεστέου με τον δεξιό.
*=	Πολλαπλασιασμός του αριστερού τελεστέου με τον δεξιό και ανάθεση αποτελέσματος στον πρώτο.
/=	Διαίρεση του αριστερού τελεστέου με τον δεξιό και ανάθεση αποτελέσματος στον πρώτο.
%=	Ανάθεση υπολοίπου ακέραιας διαίρεσης του αριστερού τελεστέου με τον δεξιό και ανάθεση αποτελέσματος στον πρώτο.

Πίνακας 10: Τελεστές ανάθεσης

Τελεστές σύγκρισης (Comparison operators), για την σύγκριση τιμών.

Τελεστής	**Περιγραφή**
==	Ελέγχει αν οι τελεστέοι είναι ίσοι.
!=	Ελέγχει αν οι τελεστέοι είναι άνισοι
>	Ελέγχει αν ο πρώτος τελεστέος είναι μεγαλύτερος του δευτέρου
<	Ελέγχει αν ο δεύτερος τελεστέος είναι μεγαλύτερος του πρώτου
>=	Ελέγχει αν ο πρώτος τελεστέος είναι μεγαλύτερος ή ίσος του δευτέρου
<=	Ελέγχει αν ο δεύτερος τελεστέος είναι μεγαλύτερος ή ίσος του πρώτου

Πίνακας 11: Τελεστές σύγκρισης

Λογικοί τελεστές (Logical operators), για την δημιουργία λογικών προτάσεων.

Τελεστής	Περιγραφή
and ή &&	Αληθής αν και οι δυο τελεστέοι είναι αληθής
or ή ||	Αληθής αν ο ένας από τους δυο τελεστέους είναι αληθής
!	Αντιστροφή της λογικής τιμής μιας λογικής έκφρασης/πρότασης

Πίνακας 12: Λογικοί τελεστές

Τελεστές αλφαριθμητικών (String operators), για την συνένωση αλφαριθμητικών.

Τελεστής	Περιγραφή
.	Συνένωση αλφαριθμητικών
.=	Συνένωση του δεξιού αλφαριθμητικού με το αριστερό και εκχώρηση του αποτελέσματος στον αριστερό τελεστέο.

Πίνακας 13: Τελεστές αλφαριθμητικών

5.5 Ροές Ελέγχου

Οι ροές ελέγχου επιτρέπουν την εκτέλεση εντολών μόνο αν ικανοποιείται η συνθήκη που τις περιλαμβάνει. Στην ψευδογλώσσα της εφαρμογής επιτρέπονται μόνο οι ροές ελέγχου if, if-then-else και if {} [else if {}]* else {}. Παραδείγματα:

- if(temperature > 10){ foo(); }
- if(humidity < 70){ foo1(); } else { foo2(); }.

5.6 Συναρτήσεις Συστήματος

Οι συναρτήσεις συστήματος αποτελούν λειτουργίες, οι οποίες επιτρέπουν στον χρήστη της εφαρμογής να διαχειριστεί πηγές δεδομένων, συσκευές και δράσεις προκειμένου να σχεδιάσει σενάρια που ικανοποιούν σε βάθος τις ανάγκες του. Οι συναρτήσεις αυτές κατηγοριοποιούνται σε δράσεως, ανάκτησης και βοηθητικές.

5.6.1 Συναρτήσεις Ανάκτησης

Με αυτές τις συναρτήσεις ο χρήστης μπορεί να συλλέξει μετρήσεις από τις συσκευές του ή από οποιαδήποτε πηγή βασιζόμενη σε API. Θα μπορούσαν να παρομοιαστούν και ως είσοδοι μιας διαδικασίας. Το σύστημα απαιτεί τουλάχιστον την χρήση μιας τέτοιας συνάρτησης για την επιτυχή εισαγωγή του σεναρίου, διαφορετικά εμφανίζεται σχετικό μήνυμα σφάλματος. Παρακάτω παρουσιάζεται η σύνταξη αυτών, όπως απαιτείτε από το σύστημα, και δίνεται επεξήγηση για κάθε συνάρτηση ξεχωριστά.

- **get_max($device, $type, $range, $req_every)** → Επιστρέφει την μεγαλύτερη τιμή ενός συνόλου. Δέχεται ως ορίσματα το μοναδικό αναγνωριστικό της συσκευής, το είδος της μέτρησης, το χρονικό διάστημα από το οποίο συλλέγονται οι σχετικές μετρήσεις του συνόλου και τον περιοδικό χρόνο εκτέλεσης.
- **get_min($device, $type, $range, $req_every)** → Επιστρέφει την μικρότερη τιμή ενός συνόλου. Δέχεται ως ορίσματα το μοναδικό αναγνωριστικό της συσκευής, το είδος της μέτρησης, το χρονικό διάστημα από το οποίο συλλέγονται οι σχετικές μετρήσεις του συνόλου και τον περιοδικό χρόνο εκτέλεσης.
- **get_mean($device, $type, $range, $req_every)** → Επιστρέφει τον μέσο όρο τιμών από ένα σύνολο. Δέχεται ως ορίσματα το μοναδικό αναγνωριστικό της συσκευής, το είδος της μέτρησης, το χρονικό διάστημα από το οποίο συλλέγονται οι σχετικές μετρήσεις του συνόλου και τον περιοδικό χρόνο εκτέλεσης.
- **get_median($device, $type, $range, $req_every)** → Επιστρέφει την τιμή που βρίσκεται στη μέση μια ταξινομημένης σειράς δεδομένων. Δέχεται ως ορίσματα το μοναδικό αναγνωριστικό της συσκευής, το είδος της μέτρησης, το χρονικό διάστημα από το οποίο συλλέγονται οι σχετικές μετρήσεις του συνόλου και τον περιοδικό χρόνο εκτέλεσης.
- **get_mode($device, $type, $range, $req_every)** → Επιστρέφει την τιμή με τις περισσότερες εμφανίσεις από ένα σύνολο. Δέχεται ως ορίσματα το μοναδικό αναγνωριστικό της συσκευής, το είδος της μέτρησης, το χρονικό διάστημα από το οποίο συλλέγονται οι σχετικές μετρήσεις του συνόλου και τον περιοδικό χρόνο εκτέλεσης.
- **get_sum($device, $type, $range, $req_every)** → Επιστρέφει το άθροισμα των τιμών ενός συνόλου. Δέχεται ως ορίσματα το μοναδικό αναγνωριστικό της συσκευής, το είδος της μέτρησης, το χρονικό διάστημα από το οποίο συλλέγονται οι σχετικές μετρήσεις του συνόλου και τον περιοδικό χρόνο εκτέλεσης.

- **get_stddev($device, $type, $range, $req_every)** → Επιστρέφει την τυπική απόκλιση ενός συνόλου. Δέχεται ως ορίσματα το μοναδικό αναγνωριστικό της συσκευής, το είδος της μέτρησης, το χρονικό διάστημα από το οποίο συλλέγονται οι σχετικές μετρήσεις του συνόλου και τον περιοδικό χρόνο εκτέλεσης.
- **get_spread($device, $type, $range, $req_every)** → Επιστρέφει την διαφορά ανάμεσα στην μέγιστη και ελάχιστη τιμή ενός συνόλου. Δέχεται ως ορίσματα το μοναδικό αναγνωριστικό της συσκευής, το είδος της μέτρησης, το χρονικό διάστημα από το οποίο συλλέγονται οι σχετικές μετρήσεις του συνόλου και τον περιοδικό χρόνο εκτέλεσης.
- **get_last($device, $type, $req_every)** → Επιστρέφει την τελευταία τιμή που εισήχθη στο σύστημα. Δέχεται ως ορίσματα το μοναδικό αναγνωριστικό της συσκευής, το είδος της μέτρησης και τον περιοδικό χρόνο εκτέλεσης.
- **sys_current_time($req_every)** → Επιστρέφει την τρέχουσα ώρα συστήματος. Το όρισμα είναι προαιρετικό και δηλώνει τον περιοδικό χρόνο εκτέλεσης. Αν δεν δοθεί όρισμα η συχνότητα εκτέλεσης είναι η προκαθορισμένη, δηλαδή ένα λεπτό.
- **fetch_endpoint($endpointId, $method, $params, $every)** → Επιστρέφει δεδομένα από μια υπηρεσία ιστοτόπου σε μορφή JSON. Δέχεται σαν όρισμα το μοναδικό αναγνωριστικό της υπηρεσίας ιστοτόπου, την μέθοδο του ερωτήματος (GET ή POST), τις τυχόν παραμέτρους και τέλος το χρονικό διάστημα κατά τον οποίο θα επαναλαμβάνεται η ανάκτηση.

5.6.2 Βοηθητικές Συναρτήσεις

Αφού έχει πραγματοποιηθεί η ανάκτηση δεδομένων, ο χρήστης είναι σε θέση να χρησιμοποιήσει συναρτήσεις για περεταίρω επεξεργασία και διεξαγωγή συμπερασμάτων. Οι συναρτήσεις αυτές δεν είναι υποχρεωτικές για την δημιουργία του σεναρίου.

- **str_contains($text, $subtext)** → Επιστρέφει true αν το δεύτερο όρισμα αλφαριθμητικού περιέχεται στο πρώτο, διαφορετικά επιστρέφει false.
- **my_time($time)** → Επιστρέφει ένα αντικείμενο χρόνου που μπορεί να χρησιμοποιηθεί από την εφαρμογή. Δέχεται σαν όρισμα μια ώρα ή ημερομηνία σύμφωνα με την προδιαγραφή ISO8601. Μερικά παραδείγματα ορισμάτων είναι τα ακόλουθα: “10:40”, “01/01/2021 12:00”, “Friday, 10 February” κτλ.

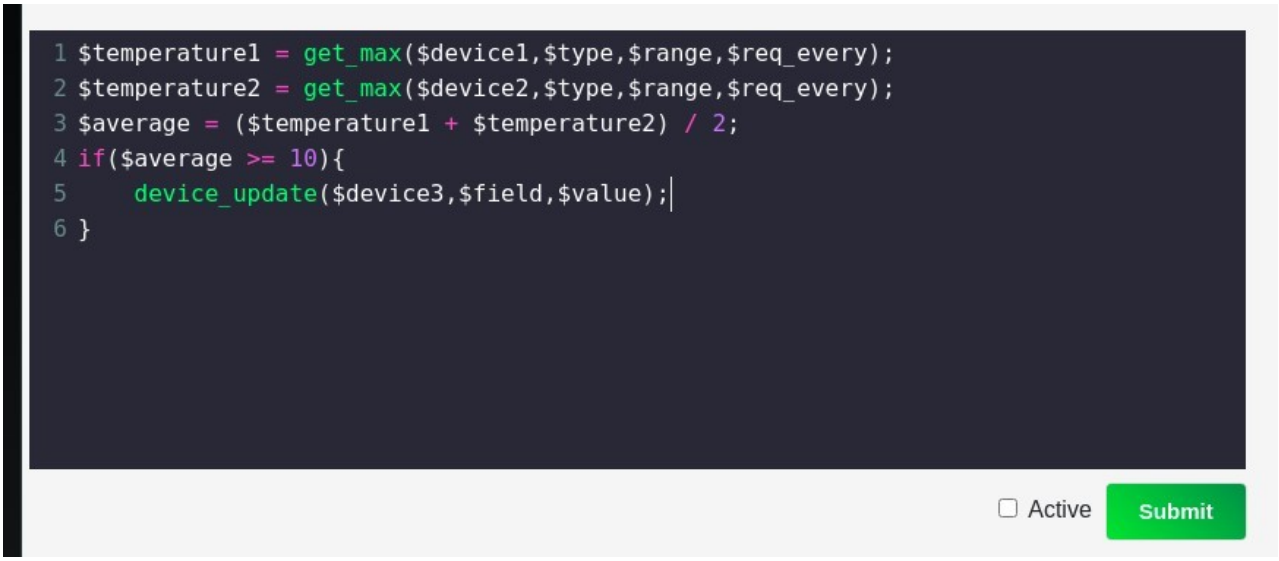
5.6.3 Συναρτήσεις Δράσεων

Έπειτα από την ανάκτηση και επεξεργασία των δεδομένων είναι απαραίτητο να πραγματοποιηθεί κάποια ενέργεια ή αλλιώς δράση. Το σύστημα απαιτεί την ύπαρξη τουλάχιστον μιας συνάρτησης δράσης προκειμένου να εκχωρήσει με επιτυχία το σενάριο, διαφορετικά τυπώνει ένα μήνυμα σφάλματος. Οι συναρτήσεις που υποστηρίζονται είναι οι ακόλουθες:

- **device_update($device, $field, $value)** → Τροποποιεί κάποιο χαρακτηριστικό της συσκευής. Αυτό μπορεί να είναι είτε η κατάσταση (on / off), είτε ο περιοδικός χρόνος ενημέρωσης της. Δέχεται ως ορίσματα το μοναδικό αναγνωριστικό της συσκευής, το γνώρισμα που πρόκειται να τροποποιηθεί καθώς και την νέα τιμή.
- **notify_discord($webhook,$msg)** → Δημιουργεί μια ειδοποίηση σε ένα κανάλι συζήτησης στην πλατφόρμα Discord. Δέχεται ως όρισμα ένα μοναδικό αναγνωριστικό hash που αντιπροσωπεύει το κανάλι συζήτησης και το επιθυμητό μήνυμα.
- **notify_telegram($token, $chat_id, $msg)** → Στέλνει ένα μήνυμα σε ένα προκαθορισμένο παράθυρο διαλόγου (chat) της πλατφόρμας Telegram. Δέχεται ως όρισμα ένα κλειδί χρήστη, ένα μοναδικό αναγνωριστικό hash που αντιπροσωπεύει το παράθυρο διαλόγου και το επιθυμητό μήνυμα.

5.6.4 Παράδειγμα σεναρίου

Σε αυτή την ενότητα, δίνεται ένα απλό παράδειγμα σεναρίου ως μια πρώτη εξήγηση για το τι είναι σενάριο και πως χρησιμοποιούνται τα παραπάνω δομικά συστατικά για τον ορισμό του.



Σχήμα 20 - Παράδειγμα σεναρίου

Αρχικά χρησιμοποιούμε μεταβλητές για να αποθηκεύσουμε προσωρινά τις μέγιστες θερμοκρασίες που παράγονται από δυο αισθητήρες. Η ανάκτηση επιτυγχάνεται μέσω της στατιστικής συνάρτησης ανάκτησης get_max(). Στην συνέχεια, υπολογίζουμε τον μέσο όρο των μέγιστων τιμών χρησιμοποιώντας αριθμητικούς τελεστές. Τέλος, γίνεται σύγκριση με μια προκαθορισμένη τιμή: αν η συνθήκη ικανοποιηθεί, καλείται η συνάρτηση δράσης device_update() για την ενημέρωση μιας τρίτης συσκευής.

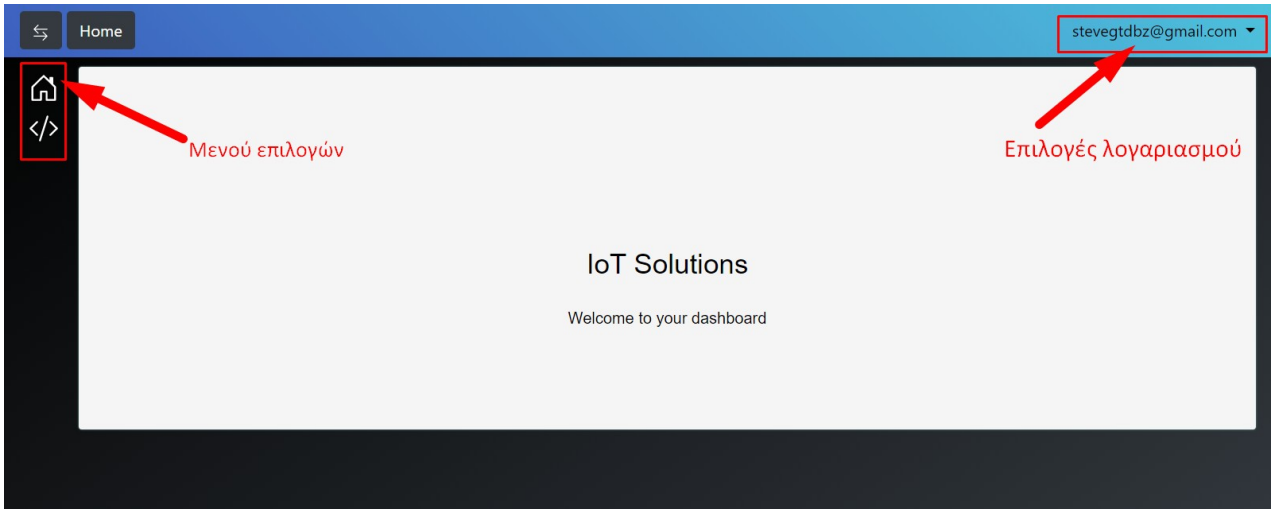
6 *Επίδειξη εφαρμογής*

6.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζονται αναλυτικά όλες οι λειτουργίες της εφαρμογής που αναπτύχθηκαν. Συγκεκριμένα παρέχεται ένας πλήρης οδηγός, βήμα προς βήμα, για την διαχείριση των έξυπνων τόπων (smart places) του χρήστη. Ο οδηγός περιλαμβάνει την δημιουργία έξυπνων τόπων, την εισαγωγή – επεξεργασία - διαγραφή συσκευών από έξυπνο τόπο, την εισαγωγή - επεξεργασία API, την δημιουργία σεναρίων και την οπτικοποίηση δεδομένων (data visualization). Επιπρόσθετα, παρουσιάζονται ορισμένα σενάρια IoT που υποστηρίζονται από την εφαρμογή.

6.2 Περιοχή Χρήστη

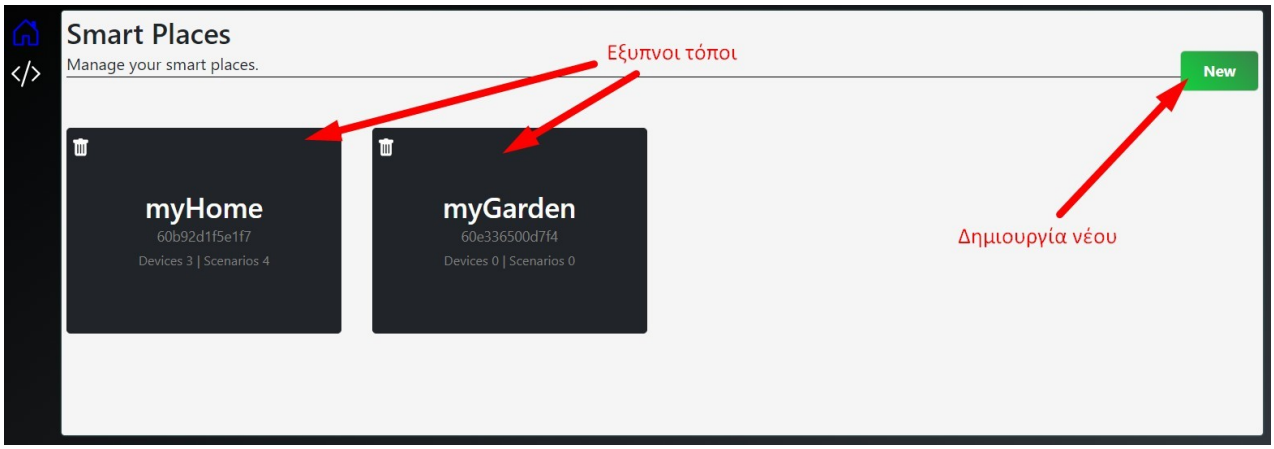
Μετά την επιτυχή ταυτοποίηση, ο χρήστης μεταφέρεται στο πάνελ διαχείρισης και από εκεί έχει πρόσβαση σε αγαθά και υπηρεσίες. Στην παρακάτω εικόνα, στα αριστερά φαίνεται το μενού επιλογών για την διαχείριση των έξυπνων τόπων και των υπηρεσιών ιστού ενώ στα δεξιά είναι τοποθετημένες οι επιλογές λογαριασμού.



Σχήμα 21 - Περιοχή χρήστη

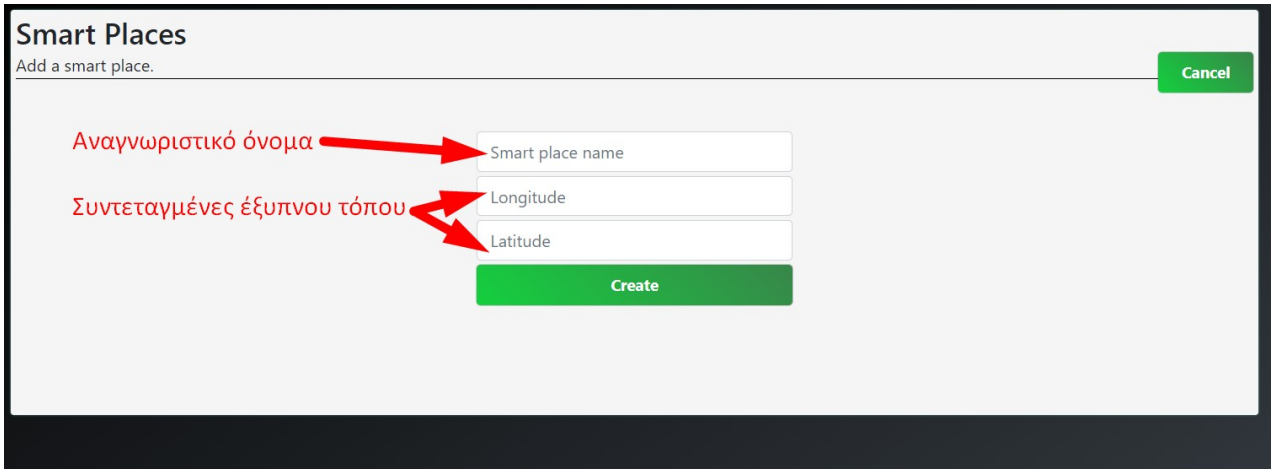
6.3 Διαχείριση έξυπνων τόπων

Αρχικά, ο χρήστης έχει την δυνατότητα εμφάνισης των έξυπνων τόπων που έχει στην κατοχή του ή άλλων όπου του έχει παραχωρηθεί η σχετική εξουσιοδότηση από τον διαχειριστή. Με την διαγραφή του έξυπνου τόπου (πατώντας το σχετικό εικονίδιο κάδου στο αριστερό μέρος του πλαισίου του έξυπνου τόπου) αυτόματα διαγράφονται όλα τα συστατικά μέρη που το συγκροτούν, όπως συσκευές, σενάρια, μετρήσεις και εγγραφές ιχνηλασιμότητας.



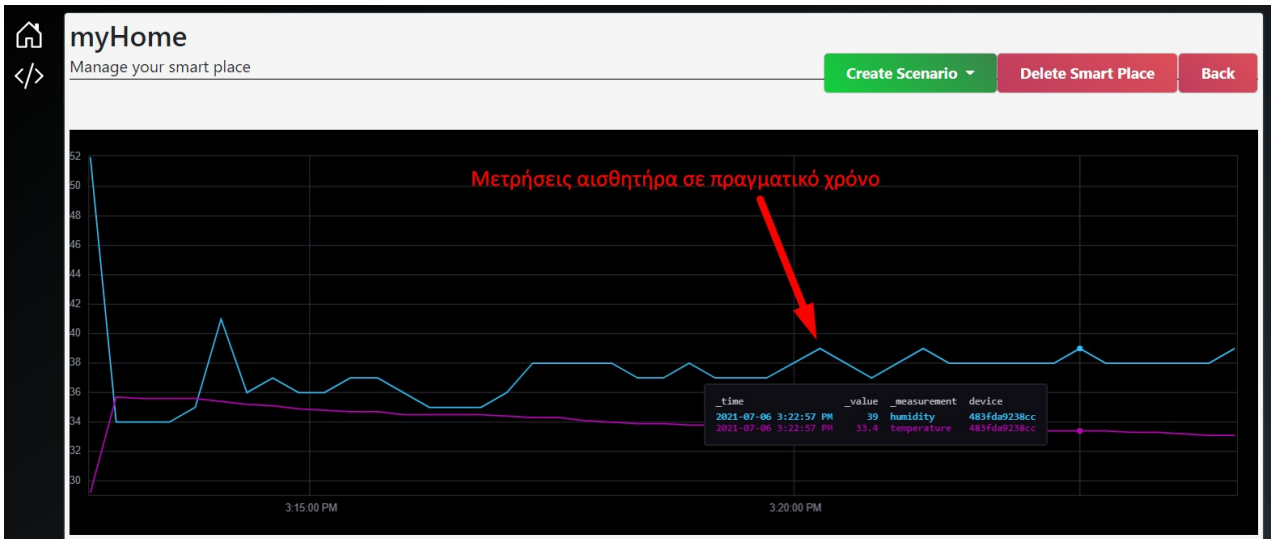
Σχήμα 22 - Διαθέσιμοι έξυπνοι τόποι

Πατώντας το κουμπί “New”, εμφανίζεται η παρακάτω φόρμα εισαγωγής νέου έξυπνου τόπου, όπου απαιτούνται οι συντεταγμένες και ένα αναγνωριστικό όνομα.



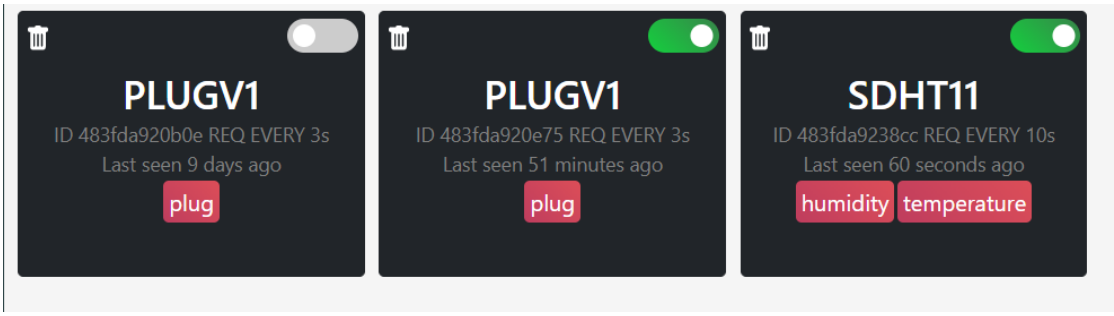
Σχήμα 23 - Δημιουργία έξυπνου τόπου

Σε έναν έξυπνο τόπο, ο χρήστης έχει άμεσα πρόσβαση σε όλες τις υπηρεσίες και αγαθά που τον απαρτίζουν. Στα παρακάτω στιγμιότυπα οθόνης φαίνεται η κεντρική σελίδα διαχείρισης ενός τόπου. Εμφανίζεται η περιοχή μετρήσεων, συσκευών, σεναρίων και εγγραφών ιχνηλασιμότητας για τον εντοπισμό αλλαγών κατάστασης των συσκευών.



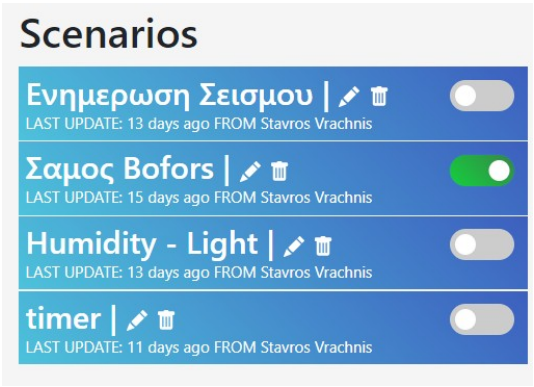
Σχήμα 24 - Γράφημα μετρήσεων

Στην παραπάνω εικόνα, εμφανίζονται μετρήσεις θερμοκρασίας και υγρασίας σε πραγματικό χρόνο. Ο χρήστης τοποθετώντας τον κέρσορα στο διάγραμμα, μπορεί να ενημερωθεί για τον ακριβή χρόνο, τιμή, είδος της μέτρησης καθώς και το μοναδικό αναγνωριστικό της πηγής. Στην συνέχεια εμφανίζονται οι διαθέσιμες συσκευές του τόπου. Παρατηρούνται δυο έξυπνες πρίζες και ένας αισθητήρας θερμοκρασίας - υγρασίας



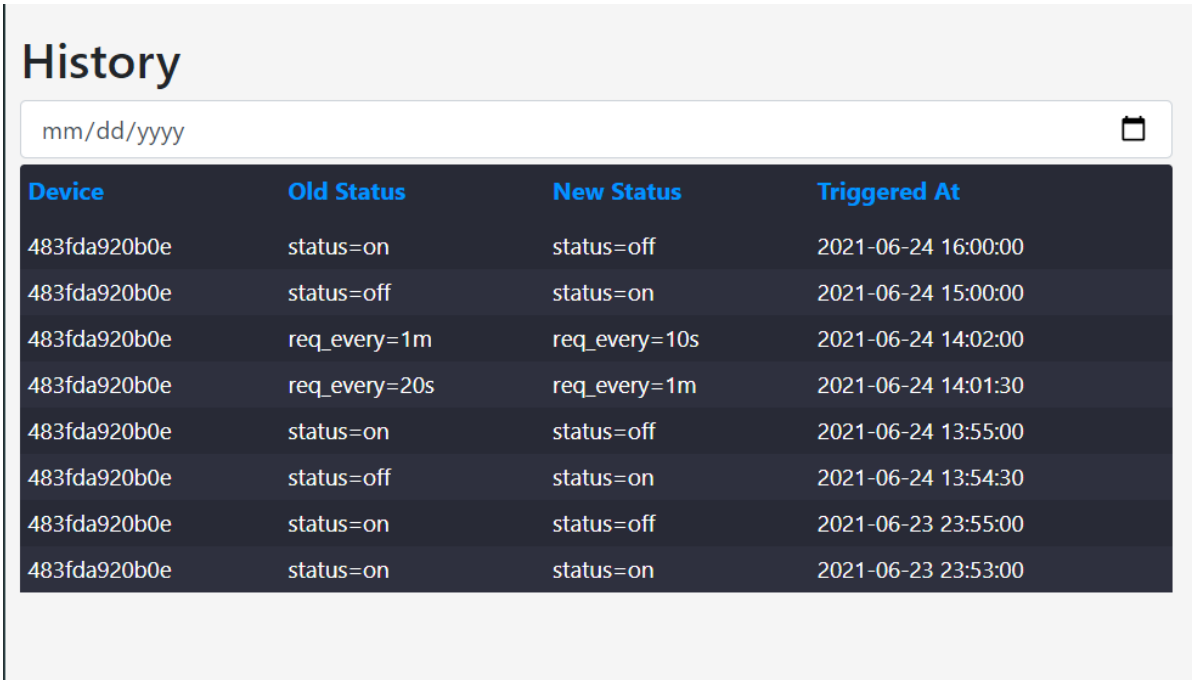
Σχήμα 25 - Περιοχή συσκευών IoT χρήστη

Στην παρακάτω εικόνα, εμφανίζονται τα διαθέσιμα σενάρια που εκτελούνται από τον έξυπνο τόπο. Ο χρήστης μπορεί να τα επεξεργαστεί, διαγράψει, απενεργοποιήσει ή να τα ενεργοποιήσει αντίστοιχα πατώντας τα αντίστοιχα εικονίδια. Επίσης, τυπώνεται η χρονοσφραγίδα τελευταίας ενημέρωσης καθώς και το όνομα του χρήστη που την επιτέλεσε.



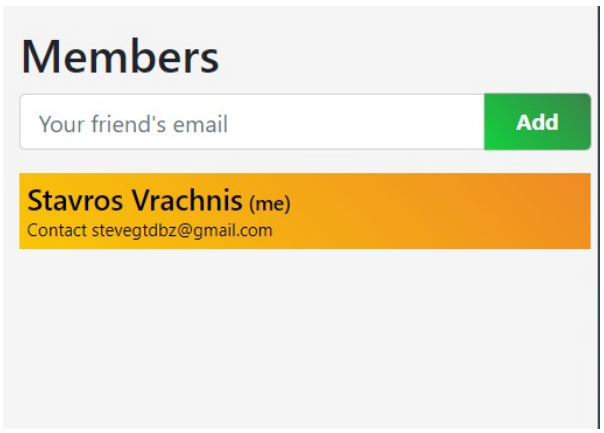
Σχήμα 26 - Περιοχή σεναρίων χρήστη

Κάθε φορά που τροποποιείται κάποια ιδιότητα μιας συσκευής, το σύστημα φροντίζει να καταγράφει την αλλαγή αυτή σε ανάλογες εγγραφές ιχνηλασιμότητας (traceability records). Στο παρακάτω πίνακα, τυπώνονται οι τελευταίες είκοσι τροποποιήσεις όλων των συσκευών που ανήκουν στον έξυπνο τόπο. Οι πληροφορίες που είναι διαθέσιμες είναι το μοναδικό αναγνωριστικό της συσκευής, η παλιά και καινούρια κατάσταση και τέλος η χρονοσφραγίδα κατά την οποία πραγματοποιήθηκε το συμβάν.

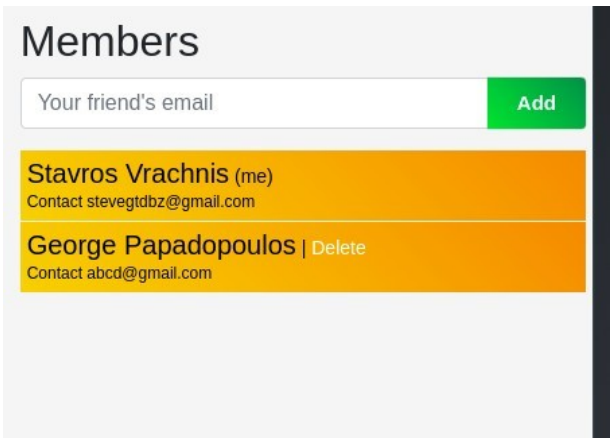


Σχήμα 27 - Εγγραφές ιχνηλασιμότητας

Τέλος, μέσω της παρακάτω φόρμας, ο χρήστης μπορεί να προσθέσει άμεσα διαχειριστές του τόπου, εισάγοντας την ηλεκτρονική διεύθυνση άλλων μελών. Τα μέλη μπορούν να διαχειριστούν όλα τα συστατικά μέρη και τις υπηρεσίες του τόπου εκτός από την διαγραφή άλλων μελών. Το τελευταίο δικαίωμα το έχει μόνο ο ιδρυτής του τόπου.



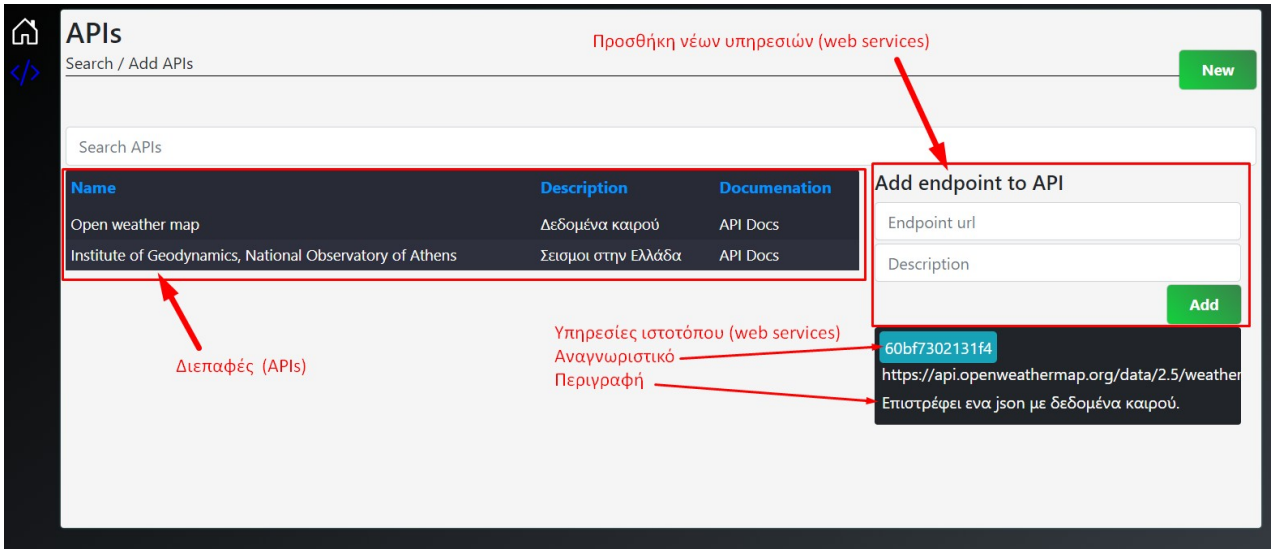
Σχήμα 28 - Περιοχή μελών έξυπνου τόπου 1



Σχήμα 29 - Περιοχή μελών έξυπνου τόπου 2

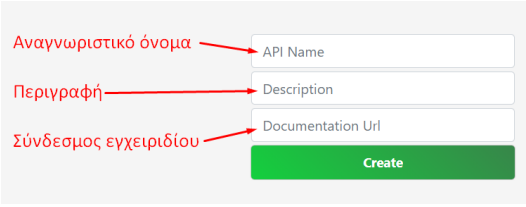
6.4 Διαχείριση διεπαφών APIs

Ο χρήστης μπορεί να μεταβεί στην περιοχή διαχείρισης διεπαφών για αναζήτηση ή εισαγωγή υπηρεσιών ιστού που τον ενδιαφέρουν. Οι εγγραφές αυτές είναι διαθέσιμες από όλους τους χρήστες της εφαρμογής. Η περιοχή αυτή λειτουργεί ως ευρετήριο πηγών δεδομένων του διαδικτύου.



Σχήμα 30 - Περιοχή διεπαφών

Για την δημιουργία μιας νέας διεπαφής, θα πρέπει να πατηθεί το αντίστοιχο κουμπί “New”. Στην παρακάτω εικόνα φαίνεται η σχετική φόρμα. Το αναγνωριστικό όνομα, η περιγραφή και ο σύνδεσμος τεκμηρίωσης είναι υποχρεωτικά, προκειμένου να επιτραπεί η καταχώριση. Επίσης, ο χρήστης μπορεί να προσθέσει υπηρεσίες ιστού σε μια διεπαφή από την φόρμα “Add endpoint to API”, εισάγοντας το URL και μια σύντομη περιγραφή.



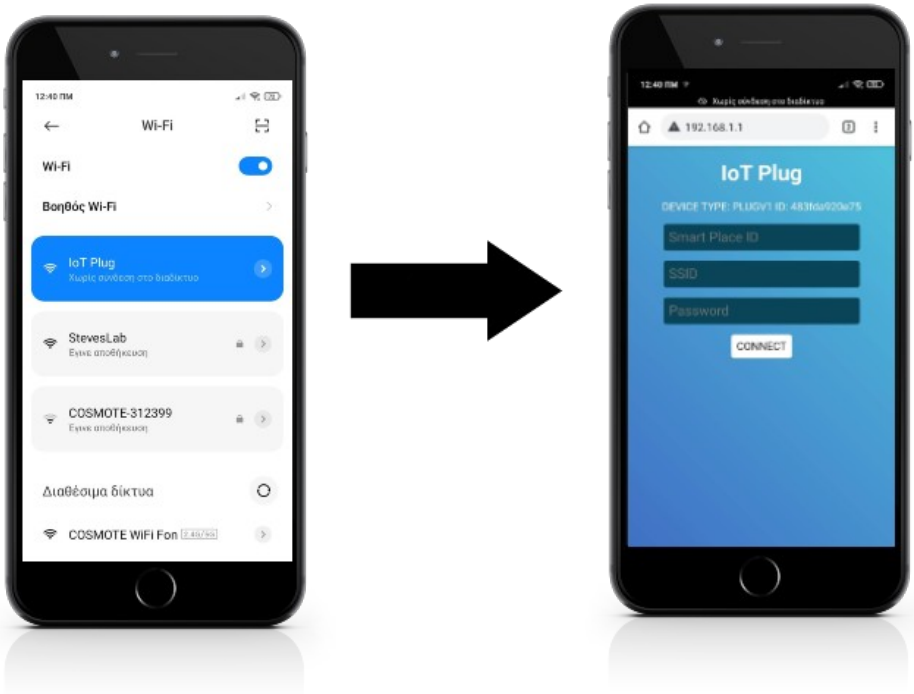
Σχήμα 31 - Εισαγωγή διεπαφής

6.5 Διαχείριση Συσκευών

Σε αυτή την ενότητα παρουσιάζεται η διαδικασία εγκατάστασης, επαναφοράς και ενημέρωσης των συσκευών. Κάθε συσκευή υποχρεωτικά θα πρέπει να ανήκει σε έναν έξυπνο τόπο, προκειμένου να είναι εφικτή η εκμετάλλευση και διαχείριση της από την εφαρμογή. Οι παρακάτω ενότητες προϋποθέτουν την ύπαρξη έξυπνων τόπων.

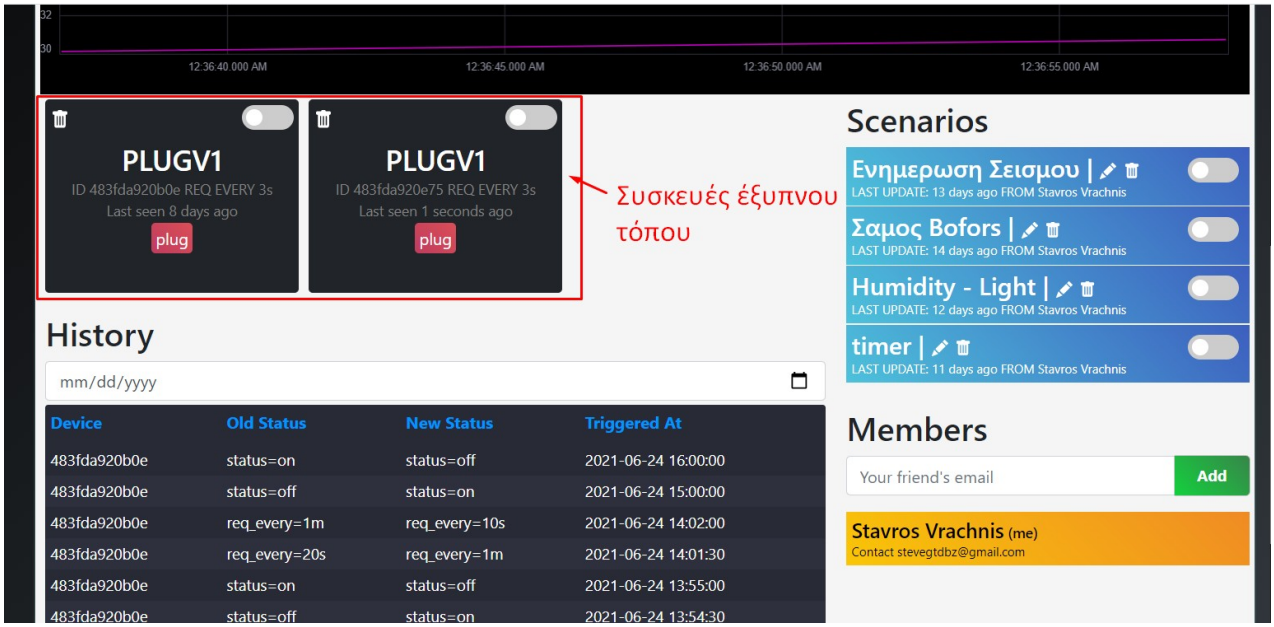
6.5.2 Ροή Εγκατάστασης Συσκευής

Όλες οι συσκευές ακολουθούν την ίδια ροή εγκατάστασης. Αρχικά θα πρέπει να τροφοδοτήσουμε τα αντικείμενα με ρεύμα. Για τις έξυπνες πρίζες αρκεί η σύνδεση με την πρίζα ενώ για τους αισθητήρες θα πρέπει να τοποθετηθεί η ανάλογη μπαταρία. Στην συνέχεια, επιλέγουμε οποιαδήποτε συσκευή που μπορεί να συνδεθεί σε οικιακό δίκτυο (WiFi) και παρέχει κάποιον φυλλομετρητή (Browser). Κάνουμε σάρωση για τοπικά δίκτυα, και εντοπίζουμε δίκτυα της μορφής “IoT <συσκευή>”, όπου συσκευή το είδος του διαθέσιμου αντικειμένου που θέλουμε να εγκαταστήσουμε. Αφού συνδεθούμε στο δίκτυο, ανοίγοντας τον φυλλομετρητή γίνεται ανακατεύθυνση στην σχετική φόρμα εγκατάστασης. Στην συνέχεια ζητούνται το μοναδικό αναγνωριστικό του έξυπνου τόπου που θα διαχειρίζεται την συσκευή, το όνομα και ο κωδικός του οικιακού μας δικτύου, προκειμένου να πραγματοποιηθεί σύνδεση με το διαδίκτυο. Μετά την επιτυχή εισαγωγή των παραπάνω στοιχείων δεν απαιτείται καμία επιπλέον ενέργεια.



Σχήμα 32 - Φόρμα εγκατάστασης συσκευής IoT

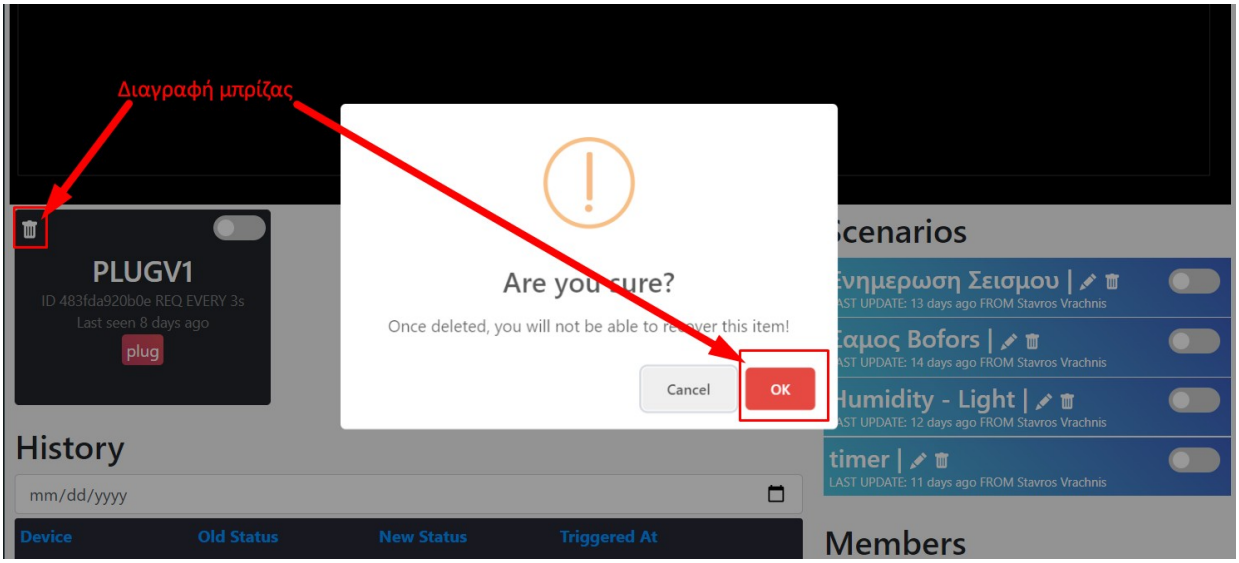
Αν τα στοιχεία που δόθηκαν είναι σωστά, πραγματοποιείται σύνδεση με την εφαρμογή και πλέον η συσκευή είναι διαθέσιμη επιλέγοντας τον έξυπνο τόπο όπου ανήκει.



Σχήμα 33 - Περιοχή συσκευών IoT

6.5.3 Επαναφορά Συσκευής

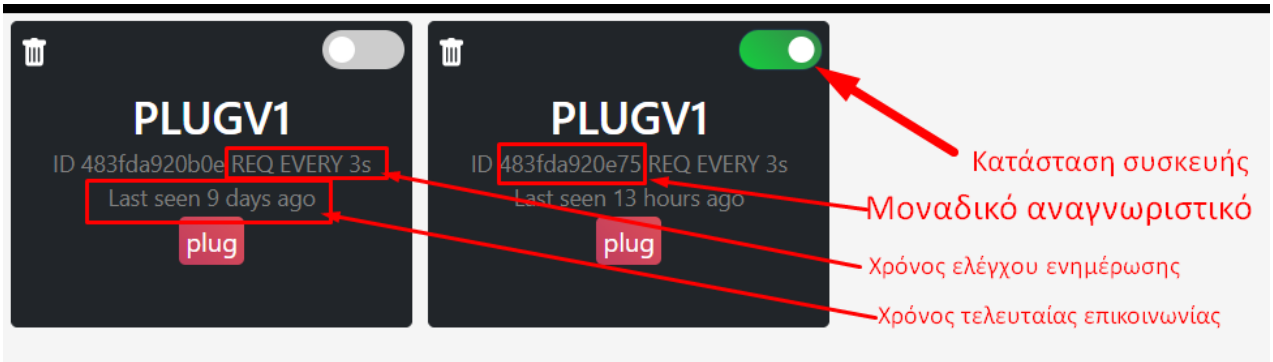
Κάθε συσκευή μπορεί να ανήκει μόνο σε έναν έξυπνο τόπο την φορά. Για την διαγραφή μιας συσκευής, ο χρήστης δεν έχει πάρα μόνο να επαναφέρει την συσκευή στην εργοστασιακή της κατάσταση. Μπορεί να επιτύχει το παραπάνω με δυο τρόπους, είτε πιέζοντας το σχετικό εικονίδιο διαγραφής στην εφαρμογή, είτε πιέζοντας το κουμπί (pushbutton) που βρίσκεται στο κάτω μέρος της κατασκευής για μερικά δευτερόλεπτα. Αν η επαναφορά είναι επιτυχής, το πλαίσιο συσκευής δεν θα είναι πλέον ορατό στην περιοχή αντικειμένων.



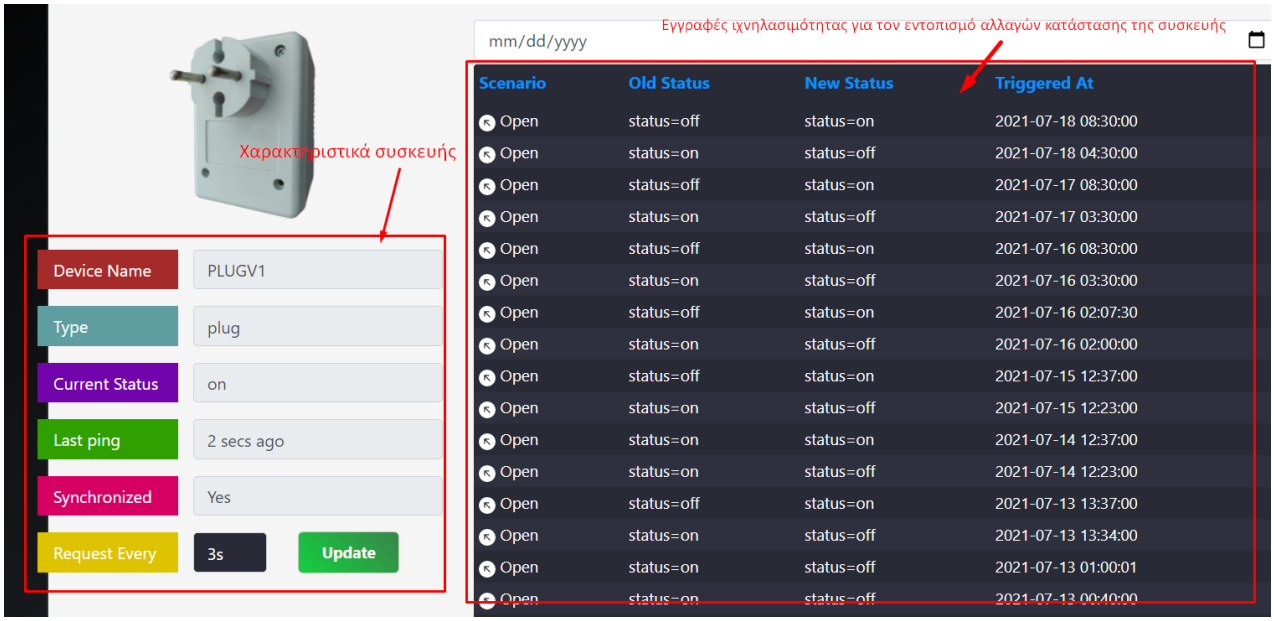
Σχήμα 34 - Διαγραφή συσκευής IoT

6.5.4 Ενημέρωση Συσκευής

Κάθε συσκευή επικοινωνεί με την εφαρμογή ανά τακτά χρονικά διαστήματα, προκειμένου να συγχρονίζεται με νέα δεδομένα. Ο χρήστης μπορεί άμεσα να τροποποιήσει αυτά τα χαρακτηριστικά, από τις παρακάτω οθόνες.



Σχήμα 35 - Διαχείριση συσκευών IoT

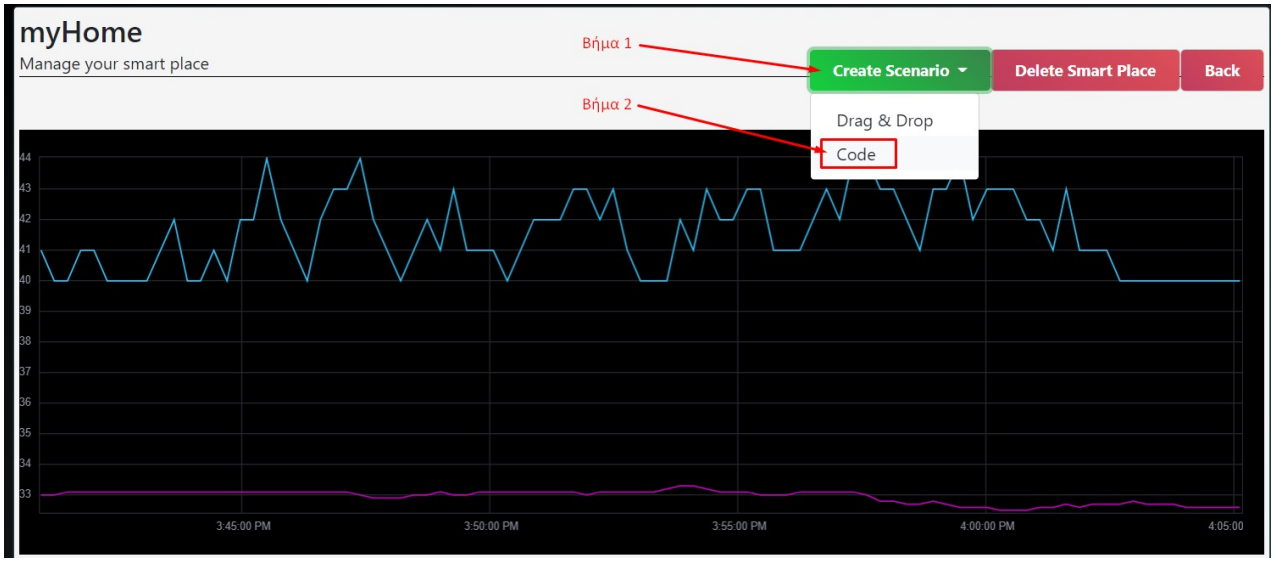


Σχήμα 36 - Χαρακτηριστικά συσκευών IoT

Στην τελευταία οθόνη, φαίνονται αναλυτικά όλα τα διαθέσιμα χαρακτηριστικά των συσκευών. Το όνομα και το είδος συσκευής, εισάγονται αυτόματα από το σύστημα και δεν μπορούν να τροποποιηθούν. Το πεδίο κατάστασης δηλώνει πότε το αντικείμενο είναι ενεργό ή όχι, παίρνοντας τιμές “on” ή “off” αντίστοιχα. Κάθε φορά που πραγματοποιείτε επιτυχή επικοινωνία με την εφαρμογή ανανεώνεται κατάλληλα η χρονοσφραγίδα “last ping”, γνωστοποιώντας οτι η συσκευή λειτουργεί ομαλά. Τέλος, το πεδίο “synchronized”, δηλώνει οτι το αντικείμενο λειτουργεί σύμφωνα με τις τελευταίες ρυθμίσεις. Αξίζει να αναφερθεί οτι για κάθε αλλαγή των ιδιοτήτων που πραγματοποιείται μέσω σεναρίων, το σύστημα αποθηκεύει σχετικές εγγραφές ιχνηλασιμότητας (traceabillity) που φαίνονται στο δεξιό μέρος της οθόνης.

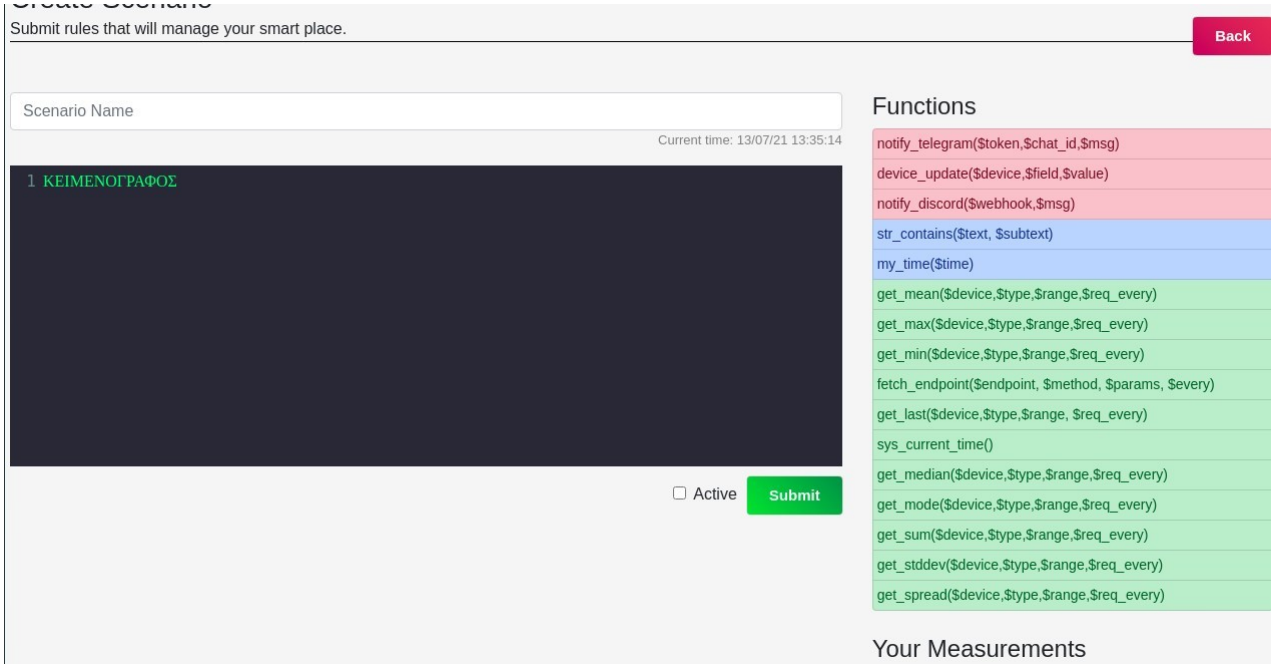
6.6 Δημιουργία Σεναρίων

Για την δημιουργία σεναρίων, ο χρήστης αφού έχει επιλέξει τον έξυπνο τόπο που τον ενδιαφέρει θα πρέπει να πατήσει το κουμπί “Create Scenario” και στην συνέχεια την επιλογή “Code” όπως φαίνεται στην παρακάτω εικόνα.



Σχήμα 37 - Εισαγωγή σεναρίου με ψευδοκώδικα

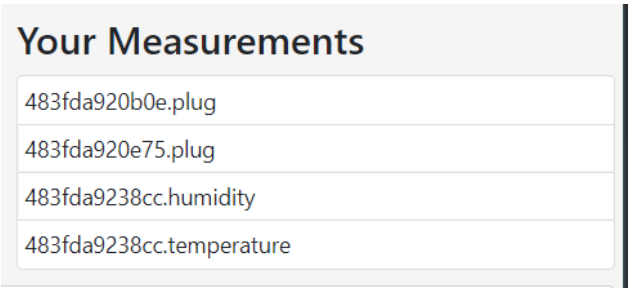
Έπειτα το σύστημα τον κατευθύνει στην σελίδα σύνταξης και υποβολής σεναρίων με χρήση ψευδοκώδικα. Όπως έχει αναφερθεί και σε προηγούμενη ενότητα η δημιουργία των σεναρίων αρχικά είχε σχεδιαστεί να πραγματοποιείται και μέσω της τεχνικής “Drag & Drop”, λόγω όμως χρονικών περιορισμών και απόκλισης από τα πλαίσια της εργασίας, δεν κρίθηκε αναγκαίο.



Σχήμα 38 - Κειμενογράφος ψευδοκώδικα

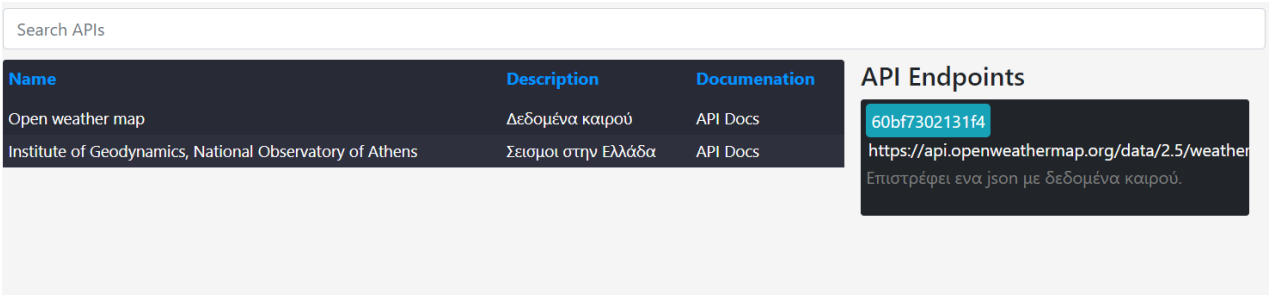
Για την επιτυχή υποβολή του σεναρίου στο σύστημα απαιτείται ένας τίτλος και κώδικας, συντακτικά ορθός, που περιλαμβάνει τουλάχιστον μια είσοδο και μια έξοδο. Σε περίπτωση μη

αποδεκτού κώδικα, εμφανίζεται κατάλληλο μήνυμα κατά την υποβολή. Στα δεξιά της παραπάνω εικόνας εμφανίζονται οι διαθέσιμες συναρτήσεις συστήματος, οι οποίες αναλύθηκαν στο προηγούμενο κεφάλαιο. Το παρακάτω παράθυρο, υπενθυμίζει στον χρήστη τις διαθέσιμες πηγές μετρήσεων. Κάθε γραμμή αποτελείται από το μοναδικό αναγνωριστικό και το είδος της συσκευής. Αξίζει να σημειωθεί οτι υπάρχουν συσκευές που παράγουν πάρα πάνω από μια μέτρηση, όπως ο παρακάτω αισθητήρας με κωδικό 483fda9238cc.



Σχήμα 39 - Περιοχή διαθέσιμων συσκευών IoT

Τέλος, στην παρακάτω εικόνα φαίνεται το παράθυρο διεπαφών για την εύκολη εισαγωγή υπηρεσιών ιστού στον ψευδοκώδικα. Ο χρήστης χρησιμοποιεί το μοναδικό αναγνωριστικό του endpoint ως πρώτο όρισμα στην συνάρτηση fetch_endpoint($endpointId, $method, $params, $every). Στο όρισμα $method εισάγει τον τύπο, ενώ στο όρισμα $params τις παραμέτρους της αίτησης, οι οποίες συνήθως δίνονται από το εγχειρίδιο χρήσης της διεπαφής. Στο τελευταίο όρισμα εισάγεται ο επιθυμητός περιοδικός χρόνος για την ανάκτηση δεδομένων από την υπηρεσία.



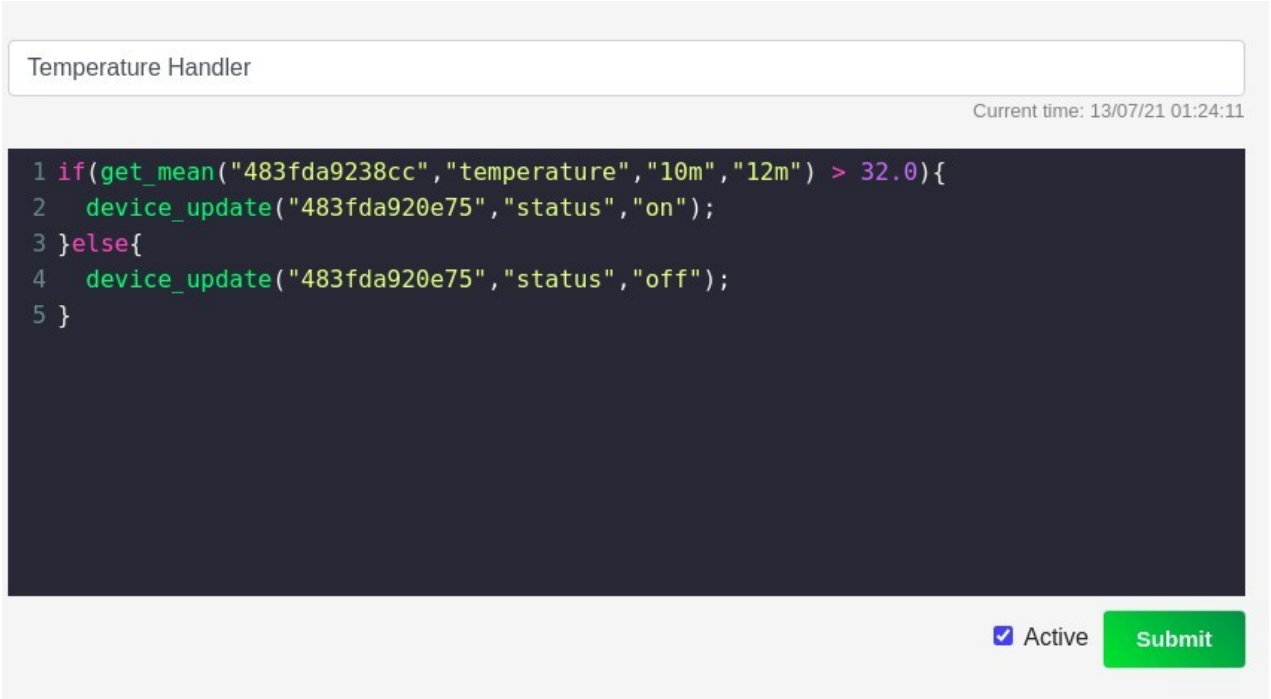
Σχήμα 40 - Περιοχή αναζήτησης διεπαφών

6.7 Επίδειξη Σεναρίων

Σε αυτή την ενότητα, παρουσιάζονται σενάρια IoT που μπορεί να υποστηρίξει η εφαρμογή. Για κάθε σενάριο δίνεται και εξηγείται αναλυτικά ο ψευδοκώδικας.

6.7.1 Χρήση αισθητήρα θερμοκρασίας για διαχείριση συσκευών ψύξης

Σε αυτό το σενάριο θα εκμεταλλευτούμε τα δεδομένα του αισθητήρα θερμοκρασίας για την διαχείριση έξυπνης πρίζας. Όταν ο μέσος όρος θερμοκρασίας ξεπεράσει το επιθυμητό όριο (32 °C), θα ενεργοποιείται μια οικιακή συσκευή ψύξης, όπως για παράδειγμα ένας ανεμιστήρας. Παρακάτω φαίνεται ο ψευδοκώδικας του σεναρίου.



Σχήμα 41 - Σενάριο διαχείρισης συσκευής αξιοποιώντας μετρικές θερμοκρασίας

Αρχικά γίνεται χρήση της στατιστικής συνάρτησης get_mean() ανά 12 λεπτά, για την ανάκτηση του μέσου όρου θερμοκρασίας των τελευταίων δέκα λεπτών. Στην περίπτωση που ο μέσος όρος είναι μεγαλύτερος των 32 βαθμών Κελσίου η πρίζα ενεργοποιείται, διαφορετικά απενεργοποιείται.

6.7.2 Χρονοπρογραμματισμός συσκευής

Σε αυτό το σενάριο, θα προγραμματίσουμε την πρίζα να ενεργεί βάση του χρόνου. Έστω οτι είναι αναγκαίο να ενεργοποιούμε την πρίζα καθημερινά στο διάστημα 19:00 – 24:00 για την ενεργοποίηση εξωτερικού φωτισμού σε ένα κτήριο. Παρακάτω φαίνεται ο σχετικός ψευδοκώδικας.

Σχήμα 42 - Σενάριο χρονοπρογραμματισμού

Χρησιμοποιώντας την συνάρτηση sys_current_time() παίρνουμε την τρέχουσα ώρα ανά λεπτό, την οποία και συγκρίνουμε με τις επιθυμητές ώρες. Στην περίπτωση που η τρέχουσα ώρα είναι εντός των ορίων που θέσαμε ενημερώνουμε την κατάσταση της συσκευής σε “on”, δηλαδή ενεργοποιημένη. Διαφορετικά η συσκευή παραμένει απενεργοποιημένη.

6.7.3 Χρήση διεπαφής ακραίων καιρικών φαινομένων.

Σε αυτό το σενάριο, θα χρησιμοποιηθεί η υπηρεσία ιστού (web service) Open Weather Map για την επεξεργασία μετρήσεων του αέρα σε περιοχή που μας ενδιαφέρει. Στην περίπτωση που ικανοποιεί την συνθήκη μας (ταχύτητα ανέμου μεγαλύτερη ή ίση των 4 μποφόρ), ως δράση ορίζεται η ενημέρωση μας μέσω μηνύματος Discord και το κλείσιμο παραθύρων που είναι συνδεδεμένα με την έξυπνη πρίζα.

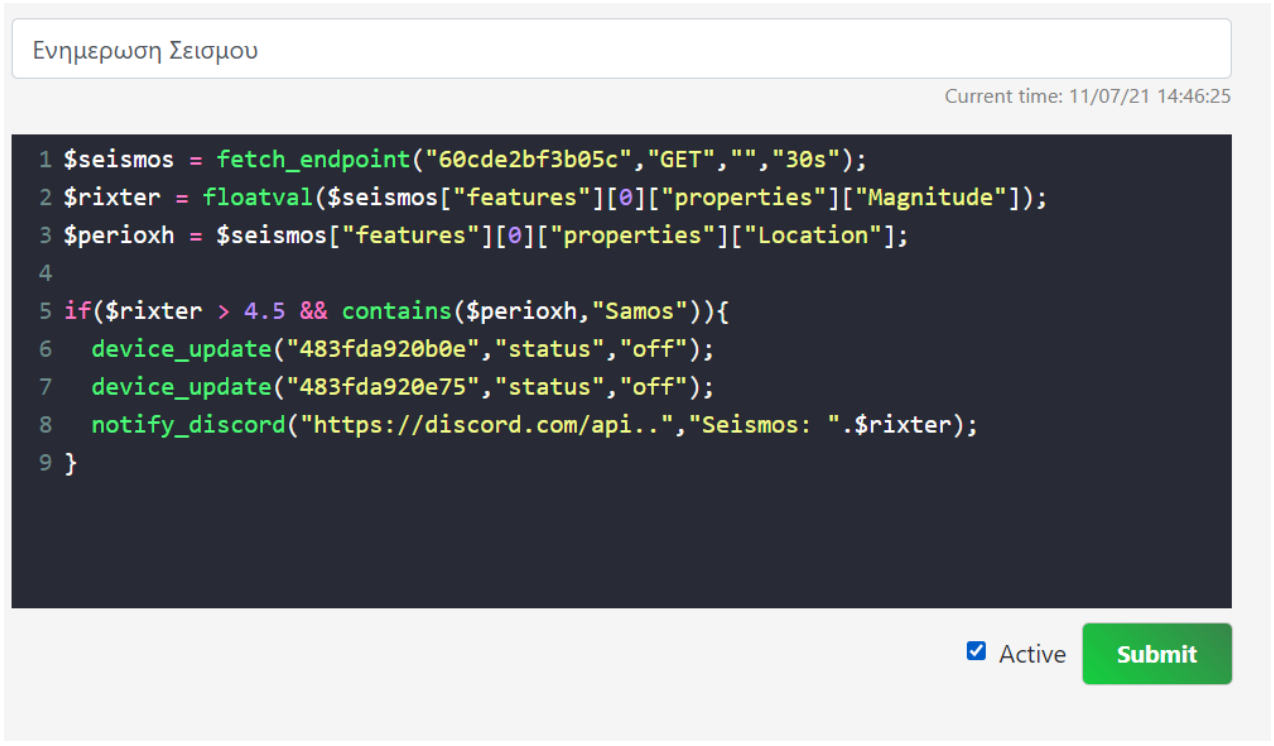
Σχήμα 43 - Σενάριο με χρήση διεπαφής ανέμου για διαχείριση συσκευών IoT & λήψη ειδοποιήσεων μέσω Discord

Αρχικά γίνεται ανάκτηση καιρικών δεδομένων της Σάμου από την υπηρεσία ιστού της Open Weather Map και τα οποία αποθηκεύονται στην μεταβλητή $wind. Στην συνέχεια, εξάγουμε την πληροφορία που μας ενδιαφέρει στην προσωρινή μεταβλητή $speed. Στην περίπτωση μας παίρνουμε την ταχύτητα του αέρα σε μποφόρ. Έπειτα, πραγματοποιείται ο εξής έλεγχος: αν η ταχύτητα του ανέμου είναι ίση ή ξεπερνά τα τέσσερα μποφόρ, η πρίζα κλείνει (τα παράθυρα κατεβαίνουν) και γίνεται αποστολή μηνύματος στο προκαθορισμένο δωμάτιο συνομιλίας στην πλατφόρμα Discord. Μεγάλο ενδιαφέρον παρουσιάζει το μήνυμα που αποστέλλεται, καθώς δεν είναι ένα στατικό κείμενο, αλλά περιλαμβάνει και την μεταβλητή $speed, όπου κάθε φορά αποδίδει την τιμή που εμπεριέχει. Σε κάθε άλλη περίπτωση η πρίζα ενεργοποιείτε κρατώντας τα παράθυρα ανοιχτά. Παρακάτω βλέπουμε το σχετικό μήνυμα που στάλθηκε στην προκαθορισμένη πλατφόρμα κατά την διάρκεια εκτέλεσης του σεναρίου αυτού.

Σχήμα 44 - Ειδοποίηση Discord

6.7.4 Χρήση διεπαφής σεισμού και απενεργοποίηση συσκευών

Σε αυτό το σενάριο θα χρησιμοποιήσουμε μια ανεπίσημη υπηρεσία ιστού του εθνικού αστεροσκοπείου Αθηνών για την ανάκτηση πληροφοριών σεισμικών δονήσεων. Στην περίπτωση ανίχνευσης σεισμικής δόνησης μεγαλύτερης της προκαθορισμένης, οι επιθυμητές συσκευές θα απενεργοποιούνται και θα στέλνεται σχετική ειδοποίηση μέσω Discord.



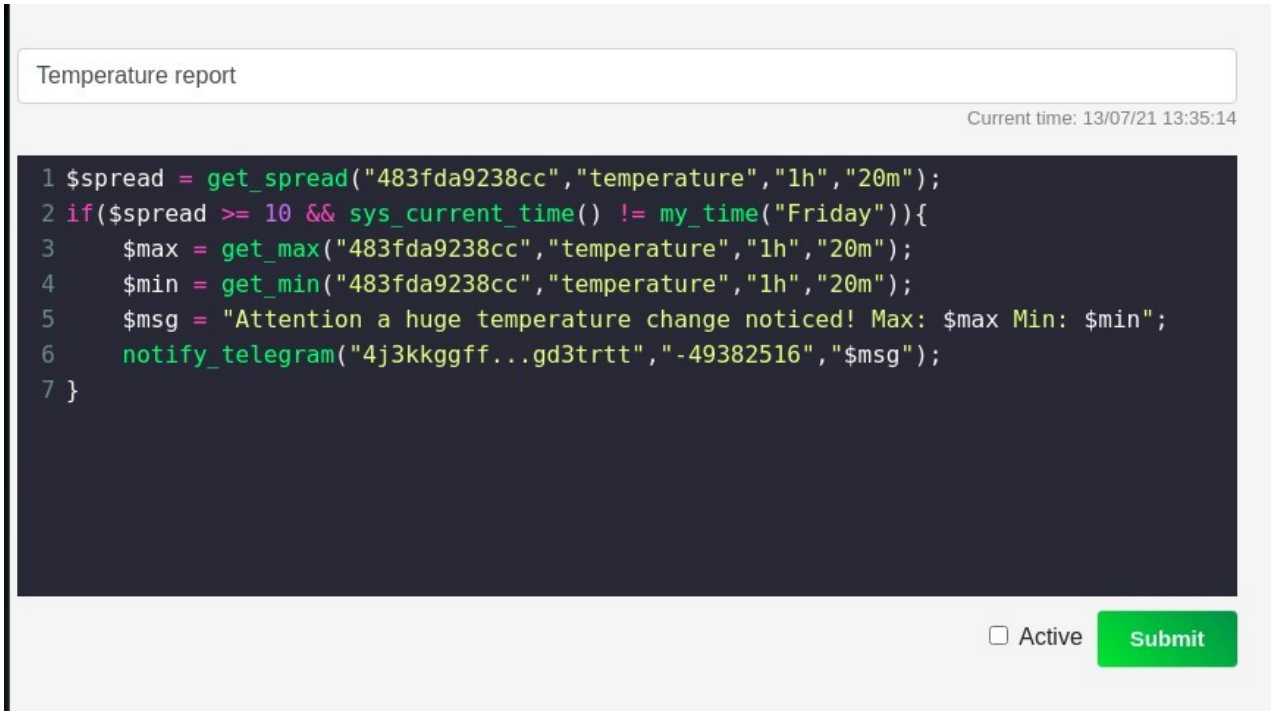
Σχήμα 45 - Σενάριο με χρήση διεπαφής σεισμού για την απενεργοποίηση συσκευών IoT & λήψη ειδοποιήσεων μέσω Discord

Αρχικά, γίνεται ανάκτηση πληροφοριών σεισμικών δονήσεων ανά 30 δευτερόλεπτα. Η απάντηση της υπηρεσίας αποθηκεύεται στην μεταβλητή $seismos. Έπειτα εξάγουμε τις πληροφορίες που μας ενδιαφέρουν σε δυο προσωρινές μεταβλητές για να μπορούμε να τις διαχειριστούμε καλύτερα στην συνέχεια. Στην μεταβλητή $rixter αποθηκεύουμε του βαθμούς ενώ στην $perioxh ένα αλφαριθμητικό που περιέχει την τοποθεσία που σημειώθηκε η δόνηση. Παρακάτω ακολουθεί

συνθήκη διπλής σύγκρισης. Στην περίπτωση που η δόνηση είναι εντός Σάμου και ξεπερνά τα 4.5 Ρίχτερ, όλες οι συσκευές απενεργοποιούνται και στέλνεται ανάλογη ειδοποίηση.

6.7.5 Επίδειξη σύνθετου σεναρίου

Σκοπός αυτής της ενότητας είναι η παρουσίαση ενός πιο σύνθετου σεναρίου. Στο σενάριο αυτό χρησιμοποιούνται στατιστικές συναρτήσεις με σκοπό την διεξαγωγή συμπερασμάτων.



Σχήμα 46 - Σενάριο χρήσης στατιστικών συναρτήσεων

Αρχικά ανακτούμε το εύρος τιμών θερμοκρασίας στο χρονικό εύρος της μιας ώρας ανά 20 λεπτά. Σε περίπτωση που η διαφορά της μέγιστης με την ελάχιστη θερμοκρασία είναι ίση ή μεγαλύτερη των 10 βαθμών Κελσίους και δεν είναι ημέρα Παρασκευή εκτελούνται οι εντολές εντός της συνθήκης. Εξάγεται η μέγιστη και ελάχιστη τιμή θερμοκρασίας από το ίδιο σύνολο τιμών και στέλνεται αναφορά στο προκαθορισμένο παράθυρο διαλόγου της πλατφόρμας Telegram.

7 Συμπεράσματα & Μελλοντική Εργασία

7.1 Συμπεράσματα

Σε αυτή την εργασία παρουσιάστηκαν κρίσιμα ζητήματα των εφαρμογών IoT της σημερινής εποχής. Με αφορμή αυτά τα ζητήματα πραγματοποιήθηκε σχεδίαση και υλοποίηση ενός ευέλικτου, ασφαλούς συστήματος διαχείρισης συσκευών Internet of Things με τη χρήση σεναρίων (Scenarios). Για την εγγραφή κανόνων διαχείρισης συσκευών, το σύστημα επιτρέπει την χρήση πηγών δεδομένων από το διαδίκτυο, επιτρέποντας υλοποίηση έργων IoT μικρότερου κόστους χωρίς κανένα περιορισμό. Συνεπώς, ο χρήστης μπορεί να έχει πρόσβαση σε ακόμα περισσότερα είδη δεδομένων. Επιπρόσθετα, παρέχεται η δυνατότητα επεξεργασίας των δεδομένων από ένα μεγάλο πλήθος στατιστικών συναρτήσεων. Τέλος, έγινε αναλυτική παρουσίαση της κατασκευής έξυπνων αντικειμένων.

7.2 Μελλοντική εργασία

Στα πλαίσια της εργασίας αναπτύχθηκε ένα πλήρες λειτουργικό πληροφοριακό σύστημα διαχείρισης συσκευών IoT. Το σύστημα αυτό μπορεί να αποτελέσει μια ρεαλιστική λύση στην βιομηχανία του IoT εφόσον ακολουθηθούν απαραίτητες τροποποιήσεις και επεκτάσεις.

Αρχίζοντας από τα χαμηλότερα στρώματα υλοποίησης, δηλαδή την κατασκευή και διάθεση των συσκευών, παρουσιάζονται οι ανάλογες προσθήκες και βελτιστοποιήσεις. Χρησιμοποιώντας το ίδιο πρότυπο κατασκευής μπορούν να αναπτυχθούν πλήθος νέων συσκευών, όπως αισθητήρες κίνησης, καπνού, φωτός, αέρα, διαρροής υγρών - αερίων και άλλων. Σημαντική είναι επίσης η προσθήκη νέων συσκευών δράσεων, όπως λάμπες και ρελέ γενικού πίνακα (relays). Με τους διακόπτες γενικού πίνακα μπορεί να πραγματοποιηθεί έλεγχος συσκευών μεγαλύτερης ισχύος όπως του θερμοσίφωνα, ψυγείων, κουζίνας και άλλων. Επίσης είναι απαραίτητη η

πραγματοποίηση έρευνας για την κατανάλωση ηλεκτρικού ρεύματος των αυτόνομων συσκευών. Έπειτα, θα πρέπει να ακολουθήσει εντοπισμός εξαρτημάτων που μπορούν να αντικατασταθούν με υλικά χαμηλότερης κατανάλωσης διατηρώντας την σχέση κόστους – απόδοσης.

Στο επίπεδο του λογισμικού κύρια βελτιστοποίηση αποτελεί η ευκολία χρήσης του συστήματος. Για την δημιουργία σεναρίων θα πρέπει να υπάρχει κατάλληλη γραφική διεπαφή. Ιδανικά η διεπαφή αυτή θα μπορούσε να μοιάζει με την εικονική γλώσσα προγραμματισμού Scratch, όπου ο χρήστης δημιουργεί τμήματα κώδικα με την τεχνική Drag & Drop. Φυσικά, είναι λογικό η παραπάνω τεχνική να μην μπορεί να καλύψει όλες τις δυνατότητες που παρέχονται από το σύστημα. Για την πλήρη εκμετάλλευση των υπηρεσιών της εφαρμογής, θα είναι παράλληλα δυνατή η χρήση του ήδη υπάρχων κειμενογράφου.

Θα μπορούσαμε, επίσης στο επίπεδο του λογισμικού να εμπλουτίσουμε τα δεδομένα ιχνηλάτησης ακόμα περισσότερο με πληροφορίες όπως τον τίτλο του σεναρίου, τις τιμές και τον κώδικα που προκάλεσαν τις αλλαγές. Επίσης, θα ήταν χρήσιμο να δημιουργούνται εγγραφές ιχνηλασιμότητας, όχι μόνο από αλλαγές που πραγματοποιούνται μέσω των σεναρίων αλλά και από αλλαγές που λαμβάνουν χώρα μέσω της γραφικής διεπαφής.

Είναι επίσης απαραίτητο να παρέχεται μια εξατομικευμένη υλοποίηση εφαρμογής κινητού, για την εύκολη διαχείριση συσκευών από οπουδήποτε. Δεν είναι τυχαίο άλλωστε που οι εταιρίες που αναφέρθηκαν στην αρχή της εργασίας, προσφέρουν μόνο εφαρμογές κινητού. Οι τελευταίες, συγκριτικά με τις εφαρμογές ιστού, προσφέρουν καλύτερη εμπειρία χρήστη, ασφάλεια, και πρόσβαση σε ευαίσθητους πόρους της συσκευής. Η προτεινόμενη εφαρμογή θα πρέπει να παρέχει τις ίδιες δυνατότητες με αυτές που αναφέρθηκαν παραπάνω.

Συνοψίζοντας, είναι απαραίτητη η πραγματοποίηση τροποποιήσεων τόσο στο κομμάτι του υλικού όσο και στου λογισμικού. Το υλικό σίγουρα μπορεί να λειτουργήσει ακόμα πιο αποδοτικά σε ακόμη μικρότερα μεγέθη, ακολουθώντας μια πιο μινιμαλιστική σχεδίαση. Τέλος, υπάρχει ανάγκη για προσθήκη νέων λειτουργιών λογισμικού, προκειμένου να εξασφαλιστεί ευκολία χρήσης.

Βιβλιογραφία

[1] Internet of things – Wikipedia, 2021, Accessed on: 27/07/2021, Available: [Online] https://en.wikipedia.org/wiki/Internet_of_things

[2] D. P. Acharjya and M. K. Geetha, Eds., *Internet of Things: Novel Advances and Envisioned Applications*, vol. 25. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-53472-5.

[3] Knud Lasse Lueth, Why it is called Internet of Things: Definition, history, disambiguation, 2014, Accessed on: 27/07/2021, Available: [Online] https://iot-analytics.com/internet-of-things-definition/

[4] "Internet of Things (IoT) Market Size, Growth| Industry Analysis 2021 to 2026 – Mordor Intelligence", Accessed on: 27/07/2021, Available: [Online] https://www.mordorintelligence.com/industry-reports/internet-of-things-moving-towards-a-smarter-tomorrow-market-industry

[5] "David Davies, The internet of things: The start of a slow revolution | Industry Trends | IBC, 2012", Accessed on: 27/07/2021, Available: [Online] https://www.ibc.org/trends/the-internet-of-things-the-start-of-a-slow-revolution/5145.article

[6] M. A. Imran, A. Zoha, L. Zhang, and Q. H. Abbasi, 'Grand Challenges in IoT and Sensor Networks', *Front. Commun.* Netw., vol. 1, p. 619452, Dec. 2020, doi: 10.3389/frcmn.2020.619452.

[7] D. Bandyopadhyay and J. Sen, 'Internet of Things: Applications and Challenges in Technology and Standardization', *Wirel. Pers. Commun.*, vol. 58, no. 1, pp. 49–69, May 2011, doi: 10.1007/s11277-011-0288-5.

[8] IKEA, IKEA invests heavily in the smart home going forward, 2019, Accessed on: 27/07/2021, Available: [Online] https://about.ikea.com/en/newsroom/2019/08/16/ikea-invests-heavily-in-the-smart-home-going-forward

[9] IKEA, The IKEA TRDFRI app gets a new name for the future of IKEA Home smart, 2019, Accessed on: 27/07/2021, Available: [Online] https://about.ikea.com/en/newsroom/2019/06/25/the-ikea-trdfri-app-gets-a-new-name-for-the-future-of-ikea-home-smart

[10] IKEA, About Us | Kasa Smart, Accessed on: 27/07/2021, Available: [Online] https://www.kasasmart.com/about

[11] “Top IoT security issues and challenges (2021) – Thales”, Accessed on: 27/07/2021, Available: [Online] https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/magazine/internet-threats

[12] “What Is a Relational Database | Oracle”, Accessed on: 27/07/2021, Available: [Online] https://www.oracle.com/database/what-is-a-relational-database

[13] Relational database – Wikipedia, 2021, Accessed on: 27/07/2021, Available: [Online] https://en.wikipedia.org/wiki/Relational_database

[14] MySQL – Wikipedia, 2021, Accessed on: 27/07/2021, Available: [Online] https://en.wikipedia.org/wiki/MySQL

[15] MariaDB – Wikipedia, 2021, Accessed on: 27/07/2021, Available: [Online] https://en.wikipedia.org/wiki/MariaDB

[16] PostgreSQL – Wikipedia, 2021, Accessed on: 27/07/2021, Available: [Online] https://en.wikipedia.org/wiki/PostgreSQL

[17] Soufiane L., MariaDB vs MySQL vs PostgreSQL | What are the differences?, 2021, Accessed on: 27/07/2021, Available: [Online] "https://stackshare.io/stackups/mariadb-vs-mysql-vs-postgresql"

[18] Kitty Gupta, MariaDB vs. MySQL vs. PostgreSQL In-Depth Comparison - FreelancingGig Blog - Freelancer Job Tips and Hiring Insights, Accessed on: 27/07/2021, Available: [Online] https://www.freelancinggig.com/blog/2017/07/22/mariadb-vs-mysql-vs-postgresql-depth-comparison/

[19] Helena Williams, The Complete Guide to Time Series Analysis and Forecasting | by Marco Peixeiro | Towards Data Science, 2019, Accessed on: 27/07/2021, Available: [Online] https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775

[20] Time Series – Wikipedia, 2021, Accessed on: 27/07/2021, Available: [Online] https://en.wikipedia.org/wiki/Time_series

[21] InfluxDB – Wikipedia, 2021, Accessed on: 27/07/2021, Available: [Online] https://en.wikipedia.org/wiki/InfluxDB

[22] InfluxData, InfluxDB: Purpose-Built Open Source Time Series Database | InfluxData, 2021, Accessed on: 27/07/2021, Available: https://www.influxdata.com/

[23] Prometheus (Software) – Wikipedia, 2021, Accessed on: 27/07/2021, Available: [Online] https://en.wikipedia.org/wiki/Prometheus_(software)

[24] “Prometheus - Monitoring system & time series database”, Accessed on: 27/07/2021, [Online] Available: https://prometheus.io/

[25] Mike Elsmore, Prometheus vs. InfluxDB: A Monitoring Comparison | Logz.io, 2020, Accessed on: 27/07/2021, Available: [Online] https://logz.io/blog/prometheus-influxdb/

[26] Igor Bobkriakov, Prometheus vs InfluxDB | MetricFire Blog, 2020, Accessed on: 27/07/2021, Available: [Online] "https://www.metricfire.com/blog/prometheus-vs-influxdb/"

[27] “PHP: eval – Manual”, Accessed on: 27/07/2021, Available: [Online] https://www.php.net/manual/en/function.eval.php

[28] Ben Bassi, 6 Common Website Security Vulnerabilities, Accessed on: 27/07/2021, Available: [Online] https://www.commonplaces.com/blog/6-common-website-security-vulnerabilities/

[29] Todd Henderson, Top 6 ESP8266 Modules for IoT Projects, 2016, Accessed on: 27/07/2021, [Online] Available: https://www.losant.com/blog/top-6-esp8266-modules

[30] ESP8266 – Wikipedia, 2021, Accessed on: 27/07/2021, Available: [Online] https://en.wikipedia.org/wiki/ESP8266

[31] Ida Hubschmann, ESP8266 for IoT: A Complete Guide, Accessed on: 27/07/2021, Available: [Online] https://www.nabto.com/esp8266-for-iot-complete-guide/

[32] "ESP8266 pinout reference and how to use GPIO pins", Accessed on: 27/07/2021, Available: [Online] https://microcontrollerslab.com/esp8266-pinout-reference-gpio-pins/

[33] "ESP8266 Temperature / Humidity Webserver with a DHT11 sensor | Elec-Cafe.Com", Accessed on: 27/07/2021, Available: [Online] https://www.elec-cafe.com/esp8266-temperature-humidity-webserver-with-a-dht11-sensor/

[34] Aswinth Raj, ESP8266 based DIY Smart Plug to Make Your Home Appliances IoT Enabled, Accessed on: 27/07/2021, Available: [Online] https://circuitdigest.com/microcontroller-projects/diy-smart-plug-using-esp8266

Παράρτημα Ι Οδηγός εγκατάστασης εφαρμογής

Σε αυτό το παράρτημα δίνονται οδηγίες ως προς τον διαχειριστή, για την εγκατάσταση της πλατφόρμας. Ο τρόπος εγκατάστασης ποικίλει ανάλογα το λειτουργικό σύστημα που διατίθεται κάθε φορά. Στο παρακάτω κείμενο, δίνονται γενικές οδηγίες και δεν γίνεται αναφορά σε εντολές ή παραμετροποιήσεις εγκατάστασης.

Αρχικά θα πρέπει να πραγματοποιηθεί σύνδεση ssh με λογαριασμό πλήρη δικαιωμάτων (root / admin) στον επιθυμητό διακομιστή. Μετά από την επιτυχή σύνδεση, θα πρέπει να εγκατασταθεί λογισμικό διακομιστή ιστού (web server), όπως Apache, nginx, litespeed ή άλλο. Στην συνέχεια μεταφερόμαστε στον φάκελο ιστού (πχ. για unix συστήματα /var/www/html), κατεβάζουμε όλα τα αρχεία της εφαρμογής που διατίθενται σε ένα αρχείο zip και τα εξάγουμε. Εγκαθιστούμε την σχεσιακή βάση δεδομένων MySQL και ακολουθούμε την ροή εγκατάστασης για την δημιουργία κεντρικού λογαριασμού χρήστη. Χρησιμοποιώντας οποιαδήποτε διεπαφή της βάσης (cli, phpmyadmin κτλ) εισάγουμε το αρχείο db.sql στην σχεσιακή βάση και στην συνέχεια το διαγράφουμε. Σε μια μια μη κατανεμημένη σχεδίαση εγκαθιστούμε την InfluxDB στον ίδιο εξυπηρετητή, διαφορετικά μπορούμε να την πραγματοποιήσουμε σε οποιονδήποτε άλλο κόμβο. Τέλος, ανοίγουμε το αρχείο παραμετροποιήσεων και κάνουμε τις απαραίτητες αλλαγές.

config.php

```php
<?php

$projectName = "IoT Solutions";
$projectPath = "http://185.243.214.236/api.php";
$version = "v21.02.17";
$debug = "true";

$influxdb_url = "http://185.243.214.236:8086";
$influxdb_token = "ELM04oiOmNaDApwUjYmMRMojqWkYugmPCoeCi88flAn-uYrW8qvxEbFdf5rH285D_Uho9IkN_ps4rGt-Pi6mnig==";
$organization = "iot";

$dbHost = "localhost";
$dbName = "iot";
$dbUsername = "iot";
$dbPassword = "iot123";
$timezone = "Europe/Athens";

$whitelisted_functions = array("intval", "floatval", "str_contains");
?>

```

Σχήμα 47 - Αρχείο παραμετροποιήσεων της εφαρμογής

Στην παραπάνω εικόνα φαίνεται ένα παράδειγμα παραμετροποίησης. Στην συνέχεια περιγράφονται αναλυτικά όλες οι δυνατές ρυθμίσεις.

- $projectName → Ο τίτλος της εφαρμογής
- $projectPath → Το URL διεπαφής (πχ https://myiot.gr/api.php)
- $version → Η τρέχουσα έκδοση της εφαρμογής
- $debug → Λογική μεταβλητή για ενεργοποίηση / απενεργοποίηση αποσφαλμάτωσης
- $influxdb_url → Σύνδεσμος της InfluxDB (πχ https://node1.myiot.gr/8086)
- $influxdb_token → Συμβολοσειρά ασφάλειας της InfluxDB
- $organization → Ο οργανισμός που έχει τεθεί στην InfluxDB
- $dbHost → Ο κόμβος που φιλοξενεί την MySQL
- $dbName → Το όνομα της βάσης MySQL
- $dbUsername → Το όνομα του χρήστη βάσης MySQL
- $dbPassword → Το συνθηματικό του χρήστη βάσης MySQL
- $timezone → Ζώνη ώρας
- $whitelisted_functions = array("intval", "floatval", "str_contains"); → Ενσωματωμένες συναρτήσεις php που επιτρέπονται στα σενάρια IoT.

Παράρτημα ΙΙ Κώδικας συσκευής αισθητήρα

Σε αυτό το παράρτημα δίνεται ο κώδικας της συσκευής αισθητήρα θερμοκρασίας - υγρασίας. Για την συγγραφή του προγράμματος χρησιμοποιήθηκε το ολοκληρωμένο περιβάλλον ανάπτυξης Arduino IDE.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPClient.h>
#include <EEPROM.h>
#include <DNSServer.h>
#include <DHT.h>
#include <Arduino_JSON.h>

#define DHTTYPE DHT11
#define DHT_DPIN 2

String DEF_AP_SSID = "IoT Sensor";
String DEF_AP_PSWD = "";
bool DEF_AP_RUNNING = true;
bool CONNECTED_TO_HOME_WIFI = false;
String API_RESPONSE = "";
String HOST_DASH = "http://185.243.214.236";
String HOST_INFLUXDB = "http://185.243.214.236";
String INFLUXDB_TOKEN = "lUlAfKlsS5Vy2bHJEGeigJZUjnXK24q1VOrksFxj4q_1Q==";

String DEV_ID = WiFi.macAddress();
String DEV_NAME = "SDHT11";
String DEV_TYPE = "humidity,temperature";
String REQ_EVERY = "10s";

String SMART_PLACE = "";
String HOME_WIFI_SSID = "";
String HOME_WIFI_PSWD = "";
```

```
float CURRENT_TEMPERATURE = 0;
float CURRENT_HUMIDITY = 0;



ESP8266WebServer server(80);
DHT dht(DHT_DPIN, DHTTYPE);

const byte DNS_PORT = 53;
DNSServer dnsServer;
IPAddress apIP(192, 168, 1, 1);
IPAddress netMsk(255, 255, 255, 0);

long device_sync_timer = 0;
void(* resetFunc) (void) = 0;  // declare reset fuction at address 0

void setup(){
  Serial.begin(9600);
  Serial.println();
  delay(500);

  DEV_ID.replace(":", "");
  DEV_ID.toLowerCase();

  // EEPROM initialize
  EEPROM.begin(512);
  loadSavedWiFiCredentials();

  // scan & connect to wifi
  if(HOME_WIFI_SSID.length() > 0 && HOME_WIFI_PSWD.length() > 0){
Serial.println("Saved Data Found: "+HOME_WIFI_SSID+" "+HOME_WIFI_PSWD+"
"+SMART_PLACE);
    int numberOfNetworks = WiFi.scanNetworks();
    for(int i=0; i<numberOfNetworks; i++){
        if(WiFi.SSID(i)==HOME_WIFI_SSID){
          if(connectToWiFi()){ // connected to WiFi
            Serial.println("Connected to saved WiFi");
            dht.begin();
            return;
          }else{ // Wrong credentials
            Serial.println("WiFi Access Point, changed credentials");
            eraseEEPROM();
            startApAndWebServer();
            return;
```

```
          }
        }
    }

    Serial.println("Saved WiFi not found within range");
    eraseEEPROM();
    startApAndWebServer();

  }else{ // first time
    Serial.println("First launch");
    eraseEEPROM();
    startApAndWebServer();
  }
}

void loop(){
  if(DEF_AP_RUNNING){
    server.handleClient();
    Serial.printf("Stations connected = %d\n", WiFi.softAPgetStationNum());
  }

  if(CONNECTED_TO_HOME_WIFI){

      CURRENT_HUMIDITY = dht.readHumidity();
      CURRENT_TEMPERATURE = dht.readTemperature();
      Serial.println("Temperature: "+String(CURRENT_TEMPERATURE));
      Serial.println("Humidity: "+String(CURRENT_HUMIDITY)+"%");

      String payload = "temperature,host="+SMART_PLACE+",device="+DEV_ID+"
_value="+String(CURRENT_TEMPERATURE)+"\
nhumidity,host="+SMART_PLACE+",device="+DEV_ID+"
_value="+String(CURRENT_HUMIDITY);

      pushMetricsInfluxDB(payload);
      Serial.println("Response: "+API_RESPONSE);

      long time_to_sleep = influxTimeToTime(REQ_EVERY);
      device_sync_timer += time_to_sleep;
      if(device_sync_timer > (60*1*1000)){ // 1m Check for device settings
updates in our dash
        device_sync_timer = 0;

        getDeviceInfo("device_id="+DEV_ID);
```

```
        JSONVar myObject = JSON.parse(API_RESPONSE);
          if (JSON.typeof(myObject) == "undefined") {
          Serial.println("Parsing input failed!");
          return;
        }


        if((int)(myObject["code"]) == 404){
          eraseEEPROM();
          resetFunc(); //call reset
        }
        REQ_EVERY = (const char*) myObject["data"]["req_every"];
      }

      delay(influxTimeToTime(REQ_EVERY)); // wait until next ping
  }


  delay(1000);
}

// HTTPS Requests ============================================
void addDeviceToSmartPlace(String payload){

  WiFiClient wifiClient;
  HTTPClient http;
  http.begin(wifiClient, HOST_DASH+"/api.php?add-device");
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
  int httpResponseCode = http.POST(payload);
  if(httpResponseCode){
    API_RESPONSE = http.getString();
  }else{
    API_RESPONSE="error";
  }

  http.end();
}


void getDeviceInfo(String payload){

  WiFiClient wifiClient;
  HTTPClient http;
```

```
  http.begin(wifiClient, HOST_DASH+"/api.php?get-device");
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
  int httpResponseCode = http.POST(payload);
  if(httpResponseCode){
    API_RESPONSE = http.getString();
  }else{
    API_RESPONSE="error";
  }

  http.end();
}

void pushMetricsInfluxDB(String data){

  WiFiClient wifiClient;
  HTTPClient http;
  http.begin(wifiClient, HOST_INFLUXDB+":8086/api/v2/write?
org=iot&bucket=metrics&precision=s");
  http.addHeader("Content-Type", "text/plain");
  http.addHeader("Authorization", "Token "+INFLUXDB_TOKEN);
  int httpResponseCode = http.POST(data);
  if(httpResponseCode){
    API_RESPONSE = http.getString();
  }else{
    API_RESPONSE="error";
  }

  http.end();
}


// WiFi Functions ================================================
bool connectToWiFi(){

  byte seconds = 0;
  WiFi.begin(HOME_WIFI_SSID, HOME_WIFI_PSWD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    seconds+=1;
    Serial.print(".");
    if(seconds==10) break;
  }
```

```
  if(seconds < 10){ // Correct WiFi credentials

    Serial.println("");
    Serial.println("WiFi connected..!");
    Serial.print("Got IP: ");  Serial.println(WiFi.localIP());
    CONNECTED_TO_HOME_WIFI = true;
    DEF_AP_RUNNING = false;
    return true;

  }else{// Wrong WiFi credentials
    CONNECTED_TO_HOME_WIFI = false;
    DEF_AP_RUNNING = false;
    return false;
  }
}

// EEPROM Functions ============================================
void eraseEEPROM(){
  for(int i=0; i < 512; i++) EEPROM.write(i, 0);
  EEPROM.commit();
}

void loadSavedWiFiCredentials(){
  short seperatorFound = 0;
  for(int i=0; i < 512; i++) {
    int inp = EEPROM.read(i);
    if(inp==0) return; // empty data
    if(char(inp)==':'){
      seperatorFound+=1;
      continue;
    }

    if(seperatorFound==0) HOME_WIFI_SSID+=char(inp);
    if(seperatorFound==1) HOME_WIFI_PSWD+=char(inp);
    if(seperatorFound==2) SMART_PLACE+=char(inp);

  }

}

// Access Point & Web Server Functions ==========================
void startApAndWebServer(){
  DEF_AP_RUNNING = true;
```

```
  Serial.println("Setting AP");
  WiFi.softAPConfig(apIP, apIP, netMsk);
  boolean result = WiFi.softAP(DEF_AP_SSID, DEF_AP_PSWD);
  delay(500);
  if(result == true){
    Serial.println("Default AP is ready");
  }else{
    Serial.println("Default AP is failed");
  }

  // Setup the DNS server redirecting all the domains to the apIP
  dnsServer.setErrorReplyCode(DNSReplyCode::NoError);
  dnsServer.start(DNS_PORT, "*", apIP);

  // Start Web Server
  Serial.print("Gateway IP: ");  Serial.println(WiFi.softAPIP());
  server.on("/", handle_OnConnect);
  server.on("/init", handle_Init);
  server.onNotFound(handle_NotFound);
  server.begin();
  Serial.println("HTTP server started");

}

void handle_OnConnect(){
 String html = "<!DOCTYPE html> <html>";
    html += "<head><meta name='viewport' content='width=device-width, initial-
scale=1.0, user-scalable=no'>";
    html += "<title>"+DEF_AP_SSID+"</title>";
    html += "<style>";
    html += "body{height:100vh;background: linear-gradient(45deg, rgb(61 95
191) 0%, rgb(78 194 220) 100%);text-align:center;color:white;font-
family:arial;}";
    html += "input[type='text'], input[type='password']{padding:8px;margin-
bottom:15px;border-radius:4px;border:0px;font-
size:20px;background:#0a4458;color:white;}";
    html += "input[type='submit']{padding:8px;border-
radius:4px;background:white;color:black;border:0px;font-size:16px;}";
    html += "</style>";
    html += "</head>";
    html += "<body>";
    html += " <h1>"+DEF_AP_SSID+"</h1>";
```

```
    html += " <p>DEVICE TYPE: "+DEV_NAME+" ID: "+DEV_ID+"</p>";
    html += " <center><form action='/init' method='get'>";
    html += "   <input type='text' name='smart_place' placeholder='Smart Place
ID' /><br>";
    html += "   <input type='text' name='ssid' placeholder='SSID' /><br>";
    html += "   <input type='password' name='pswd' placeholder='Password'
/><br>";
    html += "   <input type='submit' value='CONNECT' />";
    html += " </form></center>";
    html += "</body>";
    html += "</html>";

  server.send(200, "text/html", html);
}


void handle_Init(){
  for(int i = 0; i < server.args(); i++){
    if(server.argName(i)=="smart_place") SMART_PLACE = server.arg(i);
    if(server.argName(i)=="ssid") HOME_WIFI_SSID = server.arg(i);
    if(server.argName(i)=="pswd") HOME_WIFI_PSWD = server.arg(i);
  }

  // Disconnect AP
  WiFi.softAPdisconnect(true);
  DEF_AP_RUNNING = false;

  if(connectToWiFi()){

    //save wifi credentials
    String toWrite = HOME_WIFI_SSID+":"+HOME_WIFI_PSWD+":"+SMART_PLACE;
    for(int i=0; i<toWrite.length(); i++) EEPROM.write(i, toWrite [i]);
    EEPROM.commit();

    // Add device to smart place

addDeviceToSmartPlace("dev_id="+DEV_ID+"&smart_place="+SMART_PLACE+"&name="+DEV
_NAME+"&type="+DEV_TYPE+"&req_every="+REQ_EVERY);
    Serial.println("Response: "+API_RESPONSE);

    dht.begin();
  }else{
    startApAndWebServer();
```

```
  }

}

void handle_NotFound(){
  server.send(404, "text/plain", "Not found");
}


// Helping functions
long influxTimeToTime(String tm){
  if(tm.indexOf("s") > 0){
    tm.replace("s","");
    return tm.toInt() * 1000;
  }
  if(tm.indexOf("m") > 0){
    tm.replace("m","");
    return tm.toInt() * 60 * 1000;
  }
  if(tm.indexOf("h") > 0){
    tm.replace("h","");
    return tm.toInt() * 60 * 60 * 1000;
  }

  return 10000;
}
```

Παράρτημα III Κώδικας έξυπνης πρίζας

Παρακάτω δίνεται ο κώδικας C της συσκευής έξυπνης πρίζας. Για την συγγραφή του προγράμματος χρησιμοποιήθηκε το ολοκληρωμένο περιβάλλον ανάπτυξης Arduino IDE.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266HTTPClient.h>
#include <EEPROM.h>
#include <DNSServer.h>
#include <Arduino_JSON.h>

#define RELAY_PIN 2

String DEF_AP_SSID = "IoT Plug";
String DEF_AP_PSWD = "";

bool DEF_AP_RUNNING = true;
bool CONNECTED_TO_HOME_WIFI = false;
String API_RESPONSE = "";
String HOST_DASH = "http://185.243.214.236";

String DEV_ID = WiFi.macAddress();
String DEV_NAME = "PLUGV1";
String DEV_TYPE = "plug";
String REQ_EVERY = "3s"; // default value

String SMART_PLACE = "";
String HOME_WIFI_SSID = "";
String HOME_WIFI_PSWD = "";

ESP8266WebServer server(80);

const byte DNS_PORT = 53;
```

```
DNSServer dnsServer;
IPAddress apIP(192, 168, 1, 1);
IPAddress netMsk(255, 255, 255, 0);

void(* resetFunc) (void) = 0;  // declare reset function at address 0

void setup(){
  Serial.begin(9600);
  Serial.println();
  delay(500);

  DEV_ID.replace(":", "");
  DEV_ID.toLowerCase();

  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, HIGH);

  // EEPROM initialize
  EEPROM.begin(512);
  loadSavedWiFiCredentials();

  if(HOME_WIFI_SSID.length() > 0 && HOME_WIFI_PSWD.length() > 0 &&
SMART_PLACE.length() > 0){ // scan & connect to wifi
    Serial.println("Saved Data Found: "+HOME_WIFI_SSID+" "+HOME_WIFI_PSWD+"
"+SMART_PLACE);
    int numberOfNetworks = WiFi.scanNetworks();
    for(int i=0; i<numberOfNetworks; i++){
        if(WiFi.SSID(i)==HOME_WIFI_SSID){
          if(connectToWiFi()){ // connected to WiFi
            Serial.println("Connected to saved WiFi");
            return;
          }else{ // Wrong credentials
            Serial.println("WiFi Access Point, changed credentials");
            eraseEEPROM();
            startApAndWebServer();
            return;
          }
        }
    }

    Serial.println("Saved WiFi not found within range");
    eraseEEPROM();
    startApAndWebServer();
```

```
  }else{ // first time
    Serial.println("First launch");
    eraseEEPROM();
    startApAndWebServer();
  }
}

void loop(){
  if(DEF_AP_RUNNING){
    server.handleClient();
    Serial.printf("Stations connected = %d\n", WiFi.softAPgetStationNum());
  }

  if(CONNECTED_TO_HOME_WIFI){

      getDeviceInfo("device_id="+DEV_ID);
      Serial.println("Response: "+API_RESPONSE);

      JSONVar myObject = JSON.parse(API_RESPONSE);
      if (JSON.typeof(myObject) == "undefined") {
        Serial.println("Parsing input failed!");
        return;
      }

      if((int)(myObject["code"]) == 404){
        eraseEEPROM();
        resetFunc(); //call reset
      }

      if(myObject["data"]["status"] == JSONVar("on")){ // relay einai anapoda
        digitalWrite(RELAY_PIN, LOW);
      }else{
        digitalWrite(RELAY_PIN, HIGH);
      }

      REQ_EVERY = (const char*) myObject["data"]["req_every"];
      delay(influxTimeToTime(REQ_EVERY)); // sleep unril next ping

  }


  delay(1000);
```

```
}

// HTTPS Requests ============================================
void addDeviceToSmartPlace(String payload){

  WiFiClient wifiClient;
  HTTPClient http;
  http.begin(wifiClient, HOST_DASH+"/api.php?add-device");
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
  int httpResponseCode = http.POST(payload);
  if(httpResponseCode){
    API_RESPONSE = http.getString();
  }else{
    API_RESPONSE="error";
  }

  http.end();
}

void getDeviceInfo(String payload){

  WiFiClient wifiClient;
  HTTPClient http;
  http.begin(wifiClient, HOST_DASH+"/api.php?get-device");
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");
  int httpResponseCode = http.POST(payload);
  if(httpResponseCode){
    API_RESPONSE = http.getString();
  }else{
    API_RESPONSE="error";
  }

  http.end();
}

// WiFi Functions ============================================
bool connectToWiFi(){

  byte seconds = 0;
  WiFi.begin(HOME_WIFI_SSID, HOME_WIFI_PSWD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    seconds+=1;
```

```
    Serial.print(".");
    if(seconds==10) break;
  }

  if(seconds < 10){ // Correct WiFi credentials

    Serial.println("");
    Serial.println("WiFi connected..!");
    Serial.print("Got IP: ");  Serial.println(WiFi.localIP());
    CONNECTED_TO_HOME_WIFI = true;
    DEF_AP_RUNNING = false;
    return true;

  }else{// Wrong WiFi credentials
    CONNECTED_TO_HOME_WIFI = false;
    DEF_AP_RUNNING = false;
    return false;
  }
}

// EEPROM Functions ==========================================
void eraseEEPROM(){
  for(int i=0; i < 512; i++) EEPROM.write(i, 0);
  EEPROM.commit();
}

void loadSavedWiFiCredentials(){
  short seperatorFound = 0;
  for(int i=0; i < 512; i++) {
    int inp = EEPROM.read(i);
    if(inp==0) return; // empty data
    if(char(inp)==':'){
      seperatorFound+=1;
      continue;
    }

    if(seperatorFound==0) HOME_WIFI_SSID+=char(inp);
    if(seperatorFound==1) HOME_WIFI_PSWD+=char(inp);
    if(seperatorFound==2) SMART_PLACE+=char(inp);

  }

}
```

```
// Access Point & Web Server Functions ============================
void startApAndWebServer(){
  DEF_AP_RUNNING = true;

  Serial.println("Setting AP");
  WiFi.softAPConfig(apIP, apIP, netMsk);
  boolean result = WiFi.softAP(DEF_AP_SSID, DEF_AP_PSWD);
  delay(500);
  if(result == true){
    Serial.println("Default AP is ready");
  }else{
    Serial.println("Default AP is failed");
  }

  // Setup the DNS server redirecting all the domains to the apIP
  dnsServer.setErrorReplyCode(DNSReplyCode::NoError);
  dnsServer.start(DNS_PORT, "*", apIP);

  // Start Web Server
  Serial.print("Gateway IP: ");  Serial.println(WiFi.softAPIP());
  server.on("/", handle_OnConnect);
  server.on("/init", handle_Init);
  server.onNotFound(handle_NotFound);
  server.begin();
  Serial.println("HTTP server started");

}

void handle_OnConnect(){
  String html = "<!DOCTYPE html> <html>";
    html += "<head><meta name='viewport' content='width=device-width, initial-
scale=1.0, user-scalable=no'>";
    html += "<title>"+DEF_AP_SSID+"</title>";
    html += "<style>";
    html += "body{height:100vh;background: linear-gradient(45deg, rgb(61 95
191) 0%, rgb(78 194 220) 100%);text-align:center;color:white;font-
family:arial;}";
    html += "input[type='text'], input[type='password']{padding:8px;margin-
bottom:15px;border-radius:4px;border:0px;font-
size:20px;background:#0a4458;color:white;}";
    html += "input[type='submit']{padding:8px;border-
radius:4px;background:white;color:black;border:0px;font-size:16px;}";
```

```
    html += "</style>";
    html += "</head>";
    html += "<body>";
    html += " <h1>"+DEF_AP_SSID+"</h1>";
    html += " <p>DEVICE TYPE: "+DEV_NAME+" ID: "+DEV_ID+"</p>";
    html += " <center><form action='/init' method='get'>";
    html += "   <input type='text' name='smart_place' placeholder='Smart Place
ID' /><br>";
    html += "   <input type='text' name='ssid' placeholder='SSID' /><br>";
    html += "   <input type='password' name='pswd' placeholder='Password'
/><br>";
    html += "   <input type='submit' value='CONNECT' />";
    html += " </form></center>";
    html += "</body>";
    html += "</html>";


  server.send(200, "text/html", html);
}

void handle_Init(){
  for(int i = 0; i < server.args(); i++){
    if(server.argName(i)=="smart_place") SMART_PLACE = server.arg(i);
    if(server.argName(i)=="ssid") HOME_WIFI_SSID = server.arg(i);
    if(server.argName(i)=="pswd") HOME_WIFI_PSWD = server.arg(i);
  }

  // Disconnect AP
  WiFi.softAPdisconnect(true);
  DEF_AP_RUNNING = false;

  if(connectToWiFi()){

    //save wifi credentials
    String toWrite = HOME_WIFI_SSID+":"+HOME_WIFI_PSWD+":"+SMART_PLACE;
    for(int i=0; i<toWrite.length(); i++) EEPROM.write(i, toWrite [i]);
    EEPROM.commit();

    // Add device to smart place

addDeviceToSmartPlace("dev_id="+DEV_ID+"&smart_place="+SMART_PLACE+"&name="+DEV
_NAME+"&type="+DEV_TYPE+"&req_every="+REQ_EVERY);
    Serial.println("Response: "+API_RESPONSE);
```

```
  }else{
    startApAndWebServer();
  }

}

void handle_NotFound(){
  server.send(404, "text/plain", "Not found");
}

// Helping functions
long influxTimeToTime(String tm){
  if(tm.indexOf("s") > 0){
    tm.replace("s","");
    return tm.toInt() * 1000;
  }
  if(tm.indexOf("m") > 0){
    tm.replace("m","");
    return tm.toInt() * 60 * 1000;
  }
  if(tm.indexOf("h") > 0){
    tm.replace("h","");
    return tm.toInt() * 60 * 60 * 1000;
  }

  return 10000;
}
```