Mini SmartHome System

Nafsika Abatzi
School of E.C.E
National Technical University of Athens
Athens, Greece
el17198@mail.ntua.gr

Dimitrios Dimos
School of E.C.E
National Technical University of Athens
Athens, Greece
el17165@mail.ntua.gr

Georgios Themelis
School of E.C.E
National Technical University of Athens
Athens, Greece
el17131@mail.ntua.gr

I. Εισαγωγή

Η παρούσα αναφορά στοχεύει στην ανάλυση και περιγραφή του τρόπου με τον οποίο χρησιμοποιήθηκε το εργαλείο Node-Red για την υλοποίηση ενός συστήματος έξυπνου σπιτιού. Το έξυπνο σπίτι είναι η τεχνολογία του μέλλοντος που προσφέρει άνεση και οικονομία, ενώ ταυτόχρονα αλληλεπιδρά με τους χρήστες. Καλύπτει αυτοματοποιημένα μέσω ηλεκτρονικών και τεχνολογικών καινοτομιών τις διάφορες ανάγκες. Αρχικά δίνεται η δυνατότητα ελέγχου και διαχείρισης εσωτερικών και εξωτερικών χώρων. Επιπλέον, μπορεί να ελέγχεται η λειτουργία και οι καταναλώσεις των συσκευών του φωτισμού και θέρμανσης απ' όπου κι αν βρίσκεται ο χρήστης. Τέλος, ένα σημαντικό πλεονέκτημα ενός συστήματος έξυπνου σπιτιού είναι η ασφάλεια. Ο χρήστης ενημερώνεται ανά πάσα στιγμή για ύποπτες κινήσεις αλλά και για τυχόν διαρροές αερίων, ανοικτές συσκευές κ.α. Θεωρούμε ότι το σύστημα αυτό θα αφορά ένα σπίτι με μπάνιο, κουζίνα, υπνοδωμάτιο, μπαλκόνι, σαλόνι και γκαράζ. Φυσικά μπορεί να επεκταθεί και σε περιπτώσεις περισσότερων χώρων. Ο σκοπός του συστήματος που υλοποιήθηκε είναι να καλυφθούν οι παραπάνω απαιτήσεις, μέσω της προσομοίωσης της λειτουργίας των αισθητήρων που στέλνουν δεδομένα και της δυνατότητας των χρηστών να ρυθμίζουν τη λειτουργία κάποιων συσκευών από οπουδήποτε με τη χρήση απλώς μίας ηλεκτρονικής συσκευής, όπως το κινητό. Έτσι ο κάτοικος έχει πλήρη έλεγχο του κτίσματος όσο μακρυά και αν βρίσκεται.

II. Περιγραφή υποδομής και SOFTWARE

Ακολουθούν επιγραμματικά τα εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν και στη συνέχεια παρουσιάζονται αναλυτικά:

1) Το βασικό εργαλείο για την ανταλλαγή μηνυμάτων που χρησιμοποιήθηκε για την ανάπτυξη του συστήματος έξυπνου σπιτιού (Mini Smart Home) είναι το **Node-Red**.
2) Για την παραλαβή των μηνυμάτων χρησιμοποιήθηκε το **Mosquitto MQTT**.
3) Η αποθήκευση των δεδομένων γίνεται στο Relational Database Management System **PostgreSQL**.
4) Η γραφική απεικόνιση των δεδομένων γίνεται με τη χρήση του εργαλείου **Grafana**.
5) Η αποστολή μηνυμάτων για την ενημέρωση του χρήστη του συστήματος γίνεται στο **Slack**.
6) Για τη διαχείριση των διάφορων συσκευών από τον χρήστη χρησιμοποιήθηκε το Grafana (για την υλοποίηση των διαθέσιμων ενεργειών) και ένα API σε Nodejs για την παραλαβή και διαχείριση των αιτημάτων.

A. Node-Red

Το Node-Red στοχεύει στην εύκολη διασύνδεση συσκευών, API και διαδικτυακών υπηρεσιών. Έχει υλοποιηθεί πάνω σε Nodejs και οι λειτουργίες που αναπτύσονται αναπαρίστανται ως nodes (μαύρο κουτί), με αποτέλεσμα να εκμεταλλεύεται πλήρως το event-driven, non-blocking μοντέλο της Nodejs. Γι' αυτό τον λόγο θεωρείται και ένα lightweight I/O μοντέλο. Με τη χρήση του δίνεται έμφαση στον τρόπο διασύνδεσης και οργάνωσης των λειτουργιών και όχι στην παραγωγή κώδικα, αφού τα δεδομένα προωθούνται και επεξεργάζονται μέσω των nodes. Το αποτέλεσμα που προκύπτει είναι μία γραφική απεικόνιση ενός συστήματος και όχι πολλά αρχεία κώδικα. Έτσι η κατανόηση της εφαρμογής καθίσταται πολύ πιο εύκολη αφού είναι πλήρως οπτικοποιημένη.
Το Node-Red παρέχει ένα browser-based editor μέσω του οποίου με drag and drops από nodes και λίγη χρήση javascript για τη δημιουργία συναρτήσεων, κάποιος μπορεί να αναπτύξει μία ολοκληρωμένη εφαρμογή. Οι διασυνδέσεις ανάμεσα στα διάφορα nodes ονομάζονται flows. Ένα συνηθισμένο flow αποτελείται από τον συνδιασμό input, processing και output nodes. Τα flows αποθηκεύονται σε JSON μορφή και γίνοναι εύκολα import και export, με αποτέλεσμα να μοιράζονται εύκολα και γρήγορα. Το Node-Red ουσιαστικά στέλνει μηνύματα ανάμεσα στα διάφορα nodes. Τα μηνύματα αυτά είναι JavaScript objects με διάφορες ιδιότητες, με την πιο συνηθισμένη το payload.
Ακολουθεί μία εικόνα (Fig. 1) με το κύριο περιβάλλον του Node-Red. Στα αριστερά βρίσκονται οι διαθέσιμοι κόμβοι, στο κέντρο παρουσιάζεται ένα παράδειγμα ενός flow (αποτελεί μέρος των flows του παρόντος project) και στο δεξιό μέρος έχουμε συγκεντρωμένα όλα τα flows του project. Όπως βλέπουμε στο παράδειγμα του flow που απεικονίζεται έχουν χρησιμοποιηθεί διάφοροι κόμβοι για ανάγνωση, επεξεργασία και αποθήκευση ροών δεδομένων. Η επεξεργασία των δεδομένων γίνεται μέσω των συναρτήσεων (function nodes) με τη χρήση javascript. Στην επόμενη ενότητα εξηγούνται αναλυτικά οι επιλογές

και η διασύνδεση των κόμβων στο συγκεκριμένο project. Επίσης, υπάρχει η επιλογή του Manage Palette για την εύρεση και εγκατάσταση νέων κόμβων, πέραν των default. Αρκετά σημαντικός κόμβος είναι πάντα ο κόμβος με όνομα msg.payload(πράσινος στην εικόνα), ο οποίος εμφανίζει σε ένα debug window το payload του κόμβου που είναι συνδεδεμένος. Επιπλέον αρκετά χρήσιμος είναι ο κόμβος inject (ο πρώτος από τους διαθέσιμους κόμβους), μέσω του οποίου κάποιος μπορεί να ξεκινήσει την αποστολή ενός μηνύματος. Στο σύστημα που δημιουργήσαμε δεν χρησιμοποιείται βέβαια, καθώς διαχειριζόμαστε δεδομένα από αισθητήρες, ωστόσο αναφέρεται για λόγους πληρότητας.

Τέλος, επίσης για λόγους πληρότητας αναφέρεται ότι το Node-Red δίνει τη δυνατότητα δημιουργίας ενός dashboard (μέσω nodes) για καλύτερη οπτικοποίηση των καταστάσεων και των δεδομένων. Ωστόσο η προσθήκη της λειτουργίας αυτής δεν ήταν στα πλαίσια του συγκεκριμένου project, καθώς για την οπτικοποίηση χρησιμοποιήθηκε το εργαλείο Grafana.

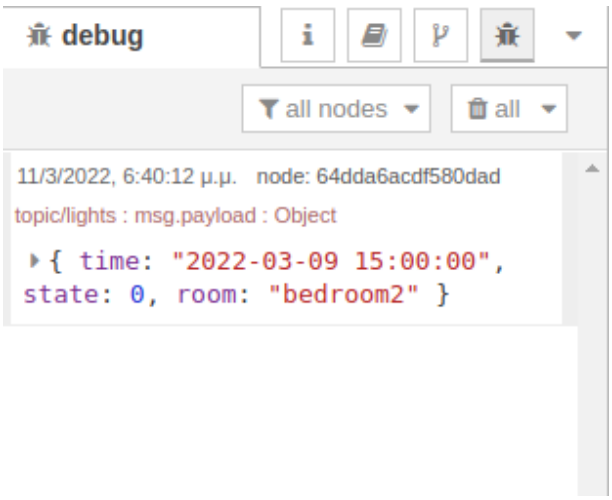

Fig. 2. Debug Window

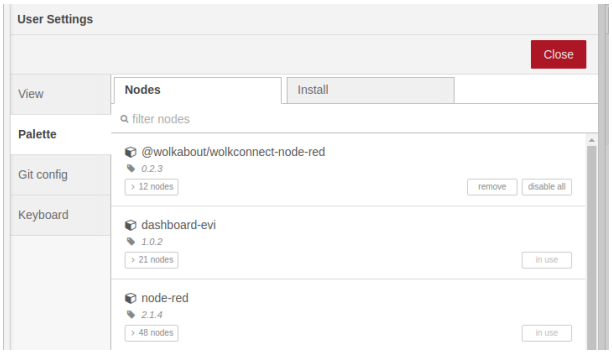

Fig. 3. Manage Palette

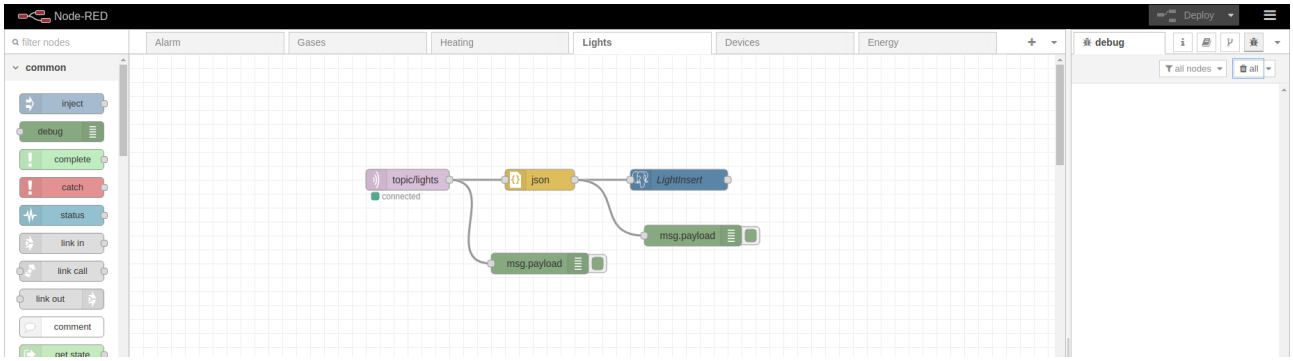

Fig. 1. Node-Red Environment

B. Mosquitto-MQTT

Το Eclipse Mosquitto είναι ένα open source message broker, το οποίο υλοποιεί το MQTT πρωτόκολλο. Ο message broker είναι μια ενδιάμεση μονάδα προγράμματος υπολογιστή που μεταφράζει ένα μήνυμα από το επίσημο πρωτόκολλο ανταλλαγής μηνυμάτων του αποστολέα στο επίσημο πρωτόκολλο ανταλλαγής μηνυμάτων του δέκτη. Το Mosquitto αποτελεί ένα "ελαφρύ" message broker και γι'αυτό είναι κατάλληλο για χρήση από low power single board computers μέχρι ολόκληρους servers. Το MQTT πρωτόκολλο περιέχει επίσης έναν ελαφρύ μηχανισμό μεταφοράς μηνυμάτων με τη χρήση του publish/subscribe model. Είναι κατάλληλο για Internet of Things messaging, όπως low power sensors. Ακολουθούν κάποιες βασικές έννοιες του MQTT και η σχηματική τους απεικόνιση:

1) Αποτελέι ένα **publish/subscribe** μηχανισμό. Μία συσκευή μπορεί να κάνει publish ένα μήνυμα σε ένα topic ή μπορεί να είναι subscribed σε ένα συγκεκριμένο topic και να λαμβάνει μηνύματα μέσω αυτού.
2) Τα μηνύματα **(messages)** είναι η πληροφορία που θέλουμε να ανταλλαχθεί ανάμεσα στις συσκευές. Μπορεί να είναι είτε δεδομένα είτε κάποια εντολή.
3) Τα **topics** αποτελούν τον τρόπο με τον οποίο ο subscriber εκδηλώνει ενδιαφέρον για εισερχόμενα μηνύματα ή ο προορισμός όπου ο publisher στέλνει κάποιο μήνυμα.
4) Ο **broker** είναι υπεύθυνος για την λήψη όλων των μηνυμάτων, το φιλτράρισμά τους, την απόφαση για το ποιος ενδιαφέρεται γι'αυτά και τέλος, το publish τους σε όλους τους subscribed clients.
5) Στο MQTT, ο publisher(client/device) κάνει publish μηνύματα σε ένα topic και ο subscriber πρέπει να κάνει subscribe στο ίδιο topic για να τα λάβει.

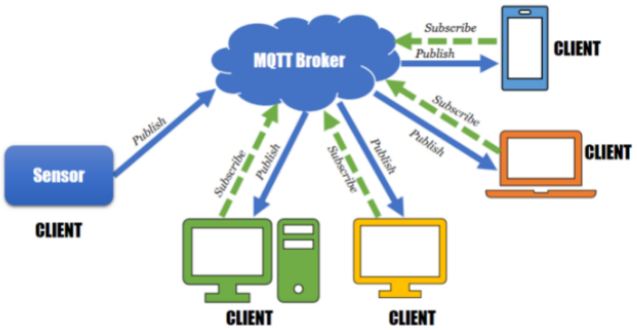

Fig. 4. MQTT Broker

Ακολουθούν μερικές ακόμα πληροφορίες σχετικά με το MQTT:

- Οι περισσότεροι MQTT clients θα συνδεθούν με τον broker και θα παραμείνουν συνδεδεμένοι ακόμα κι αν δεν στέλνουν δεδομένα. Αυτό το πετυχαίνουν στέλνοντας ένα keepalive message (κάθε 60 secs περίπου) στον broker ενημερώνοντάς τον ότι είναι ακόμα συνδεδεμένοι.
- Όλοι οι clients πρέπει να έχουν ένα client name ή ID, το οποίο είναι μοναδικό.
- Οι MQTT clients εγκαθιστούν **clean sessions** με τον broker, δηλαδή όταν αυτοί αποδυνδεθούν, ο broker δεν θα "θυμάται" κάτι γι' αυτούς. Σε περίπτωση που έχουμε unclean session, όταν ο client αποσυνδεθεί όσα subscriptions έχει θα αποθηκευτούν μέχρι να ξανασυνδεθεί.
- Το MQTT έχει τρία επίπεδα **Quality of Service(QoS)**, τα οποία ορίζουν πόσο "σκληρά" ο brocker/client θα προσπαθήσει να εξασφαλίσει ότι το μήνυμα έχει παραδωθεί.
- Υπάρχει η δυνατότητα ορισμού retained μηνυμάτων, δηλαδή η διατήρησή τους από τον broker, ακόμα και αφού αυτά σταλούν.
- Όταν ένας client συνδέεται στον broker μπορεί να τον ενημερώσει ότι έχει ένα **Will**. Έτσι εάν ο client αποσυνδεθεί ξαφνικά, ο broker θα ενημερώσει στέλνοντας αυτό το Will message.

C. PostgreSQL

Η PostgreSQL είναι ένα open source object-relational database system, που χρησιμοποιεί SQL σε συνδυασμό με αρκετά features που αποθηκεύουν και κάνουν scale με ασφαλή τρόπο περίπλοκα δεδομένα. Ο μεγάλος αριθμός αριθμός από features έχει στόχο την βοήθεια των developers στον σχεδιασμό εφαρμογών, την προστασία δεδομένων των administrators στο data integrity και τελικά την διαχείριση των δεδομένων ανεξαρτήτως από το μέγεθός τους. Επιπλέον, η PostgreSQL είναι επεκτάσιμη.

D. Grafana

Το εργαλείο Grafana είναι μία multi-platform open source analytics και interactive visualization πλατφόρμα. Δίνεται η δυνατότητα δημμιουργίας dashboards μέσω των οποία μπορούν να αναπαρασταθούν δεδομένα από πηγές όπως Kubernetes cluster, raspberry pi, cloud services, μέχρι ακόμα κα Google Sheets. Υπάρχει η δυνατότητα για share των dashboards ανάμεσα σε ομάδες χρηστών. Μέσω του Grafana μπορούμε να αναλύσουμε time series δεδομένα, τη συμπεριφορά μίας εφαρμογής, συχνότητα λαθών σε production ή pre-production περιβάλλοντα κ.α.

E. Slack

Το Slack είναι μία πλατφόρμα επιχειρηματικής επικοινωνίας με ευρεία χρήση. Χρησιμοποιήθηκε στην ανάπτυξη του συστήματος έξυπνου σπιτιού ώστε να αποστέλλονται σε αντίστοιχα κανάλια μηνύματα όπως ανίχνευση φωτιάς, ενεργοποίηση του συναγερμού κ.α., με σκοπό να ενημερώνεται άμεσα ο χρήστης για επικίνδυνες καταστάσεις οι οποίες μπορεί να έχουν δημιουργηθεί απ'όπου κι αν βρίσκεται.

III. Περιγραφή Βημάτων

Στην ενότητα αυτή θα περιγραφούν και οπτικοποιηθούν αναλυτικά τα βήματα που ακολουθήθηκαν για την δημιουργία του συστήματος Mini SmartHome.

A. Συνοπτική Περιγραφή

Θα αναφέρουμε συνοπτικά τη λογική και τη σειρά κάποιων βασικών ενεργειών για την δημιουργία του συστήματος και στη συνέχεια θα γίνει αναλυτική περιγραφή. Αρχικά ορίσαμε τα use cases και τη μορφή των δεδομένων που θα επεξεργαζόμαστε. Η λογική του συστήματός μας είναι η εξής:

1) Το εργαλείο Node-Red περιέχει MQTT Clients που έχουν κάνει subscribe σε συγκεκριμένα topics και αναμένουν μηνύματα από τους publishers.
2) Μέσω του Mosquitto MQTT στέλνονται μηνύματα με δεδομένα, θεωρητικά από αισθητήρες του σπιτιού, οι οποίοι καταγράφουν γεγονότα ανά τακτά χρονικά διαστήματα και κάνουν publish στα διάφορα topics.
3) Μόλις ένας subscriber του Node-Red λάβει ένα μήνυμα (μία κατάσταση κάποιου αισθητήρα) το επεξεργάζεται και το αποθηκεύει τελικά στη βάση δεδομένων PostgreSQL. Εάν κριθεί απαραίτητο στέλνονται προειδοποιητικά μηνύματα στο Slack.
4) Μέσω του Grafana ο χρήστης μπορεί να δει γραφικές παραστάσεις από τις καταστάσεις της βάσης δεδομένων συναρτήσει του χρόνου για τους χώρους και τις συσκευές.
5) Μέσω του Grafana, ο χρήστης μπορεί να αλλάξει καταστάσεις για κάποιες συσκευές (πχ. άνοιγμα/κλείσιμο της τηλεόρασης).

B. Προετοιμασία

Για την εκπόνηση του συγκεκριμένου project εγκαταστάθηκαν οι εφαρμογές και τα εργαλεία που αναφέρθηκαν παραπάνω και δημιουργήθηκε η βάση δεδομένων. Έπειτα σχεδιάστηκε το ER διάγραμμα των use cases, το οποίο παρουσιάζεται στη συνέχεια (Fig. 5) με την προσθήκη σχολίων για την περιγραφή κάποιων βασικών λειτουργιών. Οι λειτουργίες που θα υλοποιεί το σύστημά μας είναι οι εξής:

1) Έλεγχος από αισθητήρες για ανίχνευση φωτιάς και επικίνδυνα επίπεδα CO [1] και CO2 [2] σε όλους τους χώρους. Σε περίπτωση ανίχνευσης αποστέλεται μήνυμα στο Slack στην ενότητα gases.
2) Έλεγχος εάν η πόρτα του ψυγείου έμεινε ανοικτή για σχετικά μεγάλο χρονικό διάστημα και αποστολή σχετικής προειδοποίησης στο slack στο κανάλι home devices.
3) Έλεγχος εάν χτυπάει το κουδούνι και αποστολή σχετικού μηνύματος στο slack στο κανάλι doorbell, ώστε ο χρήστης να γνωρίζει ακόμα κι αν λείπει για την ύπαρξη κάποιας παρουσίας.
4) Καταγραφή της κατάστασης των φώτων σε όλους τους χώρους και δυνατότητα διεχείρισής τους μέσω του Grafana.
5) Καταγραφή της θερμοκρασίας και αποστολή προειδοποιητικού μηνύματος στο slack στο κανάλι gases σε περίπτωση που ανιχνευθεί υπερβολικά χαμηλή ή υπερβολικά υψηλή θερμοκρασία ενώ υπάρχει κάποιος στο σπίτι. Δυνατότητα ρύθμισης της θερμοκρασίας απ'όπου κι αν βρίσκεται ο χρήστης μέσω του Grafana. Για τη θέρμανση θεωρούμε ότι το σπίτι διαθέτει θερμοστάτη και σύγχρονο λέβητα πετρελαίου.
6) Καταγραφή της θερμοκρασίας και χωρητικότητας του λέβητα πετρελάιου. Σε περίπτωση που η θερμοκρασία ξεπεράσει τους 90°C [3] στέλνεται προειδοποιητικό μήνυμα στο slack. Επιπλέον θεωρούμε ότι το σπίτι διαθέτει λέβητα πετρελαίου χωρητικότητας 300 λίτρων. Έτσι όταν η ποσότητα του πετρελαίου πέσει κάτω από 40 λίτρα θα στέλνεται επίσης προειδοποιητικό μήνυμα στο slack στο κανάλι home devices.
7) Καταγραφή της συνολικής κατανάλωσης ρεύματος/minute. Η μέση κατανάλωση ενέργειας σε ένα σπίτι θεωρούμε ότι είναι 28kWh ανά ημέρα. [4]
8) Έλεγχος εάν είναι ανοικτός ο φούρνος ή κάποια βρύση ενώ δεν βρίσκεται κανένας στο σπίτι. Αποστολή σχετικού προειδοποιητικού μηνύματος στο slack στο home devices. Ο χρήστης μέσω του Grafana έχει τη δυνατότητα επίσης να ανοίξει ή να κλείσει τον φούρνο απ' όπου κι αν βρίσκεται. Δεν δώσαμε αυτή τη δυνατότητα για τη βρύση, καθώς θεωρούμε ότι ανοιγοκλείνει μόνο χειροκίνητα.
9) Καταγραφή της κατάστασης της τηλεόρασης, του ραδιαφώνου και της μηχανής καφέ και η δυνατότητα αλλαγής της απ'όπου κι αν βρίσκεται ο χρήστης μέσω του Grafana.
10) Συνεχής καταγραφή της κατάστασης του συναγερμού (ανοικτός ή κλειστός).
11) Συνεχής καταγραφή της ύπαρξης ή μη κάποιου στο σπίτι (αισθητήρας present).
12) Σε περίπτωση που καταγραφεί παρουσία στο σπίτι ενώ ο συναγερμός είναι ανοικτός μας έρχεται αμέσως μήνυμα στο slack στο κανάλι alarm.
13) Συνεχής καταγραφή της κατάστασης των παραθύρων και των πόρτων σε όλους τους χώρους.
14) Σε περίπτωση που ανιχνευθεί ανοικτό παράθυρο ή πόρτα ενώ είναι ανοικτός ο συναγερμός, έρχεται προειδοποιητικό μήνυμα αμέσως στο slack στο κανάλι home devices. Σχετικά με τις πόρτες, θεωρούμε ότι για να μας έρθει το μήνυμα θα πρέπει να είναι ανοικτή η κεντρική πόρτα, η πόρτα του μπαλκονιού ή η πόρτα του γκαράζ.

Αναφορικά με τη λειτουργία του συναγερμού, θεωρήσαμε μία απλοποιημένη λειτουργία του. Θεωρούμε ότι ο συναγερμός θα έχει κι άλλες λειτουργίες που θα καθορίζονται από την εταιρία κατασκευής του, όπως αναμονή για κάποιο χρονικό διάστημα πριν χτυπήσει όταν είναι ανοικτή η κεν-

τρική πόρτα (ώστε κάποιος να μπορεί να πληκτρολογήσει τον κωδικό). Φυσικά ο συναγερμός μπορεί να είναι και ενεργοποιημένος ακόμα και όταν βρισκόμαστε στο σπίτι, ωστόσο εξετάσαμε μόνο την περίπτωση που ο χρήστης τον έχει ενεργοποιήσει και θα λείπει από το σπίτι για να έχουν περισσότερο νόημα και οι ειδοποιήσεις στο slack. Δεν δημιουργήσαμε δηλαδή εκ νέου ένα σύστημα συναγερμού, αλλά "πατήσαμε" πάνω σε ένα ήδη εγκατεστημένο και επεκτείναμε κάποιες λειτουργίες του. Καθώς ο στόχος είναι η δημιουργίας ενός mini smarthome system, αποφασίσαμε να καλύψουμε έναν σημαντικό αριθμό περιπτώσεων με κάποιες παραδοχές ώστε να δείχουμε τη βασική ιδέα, αλλά να μην επαναλλαμβάνουμε λεπτομέρειες.

Αναφορικά με τη σχεδίαση της βάσης δεδομένων, δημιουργήσαμε tables για κάθε μία από τις παραπάνω περιπτώσεις, όπως φαίνεται και στο διάγραμμα. Δεν προέκυψαν σχέσεις ανάμεσα στα tables, καθώς στο καθένα θα αποθηκεύονται δεδομένα από διαφορετικούς αισθητήρες του σπιτιού. Θεωρήσαμε ότι μία διαφορετική σχεδίαση με ορισμό κάποιων εξαρτήσεων (πχ. συναγερμού και παρουσίας στο σπίτι) απλά θα περιέπλεκε τον τρόπο χειρισμό των δεδομένων.

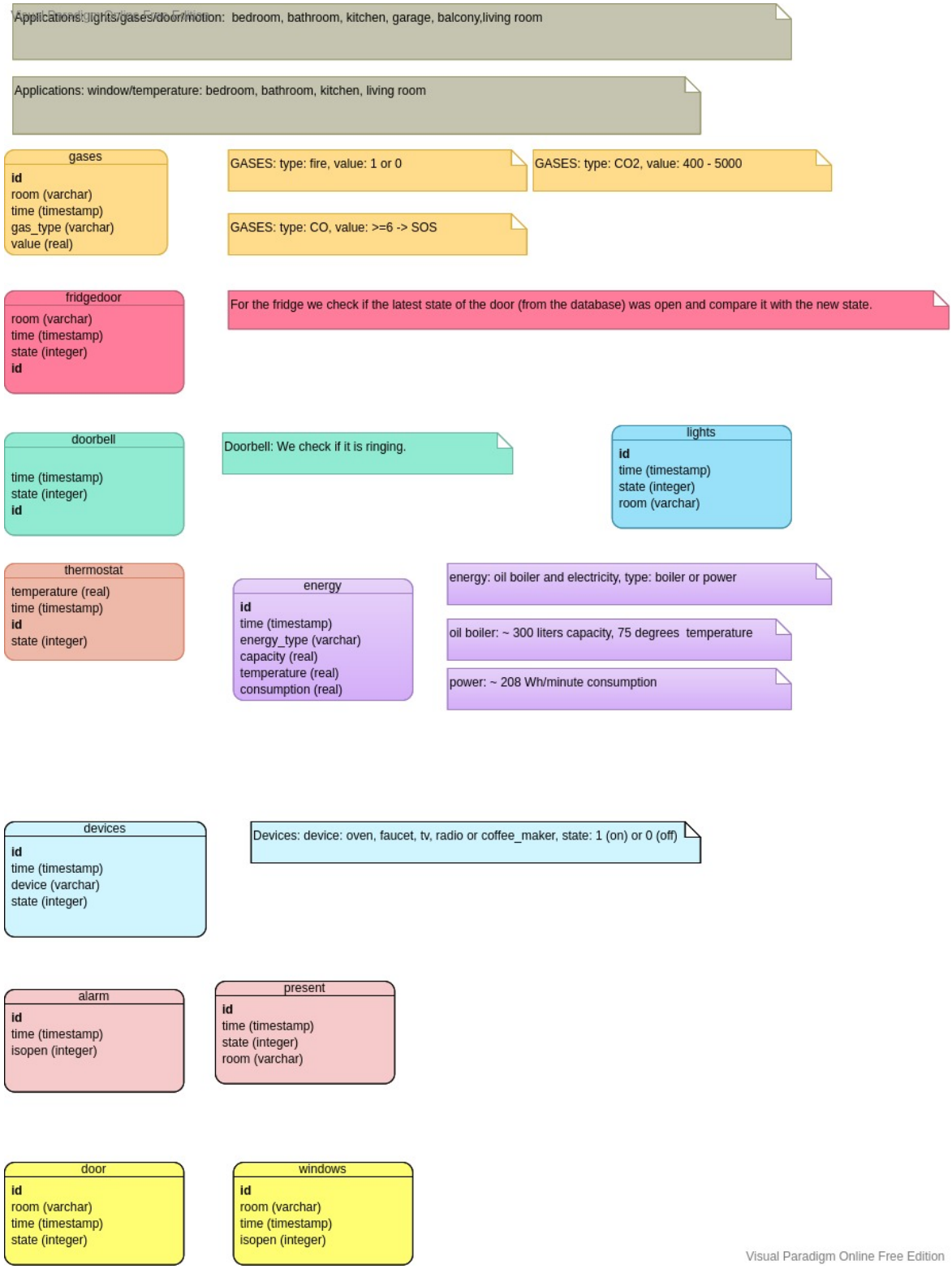

Fig. 5. ER SmartHome App

C. Αποστολή δεδομένων με το *Mosquitto MQTT*

Στη συνέχεια ορίσαμε τα μηνύματα που θα στέλνονται από τον publisher και θα θεωρούνται δεδομένα που καταγράφουν οι διάφοροι αισθητήρες του σπιτιού και έχουν συνδεθεί με το σύστημά μας. Ένα χαρακτηριστικό παράδειγμα τέτοιου μηνύματος είναι το εξής:

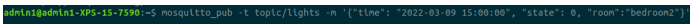

Fig. 6. MQTT Publish

Αναφορικά με το παραπάνω παράδειγμα, μέσω του mosquitto_pub γίνεται publish στο topic (-t παράμετρος) ”topic/lights” το μήνυμα (-m παράμετρος) ”’time”: ”2022-03-09 15:00:00”, ”state”: 0, ”room”:”bedroom2”’. Επιλέξαμε όλα τα μηνύματα που θα στέλνονται να είναι σε μορφή JSON, όπως σε αυτό το παράδειγμα. Συνεχίζοντας στο παράδειγμα μας, στο μήνυμα που στείλαμε έχει τα πεδία time για την καταγραφή της χρονικής στιγμής που ο αισθητήρας έστειλε μήνυμα, room για την αναφορά στο δωμάτιο που καταγράφει και state για το εάν το φως του δωματίου ήταν ανοικτό(1) ή κλειστό (0). Όλα τα μηνύματα που προσομοιώνουν τη λειτουργία των αισθητήρων του συστήματος έχουν παρόμοια μορφή.
Φυσικά με το να στέλναμε publish μηνύματα χειροκίνητα μέσω του terminal δεν ήταν πρακτικό και δεν προσομοιώνεται σωστά η λειτουργία των υποθετικών αισθητήρων. Γι’ αυτό το λόγο δημιουργήσαμε python scripts για κάθε topic, τα οποία ανά 5 λεπτά στέλνουν σχετικά μηνύματα (με κατάλληλο περιορισμό των τιμών σε κάθε περίπτωση).

D. *Node-Red Data Flows*

Με το εργαλείο Node-Red, το οποίο είναι και το πιο σημαντικό σε αυτό το σύστημα λαμβάνουμε, επεξεργαζόμαστε και αποθηκεύουμε δεδομένα στη βάση. Η εκκίνησή του γίνεται από το terminal με την εντολή node-red. Για να ανοίξουμε το περιβάλλον, ανοίγουμε τον browser και πλοηγούμαστε στη σελίδα http://127.0.0.1:1880/. Αρχικά ορίσαμε 6 κατηγορίες από flows:

1) Alarm
2) Gases
3) Heating
4) Lights
5) Devices
6) Energy

Κάθε flow ξεκινάει με έναν mqtt in node, ο οποίος είναι ένας subscriber σε κάποιο topic. Όταν κάποιος publisher στείλει μήνυμα στο topic, ο κόμβος ”ενεργοποιείται” και έχει ως έξοδο το μήνυμα που έλαβε. Σε κάθε περίπτωση η έξοδος των subscribers είναι ένα json string μήνυμα (όπως περιγράφτηκε παραπάνω) που καλείται msg.payload. Για παράδειγμα στην παρακάτω εικόνα φαίνεται ένας mqtt in node που ”ακούει” στο topic/lights στον οποίο έχουμε συνδέσει ένα debug node για να δούμε την έξοδό του. Στη συνέχεια, η κατάσταση της λάμπας του δωματίου που ορίζει το μήνυμα πρέπει να αποθηκευτεί στη βάση δεδομένων. Γι’ αυτό το λόγο το json string περνάει μέσα από ένα json node, ο οποίος το μετατρέπει σε ένα json object. Πλεόν μπορούμε να έχουμε πρόσβαση στα διάφορα πεδία του μηνύματος. Τέλος, το json αυτό object στέλνεται στον κόμβο Light Insert, ο οποίος είναι ένας PostgreSQL κόμβος μέσω του οποίου έχουμε συνδέσει τη βάση δεδομένων μας. Ορίζοντας το κατάλληλο SQL query (Fig. 10), τα δεδομένα αποθηκεύονται επιτυχώς στη βάση. Ακολουθεί η οπτικοποίηση της διαδικασίας που μόλις περιγράφηκε:

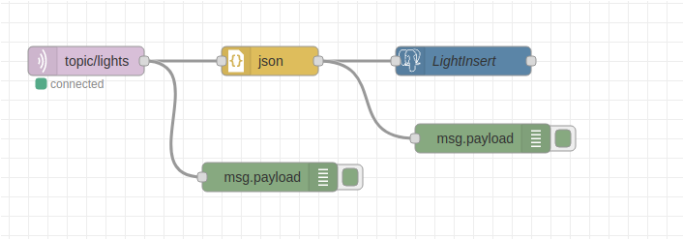

Fig. 7. Flow of Lights

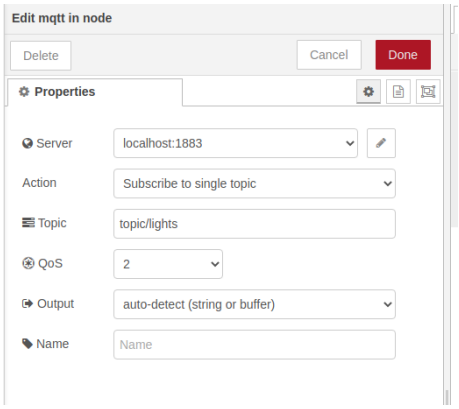

Fig. 8. MQTT node settings

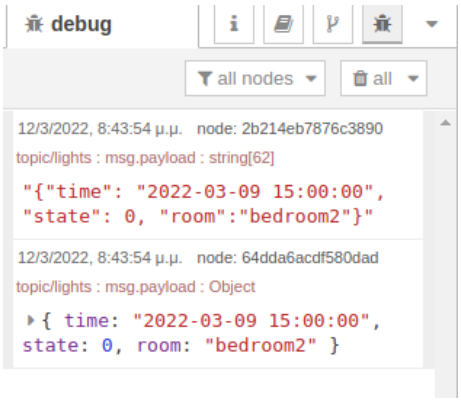

Fig. 9. Debug window output before and after json node

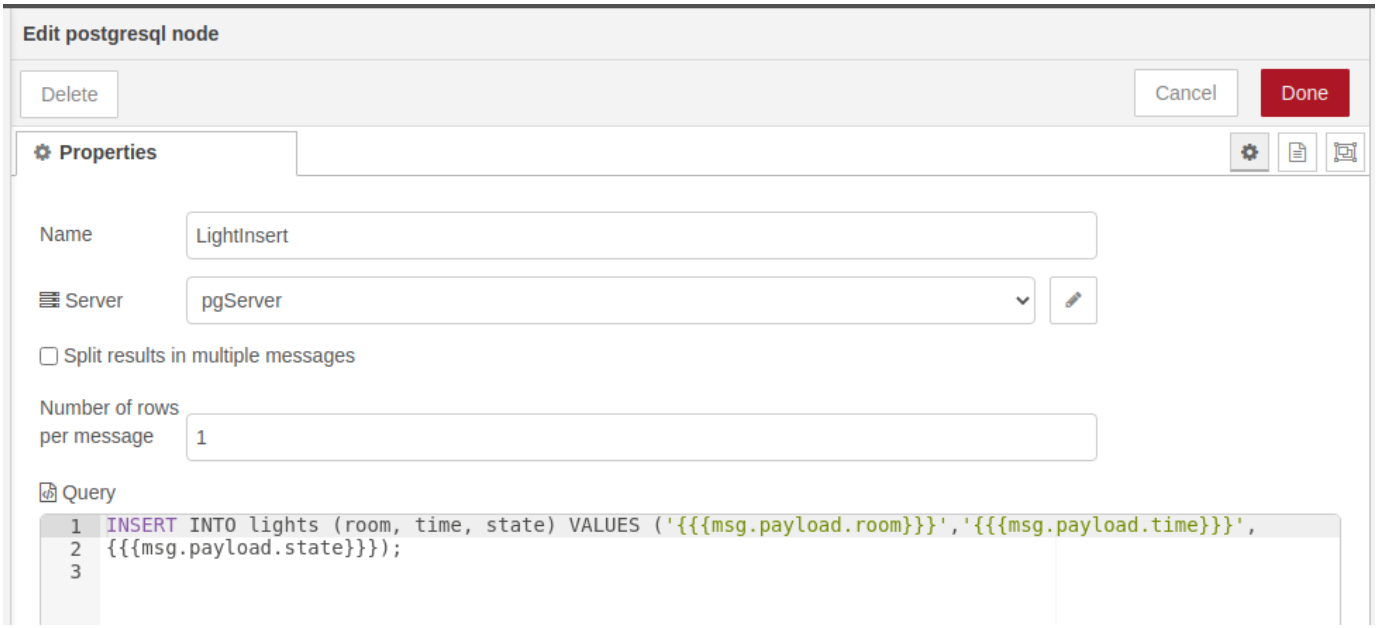

Fig. 10. Postgesql Insert from Node-Red

Σχετικά με την εικόνα 10, όπου παρουσιάζεται ένα τυπικό query εισαγωγής δεδομένων στη βάση, η μορφή των values είναι τέτοια ώστε να παίρνουμε από το msg.payload (json μήνυμα), το πεδίο που χρειαζόμαστε κάθε φορά.
Αυτή είναι και η βασική λογική όλων των καταστάσεων που στέλνονται μέσω των αισθητήρων, με τη διαφορά ότι αλλάζει το topic και τα πεδία μέσα στο json μήνυμα.
Ακολουθούν τα screenshots από κάθε flow (εκτός από το Lights Flow που παρουσιάστηκε) και μία συνοπτική περιγραφή τους. Τα δεδομένα από τους αισθητήρες θα φτάνουν ταυτόχρονα ανά 5 λεπτά:

- **Alarm**
  Στο flow αυτό ελέγχουμε εάν ενώ ήταν ενεργοποιημένος ο συναγερμός άνοιξε κάποιο παράθυρο, κάποια πόρτα ή ανιχνεύθηκε παρουσία. Αρχικά έχουμε τον subscriber που ακούει στο topic/alarm και ελέγχει εάν το πεδίο isopen του msg.payload ισούται με 1, δηλαδή είναι ανοικτός ο συναγερμός. Εάν δεν είναι, περιοριζόμαστε απλά στην αποθήκευση της κατάστασης του στη βασή. Διαφορετικά, δημιουργούμε ένα msg.topic με όνομα ”alarm” και το στέλνουμε σε όλους τους join κόμβους από τις υπόλοιπες καταστάσεις που εξετάζουμε. Κάθε subscriber από αισθητήρα παραθύρων, πορτών και παρουσίας σε περίπτωση που ανιχνύεσει δραστηριότητα στέλνει επίσης ένα msg.topic στον join κόμβο του. Σε κάθε join κόμβο είναι συνδεδεμένα 2 topics, ένα από την ανίχνευση ανοικτού συναγερμού και ένα από την ανίχνευση ανοικτού παραθύρου, πόρτας ή παρουσίας. Επομένως όταν ”φτάσουν” δύο τέτοια topics σημαίνει ότι κάτι συμβαίνει ενώ ήταν ανοικτός ο συναγερμός. Σε αυτή την περίπτωση στέλνεται μήνυμα στο κατάλλλο κανάλι του slack μέσω του slack κόμβου στο τέλος. Για την δημιουργία του μηνύματος προηγείται ένας function κόμβος που δημιουργεί μέσω javascript το επιθυμητό μήνυμα και το στέλνει στον κόμβο slack.
- **Gases**
  Στο flow αυτό ελέγχουμε εάν έχει ανιχνευθεί από τους αισθητήρες φωτιά ή επικίνδυνες τιμές από CO και CO2. Όπως και παραπάνω χρησιμοποιούμε switch κόμβους που ανάλογα με την κατάσταση έχουν διαφορετικές εξόδους. Σε αυτή την περίπτωση στέλνεται μήνυμα στο κατάλλλο κανάλι του slack μέσω του slack κόμβου στο τέλος.
- **Heating**
  Στο flow αυτό καταγράφεται και ελέγχεται η θερμοκρασία και σε περίπτωση που ξεπεράσει τους 30 °C ή πέσει κάτω από 5°C, ενώ υπάρχει κάποιος στο σπίτι στέλνεται προειδοποιητικό μήνυμα στο slack.
- **Devices**
  Σε αυτό το flow ελέγχεται και καταγράφεται εάν χτύπησε το κουδούνι, ανιχνεύθηκε ανοικτός ο φούρνος ή κάποια βρύση ενώ δεν βρίσκεται κάποιος στο σπίτι ή εάν η πόρτα του ψηγείου έμεινε ανοικτή για αρκετή ώρα. Ομοίως με τις παραπάνω περιπτώσεις στέλνεται μήνυμα στο κατάλλληλο κανάλι του slack. Επίσης καταγράφεται η κατάσταση της τηλεόρασης, του ραδιοφώνου και της μηχανής καφέ.
  Για να ελέγξουμε εάν η πόρτα του ψυγείου ήταν ανοικτή για αρκετή ώρα ελέγχουμε την τιμή που στέλνει εκείνη τη στιγμή ο αισθητήρας του ψυγείου με την τελευταία τιμή που είχε αποθηκευτεί στη βάση δεδομένων. Τα δεδομένα από τους αισθητήρες (μέσω των python scripts) αποστέλονται κάθε 5 λεπτά επομένως ένας τέτοιος έλεγχος εξυπηρετεί τον σκοπό μας. Έχει προστεθεί

ένας κόμβος καθυστέρησης κατά 1 second, ώστε να τραβήξουμε το τελευταίο γεγονός της βάσης πριν αποθηκευτεί το νέο.
Για να ελέγξουμε εάν είναι ανοικτός ο φούρνος ή κάποια βρύση ενώ λείπουμε ελέγχουμε εάν υπάρχει στο table της βάσης δεδομένων που περιέχει την ύπαρξη παρουσίας ή όχι εάν τότε δεν υπάρχει κανένας στο σπίτι.

- **Energy**
  Στο flow αυτό ελέγχεται η θερμοκρασία και η ποσότητα πετρελαίου του oil boiler, καθώς και η κατανάλωση ρεύματος συναλοκά σε όλο το σπίτι.

Τα σχετικά screenshots παρουσιάζονται στο τέλος της αναφοράς.

Για την αποστολή μηνυμάτων μέσω Slack, χρησιμοποιήθηκε το

E. Μηνύματα *Slack*

Για την αποστολή μηνυμάτων από το Node Red στο Slack, χρησιμοποιήθηκε το integration Incoming WebHooks, το οποίο προστέθηκε σε όλα τα κανάλια. Ουσιαστικά παράχθηκε ένα link για κάθε κανάλι, το οποίο το εισάγαμε στα αντίστοιχα slack nodes στο Node Red. Επομένως, κάθε μήνυμα που θα στέλνεται στα slack nodes, θα εμφανίζεται ως μήνυμα και στα αντίστοιχα κανάλια του Slack. Ακολουθούν ενδεικτικά screenshots από τα προειδοποιητικά μηνύματα που αναφέρθηκαν παραπάνω στο Slack.

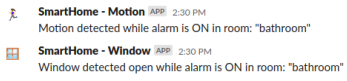

Fig. 11. Alarm Channel

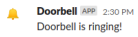

Fig. 12. Entrance Channel

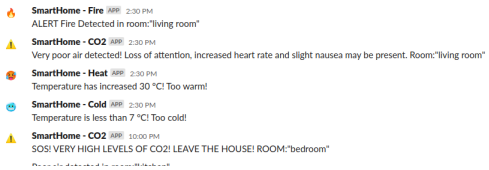

Fig. 13. Gases Channel

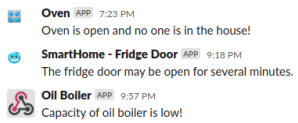

Fig. 14. Home Devices Channel

F. Granafa

Το εργαλείο Grafana χρησιμοποιήθηκε για την αναπαράσταση των γραφικών παραστάσεων και για την δυνατότητα στον χρήστη να χειρίζεται remote ορισμένες συσκευές του, όπως τα φώτα, την τηλεόραση κ.α. Για να ξεκινήσουμε τον server (τοπικά) γράφουμε σε ένα terminal την εντολή sudo systemctl start grafana-server και πληκτρολογούμε στον browser τη διεύθυνση http://localhost:3000/. Θα μας εμφανιστεί ένα παράθυρο αντίστοιχο με το παρακάτω:

Όπως φαίνεται και στην εικόνα 15, στο αριστερό μέρος υπάρχουν ορισμένα κάποια dashboards. Έχουμε δημιουργήσει τα ίδια dashboards με τα flows στο Node-Red. Μέσα σε κάθε dashboard απεικονίζουμε γραφικές παραστάσεις των καταστάσεων των αισθητήρων συναρτήσει του χρόνου και έχουμε δημιουργήσει κουμπιά για την διαχείριση ορισμένων συσκευών.
Για τις παραπάνω λειτουργίες συνδέσαμε πρώτα τη βάση δεδομένων μας στο Grafana. Για το σκοπό αυτό πλοηγηθήκαμε στο Configuration → Data Sources → Add Data Source, επιλέξαμε το Data Source PostgreSQL και ορίσαμε τις κατάλληλες μεταβλητές.

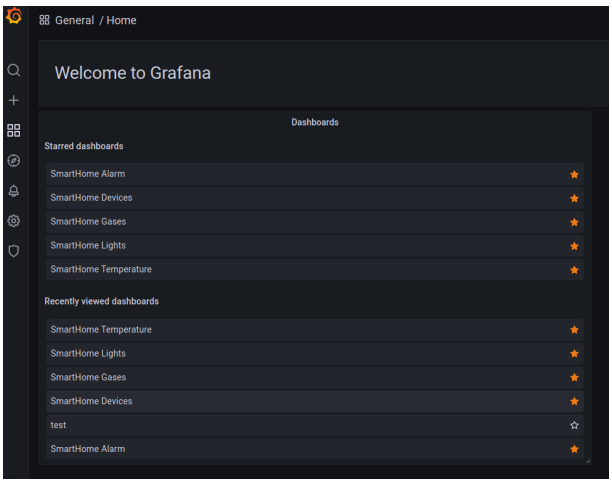

Fig. 15. Grafana Dashboard

Επιλέγουμε σε ένα dashboard το Add New Panel και μας εμφανίζεται το παραπάνω περιβάλλον. Παρατηρούμε ότι έχει ήδη δημιουργηθεί ένα query builder. Φυσικά μπορούμε να χρησιμοποιήσουμε την επιλογή Edit sql, ώστε να γράψουμε εμείς ένα query. Το ένα από τα columns που θα επιλέξουμε θα πρέπει να έχει δεδομένα σε μορφή timestamp, τα οποία θα χρησιμοποιηθούν ως άξονας του χρόνου. Επιπλέον μπορούμε να ορίσουμε το χρονικό διάστημα που επιθυμούμε και τον τύπο της αναπαράστασης. Στη συνέχεια παρουσιάζονται οι γραφικές παραστάσεις από κάθε dashboard.

1) Γραφική Αναπαράσταση των Δεδομένων*:*

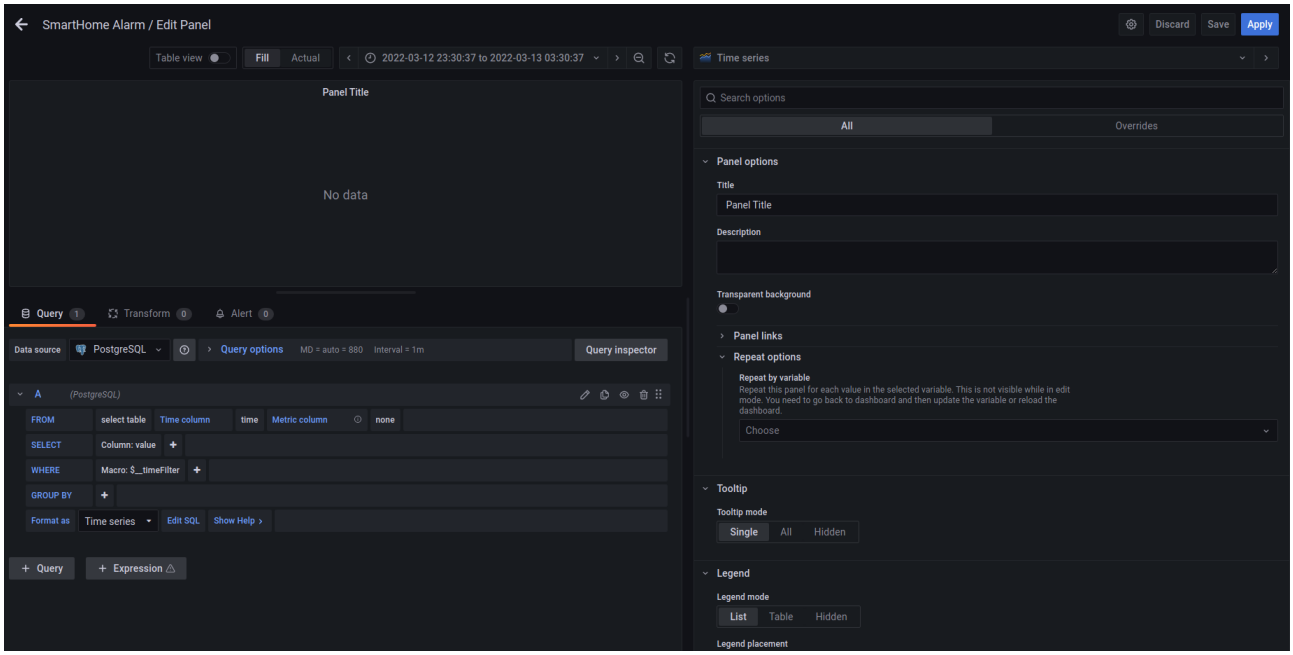

Fig. 16. Grafana New Panel

Fig. 17. Temperature Dashboard

Fig. 18. Alarm Dashboard - Alarm State - Presense at Home

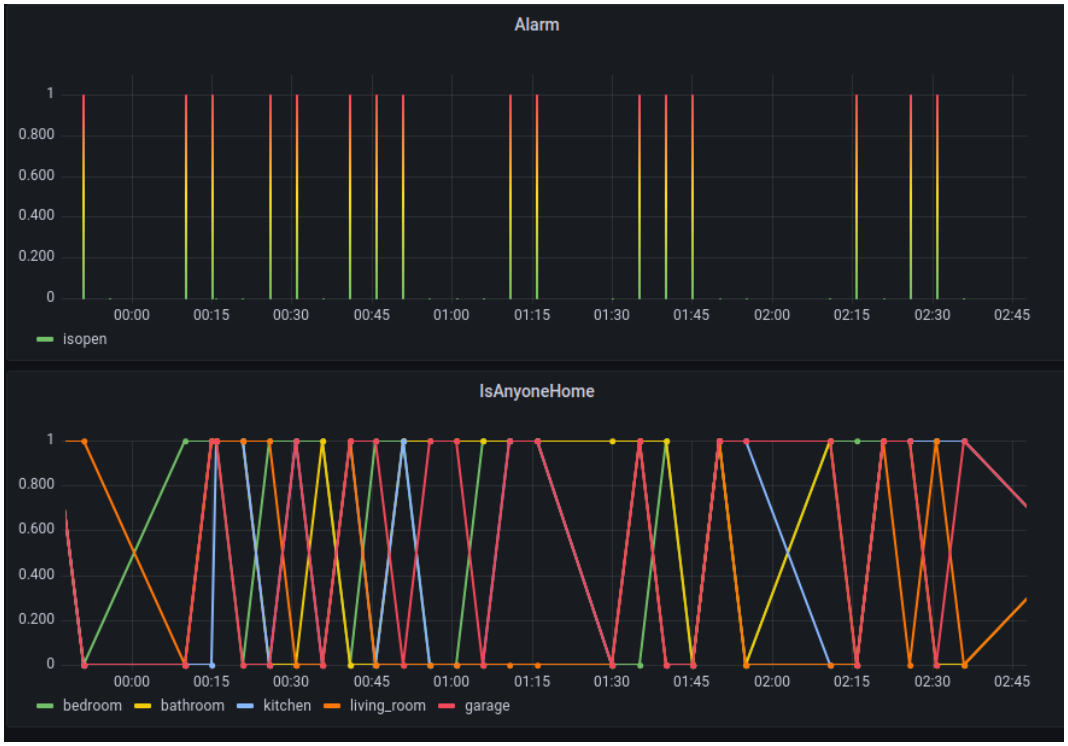

Fig. 19. Alarm Dashboard - Alarm State - Presense at Home

Fig. 20. Gases Dashboard - CO

Fig. 21. Gases Dashboard - CO2

Fig. 22. Gases Dashboard - Fire

Fig. 23. Devices Dashboard - Fridgedoor, Doorbell

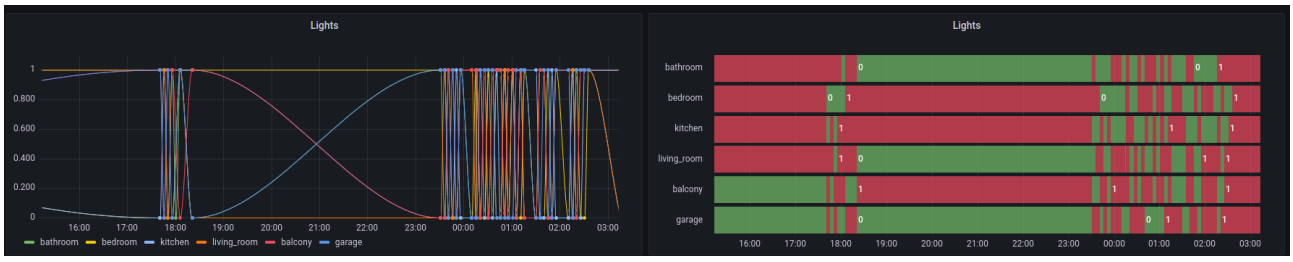

Fig. 24. Lights Dashboard

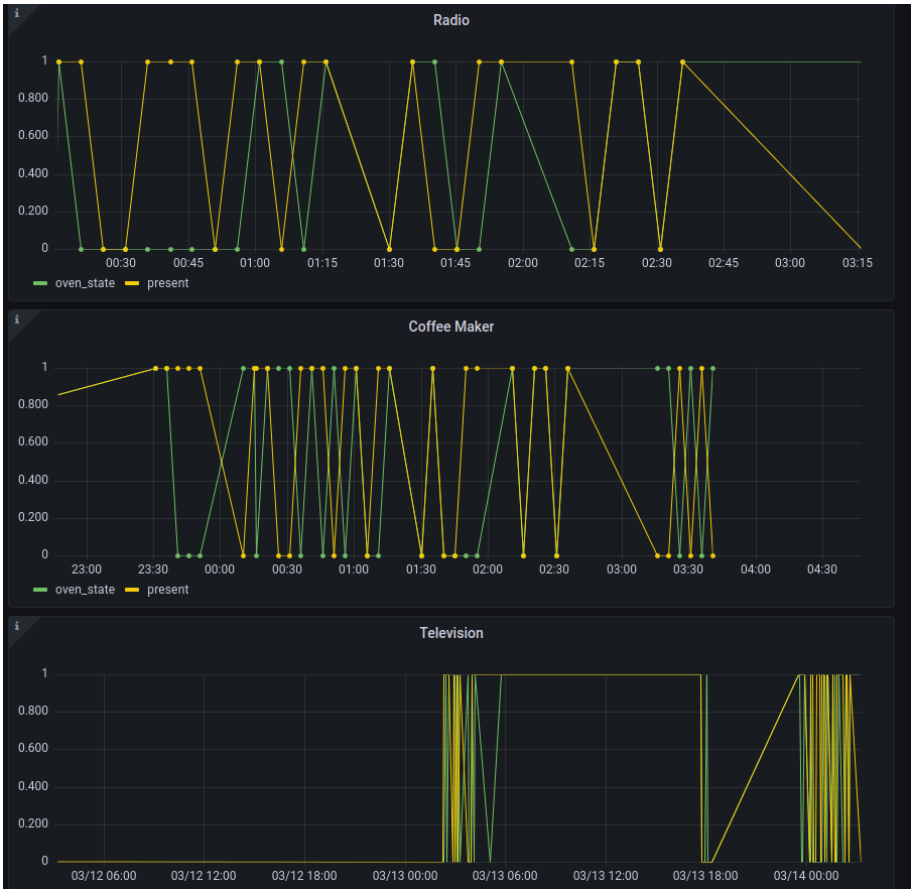

Fig. 25. Devices Dashboard - Television, Radio, Coffee Maker

Fig. 26. Devices Dashboard - Oven, Faucet

2) Διαχείριση Συσκευών*:* Το τελευταίο μέρος του συστήματός μας είναι η δυνατότητα ενός χρήστη να αλλάζει την κατάσταση του συναγερμού, του φούρνου, των φώτων, της θερμοκρασίας, της τηλεόρασης, του ραδιοφώνου και της μηχανής του καφέ. Για να το πετύχουμε αυτό δημιουργήσαμε στα αντίστοιχα dashboards κουμπιά ON/OFF. Τα κουμπιά αυτά δημιουργήθηκαν με δύο τρόπους. Ο πρώτος ήταν η επιλογή Text στο Visualizations ενός νέου Panel και εισαγωγή HTML στο κατάλληλο πεδίο και ο δεύτερος η επιλογή Button Panel. Το Button Panel χρησιμοποιήθηκε για την περίπτωση αλλαγής θερμοκαρίας, ώστε ο χρήστης να στέλνει και δεδομένα. Και στις δύο περιπτώσεις υλοποίησης τα requests που στέλνονται είναι HTTP GET και POST requests. Μέσω του Granafa δεν γινόταν να στείλουμε mqtt μηνύματα κατευθείαν στο Node-Red. Γι' αυτό αναπτύξαμε ένα απλό API με χρήση Node.js, το οποίο λαμβάνει τα requests και αντιστοίχως στέλνει MQTT μηνύματα στα διάφορα topics με τις επιλογές του χρήστη. Αναφορικά με τα κουμπιά διαχείρισης της θερμοκρασίας, δίνουμε την δυνατότητα ο χρήστης να κλείνει τον θερμοστάτη και να τον ανοίγει αφού ορίσει την επιθυμητή θερμοκρασία μέσω της μεταβλητής Thermostat στο πεδίο Edit variable value. Ακολουθεί η παρουσίαση των διαθέσιμων κουμπιών του συστήματός μας.

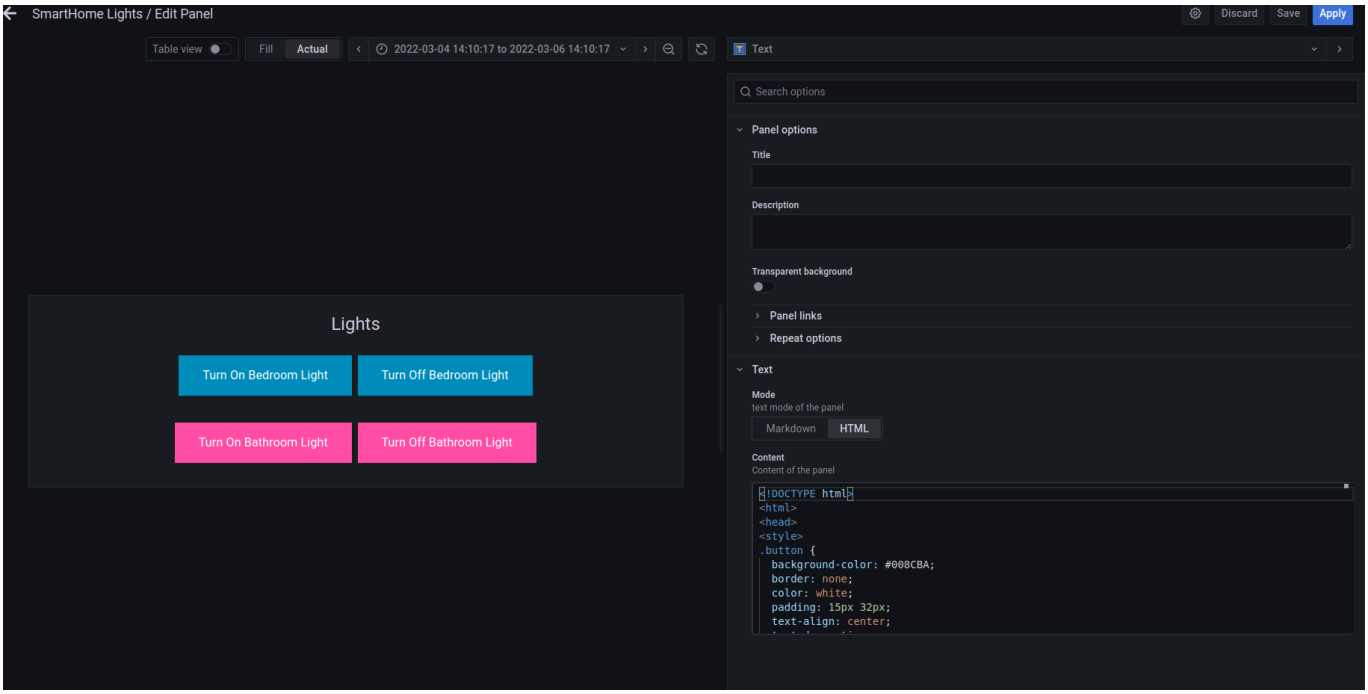

Fig. 27. Grafana Button Example

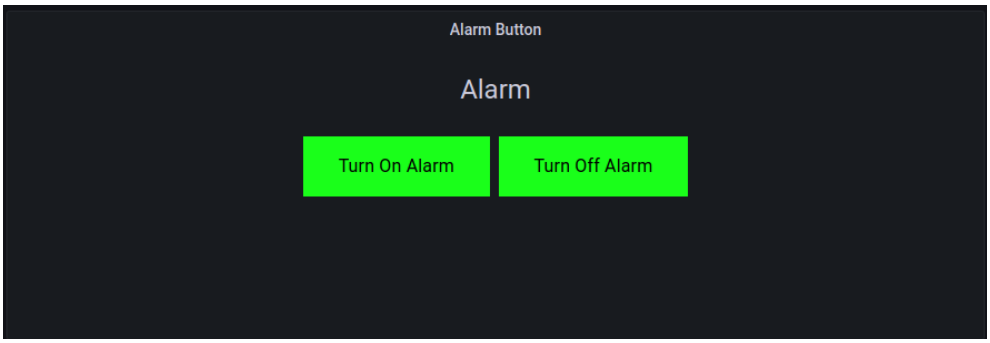

Fig. 28. Grafana Buttons Alarm

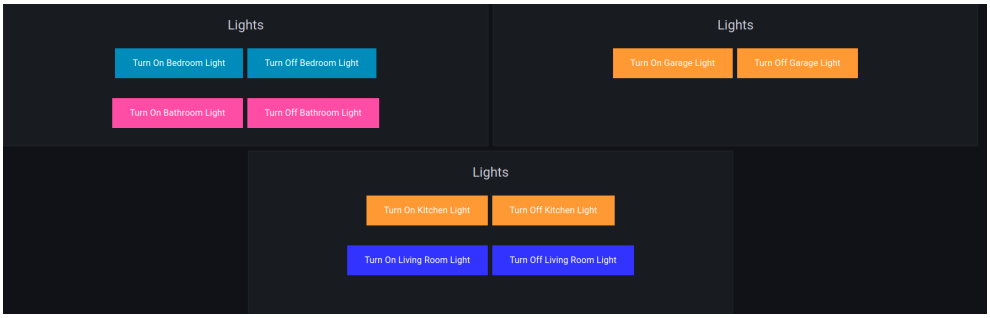

Fig. 29. Grafana Buttons Lights

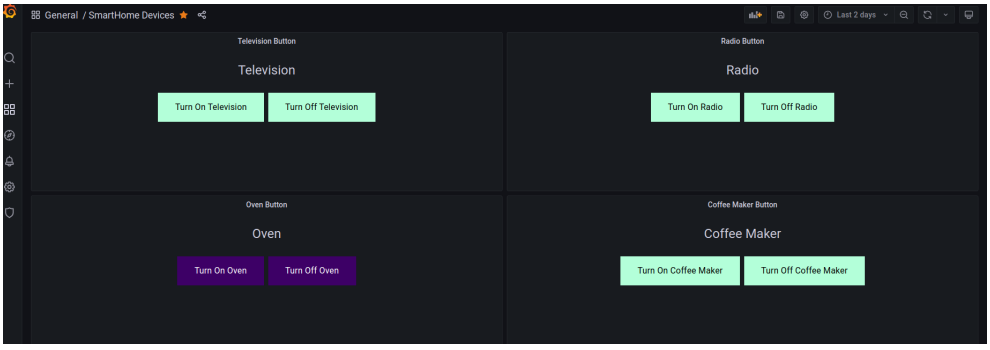

Fig. 30. Grafana Oven, Television, Radio Buttons

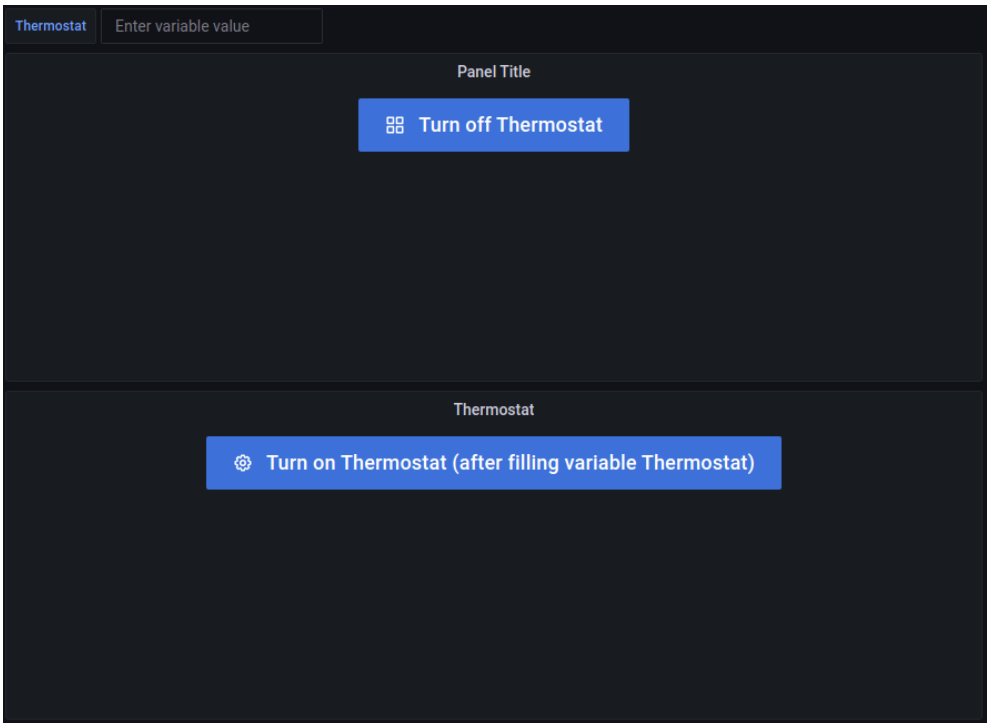

Fig. 31. Grafana Temperature Buttons

Ο κώδικας υλοποίησης του παραπάνω project βρίσκεται στο ακόλουθο Github Repository: https://github.com/nafsika24/ece_smarthome_system

References

[1] https://www.kane.co.uk/knowledge-centre/what-are-safe-levels-of-co-and-co2-in-rooms
[2] https://www.kane.co.uk/knowledge-centre/what-are-safe-levels-of-co-and-co2-in-rooms
[3] https://www.designairinc.com/blog/the-temperature-in-your-boiler-and-the-dangers-of-overheating/
[4] https://electricityplans.com/kwh-kilowatt-hour-can-power/

Node Red Flows:

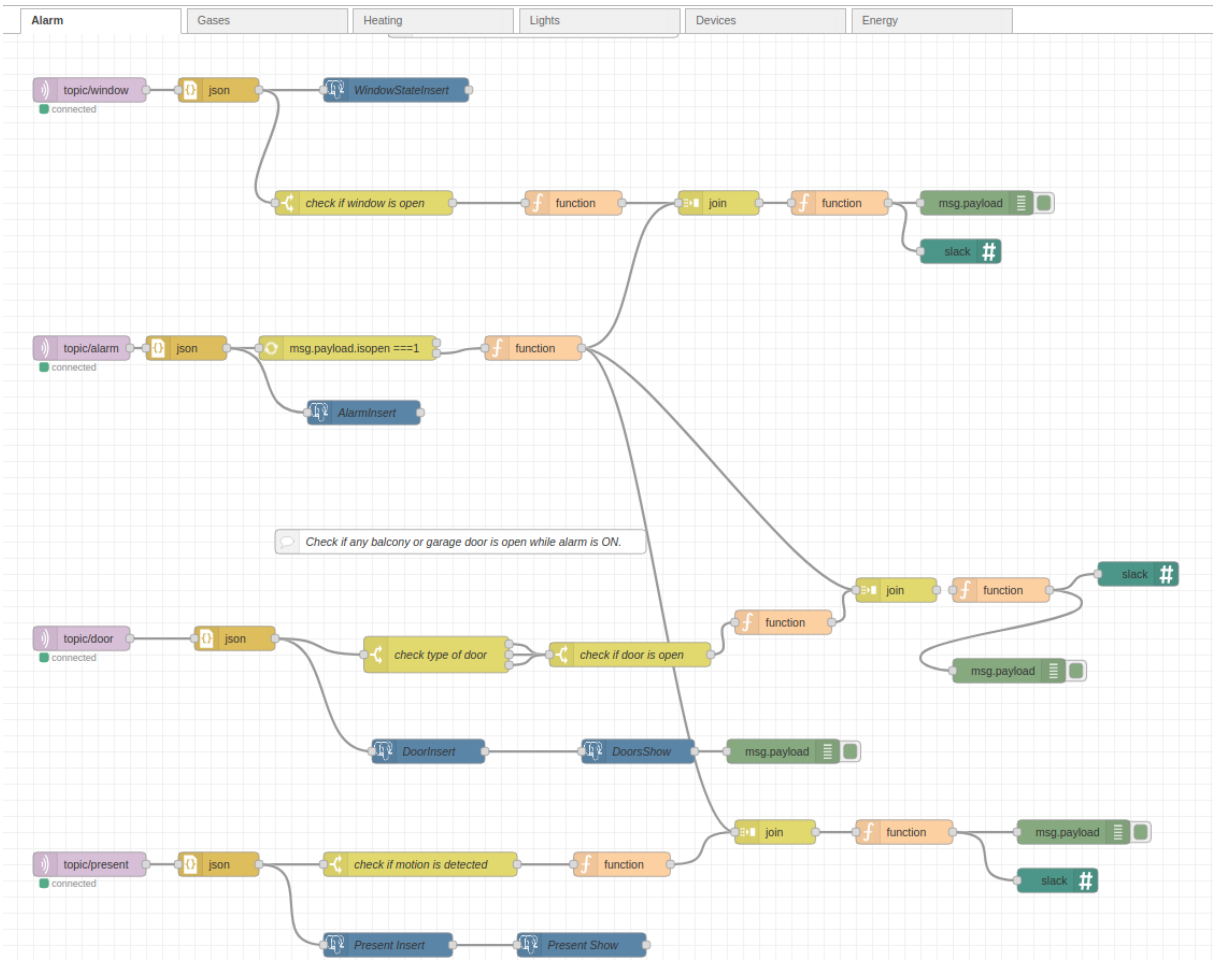

Fig. 32. Alarm Flow

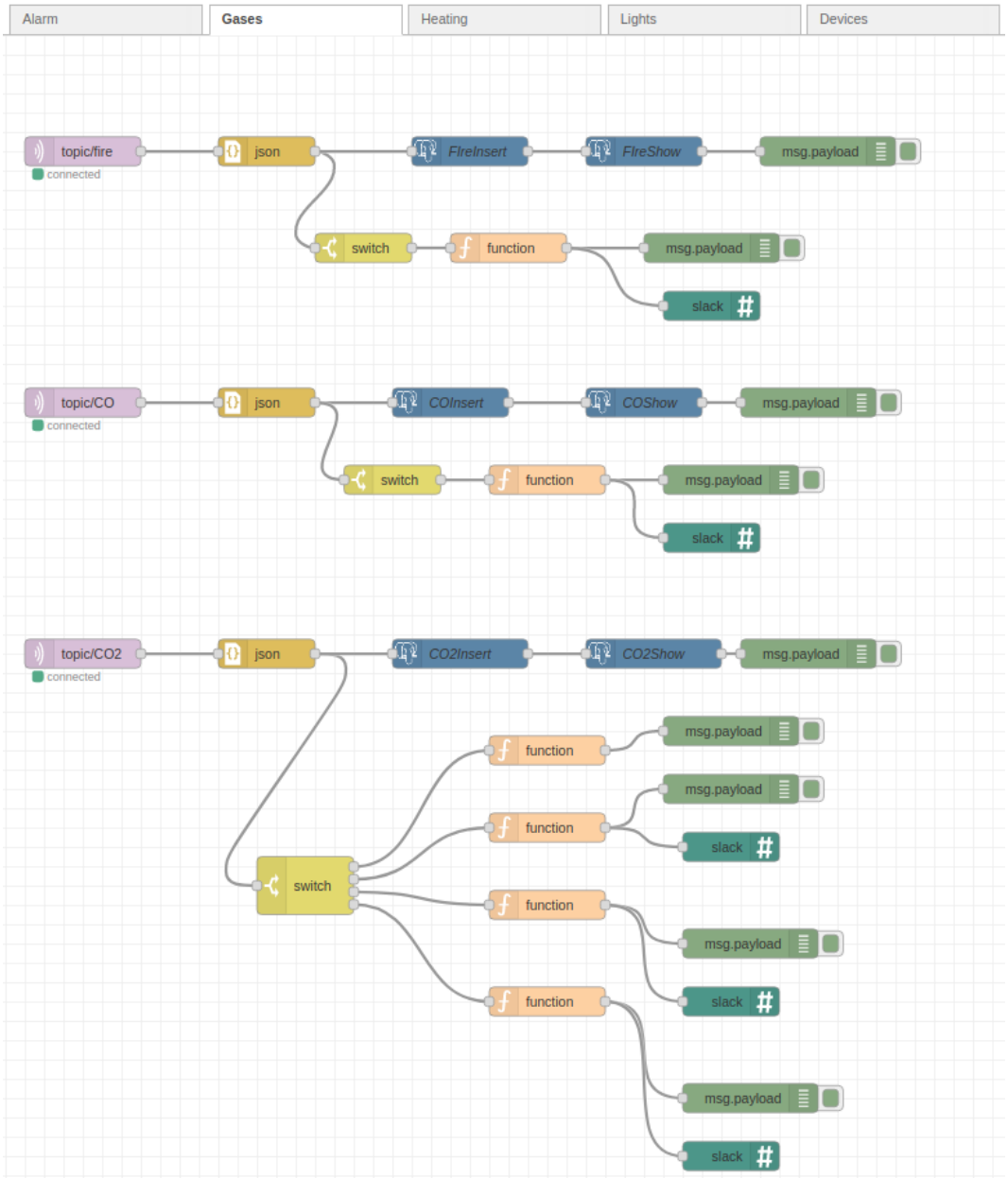

Fig. 33. Gases Flow

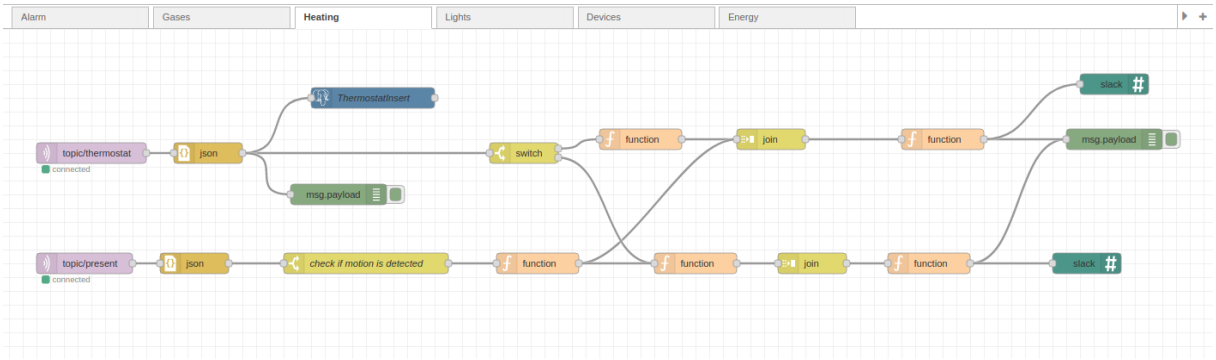

Fig. 34. Heating Flow

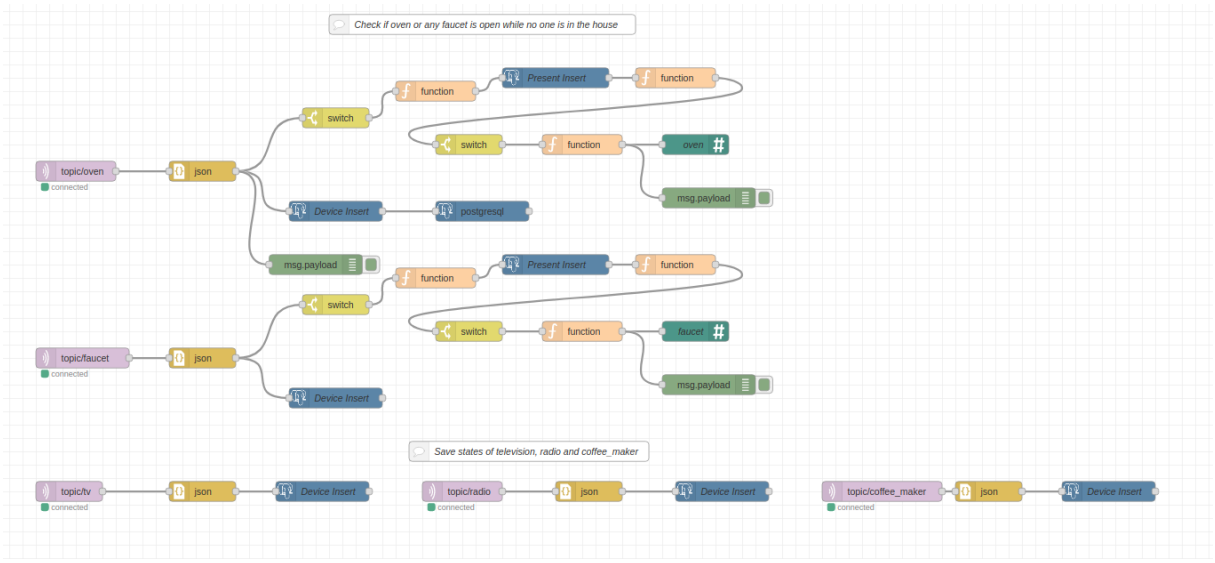

Fig. 35. Devices Flow

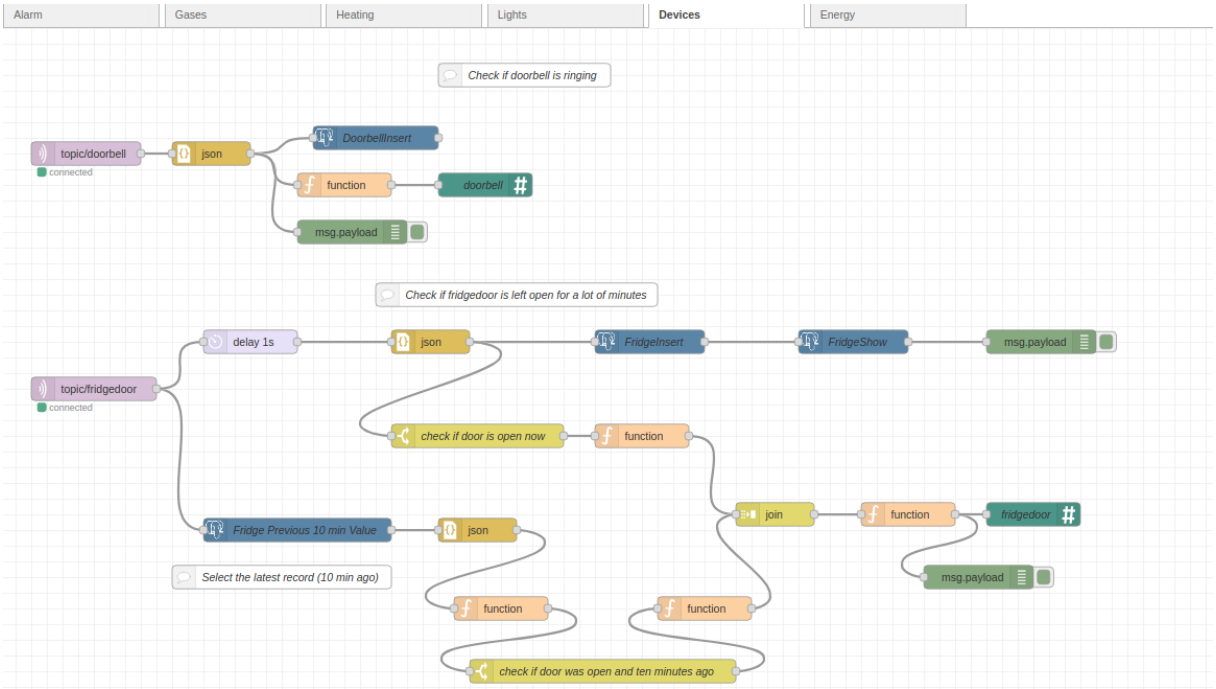

Fig. 36. Devices Flow

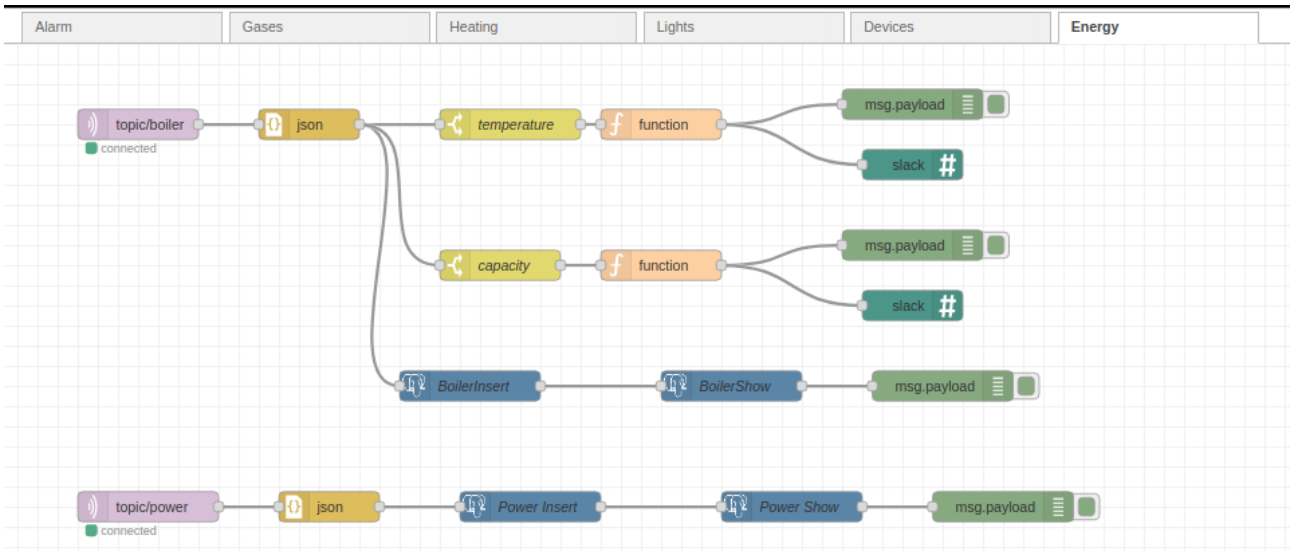

Fig. 37. Energy Flow