



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΜΕΛΛΟΝΤΙΚΟΥ ΔΙΑΔΙΚΤΥΟΥ ΜΕ ΧΡΗΣΗ ΤΕΧΝΟΛΟΓΙΩΝ HTML5

ΕΥΣΤΑΘΙΑ ΓΙΑΛΟΥΡΗ
ΑΜ. 4661

ΕΠΙΒΛΕΠΩΝ: ΚΑΘΗΓΗΤΗΣ ΧΡΗΣΤΟΣ ΖΑΡΟΛΙΑΓΚΗΣ
ΣΥΝΕΠΙΒΛΕΠΩΝ: ΔΡ. ΙΩΑΝΝΗΣ ΧΑΤΖΗΓΙΑΝΝΑΚΗΣ

ΠΑΤΡΑ, ΙΟΥΝΙΟΣ 2014

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Χρήστο Ζαρολιάγκη, καθηγητή του τμήματος Μηχανικών Η/Υ και Πληροφορικής για την εμπιστοσύνη που μου έδειξε με την ανάθεση της διπλωματικής εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω τον Δρ. Ιωάννη Χατζηγιαννάκη για την επίβλεψη και καθοδήγηση καθ'ολη τη διάρκεια εκπόνησης της εργασίας μου όπως και τον μεταπτυχιακό φοιτητή Δημήτριο Αμαξηλάτη για τη βοήθεια που μου προσέφερε και την αστείρευτη υπομονή του. Ακόμη, θέλω να ευχαριστήσω την ομάδα του P-Space που με βοήθησε να χρησιμοποιήσω το χώρο ως πλατφόρμα δοκιμών των εφαρμογών που υλοποιήθηκαν.

Τέλος, ένα θερμό ευχαριστώ στη μητέρα μου για τη στήριξή της σε κάθε βήμα της ζωής μου και φυσικά στους φίλους μου που αποτελούν αναπόσπαστο κομμάτι της κάθε μορφής επιτυχία της ζωής μου.

Περιεχόμενα

1. Εισαγωγή	
1.1 Κίνητρο και Σημασία	7
1.2 Στόχος	7
1.3 Συνεισφορά	8
1.4 Δομή	8
2. Υπηρεσίες Σημασιολογικού Ιστού για το Διαδίκτυο των Αντικειμένων	
2.1 Πρωτόκολλο Εφαρμογής για Συσκευές Περιορισμένων Πόρων (COAP)	9
2.2 Σύστημα Διαχείρισης Δεδομένων Αισθητήρων (Überdust)	11
2.3 Εφαρμογή – P-space	12
3. Τεχνολογίες	
3.1 HTML 5	17
3.2 Websockets	18
3.3 Highcharts	20
3.4 Wunderground	21
3.5 Εφαρμογή μέσω ιστοσελίδας	22
4. On{X}	
4.1 Τι είναι το On{X}	32
4.2 Το λειτουργικό σύστημα Android	34
4.3 Οδηγός για δημιουργία εφαρμογών	36
4.4 On{x} και Uberdust	38
4.5 Υλοποίηση νέων εφαρμογών	39
I. Έλεγχος φωτισμού όταν το κινητό περιστρέφεται	39
II. Ενεργοποίηση Η/Υ χρήστη όταν ο χρήστης πλησιάζει στο χώρο εργασίας του και αντίστροφα.	41
III. Ενεργοποίηση κλιματισμού όταν ο χρήστης πλησιάζει το χώρο εργασίας του αν καιρικές συνθήκες το απαιτούν	43

5. Επέκταση περιηγητή Chrome	
5.1 Εισαγωγή	47
5.2 Εφαρμογή μέσω επέκτασης περιηγητή	47
6. Drupal	
6.1 Τί είναι το Drupal	54
6.2 Τι είναι το CMS (Content Management System)	55
6.3 Ιστότοπος Uberdust Testbed – CTI Πάτρας	56
6.3.1 Γενικά	57
6.3.2 Power Sensor Module	57
6.3.3 Chart Module	69
6.3.4 Weather Module	73
Μελλοντικές Κατευθύνσεις	76
Βιβλιογραφία	77

1 Εισαγωγή

1.1 Κίνητρο και Σημασία

Η ραγδαία εξέλιξη των τεχνολογιών που σχετίζονται με τον Παγκόσμιο Ιστό και το διαδίκτυο γενικότερα, έχει δημιουργήσει νέες προοπτικές στον τρόπο χρήσης του και τις δυνατότητες που προσφέρει. Η HTML5 έφερε μια μικρή επανάσταση στο τρόπο που απεικονίζουμε το περιεχόμενο στο διαδύκτιο και παράλληλα με τις νέες τεχνολογίες που εισήγαγε (CSS3), στο τρόπο που αλληλεπιδρούμε με αυτό.

Ταυτόχρονα με τις εξελίξεις στο πεδίο του Παγκόσμιου ιστού, συνέπεσε και η αύξηση του ερευνητικού ενδιαφέροντος όσον αναφορά τα δίκτυα ασύρματων αισθητήρων που έχουν αμέτρητες χρήσεις στην καθημερινή ζωή. Το γεγονός ότι η συνδεσιμότητα με το Internet είναι πλέον μια αυτονόητη πραγματικότητα, φέρνει το Internet of Things όλο και πιο κοντάς στην καθημερινή μας ζωή. Ένα δίκτυο δηλαδή όπου όλα τα αντικείμενα και οι συσκευές γύρω μας είναι συνδεδεμένα στο διαδύκτιο και "δουλεύουν" για τον άνθρωπο.

Τέλος, η χρήση έξυπνων προσωπικών συσκευών, όπως smartphones και tablets στην καθημερινή μας ζωή συνεχώς αυξάνεται. Ενδεικτικό είναι πως υπολογίζεται ότι τον Ιούνιο του 2013 το 61% των Αμερικανών πολιτών είναι χρήστες smartphone.

Όλα αυτά λοιπόν ευνοούν στη δημιουργία ετερογενών συστημάτων, συστημάτων δηλαδή που θα χρησιμοποιούν περισσότερα από ένα είδη συσκευών, τα οποία θα συνδέονται μεταξύ τους μέσω του Παγκόσμιου Ιστού.

1.2 Στόχος

Η παρούσα διπλωματική εργασία έχει ως στόχο τη μελέτη τεχνολογιών μελλοντικού διαδικτύου και την σύνδεση τους με την έννοια του Internet of Things. Διότι η χρησιμότητα των έξυπνων συσκευών στη ζωή μας θα αυξηθεί όταν τα μέσα που χρησιμοποιούμε για να αλληλεπιδρούμε με αυτές είναι προσβάσιμα απο όλους.

1.3 Συνεισφορά

Στα πλαίσια της συγκεκριμένης διπλωματικής αναπτύθηκαν εφαρμογές που αλληλεπιδρούν με το συστημα αποθήκευσης και διαχείρισης δεδομένων σε ένα δίκτυο αισθητήρων (Uberdust). Ως αποτέλεσμα προέκυψε μια φιλικότερη προς το χρήστη οπτικοποίηση των δεδομένων σε σχέση με τους μέχρι τώρα τρόπους παρουσίασης τους. Επίσης υλοποιήθηκαν εφαρμογές που αξιοποιούν τις λειτουργίες των έξυπνων κινητών τηλεφώνων μέσω της πλατφόρμας On{x} με σκοπό να συνδέσουν το Uberdust με τις καθημερινές δραστηριότητες των χρηστών του. Τέλος, υλοποιήθηκαν δύο ειδών δικτυακές εφαρμογές για έλεγχο και παρακολούθηση των συσκευών που διαχειρίζεται το Uberdust.

1.4 Δομή

Στο κεφάλαιο 2 περιγράφεται η αρχιτεκτονική του συστήματος με το οποίο αλληλεπιδρούν οι εφαρμογές που αναπτύχθηκαν.

Στο κεφάλαιο 3 παρουσιάζονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη των εφαρμογών της παρούσας διπλωματικής εργασίας και η υλοποίηση ο ελεγκτής συσκευών του Uberdust σε μορφή εφαρμογής ιστοσελίδας.

Στο κεφάλαιο 4 παρουσιάζεται η πλατφόρμα On{x} οι εφαρμογές που υλοποιήθηκαν σ' αυτή καθώς και οδηγίες για να αναπτύξει κάποιος τις δικές του εφαρμογές.

Στο κεφάλαιο 5 παρουσιάζεται ο ελεγκτής συσκευών του Uberdust και η υλοποίηση του σε μορφή επέκτασης περιηγητή.

Στο κεφάλαιο 6 παρουσιάζεται ο ιστότοπος σε Drupal του συστήματος Uberdust για την πλατφόρμα δοκιμών του ΙΤΥΕ Διόφαντος.

2 Υπηρεσίες Σημασιολογικού Ιστού για το Διαδίκτυο των Αντικειμένων

2.1 Πρωτόκολλο Εφαρμογής για Συσκευές Περιορισμένων Πόρων (COAP)

Η χρήση των web services σε εφαρμογές διαδικτύου είναι πλέον ένα σύνηθες φαινόμενο και βασίζονται στην θεμελιώδη Representation State Transfer (REST) αρχιτεκτονική του παγκόσμιου ιστού. Η ομάδα του Constrained RESTful Environments (CoRE) [1] στοχεύει στον σχεδιασμό μιας REST αρχιτεκτονικής ιδανικής για συσκευές περιορισμένων δυνατοτήτων, όπως 8-bit microcontrollers με περιορισμένη μνήμη RAM, ROM και δυνατότητες δικτύωσης (πχ 6LoWPAN). Δίκτυα όπως το 6LoWPAN υποστηρίζουν τον κατακερματισμό IPv6 πακέτων, κάτι που κρίνεται δαπανηρό σε πόρους. Ένας σχεδιαστικός στόχος ήταν να δημιουργηθεί ένα πρωτόκολλο που θα έχει μικρό overhead ώστε να περιορίζεται όσο είναι δυνατόν η ανάγκη για κατακερματισμό των πακέτων.

Το αποτέλεσμα της ομάδας του CoRE ήταν το Constrained Application Protocol (CoAP)[2]. Ο κύριος σχεδιαστικός στόχος ήταν ο σχεδιασμός ενός πρωτοκόλλου για τον παγκόσμιο ιστό για τις συγκεκριμένες ανάγκες του περιορισμένου περιβάλλοντος και συσκευών, ιδιαίτερα τις ανάγκες για περιορισμένη κατανάλωση, αυτοματοποίησης κτηρίων και άλλες machine-to-machine (M2M) εφαρμογές. Φέρει υπηρεσίες παρόμοιες του HTTP σε συσκευές με περιορισμένες υπολογιστικές ικανότητες, το οποίο κρίθηκε ιδανικό στην περίπτωση των ασύρματων δικτύων αισθητήρων. Με την υλοποίηση και χρήση του CoAP επιτυγχάνεται ο επιθυμητός διαχωρισμός, της τελικής εφαρμογής, από το πρωτόκολλο επικοινωνίας και το πρωτόκολλο δικτύου. Πλέον οι εφαρμογές τρέχουν πάνω από το επίπεδο του CoAP αγνοώντας τις ιδιαιτερότητες του επιπέδου του δικτύου.

Το μοντέλο επικοινωνίας βασίζεται στο γενικότερο μοντέλο του πελάτη/διακομιστή

(client/server) του HTTP [3]. Μία βασική διαφορά είναι ότι έχει σχεδιαστεί και για μηχανή προς μηχανή επικοινωνία (machine to machine – M2M) όπου μία συσκευή μπορεί να δρα ως πελάτης και ως διακομιστής. Παρόμοια με το HTTP, μία αίτηση (request) στέλνεται προς έναν διακομιστή, αναφερόμενη σε ένα αντικείμενο, με μία συγκεκριμένη μέθοδο αίτησης (request method). Η απάντηση (response) περιλαμβάνει συχνά την αναπαράσταση του αντικειμένου που ζητήθηκε και τον κωδικό απάντησης (response code).

Σε αντίθεση με το HTTP η επικοινωνία είναι ασύγχρονη, χωρίς την δημιουργία κάποιας σύνδεσης και βασίζεται στο UDP που δεν παρέχει τρόπους αξιόπιστης μετάδοσης μηνυμάτων όπως στο TCP. Παρέχονται όμως ειδικά μηνύματα και τρόποι ελέγχου από το CoAP, ώστε να γίνει η επικοινωνία αξιόπιστη εφόσον αυτό είναι επιθυμητό από την εκάστοτε εφαρμογή. Ορίζονται 4 τύποι μηνυμάτων: Confirmable (CON), Non-Confirmable (NON), Acknowledgement (ACK), Reset (RST) και σε συνδυασμό με την μέθοδο αίτησης ή τον κωδικό απάντησης που περιέχεται σε κάποιους από αυτούς του τύπους μηνυμάτων, δηλώνουν αν πρόκειται για αίτηση ή για απάντηση. Υποστηρίζει αιτήσεις προς multicast IP διευθύνσεις καθώς και διάφορους τρόπους προστασίας και ασφάλισης της επικοινωνίας. Παράλληλα με το CoAP, αναπτύσσονται διάφορες επεκτάσεις του πρωτοκόλλου που προσθέτουν επιπλέον δυνατότητες και χαρακτηριστικά σε αυτό.

Αναλυτικά οι **μέθοδοι αίτησης** :

GET: Επιστρέφει το αντικείμενο που ζητήθηκε μέσω του URI της αίτησης. Αν η αίτηση περιέχει ένα ή περισσότερα Accept option, δηλώνει ποια Content-Type προτιμώνται στην απάντηση. Αν υπάρχει ETag option τότε αυτό επικυρώνεται και μόνο σε περίπτωση αποτυχίας αποστέλλεται το αντικείμενο. Σε περίπτωση επιτυχία επιστρέφει 2.05 – Content ή 2.03 Valid.

POST: Στέλνεται ένα αντικείμενο στον διακομιστή και πρέπει να επεξεργαστεί από αυτόν. Η ακριβής λειτουργία εξαρτάται από τον διακομιστή αλλά συνήθως ένα αντικείμενό του ανανεώνεται ή δημιουργείται. Στην 1η περίπτωση η απάντηση είναι 2.04 – Changed, ενώ στην 2η 2.01 – Created και στο URI του αντικειμένου που δημιουργήθηκε στο Location-Path option της απάντησης.

PUT: Στέλνεται ένα αντικείμενο προς ένα URI και ο τύπος του δηλώνεται με το Content-Type option. Επιστρέφει 2.01 – Created σε περίπτωση δημιουργίας νέου αντικειμένου και 2.04 – Changed σε περίπτωση ανανέωσης, ενώ στην αποτυχία επιστρέφει το ανάλογο σφάλμα.

DELETE: Το αντικείμενο που ορίζεται από το URI θα σβηστεί και η μέθοδος θα επιστρέψει 2.02 – Deleted σε περίπτωση επιτυχίας ή αν το αντικείμενο δεν υπήρχε εξ αρχής.

2.2 Σύστημα Διαχείρισης Δεδομένων Αισθητήρων (Überdust)

Το Überdust [4] είναι ένα σύστημα αποθήκευσης και διαμοιρασμού πληροφοριών μέσω web services. Η αλληλεπίδραση με το σύστημα γίνεται ανεξαρτήτου πλατφόρμας, με τους χρήστες να μπορούν να επικοινωνούν με αυτό μέσω REST και WebSockets [5]. Τα δεδομένα αποθηκεύονται σε μία βάση δεδομένων χρησιμοποιώντας το WiseDB. Η έξοδος του συστήματος μπορεί να είναι σε διάφορους τύπους δεδομένων, όπως απλό κείμενο, HTML, JSON, RDF κα. Σχεδιάστηκε με σκοπό να συλλέγει πληροφορίες από ασύρματα δίκτυα αισθητήρων, αλλά μπορεί να δεχτεί πληροφορίες από οποιαδήποτε πηγή. Είναι υλοποιημένο σε Java κάνοντας χρήση τεχνολογιών όπως Spring, Hibernate, JSP & JSTL κα.

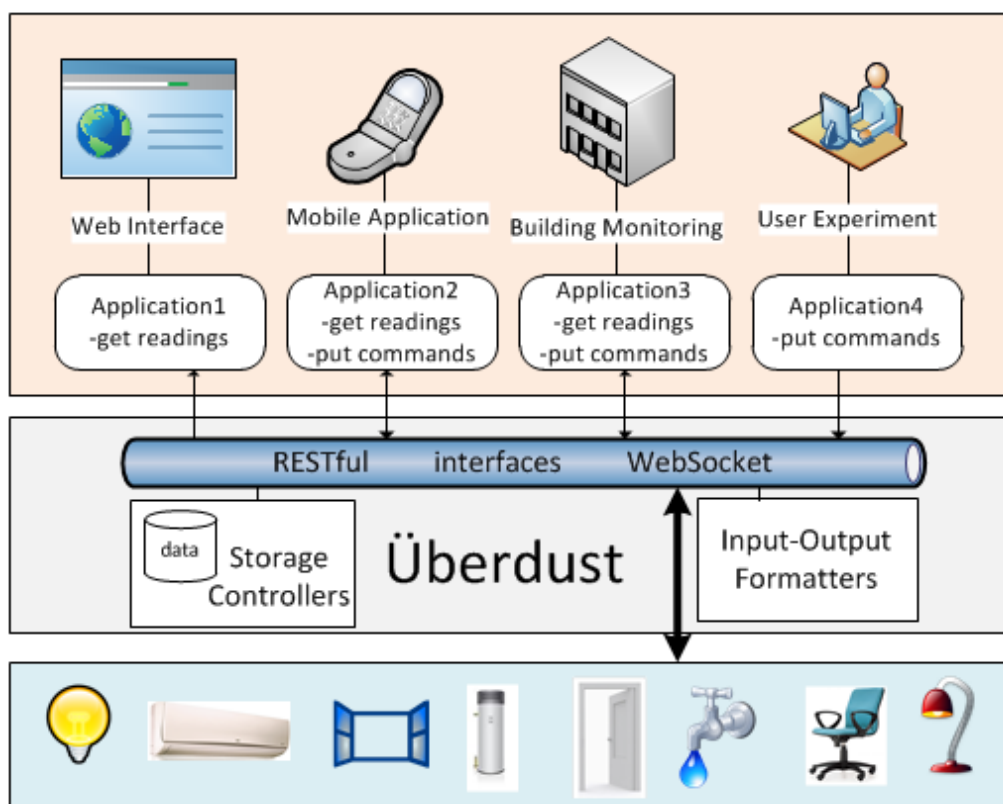
Οι συσκευές του δικτύου αισθητήρων έχουν προγραμματιστεί με ειδικές εφαρμογές, οι οποίες παρέχονται μαζί με το Überdust, μέσω του NodeFlasher που αναλαμβάνει αυτή την διαδικασία. Οι εφαρμογές αυτές κάνουν broadcast περιοδικά τις τιμές όλων των αισθητήρων τους στο δίκτυο. Κάποιοι κεντρικοί κόμβοι που έχουν οριστεί ως πύλες (Gateways) συλλέγουν αυτές τις πληροφορίες και τις τυπώνουν στην σειριακή θύρα. Οι πύλες είναι συνδεδεμένες με το TestbedRuntime, το οποίο διαβάζει τις πληροφορίες που τυπώνουν οι συσκευές. Από εκεί και έπειτα το Überdust μέσω του TestbedListener συλλέγει τις πληροφορίες, τις μορφοποιεί και τις αποθηκεύει κατάλληλα στην βάση δεδομένων του. Η πρόσβαση σε αυτές γίνεται μέσω της Web εφαρμογής του.

Ο σχεδιασμός του επιτρέπει την δημιουργία πρόσθετων εφαρμογών που λειτουργούν ως πελάτες της Web εφαρμογής. Χρησιμοποιώντας τις πληροφορίες που συλλέγονται και είναι αποθηκευμένες στην βάση δεδομένων, μπορούν να υλοποιήσουν διάφορες αυτόματες διαδικασίες που ενεργούν ανάλογα με τα δεδομένα. Παράδειγμα τέτοιας εφαρμογής είναι η διαχείριση του φωτισμού των γραφείων, όπου αν εντοπιστεί κίνηση ανάβουν τα φώτα, ενώ αν για ένα μεγάλο διάστημα δεν υπάρχει κίνηση αυτά σβήνουν.

2.3 Εφαρμογή – P-space

Το P-Space^[6] είναι ένα hackerspace στην Πάτρα, το οποίο χρησιμοποιήθηκε από φοιτητές του τμήματος Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Πατρών, στα πλαίσια του μαθήματος 'Καταμημένα Συστήματα II' με σκοπό την εφαρμογή τεχνολογιών Internet of Things στις καθημερινές ανάγκες των χρηστών του χώρου. Σχεδιάστηκαν και κατασκευάστηκαν 'έξυπνα' αντικείμενα βασισμένα στην πλακέτα ανοιχτού κώδικα Arduino. Για την επίτευξη ασύρματης επικοινωνίας των αντικειμένων ενσωματώθηκαν XBee 802.15.4 modules στις arduino πλακέτες. Επίσης χρησιμοποιήθηκαν ATmega328 μικροελεγκτές για τον έλεγχο συσκευών που εγκαταστήθηκαν σε αντικείμενα όπως γραφεία, λάμπες, πόρτες και βρύσες. Τα έξυπνα αντικείμενα που προέκυψαν είναι ασύρματα συνδεδεμένα μεταξύ τους, είναι χαμηλού κόστους και είναι συμβατά με μία μεγάλη ομάδα αισθητήρων και ενεργοποιητών.

Η επικοινωνία με τις συσκευές για την διαχείριση των μετρήσεων των αισθητήρων καθώς και η αποστολή εντολών για την ενεργοποίηση/απενεργοποίηση τους επιτυγχάνεται με τη χρήση του πρωτοκόλλου CoAP που περιγράφηκε στην προηγούμενη παράγραφο. Συνδυάζεται το πρωτόκολλο CoAP με το σύστημα διαχείρισης ασύρματων δικτύων αισθητήρων Uberrust, παρέχοντας αποθήκευση, διαμοιρασμό πραγματικού χρόνου μετρήσεων αλλά και ιστορικών δεδομένων.



Εικόνα 2.1 Αρχιτεκτονική Συστήματος

Το σύστημα σχεδιάστηκε σε τρία ξεχωριστά επίπεδα τα οποία επικοινωνούν μεταξύ τους ανταλλάσσοντας μηνύματα που περιέχουν εντολές ή μετρήσεις αισθητήρων. Η εικόνα 2.1 δείχνει τα τρία αυτά ανεξάρτητα επίπεδα και την επικοινωνία τους μέσω RESTful υπηρεσιών και Websockets.

Το κατώτατο επίπεδο περιλαμβάνει το hardware που σχεδιάστηκε και εφαρμόστηκε στα αντικείμενα του P-Space. Αυτό το επίπεδο παρέχει τις μετρήσεις των αισθητήρων ασύρματα σε μια κεντρική συσκευή ελέγχου.

Το Uberdust αποτελεί το κεντρικό επίπεδο του συστήματος καθώς είναι υπεύθυνο για την αμφίδρομη επικοινωνία των συσκευών χαμηλού επιπέδου με τις IoT εφαρμογές.

Στην κορυφή βρίσκεται το επίπεδο IoT εφαρμογών που είναι υπεύθυνο για τη συλλογή των μετρήσεων των αισθητήρων καθώς και για τον έλεγχο των ενεργοποιητών. Οι περισσότεροι χρήστες δεν απαιτείται να αλληλεπιδρούν απευθείας με τις συσκευές γι' αυτό και είναι επαρκείς οι δυνατότητες που δίνονται σ' αυτό το επίπεδο αφαίρεσης. Αυτή η αφαιρετική σκοπιά δίνει τη δυνατότητα ομαδοποίησης των συσκευών σύμφωνα με τα κύρια χαρακτηριστικά τους σε μια οντότητα που ονομάζεται Virtual node (εικονικός κόμβος). Αυτή μπορεί να περιλαμβάνει συσκευές που βρίσκονται στην ίδια περιοχή όπως γραφεία ή ορόφους ή συσκευές που έχουν σχέση ως προς τη χρήση τους. Επίσης, σ' αυτό το επίπεδο περιλαμβάνονται εφαρμογές που δεν αποσκοπούν στην απευθείας αλληλεπίδραση με το χρήστη όπως εφαρμογές που ελέγχουν τα φώτα σύμφωνα με τη χωρητικότητα του δωματίου/γραφείου.



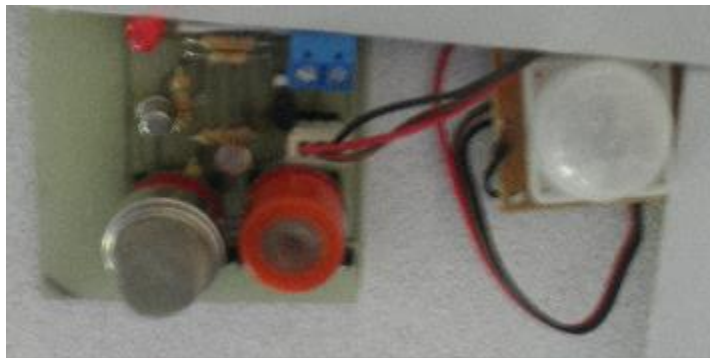
Εικόνα 2.4 Κάτοψη P-Space

Προκειμένου να ενσωματωθούν οι βασικές ηλεκτρικές και ηλεκτρονικές συσκευές στο δίκτυο, σχεδιάστηκαν πλακέτες που υποστηρίζουν ασύρματα επικοινωνία, είναι χαμηλού κόστους και υποστηρίζονται από την κοινότητα ανοιχτού κώδικα.

Έξυπνες Λάμπες

Οι πλακέτες που σχεδιάστηκαν εφάρμόστηκαν σε όλες τις λάμπες γραφείου και οροφής σε κάθε δωμάτιο του P-Space. Για τον έλεγχο των 230V AC λαμπών από τα 5V I/O pins χρησιμοποιήθηκαν ρελέ. Κάθε πλακέτα είναι ικανή να ελέγχει έως δύο διαφορετικές λάμπες έτσι ώστε να παρέχεται στον χρήστη η δυνατότητα να έχει πολλαπλά επίπεδα φωτισμού, για παράδειγμα χαμηλότερη ένταση φωτισμού μια ηλιόλουστη μέρα και υψηλότερη μία συννεφιασμένη. Επιπλέον, για την περίπτωση αποτυχίας του υλικού υπάρχει η δυνατότητα για χειροκίνητη παράκαμψη.

Οι πλακέτες που ενσωματώθηκαν στις λάμπες είναι εξοπλισμένες με αισθητήρες για μέτρηση της φωτεινότητας, θερμοκρασίας και κίνησης (μέσω PIR αισθητήρων) όπως φαίνεται στην εικόνα 2.2. Αυτές οι μετρήσεις στέλνονται περιοδικά στο Uberdust και αξιοποιούνται από εφαρμογή που τρέχει στο άνω επίπεδο. Η εφαρμογή συνδυάζει αυτές τις τιμές για να ληφθεί απόφαση για το αν πρέπει να ανάψουν ή να σβήσουν τα φώτα ώστε να παρέχονται καλύτερες συνθήκες εργασίας. Επίσης, για να επιτευχθεί η μείωση της κατανάλωσης της ενέργειας της συσκευής όταν δεν υπάρχει ανθρώπινη παρουσία στον χώρο οι λάμπες σβήνουν.



Εικόνα 2.2 Πλακέτα έξυπνης λάμπας

Έξυπνο HVAC

Ο ελεγκτής HVAC κατασκευάστηκε με Arduino για να αντικαταστήσει το 38KHz υπέρυθρο(IR) ελεγκτή που χρησιμοποιούταν(Toyotomi brand). Γι'αυτό το σκοπό

αναπτύχθηκε μια βιβλιοθήκη λογισμικού που επιτρέπει τον έλεγχο μεγάλου εύρους HVAC συσκευών που είναι συμβατές με ένα συγκεκριμένο IR πρωτόκολο που χρησιμοποιείται κατά κόρον στο συγκεκριμένο πεδίο. Η βιβλιοθήκη παρέχει μια προγραμματιστική διεπαφή (API) που υποκαθιστά όλες τις συναρτήσεις που υποστηρίζονται από το R51L1/BGE απομακρυσμένο ελεγκτή που χρησιμοποιείται κυρίως από την Toyota. Επιπλέον, το API παρέχει συναρτήσεις που επιτρέπουν αποστολή σειριακής εισόδου διαμορφωμένη στα 38KHz.



Εικόνα 2.3 Πλακέτα ελέγχου HVAC

Αισθητήρες Ασφαλείας

Ένα ακόμη σημαντικό μέρος του συστήματος είναι οι αισθητήρες που καταγράφουν την κατάσταση των θυρών και των παραθύρων (ανοιχτά/κλειστά). Για να γίνει αντιληπτό το αν τα παράθυρα είναι ανοιχτά χρησιμοποιούνται οι απλοί και ευρέως δοκιμασμένοι αισθητήρες που χρησιμοποιούνται στα συστήματα συναγερμών. Μια λειτουργία που υποστηρίζεται είναι η ενημέρωση του χρήστη μέσω SMS, email, τηλεφωνικές κλήσεις με ηχογραφημένο μήνυμα ακόμα και Skype ή Desktop ενημερώσεις. Άλλη μία σημαντική λειτουργία που υποστηρίζεται είναι η απενεργοποίηση του κλιματισμού όταν ανιχνεύονται ανοιχτά παράθυρα.

Έξυπνη βρύση

Αυτή η πλακέτα παρέχει τον έλεγχο της παροχής νερού με σκοπό την αποφυγή σπατάλης αλλά και την ανίχνευση κάποιας διαρροής. Η πλακέτα που σχεδιάστηκε λειτουργεί σε τρία επίπεδα. Τοπικά, έχει τη δυνατότητα να ελέγχει μία βαλβίδα ώστε να παρέχει νερό στο νυπτήριο του μπάνιου. Ενα δεύτερο τμήμα της πλακέτας εντοπίζει την υπερχειλίση μέσω ενός απλού κυκλώματος που βραχυκυκλώνει όταν βυθιστεί στο νερό. Αν εντοπιστεί αυτό, η παροχή νερού σταματάει για να αποφευχθούν προβλήματα των ηλεκτρικών εγκαταστάσεων και ζημιές στο κτήριο ή στα έπιπλα. Τέλος, χρησιμοποιείται αισθητήρας που ανιχνεύει την κίνηση σε κοντινή απόσταση ώστε να αρχίζει η παροχή νερού όταν κάποιος πλησιάσει τα χέρια του.

3 Τεχνολογίες

3.1 HTML5

Η HTML5 [7] είναι μία γλώσσα περιγραφής και παρουσίασης περιεχομένου στον Παγκόσμιο Ιστό και αποτελεί την πέμπτη έκδοση της HTML. Η προηγούμενη έκδοση, η οποία χρησιμοποιούμε ακόμα σήμερα, βγήκε το 1999. Από τότε όμως έχουν αλλάξει πολλά στον παγκόσμιο ιστό, και έτσι η ανάγκη για standards και πλήρη συμβατότητα οδήγησε το World Wide Web Consortium (W3C) και το Web Hypertext Application Technology Working Group (WHATWG) να συνεργαστούν και να δημιουργήσουν την νέα αυτή γλώσσα. Η HTML5 είναι ακόμα υπό ανάπτυξη, αλλά μερικοί browsers υποστηρίζουν από τώρα κάποια χαρακτηριστικά της και κάποιες λειτουργίες της.

Κάποιοι βασικοί κανόνες που έχουν οριστεί για την HTML5 είναι:

- Βάση για τα νέα χαρακτηριστικά να είναι οι HTML, CSS, DOM, και η JavaScript.
- Ελαχιστοποίηση των plugins (όπως το Flash).
- Καλύτερη λειτουργία εντοπισμού λαθών.
- Περισσότερο markup για να αντικατασταθεί το scripting.
- Πλήρη συμβατότητα ανεξαρτήτως συσκευής.

Κάποια από τα νέα χαρακτηριστικά της HTML5 είναι:

- Το στοιχείο canvas για το drawing
- Τα στοιχεία video και audio για αναπαραγωγή πολυμέσων
- Νέα στοιχεία περιεχομένου όπως τα footer, header, nav και section
- Νέα στοιχεία δημιουργίας φόρμας όπως τα calendar, date, time, email, url και search

Στόχος της HTML5 είναι να βελτιώσει τη γλώσσα και να υποστηρίξει τα σύγχρονα πολυμέσα (ήχο, εικόνα, video), ενώ παραμένει ευανάγνωστη για τους ανθρώπους και αναπαρίσταται με συνέπεια από τους υπολογιστές.

3.2 Websockets

Τα WebSockets είναι ένα πρωτόκολλο που επιτρέπει αμφίδρομη επικοινωνία πάνω από μία TCP σύνδεση, και έχει προτυποποιηθεί από την IETF ως RFC 6455. Αποτελεί ένα ανεξάρτητο, βασισμένο στο TCP, πρωτόκολλο που η μόνη του σύνδεση με το HTTP είναι πως οι διακομιστές ιστού διαχειρίζονται το handshake του πρωτοκόλλου σαν αίτημα HTTP Upgrade. Η TCP θύρα που χρησιμοποιείται από τα websockets είναι η 80 που είναι η καθιερωμένη θύρα του HTTP πρωτοκόλλου.

Για να εγκατασταθεί μια σύνδεση πρέπει ο περιηγητής ιστού να στείλει ένα WebSocket handshake αίτημα και ο web server να απαντήσει με μία WebSocket handshake απάντηση, όπως φαίνεται στο ακόλουθο παράδειγμα:

```
GET /mychat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: chat
Sec-WebSocket-Version: 13
Origin: http://example.com
```

Εικόνα 3.1 WebSocket Αίτημα

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

Εικόνα 3.2 WebSocket Απάντηση

Παρόλο που αρχικά σχεδιάστηκε για χρήση από διακομιστές ιστού, και περιηγητές ιστού αντίστοιχα, μπορεί να χρησιμοποιηθεί σε οποιαδήποτε εφαρμογή που ακολουθεί το μοντέλο πελάτη-διακομιστή. Έτσι, προσφέρεται εύκολη, αμφίδρομη διαρκής επικοινωνία, για όσο η σύνδεση παραμένει ανοιχτή.

Πολύ σημαντικό πλεονέκτημα των WebSockets είναι η ευκολία που προσφέρουν στον προγραμματιστή στην ανάπτυξη ενός προγράμματος βασισμένου σε αυτά. Πρακτικά χρειάζεται να υλοποιηθούν 4 βασικές συναρτήσεις. Σε διάφορες υλοποιήσεις υπάρχουν και πιο εξειδικευμένες συναρτήσεις αλλά οι 4 βασικές είναι οι εξής:

- **onOpen()**: Εκτελείται όταν το WebSocket ανοίγει.
- **onClose()**: Εκτελείται όταν το WebSocket κλείνει.
- **onMessage()**: Εκτελείται όταν το WebSocket δεχθεί μήνυμα. Συνήθως υλοποιούνται 2 συναρτήσεις, η **onTextMessage()** που εκτελείται όταν το WebSocket δεχθεί μήνυμα σε μορφή String, και η **onBinaryMessage()** η οποία εκτελείται όταν το WebSocket δεχθεί μήνυμα που αποτελείται από Raw Data.
- **onError()**: Εκτελείται όταν συμβεί κάποιο σφάλμα.

3.3 Highcharts

Η βιβλιοθήκη Highcharts^[8], η οποία είναι γραμμένη σε Javascript , χρησιμοποιήθηκε για την προσθήκη διαδραστικών γραφημάτων. Ο λόγος για τον οποίο επιλέχθηκε ήταν γιατί παρείχε την δυνατότητα παρουσίασης δεδομένων πραγματικού χρόνου.

Γενικά, διαθέτει ποικιλία γραφημάτων όπως line, spline, area, areaspline, column, bar, pie, scatter, angular gauges κ.α. Επίσης επιτρέπει το συνδυασμό πολλών από αυτών σε ένα γραφήμα.

- **Συμβατότητα**

Τα γραφήματα που περιλαμβάνει είναι αναγνωρίσιμα από όλους τους φυλλομετρητές (mobile και desktop) συμπεριλαμβανομένων των iPhone/iPad και Internet Explorer σε εκδόσεις μετά την 6η.

- **Ανοιχτός κώδικας**

Ενα ακόμη χαρακτηριστικό της highcharts είναι το ότι δίνεται η δυνατότητα στον προγραμματιστή να κατεβάσει τον κώδικα και να τον προσαρμόσει στις δικές του ανάγκες .

- **Javascript**

Απαιτεί μόνο τη φόρτωση του highcharts.js αρχείου και τη jQuery η οποία πιθανότατα χρησιμοποιείται ήδη.

- **Σύνταξη**

Η επεξεργασία και προσαρμογή του εκάστοτε γραφήματος είναι ευκολή καθώς απαιτείται μόνο η διαμόρφωση του αντικειμένου *options* το οποίο αποτελείται από ένα σύνολο key – values τα οποία συνδέονται με ανω και κάτω τελείες, χωρίζονται με κόμμα και ομαδοποιούνται με αγκύλες .

- **Δυναμικά γραφήματα**

Μετά τη δημιουργία του γραφήματος, υπάρχει η δυνατότητα προσθήκης δεδομένων, αφαίρεσης των ήδη υπαρχόντων και μετατροπής τους. Σε συνδυασμό με τεχνολογίες όπως jQuery, MooTools ή Prototype's Ajax API προκύπτουν γραφήματα παρουσίασης live δεδομένων.

- **Εξαγωγή και εκτύπωση**

Ο χρήστης μπορεί να εξαγάγει το γράφημα σε μορφή png, jpg, pdf και svd είτε κάνοντας κλικ στο σχετικό κουμπί είτε μέσω του περιηγητή.

- **Εστίαση**

Εστιάζοντας σε συγκεκριμένο τμήμα του γραφήματος, μπορεί να εξεταστεί με μεγαλύτερη ευκολία το τμήμα που ενδιαφέρει περισσότερο το χρήστη. Εστίαση μπορεί να γίνει στον Χ ή στον Υ άξονα, ή και στους δύο.

3.4 Wunderground



Εικόνα 3.4 Λογότυπο Weather Underground

Η υπηρεσία Weather Underground παρέχει πραγματικού χρόνου πληροφορίες για τον καιρό παγκοσμίως. Παρέχει λεπτομερείς αναφορές για τον καιρό ανα τον κόσμο οι οποίες είναι προσβάσιμες κάνοντας API αιτήσεις με χρήση του πρωτοκόλλου HTTP. Τα δεδομένα επιστρέφονται σε μορφή JSON ή XML.

```

<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.5.1/jquery.min.js"></script>

<script> jQuery(document).ready(function($) {

    $.ajax({

        url :
"http://api.wunderground.com/api/Your_Key/geolookup/conditions/q/IA/Cedar_Rapids.json",

        dataType : "jsonp",

        success : function(parsed_json) {

            var location = parsed_json['location']['city'];

            var temp_f = parsed_json['current_observation']['temp_f'];

            alert("Current temperature in " + location + " is: " + temp_f);

        } });

});

</script>

```

Παράδειγμα χρήσης wunderground API σε *Javascript* με *jQuery*.

3.5 Εφαρμογή μέσω ιστοσελίδας

Ως συνέχεια της παραπάνω εφαρμογής, αναπτύχθηκε ιστοσελίδα μέσω της οποίας ο χρήστης αλληλεπιδρά με τις συσκευές που διαχειρίζεται το σύστημα Uberdust. Η σημαντική διαφορά το ότι πλέον μπορούν να επιλεγούν παραπάνω απο μία συσκευές. Επίσης, στην παρούσα εφαρμογή, ο χρήστης έχει τη δυνατότητα να αφαιρεί τις συσκευές που δεν επιθυμεί πλέον να παρακολουθεί.

Στην εκκίνηση της εφαρμογής καλείται ο EventListener DOMContentLoaded, ο οποίος ορίζει μία σειρά απο EventListeners οι οποίοι χειρίζονται τα γεγονότα που πρόκειται να προκληθούν:

```
document.addEventListener('DOMContentLoaded', function () {  
    init();  
    document.querySelector('img').addEventListener('click', change);  
    document.querySelector('#testbed').addEventListener('change', getNodes);  
    document.querySelector('#nodes').addEventListener('change', getCapabilities);  
    document.querySelector('#addtolist').addEventListener('click', addtolist);  
});
```

Επίσης, καλεί τη συνάρτηση init() η οποία ελέγχει το αν ο χρήστης είναι συνδεδεμένος στο σύστημα έτσι ώστε να έχει πρόσβαση στις διαθέσιμες συσκευές:

```
function init() {  
  
    if(typeof(window.localStorage.getItem("server"))== "undefined" ||  
window.localStorage.getItem("server") == null){  
        window.localStorage.setItem("server","uberdust.cti.gr:80");  
    }else {  
        var inputServer=window.localStorage.getItem("server").split("/:");  
        if (inputServer[0]=="http"){  
            window.localStorage.setItem("server",inputServer[1]);  
        }  
    }  
    $("#inputServer").val(window.localStorage.getItem("server"));  
  
    $.ajax({  
        dataType: "jsonp",  
        url: "http://" + getUberdustServer() + "/rest/username",  
        jsonpCallback: 'getUsername'  
    });  
  
    $('#server').text('Server: '+window.localStorage.getItem("server"));  
    $("#loginform").attr("action","http://"+getUberdustServer()+"/j_spring_security_check");  
    $("#logout-  
btn").prop("href","http://"+getUberdustServer()+"/j_spring_security_logout");  
}
```

```
        populateMedia();
        getTestbeds();
        setDateTime();
    }
```

Όπως φαίνεται παραπάνω, η συνάρτηση `init()` καλεί τη συνάρτηση `populateMedia()` η οποία ανακτά απο τη μνήμη του browser τις αποθηκευμένες συσκευές του χρήστη:

```
function populateMedia() {
    var nodes = null;
    try {
        nodes = JSON.parse(window.localStorage.getItem("nodes"));
    } catch (err) {}
    window.nodes = nodes;
    if (nodes === null || nodes.constructor !== Array) {
        nodes = new Array();
    }
    if (nodes.length > 0) {
        $("#collapseOne").removeClass("in");
    }
    var i = 0;
    populate(i, nodes);
    var url = "";
    $.get(url, function (data) {
        var nodes = data.split("\n");
        var node = document.getElementById("nodes");
        if (testbed.value == "0") {
            node.options[0] = new Option("Select Node:", "0");
            node.options.length = 1;
        } else {
            node.options[0] = new Option("Select Node:", "0");
            for (var i = 0; i < nodes.length; i++) {
                node.options[i + 1] = new Option(nodes[i], nodes[i]);
            }
            node.options.length = nodes.length;
        }
    })
}
```

Στην παραπάνω συνάρτηση καλείται η συνάρτηση `populate()` με όρισμα την πρώτη αποθηκευμένη συσκευή με σκοπό να την προβάλλει. Η `populate()` καλεί αναδρομικά τον εαυτό της εως ότου εμφανιστούν όλες οι συσκευές του χρήστη. Η λειτουργία αυτής

της συνάρτησης είναι να διαμορφώσει την παρουσίαση των συσκευών και να ανοίξει της συνδέσεις με το websocket :

```
function populate(i, nodes) {  
    if (i >= nodes.length) return;  
    var nodename = nodes[i]["node"];  
    var testbed = nodes[i]["testbed"];  
    var capability = nodes[i]["capability"];  
    var encodedId = nodes[i]["id"];  
    var id = "li-" + encodedId;  
    var removeid = "remove-" + encodedId;  
    var imgid = "img-" + encodedId;  
    $("#controls ul").append(  
        '<li id="' + id + '" class="media"><div class="media-  
body"><h4 class="media-heading">' + nodename + '</h4>' + capability + ' Value:  
<span id="' + imgid + 'value"></span><button class="btn btn-danger removeitem" id="' +  
removeid + '">Delete</button></div></li>');  
    socket = connect(getUberdustServer(), nodename, capability, function (value2add) {  
        var val = value2add.split(" ");  
        window.nodes[window.i]["value"] = val[8];  
        var nodes = null;  
        try {  
            nodes = JSON.parse(window.localStorage.getItem("nodes"));  
        } catch (err) { }  
  
        if (nodes === null || nodes.constructor !== Array) {  
            nodes = new Array();  
        }  
        for (var i = 0; i < nodes.length; i++) {  
            var ctestbed = nodes[i]["testbed"];  
            var cnodename = nodes[i]["node"];  
            var ccapability = nodes[i]["capability"];  
            var encodedId = nodes[i]["id"];  
            var imgid = "img-" + encodedId;  
            if ((nodename == cnodename) && (capability == ccapability)) {  
                nodes[i]["value"] = val[8];  
                if (capability.match(".*:[1-9]r") || capability.match(".*:[1-9]ac") ||  
capability.match(".*:lz[1-9]") || capability.match(".*:red") ||  
capability.match(".*:green") || capability.match(".*:blue")) {  
                    $("##" + imgid).attr("src", "img/newswitch" + parseInt(val[8]) + ".0.png");  
                }else{  
                    $("##" + imgid + "value").text(val[8]);  
                }  
            }  
        }  
    }  
}
```

```

        window.localStorage.setItem("nodes", JSON.stringify(nodes));

    });

    nodes[i]['socket'] = socket;
    var url = "http://" + getUeberdustServer() + "/rest/testbed/" + testbed + "/node/" +
nodename + "/capability/" + capability + "/latestreading";
    window.nodes = nodes;
    window.i = i;
    document.querySelector('#' + imgid).addEventListener('click', change);
    $.get(url, function (data) {
        var latest = data.split('\t')[1];
        console.log(parseInt(data.split('\t')[1]));
        window.nodes[window.i]["value"] = parseInt(latest);

        if (capability.match(".*:[1-9]r") || capability.match(".*:[1-9]ac") ||
capability.match(".*:lz[1-9]")) {
            $('#' + imgid).attr("src", "img/newswitch" +
window.nodes[window.i]["value"] + ".0.png");
        } else {
            $('#' + imgid).attr("src", "img/sensor.png");
            $('#' + imgid + "value").text(latest);
        }
        window.localStorage.setItem("nodes", JSON.stringify(window.nodes));
        populate(i + 1, nodes);
    });
    document.querySelector('#' + removeid).addEventListener('click', removeElement);
}

```

Για να αφαιρεθεί μία συσκευή απο τη σελίδα, αρκεί ο χρήστης να πατήσει το remove button στο οποίο ‘ακούει’ ο remove element EventListener και καλεί τη συνάρτηση removeElement(). Η συνάρτηση αυτή αναλαμβάνει να αφαιρέσει τη συσκευή απο την μνήμη του περιηγητή καθώς και απο τη λίστα συσκευών του χρήστη:

```

function removeElement(e) {
    removeId = "li-" + e.target.id.split("-")[1];
    $('#' + removeId).remove();
    console.log("remove element " + removeId + " socket " + e.target.socket);
    var nodes = null;
    try {
        nodes = JSON.parse(window.localStorage.getItem("nodes"));
    } catch (err) {
    }
    if (nodes === null || nodes.constructor !== Array) {
        nodes = new Array();
    }
}

```

```

for (var i = 0; i < nodes.length; i++) {
    var nodename = nodes[i]["node"];
    var encodedNodeName = replaceAll("\\.", "", replaceAll(":", "", nodename));
    var capability = nodes[i]["capability"];
    var encodedCapability = replaceAll("\\.", "", replaceAll(":", "", capability));
    var encodedId = nodes[i]["id"];
    var id = "li-" + encodedId;
    var removeid = "remove-" + encodedId;
    var scheduleid = "schedule-" + encodedId;
    var imgid = "img-" + encodedId;

    if (id == removeid) {
        console.log("first:" + id);
        console.log("second:" + removeid);
        var index = nodes.indexOf(nodes[i]);
        nodes.splice(index, 1);
        break;
    }
}
window.localStorage.setItem("nodes", JSON.stringify(nodes));
}

```

Για να προστεθεί μία συσκευή στη σελίδα, πρέπει να επιλεγθεί το 'Add' ώστε να εμφανιστούν οι διαθέσιμες πλατφόρμες δοκιμών του συστήματος:

```

function getTestbeds() {
    var getAllTestbeds = "http://" + getUberdustServer() + "/rest/testbed/json";
    $.getJSON(getAllTestbeds, function (data) {
        var testbed = document.getElementById("testbed");
        testbed.options[0] = new Option("Select Testbed:", "0");
        for (var i = 0; i < data.length; i++) {
            var obj = data[i];
            testbed.options[i + 1] = new Option(obj["testbedName"], obj["testbedId"]);
        }
    });
};

```

Μόλις ο χρήστης επιλέξει πλατφόρμα δοκιμών, καλείται η συνάρτηση `getNode()` για να προβληθούν οι κόμβοι του testbed:

```

function getNode() {
    $("#select_node").show();
    var testbed = document.getElementById("testbed").value;
    var url = "http://" + getUberdustServer() + "/rest/testbed/" + testbed +
'/virtualnode/raw';
    $.get(url, function (data) {
        var nodes = data.split("\n");
    });
}

```

```

var node = document.getElementById("nodes");
if (testbed.value == "0") {
    node.options[0] = new Option("Select Node:", "0");
    node.options.length = 1;
} else {
    node.options[0] = new Option("Select Node:", "0");
    for (var i = 0; i < nodes.length; i++) {
        node.options[i + 1] = new Option(nodes[i], nodes[i]);
    }
    node.options.length = nodes.length;
}
}).fail(function (data) {
    showToast(testbed);
})
};

```

Όταν επιλεγθεί και το node, καλείται η συνάρτηση `getCapabilities()` η οποία είναι υπεύθυνη για την προβολή των διαθέσιμων συσκευών:

```

function getCapabilities() {
    $("#select_capability").show();
    var testbed = document.getElementById("testbed").value;
    var node = document.getElementById("nodes").value;
    var url = "http://" + getUberdustServer() + "/rest/testbed/" + testbed +
        "/node/raw";
    $.get(url, function (data) {
        var capability = new Array();
        var cap = document.getElementById("capabilities");
        if (node.value == "0") {
            cap.options[0] = new Option("Select Capability:", "0");
            cap.options.length = 1;
        } else {
            var testbed = document.getElementById("testbed").value;
            var url = "http://" + getUberdustServer() + "/rest/testbed/" + testbed +
                '/node/' + node + '/capabilities';
            $.get(url, function (data) {
                var lines = data.split("\n");
                var i, j, k = 0;
                cap.options[0] = new Option("Select Capability:", "0");
                for (i = 0; i < lines.length; i++) {
                    var urn = lines[i].split(":");
                    if (urn[0] == "urn") {
                        k++;
                        capability[k] = "";
                        for (j = 0; j < (urn.length - 1); j++) {
                            capability[k] = capability[k] + urn[j] + ':';
                        }
                        var name = urn[urn.length - 1];

```

```

var value = capability[k] + urn[urn.length - 1];
    cap.options[k] = new Option(name, value);
    }
    }
    });
  }
}).fail(function (data) {
    showToast(data);
})
};

```

Αφού επιλεγθούν όλα τα παραπάνω, ο χρήστης πρέπει να πατήσει το Add button ώστε να ενεργοποιηθεί ο κατάλληλος EventListener που θα καλέσει την υπεύθυνη συνάρτηση. Αυτή είναι η addToList(), η οποία θα πρόσθεσει στην τοπική μνήμη του περιηγητή (Google Local Storage) τη συσκευή, καθώς επίσης θα πραγματοποιήσει και την ανανέωση της λίστας συσκευών του χρήστη. Η υλοποίηση της φαίνεται παρακάτω:

```

function addtolist() {
    var nodes = null;
    try {
        nodes = JSON.parse(window.localStorage.getItem("nodes"));
    } catch (err) {
    }
    if (nodes === null || nodes.constructor !== Array) {
        nodes = new Array();
    }
    var encodedNodeName = replaceAll("\\.", "", replaceAll(":", "", $('#nodes
:selected').val()));
    var encodedCapability = replaceAll("\\.", "", replaceAll(":", "", $('#capabilities
:selected').val()));
    var encodedId = encodedNodeName + encodedCapability;

    nodes.push(
        {'testbed': $('#testbed :selected').val(),
        'node': $('#nodes :selected').val(),
        'capability': $('#capabilities :selected').val(),
        'value': 1,
        'id': encodedId,
        'socket': ""
        }
    );

    window.localStorage.setItem("nodes", JSON.stringify(nodes));
    window.location.reload();
}

```

Για να ενεργοποιηθεί ή να απενεργοποιηθεί κάποια συσκευή, αρκεί ένα 'κλικ' στο αντίστοιχο εικονίδιο. Έτσι καλείται η συνάρτηση `change()` μέσω της οποίας ελέγχεται η τρέχουσα κατάσταση της συσκευής και στέλνεται το κατάλληλο http request:

```
function change(e) {
  var encodedDevice = e.target.id.split("-")[1];
  var nodes = null;
  try {
    nodes = JSON.parse(window.localStorage.getItem("nodes"));
  } catch (err) {
  }
  if (nodes === null || nodes.constructor !== Array) {
    nodes = new Array();
  }
  for (var i = 0; i < nodes.length; i++) {
    var testbed = nodes[i]["testbed"];
    var nodename = nodes[i]["node"];
    var capability = nodes[i]["capability"];
    var value = window.nodes[i]["value"];
    console.log("hadValue" + value);
    var encodedId = nodes[i]["id"];
    var imgid = "img-" + encodedId;
    var server = window.localStorage.getItem("server");
    if (encodedDevice == encodedId) {
      window.nodes = nodes;
      window.i = i;
      if (window.nodes[i]["value"] == 1.0) {
        value = 0;
      } else {
        value = 1;
      }
      $("#" + imgid).attr("src", "/img/loading.gif");
      $.post("http://" + getUberdustServer() + "/rest/testbed/" + testbed + "/node/"
+ nodename + "/capability/" + capability + "/" + value + "/", function () {
        });
    }
  }
}
```

Sensor Control Panel beta :)

[Items](#)[Reload](#)

Add Item



0013a200408d358e/0x2eb/lz1

X Latest 1 | 5Min 10 updated at 2014-01-20T21:07:52.927Z



0013a200408d358e/0x319/lz1

X Latest 1 | 5Min 1 updated at 2014-01-20T21:05:01.296Z

Εικόνα 3.5 Ελεγκτής Uberdust μέσω ιστοσελίδας

4 On{X}

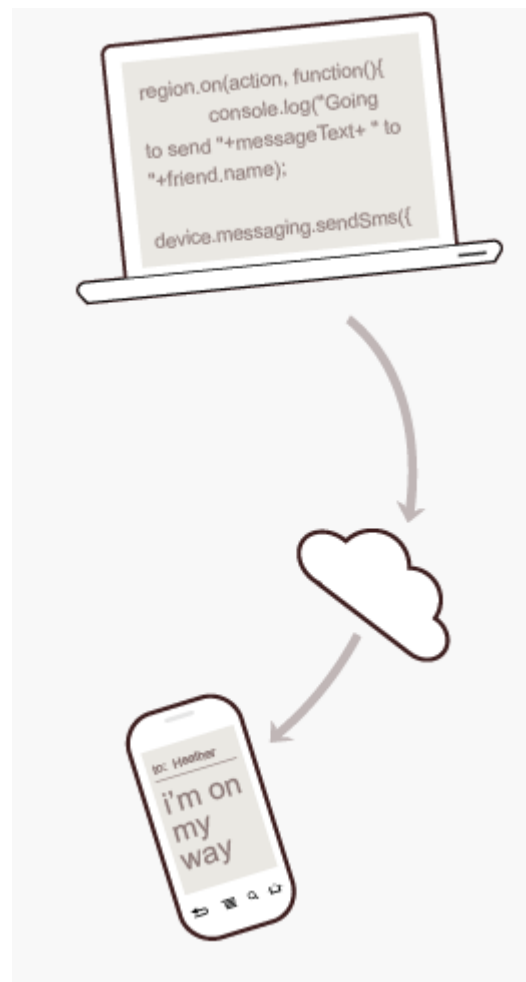
4.1 Τι είναι το On{X}

Πρόκειται για ένα site και μία Android εφαρμογή που παρέχει στους χρήστες του εν λόγω λειτουργικού συστήματος την ευκαιρία να επεκτείνουν τις δυνατότητες της συσκευής τους. Οι προγραμματιστές-χρήστες του Android μπορούν να δημιουργούν τις δικές τους εφαρμογές (rules) γράφοντας Javascript, ενώ οι υπόλοιποι χρήστες μπορούν να χρησιμοποιούν τις εφαρμογές που έχουν δημιουργήσει οι προηγούμενοι ως δημόσιες (recipes).

Rule ονομάζεται ο κώδικας σε Javascript ο οποίος τρέχει στο κινητό. Η λογική των rules είναι ότι 'εκμεταλεύονται' ένα ερέθισμα του κινητού και ενεργοποιούν μια λειτουργία - όταν συμβεί το X, κάνε το Y (on{X} do {Y}).

Τι χρειάζεται:

- Εγκατάσταση της εφαρμογής στο Android smartphone.
- Σύνδεση στην ιστοσελίδα χρησιμοποιώντας τον λογαριασμό του facebook).
- Αποστολή των rules στο κινητό.



Κάθε rule που δημιουργείται στο site αποστέλλεται άμεσα στο κινητό χρησιμοποιώντας το `on{X}[10]` application. Ο κώδικας που δημιουργείται εκτελείται στο τηλέφωνο χρησιμοποιώντας τις δυνατότητες-λειτουργίες του κινητού όπως το GPS, τα γραπτά μηνύματα, τηλεφωνικές κλήσεις κ.α. Στο `on{X}` API αναφέρονται όλες οι λειτουργίες της συσκευής που είναι διαθέσιμες στους προγραμματιστές ως Triggers and Actions (ώθηση και αποτέλεσμα).

Triggers

Μέσω του `on{X}` ο προγραμματιστής μπορεί να ορίσει 'γεγονότα' που βασίζονται στις λειτουργίες και αισθήσεις του κινητού. Μερικά απο αυτά φαίνονται στην εικόνα 4.1.2.



Εικόνα 4.1 Triggers

Actions

Όταν συμβαίνει ένα trigger, ο προγραμματιστής έχει ορίσει τί θα συμβεί στη συνέχεια. Μερικά παραδείγματα αποτελούν τα παρακάτω:

- Εκίνηση κάποιας εφαρμογής
- Προβολή ειδοποίησης
- Μετάβαση σε σύνδεσμο
- Αποστολή SMS

Τα παραπάνω μπορούν να γίνουν αμέτρητα, καθώς ο κάθε προγραμματιστής μπορεί να δημιουργήσει τα δικά του javascript actions.

4.2 Το λειτουργικό σύστημα Android

Το Android είναι λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google.

Το Android είναι μια μοναδική πλατφόρμα που επιτρέπει την ανάπτυξη εφαρμογών λογισμικού το οποίο εκμεταλλεύεται πλήρως τις δυνατότητες μιας συμβατής συσκευής. Για παράδειγμα, οι προγραμματιστές εφαρμογών είναι ελεύθεροι να δημιουργήσουν εφαρμογές που χρησιμοποιούν οποιαδήποτε από τις βασικές λειτουργίες του τηλεφώνου όπως η αποστολή SMS, τηλεφωνικές κλήσεις, τη λήψη φωτογραφιών, το GPS κτλ. Έτσι διευκολύνονται στην ανάπτυξη πιο περίπλοκου και πιο πλούσια λειτουργικού λογισμικού. Αυτό το λειτουργικό σύστημα κινητών τηλεφώνων (ή άλλων μικρών φορητών συσκευών που συνδέονται στο διαδίκτυο) στηρίζεται στον ελεύθερο πυρήνα του Linux. Επιπλέον, η πλατφόρμα ανάπτυξης Android είναι μια πλατφόρμα multi tasking, πράγμα που σημαίνει ότι κάθε εφαρμογή μπορεί να τρέξει στο τηλέφωνο ταυτόχρονα κάποια άλλη χωρίς να επηρεαστεί η απόδοσή τους, και αυτό είναι καλύτερο από το να περιορίζεται σε μία εφαρμογή κάθε φορά. Το Android είναι μια πλατφόρμα ανοικτού κώδικα, πράγμα που σημαίνει ότι μπορεί εύκολα να επεκταθεί και να τροποποιηθεί για να συμβαδίζει και να υιοθετεί τις τελευταίες τεχνολογίες και εξελίξεις. Το γεγονός ότι και η πηγή της πλατφόρμας είναι ανοικτή διασφαλίζει ότι η ανάπτυξη το Android θα έχει συνεχή πρόοδο και θα εξελίσσεται αφού ένας μεγάλος αριθμός ικανών android προγραμματιστών εργάζεται για τη δημιουργία ελεύθερων για χρήση προηγμένων εργαλείων λογισμικού.

Πλήρης παραμετροποίηση

Δεν υπάρχει διαφορά μεταξύ των λειτουργιών/εφαρμογών οι οποίες είναι ενσωματωμένες στο τηλέφωνο από τις εφαρμογές που δημιουργούνται και προστίθενται από τρίτους προγραμματιστές Android. Οι τελευταίες μπορούν και έχουν την ίδια πρόσβαση σε όλες τις κύριες λειτουργίες της συσκευής κάτι που επιτρέπει στους

τελικούς χρήστες να απολαμβάνουν ένα ευρύ φάσμα εφαρμογών Android που μπορούν να χρησιμοποιηθούν για τη σχεδόν απεριόριστους σκοπούς. Με συσκευές χτισμένες στην πλατφόρμα Android, οι χρήστες έχουν τη δυνατότητα να προσαρμόσουν πλήρως τη συσκευή τους ανάλογα με τις ανάγκες και τις απαιτήσεις τους. Τυχόν εφαρμογές ακόμα και οι βασικές λειτουργίες μπορεί να τροποποιηθούν ή να αντικατασταθούν πλήρως από άλλες. Για παράδειγμα, ο χρήστης μπορεί να χρησιμοποιήσει την επιθυμητή του εφαρμογή για να εμφανίσει τις φωτογραφίες που είναι αποθηκευμένες στο τηλέφωνό του, ή για να έχει πρόσβαση στην αλληλογραφία του.

Διαδραστικότητα

Οι προγραμματιστές Android μπορούν να δημιουργήσουν πολύπλοκες καινοτόμες εφαρμογές με σχεδόν απεριόριστη λειτουργικότητα. Για παράδειγμα, μια εφαρμογή μπορεί να μεταδώσει τα δεδομένα από το κινητό σας με το διαδίκτυο (κάτι που μπορεί να περιλαμβάνει το ημερολόγιο σας και τις προγραμματισμένες εκδηλώσεις, λίστα με τις επαφές, τις φωτογραφίες σας και ακόμη και την τρέχουσα θέση σας, αλλά και παραγγελίες, τιμολόγια κτλ) και να λάβει όλα όσα μπορεί να χρειαστεί online και να εμφανίζονται στην οθόνη της συσκευής.

Απλούστερη Ανάπτυξη Εφαρμογών Κινητών

Η πλατφόρμα παρέχει στον καθένα που ασχολείται με την ανάπτυξη εφαρμογών τη δυνατότητα χρησιμοποίησης μια μεγάλης ποικιλίας από βιβλιοθήκες και τα χρήσιμα εκείνα εργαλεία που μπορούν να χρησιμοποιηθούν για τη δημιουργία του πιο εξελιγμένου λογισμικού. Αυτή η ολοκληρωμένη δέσμη από έτοιμα εργαλεία αυξάνει σημαντικά την παραγωγικότητα των προγραμματιστών Android εφαρμογών και τους βοηθά να δημιουργήσουν εκπληκτικά πλούσιο λογισμικό γρηγορότερα και με λιγότερα λάθη.

4.3 Οδηγός για τη δημιουργία εφαρμογών

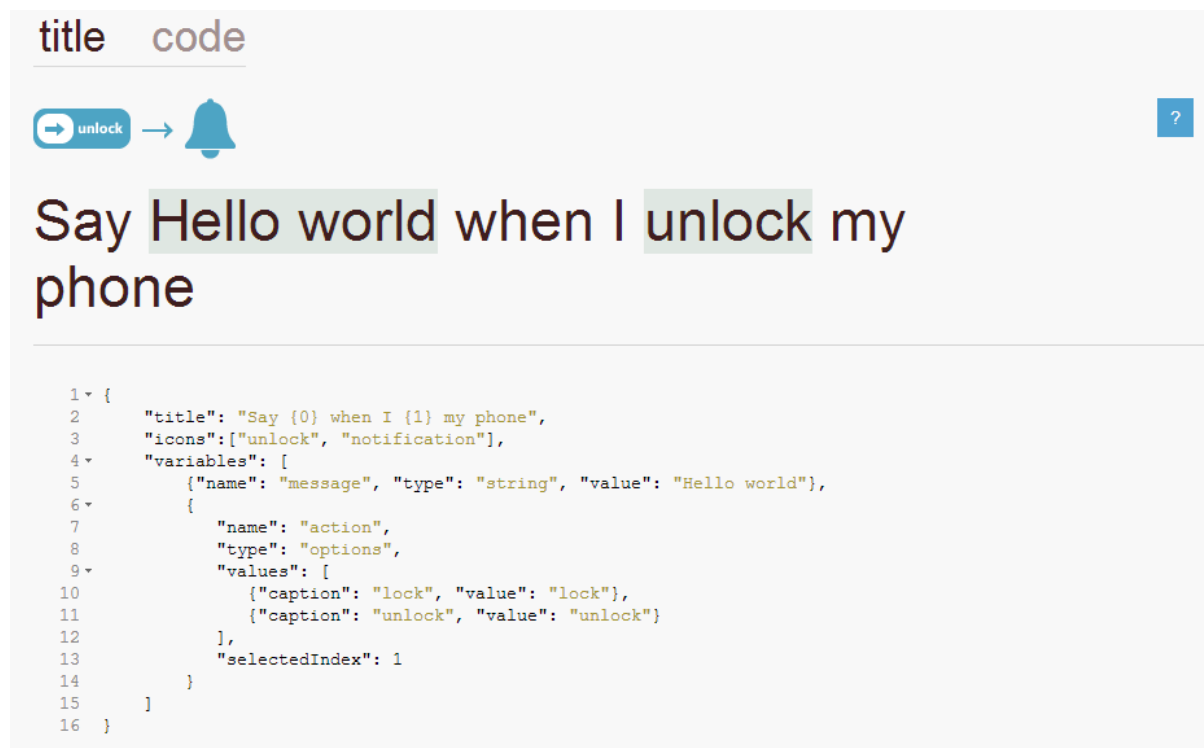
Το πρώτο βήμα είναι η λήψη του On{x} application στο κινητό. Η σύνδεση στην ιστοσελίδα απαιτεί τα στοιχεία του facebook.

Δημιουργία εφαρμογών (rules)

Για τη δημιουργία μιας On{x} εφαρμογής, αρκεί ο χρήστης να δημιουργήσει ένα νέο rule και να επεξεργαστεί τον υπάρχον κώδικα.

Δημιουργία 'συνταγών' (recipes)

Στην πλατφόρμα On{x}, οι εφαρμογές προς δημοσίευση και λήψη αποκαλούνται 'συνταγές'. Η διαφορά τους από τα rules είναι ότι στις συνταγές μπορεί ο χρήστης να αλλάξει τις παραμέτρους της εφαρμογής. Για παράδειγμα, στο recipe που φαίνεται στην εικόνα 4.3.1 οι παράμετροι της εφαρμογής είναι το μήνυμα (κειμενο) που θα λάβει ο χρήστης και το πότε(κλείδωμα ή ξεκλείδωμα τηλεφώνου) Για την μετατροπή ενός rule σε recipe αρκεί ένα επιπλέον script (στο On{x} ονομάζεται title) με τις παραμέτρους εισόδου της εφαρμογής και στη συνέχεια η χρήση τους ως μεταβλητές στο κυρίως πρόγραμμα.



Εικόνα 4.3 Παράδειγμα title συνταγής

Έλεγχος - Αποσφαλμάτωση

Για να ελεγχθεί μια εφαρμογή στην πλατφόρμα On{x} υπάρχουν δυο τρόποι. Ο ένας είναι ο γνωστός – απλά τυπώνοντας στην κονσόλα. Ο άλλος είναι η δημιουργία ‘ψεύτικων’ γεγονότων χρησιμοποιώντας τη λειτουργία emit ώστε η εφαρμογή να ‘πιστέψει’ πως κάποιο από τα γεγονότα που περιμένει συνέβη και να δράσει ανάλογα. Για παράδειγμα, σε κάποια από τις εφαρμογές που υλοποιήθηκαν στα πλαίσια της παρούσας εργασίας χρησιμοποιήθηκε το `trigger device.region.on(‘enter’)` το οποίο καλεί μια συνάρτηση όταν η συσκευή εισέλθει σε μια συγκεκριμένη γεωγραφική περιοχή. Πρακτικά, η εφαρμογή πιστεύει ότι η συσκευή όντως εισήλθε στην περιοχή ενδιαφέροντος. Παρακάτω φαίνεται πως χρησιμοποιείται η εντολή emit για το συγκεκριμένο παράδειγμα. Οι συντεταγμένες `locationSignal` αφορούν τις εφαρμογές της επόμενης ενότητας.

```
var locationSignal = { latitude : 38.291061 ,longitude : 21.796647};  
region.emit(‘exit’, locationSignal);
```

Υπήρξε πολύ εξυπηρετικό το γεγονός ότι κατά τη διάρκεια ανάπτυξης της εφαρμογής χρησιμοποιήθηκε η συγκεκριμένη λειτουργία. Αν δεν υπήρχε αυτός ο τρόπος ελέγχου της θα έπρεπε να δοκιμασθεί με πραγματική είσοδο/έξοδο του χρήστη στην περιοχή.

Λειτουργίες του On{X}

Η κύρια κλάση αντικειμένων του On{x} είναι η `device` η οποία περιλαμβάνει όλες τις δυνατές λειτουργίες του τηλεφώνου στις οποίες έχει πρόσβαση το On{x} είτε πρόκειται για άντληση γεγονότων (μπαταρία, ανίχνευση ασύρματου δικτύου, αίσθηση κίνησης και θέσης, τηλεφωνικές κλήσεις, gps κ.α) είτε για να προκαλέσει λειτουργίες όπως κλήσεις, γραπα μηνύματα, αναπαραγωγή πολυμέσων, εκτέλεση εφαρμογών και άλλα.

Μια επιπλέον κλάση αντικειμένων αποτελεί η `feeds` η οποία περιλαμβάνει αντικείμενα εξωτερικών πληροφοριών όπως ο καιρός, το ωροσκόπιο και τα τοπικά νέα.

Παράδειγμα Εφαρμογής

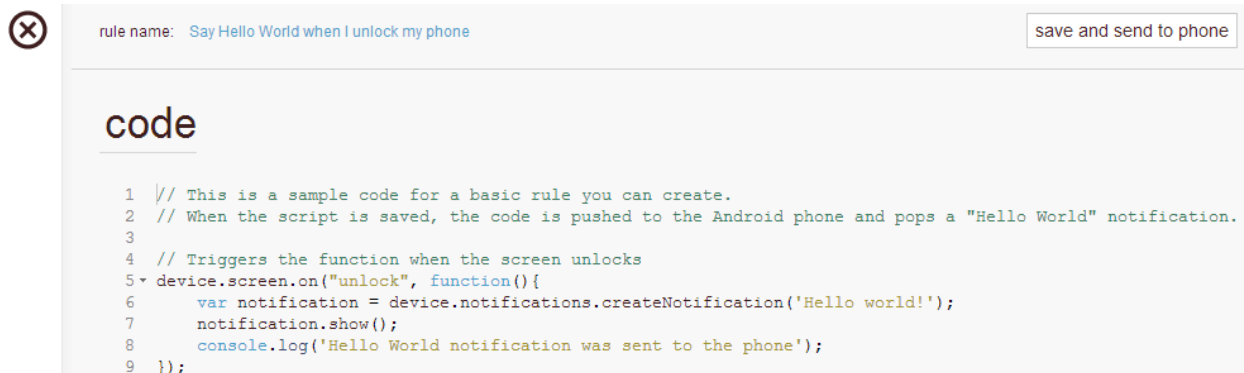
Το παράδειγμα που ακολουθεί ενημερώνει το χρήστη μόλις συνδεθεί σε ένα δίκτυο wifi. Εκτελείται η `startWifiScan()` του αντικειμένου `network`. Όταν η συσκευή συνδεθεί

σε κάποιο ασύρματο δίκτυο εκτελείται η `device.network.on()` με όρισμα: το trigger `wifiScan` και την συνάρτηση callback που στη συγκεκριμένη περίπτωση ενημερώνει το χρήστη για τη νέα σύνδεση που πραγματοποιήθηκε.

Για τον έλεγχο της παραπάνω εφαρμογής αρκεί να προσθέσουμε τη γραμμή που δημιουργεί το event συνδεσης σε σύνδεσης σε ασύρματο δίκτυο:

```
device.network.startWifiScan();

device.network.on("wifiScan", function ()
{
    device.notifications.createNotification("wifiscan...").show();
    var results = device.WifiScanSignal.scanResults[0];
    device.notifications.createNotification('Wifi connection was made').show();
    console.info(results);
});
```



Εικόνα 4.4 Παράδειγμα δημιουργίας Rule

4.4 On{x} και Uberdust

Η πλατφόρμα On{x} προσφέρει την δυνατότητα δημιουργίας ρουτινών που εκτελούνται ως αποτέλεσμα ενεργειών και συμβάντων στο κινητό τηλέφωνο του χρήστη της

εφαρμογής. Τέτοια συμβάντα μπορούν να προέρχονται από ένα μεγάλο αριθμό ερεθισμάτων όπως η μεταβολή της θέσης του τηλεφώνου, κάποια εισερχόμενη κλήση ή ακόμη και μια μεταβολή στις καιρικές συνθήκες. Σε κάθε ένα από αυτά τα συμβάντα υπάρχει η δυνατότητα να προστεθεί ένα κομμάτι κώδικα Javascript και με τη χρήση κατάλληλου API να έχει πρόσβαση είτε στις δυνατότητες του ίδιου του τηλεφώνου είτε να αλληλεπιδράσει με μια εξωτερική εφαρμογή.

Στα πλαίσια της παρούσας εργασίας χρησιμοποιήθηκε το On{X} για να δημιουργηθούν εφαρμογές που αλληλεπιδρούν με το Uberdust ανοίγοντας και κλείνοντας τα φώτα και τον εξοπλισμό του γραφείου όταν ο χρήστης πλησιάζει στην θέση εργασίας του.

Για την επικοινωνία των δεδομένων από το σύστημα Überdust γίνεται χρήση 2 μεθόδων συλλογής δεδομένων.

- WebSockets, για την αδιάκοπη λήψη των νεότερων μετρήσεων από τις συσκευές.
- Ajax http αιτήσεις, συγκεκριμένα http post για την εκτέλεση on/off λειτουργιών.

4.5 Υλοποίηση νέων εφαρμογών

Παρακάτω παρουσιάζονται οι εφαργονές (rules) που υλοποιήθηκαν:

I. Έλεγχος φωτισμού όταν το κινητό περιστρέφεται.

Περιγραφή:

Κάθε φορά που η συσκευή περιστρέφεται, το φως του χώρου σβήνει ή ανάβει ανάλογα με το αν η οθόνη είναι στραμμένη προς τα κάτω ή προς τα πάνω. Η εφαρμογή χρησιμοποιεί το `trigger device.position.on()` το οποίο καλείται κάθε φορά που αλλάζει θέση και στη συνέχεια εκτελείται μια `http post` μέθοδος προς το Uberdust για να ανάψει ή να σβήσει το φως.

Υλοποίηση:

Για να εντοπιστεί η αλλαγή της θέσης της έξυπνης συσκευής χρησιμοποιείται η `device.position.on()` η οποία παίρνει 2 ορίσματα : τη θέση που πρέπει να ανιχνεύσει και τη συνάρτηση (callback) που θα εκτελέσει.

```
device.position.on('faceup', up);  
device.position.on('facedown', down);
```

Κάθε μία απο τις callback συναρτήσεις (up,down) εκτελούν μια μέθοδο POST. Στο On{x} υπάρχει υλοποιημένος Http Client οπότε αυτό που μένει είναι να χρησιμοποιηθεί:

```
HttpScriptClientPromise ajax(Object args, Object onSuccess, Object onError)
```

Η ανίχνευση αλλαγής θέσης της συσκευής γίνεται κάθε 30 δευτερόλεπτα:

```
device.position.startDetection(30*60*1000);
```

Για να αποφευχθεί η επανάληψη αποστολής όμοιων αιτήσεων χρησιμοποιείται μεταβλητή στην οποία αποθηκεύεται η τρέχουσα θέση του κινητού (realState). Αν ή αίτηση η οποία πρόκειται να αποσταλεί θα ανάψει/σβήσει το ήδη αναμένο/σβησμένο φως τότε αγνοείται.

Για την ενημέρωση του χρήστη χρησιμοποιείται η προβολή ενημερώσεων του On{x}:

```
device.notifications.createNotification('This is a notification').show();
```

Η συνάρτηση που καλείται όταν εντοπιστεί «faceup»:

```
function down(signal) {  
  if (realState>-1){  
    realState=-1;  
    device.ajax(  
      {  
        url: "http://" + username + ":" + password + "@" + sspurl + "/" + "?uri=" + deviceURI,  
        type: 'POST',  
        data: 'on'  
      },  
      function onSuccess(body, textStatus, response) {  
        console.info('successfully received http response!');  
      },  
      function onError(textStatus, response) {  
        console.error('error: ',textStatus);  
      });  
  }  
}
```

Αντίστοιχη είναι και η συνάρτηση down που καλείται όταν εντοπιστεί «facedown»:

```
function up(signal) {  
  
    if (realState<1){  
        realState=1;  
        device.ajax(  
            {  
                url: url,  
                type: 'POST',  
                data: 'off'  
            },  
            function onSuccess(body, textStatus, response) {  
                device.notifications.createNotification('Turn on!').show();  
            },  
            function onError(textStatus, response)  
                device.notifications.createNotification('Something went  
wrong...').show();  
            });  
    }  
}
```

II. Ενεργοποίηση Η/Υ χρήστη όταν ο χρήστης πλησιάζει στο χώρο εργασίας του και αντίστροφα.

Περιγραφή:

Κάθε φορά που ο χρήστης πλησιάζει το χώρο εργασίας του ενεργοποιείται η πρίζα στην οποία είναι συνδεδεμένη η οθόνη του. Ορίζονται οι συντεταγμένες του τόπου εργασίας και η ακτίνα στην οποία είναι επιθυμητό να ανιχνεύεται ο χρήστης. Έπειτα χρησιμοποιείται το `trigger device.region.on()` το οποίο καλείται κάθε φορά που η συσκευή (δηλαδή ο χρήστης) εισέρχεται στην περιοχή που έχει οριστεί ή εξέρχεται από αυτή. Στη συνέχεια αποστέλεται η αντίστοιχη http αίτηση για το αν θα ενεργοποιηθεί ή θα απενεργοποιηθεί η πρίζα που τροφοδοτεί την οθόνη του.

Υλοποίηση:

Αρχικά ορίζεται η περιοχή του τόπου εργασίας. Δημιουργείται ένα αντικείμενο json

(mLocation) το οποίο περιέχει το όνομα της περιοχής, τις συντεταγμένες (latitude, longitude) και την ακτίνα που θα ορίζει την περιοχή:

```
var region = device.regions.createRegion({
  name : mLocation.name,
  latitude : parseFloat(mLocation.latitude),
  longitude : parseFloat(mLocation.longitude),
  radius: 500
});
```

Για να τεθεί σε λειτουργία η παρακολούθηση τοποθεσίας του χρήστη χρησιμοποιείται η device.regions.startMonitoring() η οποία παίρνει ως όρισμα το αντικείμενο region που ορίστηκε προηγουμένως.

Όταν πραγματοποιηθεί η είσοδος ή η έξοδος της συσκευής στην περιοχή ενδιαφέροντος καλείται η region.on() η οποία δέχεται ως παραμέτρους τη δράση που ανιχνεύθηκε (enter/exit) και τη συνάρτηση που θα εκτελεστεί:

```
region.on('enter', function(signal){
  var notification = {};
  device.notifications.createNotification('Welcome!').show();
  device.ajax(
    {
      url: 'http://uberdust.cti.
gr/rest/testbed/1/node/urn:wisebed:ctitestbed:150.140.5.67/capability/urn:wisebed:n
ode:capability:3r/1/',
      type: 'POST',
      data : '1',
      headers: {
        'Content-Type': 'application/xml' }
    },
    function onSuccess(body, textStatus, response) {
      device.notifications.createNotification('successfully received http
response!').show();

    },
    function onError(textStatus, response) {
      var error = {};
      error.message = textStatus;
      error.statusCode = response.status;
      console.error('error: ',error);
    })
  );
```

Όπως φαίνεται στο τμήμα του κώδικα που παρουσιάζεται, αυτό που συμβαίνει όταν ο χρήστης εισέλθει στην περιοχή που έχει οριστεί είναι η πραγματοποίηση μιας http POST αίτησης ώστε να ενεργοποιηθεί η οθόνη του. Αντίστοιχη είναι και η υλοποίηση της συνάρτησης που καλείται στο `region.on('exit')`. Και στις δυο περιπτώσεις, ο χρήστης λαμβάνει ενημέρωση σχετική με τη λειτουργία που πραγματοποιήθηκε.

III. Ενεργοποίηση κλιματισμού όταν ο χρήστης πλησιάζει το χώρο εργασίας του αν καιρικές συνθήκες το απαιτούν

Περιγραφή:

Κάθε φορά που ο χρήστης πλησιάζει το χώρο εργασίας του ενεργοποιείται το κλιματιστικό του γραφείου του, αν η θερμοκρασία είναι πάνω από μία συγκεκριμένη τιμή. Ορίζονται οι συντεταγμένες του τόπου εργασίας και η ακτίνα στην οποία είναι επιθυμητό να ανιχνεύεται ο χρήστης. Έπειτα χρησιμοποιείται το `trigger device.region.on()` το οποίο καλείται κάθε φορά που η συσκευή (δηλαδή ο χρήστης) εισέρχεται στην περιοχή που έχει οριστεί. Στη συνέχεια ελέγχονται οι καιρικές συνθήκες στην περιοχή εργασίας του και αποστέλεται η αντιστοιχη http αίτηση για να ενεργοποιηθεί η συσκευή κλιματισμού. Στην περίπτωση που οι καιρικές συνθήκες δεν το απαιτούν δεν πραγματοποιείται η http αίτηση.

Υλοποίηση:

Όπως και στην προηγούμενη εφαρμογή, ορίζεται ή περιοχή του τόπου εργασίας. Τείθεται σε λειτουργία η παρακολούθηση τοποθεσίας του χρήστη με τη χρήση της `device.regions.startMonitoring()` η οποία παίρνει ως όρισμα το αντικείμενο `region` που ορίστηκε προηγουμένως.

Όταν πραγματοποιηθεί η είσοδος της συσκευής (χρήστη) στην περιοχή ενδιαφέροντος καλείται η `region.on('enter')` η οποία πραγματοποιεί μια HTTP GET αίτηση στο `wunderground` για να λάβει ένα xml αρχείο που περιέχει μια λεπτομερή αναφορά των καιρικών συνθηκών στην συγκεκριμένη περιοχή. Επεξεργάζεται το xml αρχείο με σκοπό να βρεθεί η τιμή της θερμοκρασίας σε βαθμούς Κελσίου και ελέγχει το αν ξεπερνάει την ορισμένη τιμή (25 ° C). Σε αυτή την περίπτωση αποστέλεται η σχετική HTTP POST αίτηση για την ενεργοποίηση του κλιματιστικού.

```

region.on('enter', function(signal){
    var notification = {};
    var temp_c={};
    device.ajax(
        {
            url:
'http://api.wunderground.com/auto/wui/geo/WXCurrentObXML/index.xml?query='+
mLocation.latitude+', '+mLocation.longitude,
            type: 'GET',
            data: 'xml',
            dataType: 'xml',
            headers: {
                'Content-Type': 'application/xml'
            }
        },
        function onSuccess(body, textStatus, response) {

            var start=body.search("<temp_c>");
            var finish=body.search("</temp_c>");
            var count=finish-start-8;
            for(i=0;i<count;i++){
                temp_c = temp_c + body[start+8+i];
            }
            temp=JSON.stringify(temp_c);
            temp_c = temp.split("[");
            temp = parseInt(temp_c[1],10);
            console.info("Temperature : "+temp);
            device.notifications.createNotification("Temperature : "+temp).show();
            if(temp>25){
                device.notifications.createNotification('The weather is hot! Turn on A /C!')
                .show();
            }
            device.ajax(
                {
                    url:
'http://uberdust.cti.gr/rest/testbed/1/node/urn:wisebed:ctitestbed:'+node+'/capabilit
y/urn:wisebed:node:capability:'+capability+'/1/',
                    type: 'POST',
                    data : '1',
                    headers: {
                        'Content-Type': 'application/xml'
                    }
                },
                function onSuccess(body, textStatus, response) {
                    console.info('successfully received http response!');
                    var notification = {};
                    notification.content = 'successfully received http response!';
                    notification.show();
                },

```

```
        function onError(textStatus, response) {
            var error = {};
            error.message = textStatus;
            error.statusCode = response.status;
            console.error('error: ',error);
        };

    },
    function onError(textStatus, response) {
        var error = {};
        error.message = textStatus;
        error.statusCode = response.status;
        console.error('error: ',error);
    };
};
```

Αντιστοιχη είναι και η υλοποίηση για την περίπτωση που η θερμοκρασία είναι χαμηλή.

5 Επέκταση περιηγητή Chrome

5.1 Εισαγωγή

Στα πλαίσια της παρούσας εργασίας υλοποιήθηκαν εφαρμογές για την παρακολούθηση και τον απομακρυσμένο έλεγχο των αντικειμένων που διαχειρίζεται το Uberdust. Μια από αυτές αναπτύχθηκε σε μορφή επέκτασης του φυλομετρητή Google Chrome. Η υλοποίηση της επέκτασης έγινε με χρήση τεχνολογιών όπως JavaScript και HTML5 καθώς και της βιβλιοθήκης jQuery^{3[11]} για την αλληλεπίδραση με τα αντικείμενα της εφαρμογής. Η εφαρμογή προσφέρει τη δυνατότητα στους χρήστες του συστήματος να αναζητήσουν την συσκευή ή την οντότητα που επιθυμούν να χειριστούν ή να παρακολουθήσουν και στην συνέχεια με κατάλληλης διεπαφής παρουσιάζονται οι μετρήσεις που παρέχουν οι συσκευές. Για τον σκοπό αυτό γίνεται χρήση της τεχνολογίας WebSocket με αποτέλεσμα η εφαρμογή να εγγράφεται ως παρατηρητής για να λαμβάνει ενημερώσεις από το Uberdust για αλλαγές στην κατάσταση του αντικειμένου ενδιαφέροντος.

5.2 Υλοποίηση Εφαρμογή μέσω επέκτασης περιηγητή

Αμέσως μετά την προσθήκη της επέκτασης στον Google Chrome, ο χρήστης πρέπει να επιλέξει τη συσκευή που θέλει να παρακολουθεί και να ελέγχει. Κάνοντας δεξί κλικ στο εικονίδιο που έχει εμφανιστεί στον περιηγητή, επιλέγει το 'Options' από το μενού που εμφανίζεται και οδηγείται στη σελίδα που επιλέγεται η συσκευή. Αρχικά επιλέγει το 'testbed' από τη λίστα των διαθέσιμων :

```

var testbed=document.getElementById("testbed");
var getAllTestbeds = "http://uber dust.cti.gr/rest/testbed/json";
$.get(getAllTestbeds,function(data){
    testbed.options[0]=new Option("Select Testbed:", "0");
    for(var i=0;i<data.length;i++){
        var obj = data[i];
        testbed.options[i+1]=new Option(obj["testbedName"], obj["testbedId"]);
    }
});

```

Όταν επιλέξει το testbed καλείται η συνάρτηση που του εμφανίζει τα διαθέσιμα virtual nodes μέσω του EventListener 'change' :

```

testbed.addEventListener('change', function(){
    var url='http://uber dust.cti.gr/rest/testbed/'+testbed.value+'/virtualnode/raw';
    $.get(url,function(data){
        var nodes=data.split("\n");
        var node = document.getElementById("nodes");
        if(testbed.value == "0"){
            node.options[0]=new Option("Select Node:", "0");
            node.options.length=1;
        }else{
            node.options[0]=new Option("Select Node:", "0");
            for(var i=0;i<nodes.length;i++){
                node.options[i+1]=new Option(nodes[i], nodes[i]);
            }
            node.options.length=nodes.length;
        }
    }).fail(function() {
        alert("error");
    })
}, false);

```

Αντίστοιχα, όταν επιλεγθεί το virtual node ενεργοποιείται άλλος EventListener που καλεί συνάρτηση που εκτελεί τα απαραίτητα AJAX Requests ώστε να προβληθούν οι διαθέσιμες προς έλεγχο και παρακολούθηση συσκευές:

```

var node =document.getElementById("nodes");
node.addEventListener('change', function(){
    var url='http://uber dust.cti.gr/rest/testbed/'+testbed.value+'/node/raw';
    $.get(url,function(data){
        var capability= new Array();
        var cap = document.getElementById("capabilities");
        if(node.value == "0"){
            cap.options[0]= new Option("Select Capability:", "0");
            cap.options.length=1;
        }else{
            var testbed= document.getElementById("testbed").value;

            url='http://uber dust.cti.gr/rest/testbed/'+testbed+'/node/'+node.value+'/capa
            bilities';
            $.get(url,function(data){
                var lines=data.split('\n');
                var i, j, k=0;
                cap.options[0]=new Option("Select Capability:", "0");
                for(i=0;i<lines.length;i++){
                    var urn = lines[i].split(":");
                    if(urn[0]=="urn"){
                        k++;
                        capability[k]="";
                        for(j=0;j<(urn.length-1);j++){
                            capability[k]=capability[k]+urn[j]+'.';
                        }
                        var name = urn[urn.length-1];
                        var value = capability[k]+urn[urn.length-1];
                        cap.options[k]= new Option(name,value);
                    }
                }
                document.capform.capabilities.options.length=capability.length;
            });
        }
    }).fail(function() {
        alert("error");
    })
}, false);

```

Έπειτα ο χρήστης πρέπει να πατήσει το submit button ώστε η συσκευή που επιλέχθηκε να αποθηκευτεί στη μνήμη του περιηγητή (Chrome Local Storage^[12]):

```

var form=document.getElementById("theForm");
form.addEventListener("submit", function(){

```

```

var data =
document.getElementById("testbed").value+" "+document.getElementById("nodes").
value+" "+document.getElementById("capabilities").value;
    chrome.storage.local.set({'data': data});
    chrome.extension.getBackgroundPage().window.location.reload()
});

```

Αφού γίνει μία φορά αυτή η ενέργεια, η επιλεγμένη συσκευή θα 'φορτώνεται' κάθε φορά που ο χρήστης εκκινεί τον browser. Φυσικά ρυθμίσεις αυτές μπορούν να επαναληφθούν οποτεδήποτε είναι επιθυμητή η αλλαγή συσκευής.

Για να ανακτηθούν τα αποθηκευμένα δεδομένα απο τη μνήμη του περιηγητή χρησιμοποιείται η κατάλληλη μέθοδος get του chrome.storage.local:

```

chrome.storage.local.get('data', function (obj) {
    info = obj.data.split(" ");
    var testbed = info[0];
    var node = info[1];
    var capability = info[2];
    document.getElementById("selected").innerHTML="Selected Device:";
    document.getElementById("t").innerHTML=" Testbed:"+testbed;
    document.getElementById("n").innerHTML=" Node:"+node;
    document.getElementById("c").innerHTML=" Capability:"+capability;
});

```

Στην παρακάτω εικόνα φαίνεται η σελίδα options μετά απο επιλογή συσκευής.


Options:

Selected Device:
 Testbed:5
 Node:urn:pspace:virtual:ROOM3
 Capability:urn:node:capability:lz1

Εικόνα 5.1 Σελίδα ρυθμίσεων της επέκτασης browser

Με την εκκίνηση του browser ελέγχεται το αν έχει επιλεχθεί κάποια συσκευή. Αν στη μνήμη του περιηγητή δεν βρεθεί επιλεγμένη συσκευή τότε ο χρήστης ενημερώνεται. Σε αντίθετη περίπτωση, η επιλεγμένη συσκευή φορτώνεται:

```
chrome.storage.local.get('data', loadSettings );

function loadSettings(obj){

    info = obj.data.split("+");
    window.testbed = info[0];
    window.node = info[1];
    window.capability = info[2];

    var url="http://uberdust.cti.gr/rest/testbed/"+ window.testbed +"/node/"+
    window.node +"/capability/"+ window.capability +"/latestreading";
    $.get(url,function(data){
        var latest=data.split('\t');
        window.current= latest[1];
        chrome.browserAction.setIcon({path:"newswitch" + window.current + ".png"});
    });
    if(window.websocket=="closed"){
        console.log("first connection "+window.node);
        connect("uberdust.cti.gr",window.node,window.capability,function(value2add){
            console.log(value2add);
            var val=value2add.split(" ");
            chrome.browserAction.setIcon({path:"newswitch"+val[8]+".png"});
            if (window.current!=val[8]){
                show(val[8]);
            }
            window.current=val[8];
        });
        window.websocket="open";
    }
}
```

Όπως φαίνεται παραπάνω, η συνάρτηση loadSettings είναι υπεύθυνη για τη σύνδεση της συσκευής στο κατάλληλο websocket. Όταν έρχεται ένα νέο μήνυμα στο websocket το εικονίδιο της επέκτασης ανανεώνεται ανάλογα και εμφανίζεται στο χρήστη notification για την αλλαγή που έγινε. Η συνάρτηση show() είναι υπεύθυνη για την ενημέρωση του χρήστη:

```
function show(val) {
    var time = /(..)(:..)/.exec(new Date());
    var hour = time[1] % 12 || 12;
    var period = time[1] < 12 ? 'a.m.' : 'p.m.';
    var notification = window.webkitNotifications.createNotification(
        "newswitch"+val+".png",
        hour + time[2] + ' ' + period, // The title.
        'Device Status Changed: '+val    // The body.
    );
    notification.show();
}
```

Η ενημέρωση εξαφανίζεται μετά απο τρία δευτερόλεπτα:

```
notification.ondisplay = function(event) {
    setTimeout(function() {
        event.currentTarget.cancel();
    }, 3000);
};
}
```

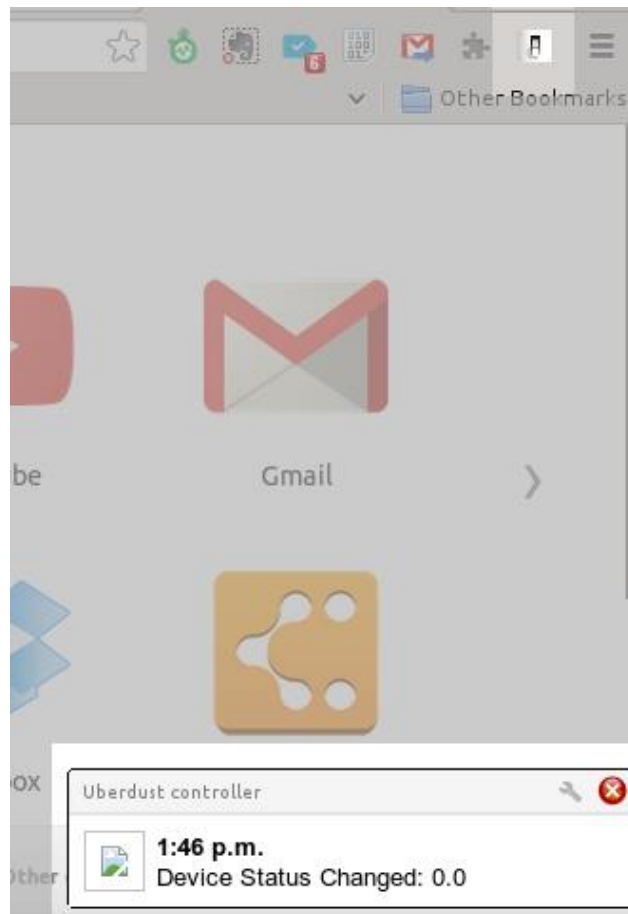
Επίσης, ο χρήστης έχει τη δυνατότητα ενεργοποίησης/απενεργοποίησης της συσκευής. Αυτό επιτυγχάνεται με απλό 'κλικ' στο εικονίδιο της επέκτασης. Όταν συμβεί αυτό, καλείται η συνάρτηση `updateIcon()` που ελέγχει την τρέχουσα κατάσταση της συσκευής και πραγματοποιεί το κατάλληλο http post request:

```
function updateIcon() {
    chrome.storage.local.get('data', loadSettings );
    if(window.current==null){alert("null!");}
    if ( window.current == "0.0" ){

        $.post(
            "http://uber dust.cti.gr/rest/testbed/"+window.testbed+"/node/"+window.node+
            "/capability/" +window.capability+"/1/",function({})
            .done(function() {
            })
            .fail(function() {
                alert("error");
            });
        }
        else if(window.current == "1.0")
        {

```

```
$.post(
  "http://uberdust.cti.gr/rest/testbed/"+window.testbed+"/node/"+window.node+"/cap
  ability/" +window.capability +"/0/",function({})
    .done(function() {
    })
    .fail(function() { alert("error"); });
  } else if(window.current=="")
  {
    alert("No device is selected.Please select a device from the options
    menu.");
  }
}
```



Εικόνα 5.2 Αλλαγή κατάστασης ελεγχόμενης συσκευής

6 Drupal

6.1 Τί είναι το Drupal

Το Drupal χρησιμοποιείται για τη δημιουργία web sites. Είναι ένα Web Content Management System (WCMS) ανοικτού κώδικα με έμφαση στη συνεργασία. Το Drupal, όπως πολλά σύγχρονα CMS, επιτρέπει στο διαχειριστή συστήματος να οργανώνει το περιεχόμενο, να προσαρμόζει την παρουσίαση, να αυτοματοποιεί διαχειριστικές εργασίες και να διαχειρίζεται τους επισκέπτες του ιστοτόπου και αυτούς που συνεισφέρουν. Παρόλο που υπάρχει μια πολύπλοκη προγραμματιστική διεπαφή, οι περισσότερες εργασίες μπορούν να γίνουν με λίγο ή και καθόλου προγραμματισμό.

Το Drupal μπορεί να χρησιμοποιηθεί για την δημιουργία ενός internet portal (προσωπικό ή τμήμα εταιρείας), ενός εταιρικού web site ή εμπορικού site, μιας online εφημερίδας, ή μιας gallery εικόνων, ενός intranet. Αυτές είναι μερικές από τις χρήσεις του. Τέλος μπορεί να χρησιμοποιηθεί ακόμη και για τη εξ αποστάσεως εκπαίδευση (e-learning). Μία ομάδα που είναι αφιερωμένη στην ασφάλεια δουλεύει για να κρατήσει το Drupal προστατευμένο επιλύοντας και θέματα εκσυγχρονισμού της ασφάλειας. Ένας μη κερδοσκοπικός οργανισμός επονομαζόμενος Drupal Assosiation υποστηρίζει το Drupal παρέχοντας την υποδομή στο drupal.org και οργανώνοντας διασκέψεις και εκδηλώσεις. Παράλληλα μια ολοένα αναπτυσσόμενη κοινότητα χρηστών, διαχειριστών site, σχεδιαστών, και κατασκευαστών λογισμικού δουλεύουν σκληρά με σκοπό να βελτιωθεί το λογισμικό.

Για την εγκατάσταση του Drupal:

Για την εγκατάσταση και δημιουργία ιστοσελίδας απαιτείται η εγκατάσταση του Drupal. Αυτή όμως προϋποθέτει ένα περιβάλλον ανάπτυξης δηλαδή την ύπαρξη μιας βάσης δεδομένων MYSQL, την γλώσσα προγραμματισμού δυναμικών ιστοσελίδων PHP

και έναν Apache Server ώστε να φορτώνονται τα αρχεία της ιστοσελίδας στο πρόγραμμα πλοήγησης (browser). Όλα αυτά παρέχονται εντελώς δωρεάν από το ανοιχτού κώδικα διαδικτυακό πακέτο XAMPP. Το όνομα του Xampp είναι ένα ακρωνύμιο των:

- X (σημαίνει cross-platform, που λειτουργεί σε πολλές πλατφόρμες)
- Apache HTTP Server
- MySQL
- PHP
- Perl

Το XAMPP είναι διαθέσιμο διαδικτυακά στη διεύθυνση <http://www.apachefriends.org/en/xampp.html>. Αυτή τη στιγμή η παρούσα έκδοση 1.8 εμπεριέχει

- Apache 2.4.2
- MySQL 5.5.25
- PHP 5.4.4
- phpMyAdmin 3.5.2
- FileZilla FTP Server 0.9.41
- XAMPP Control Panel

6.2 Τι είναι το CMS (Content Management System)

Το Content Management System (Σύστημα Διαχείρισης Περιεχομένου) είναι ένας από τους συνηθέις τρόπους για τη δημιουργία και διαχείριση περιεχομένου στο Internet¹, στα intranets² και γενικότερα σε περιβάλλοντα όπου υπάρχουν περισσότερα από ένα άτομα που εργάζονται επί του περιεχομένου, ή όπου εμπλέκεται πολύ περιεχόμενο. Είναι συστήματα τα οποία δίνουν τη δυνατότητα στους χρήστες να :

- Δημιουργούν περιεχόμενο
- Επεξεργάζονται, διαγράφουν και γενικά διαχειρίζονται περιεχόμενο
- Διαχειρίζονται, παρέχουν περιορισμένη πρόσβαση και επεξεργασία στο περιεχόμενο.
- Έχουν ξεχωριστό επίπεδο για τη σχεδίαση
- Συνεργάζονται δραστικά
- Διαχειρίζονται τύπους περιεχομένου

Με ένα CMS, το περιεχόμενο και η σχεδίαση κρατούνται ξεχωριστά, το οποίο σημαίνει ότι η σχεδίαση ενός site (ιστοτόπου) μπορεί να αλλάξει ριζικά χωρίς να επηρεαστεί το

περιεχόμενο του. Αυτό είναι ένα ιδιαίτερα σημαντικό χαρακτηριστικό το οποίο σημαίνει πως η σχεδίαση του site χρειάζεται να γίνει μόνο μια φορά και όχι σε κάθε σελίδα ξεχωριστά, κάνοντάς το έτσι εύκολο για websites (ιστοσελίδες) κάθε μεγέθους να αλλάζουν εύκολα και γρήγορα σχεδίαση. Έτσι προστατεύεται και η σχεδίαση, καθώς οι διαχειριστές περιεχομένου δε χρειάζεται να ενσωματώσουν τη σχεδίαση – πράγμα που θα δημιουργούσε πρόβλημα .

Οι εφαρμογές CMS χρησιμοποιούνται για δημιουργία blogs, επίσης για δημιουργία forums συζητήσεων, δημιουργία «Gallery sites»(ιστοσελίδων μόνο με εικόνες). Παρέχονται ευκολίες δημιουργίας δημοψηφισμάτων ή ένδειξη στατιστικών στοιχείων. Υπάρχει η δυνατότητα να δίνονται διαφορετικοί ρόλοι ανά κατηγορία περιεχομένου ή ανά επίπεδο χρηστών. Επίσης παρέχουν ευκολία στην δημιουργία «new feeds» τροφοδοσίας νέων στον σελιδοδείκτη «browser». Όλα σχεδόν τα συστήματα CMS παρέχουν δύο επίπεδα διαχείρισης το «front-end manager» (το εμπρός μέρος διαχείρισης) και το «back-end manager» (το πίσω μέρος διαχείρισης) που συνήθως ο διαχειριστής συστήματος έχει τα δικαιώματα επέμβασης στον ιστοχώρο, ενώ η πρόσβαση σε αυτά γίνεται απλώς και μόνο με την χρήση του φυλλομετρητή (browser).

6.3 Ανάπτυξη ιστότοπου Uberdust Testbed – CTI Πάτρας

Η διαδικτυακή πύλη παρουσίασης των δεδομένων που προέρχονται από τους εγκατεστημένους αισθητήρες στις συσκευές του ΙΤΥΕ έχει ως στόχο την φιλικότερη προς τους χρήστες παρουσίαση των δεδομένων μέσω μιας εύχρηστης και οικείας διαδικασίας όπως η πλοήγηση σε μια θεματική ιστοσελίδα. Πιο συγκεκριμένα η ιστοσελίδα έχει ως στόχο την παροχή τόσο συνδυασμένων πληροφοριών μέσω των σημασιολογικών οντοτήτων του Überdust όσο και του εύκολου χειρισμού των εγκατεστημένων αυτοματισμών για τον φωτισμό ή την διαχείριση συσκευών του κεντρικού πίνακα (π.χ. θερμοσίφωνο). Για την επίτευξη του στόχου αυτού υλοποιήθηκαν διεπαφές που προσφέρουν τα εξής:

- Γραφικές παραστάσεις ιστορικών δεδομένων (π.χ., κατανάλωση την τελευταία μέρα)
- Γραφικές αναπαραστάσεις των τελευταίων μετρήσεων με χρήση κατάλληλων εικόνων (θερμόμετρα, βαρόμετρα, ταχύμετρα).
- Διακόπτες για τον χειρισμό των ελεγκτών των συσκευών
- Συνδυαστικά γραφήματα για την σύγκριση της κατανάλωσης και της παραγωγής ενέργειας σε επίπεδο κτηρίου.

Για την άντληση των δεδομένων από το σύστημα Überdust γίνεται χρήση 2 μεθόδων συλλογής δεδομένων.

- REST, για την άντληση ιστορικών δεδομένων για την κατασκευή γραφημάτων και την λήψη της τελευταίας τιμής από τις συσκευές αλλά και την αποστολή ενεργειών σε αυτές.
- WebSockets, για την αδιάκοπη λήψη των νεότερων μετρήσεων από τις συσκευές και την αποφυγή του polling.

Η παρουσίαση των δεδομένων γίνεται με χρήση της βιβλιοθήκης Highcharts και του συστήματος CMS Drupal για την δημιουργία άρθρων και τον έλεγχο της πρόσβασης στο σύστημα. Επίσης, όλες οι μέθοδοι παρουσίασης γίνονται με χρήση modules της Drupal, για την ευκολότερη παροχή των υλοποιημένων δυνατοτήτων σε τρίτους που ενδιαφέρονται για την χρήση αντίστοιχου συστήματος σε δική τους εγκατάσταση.

6.3.1 Γενικά

Για κάθε σελίδα που δημιουργείται στο Drupal ορίζεται το περιεχόμενο της (content), δηλαδή τα html στοιχεία και ο php κώδικας της σελίδας. Πιο συγκεκριμένα, καλούνται οι συναρτήσεις (Drupal modules) που υλοποιούν το εκάστοτε περιεχόμενο της σελίδας. Κάθε σελίδα έχει διαφορετικό περιεχόμενο παρ'ότι μπορεί να καλεί τα ίδια modules με κάποια άλλη. Επίσης Για παράδειγμα, η σελίδα που παρουσιάζει τα γραφήματα με το ιστορικό των μετρήσεων των αισθητήρων ενός εικονικού δωματίου καλεί κάποια modules που καλούνται και απο τη σελίδα που παρουσιάζονται οι μετεωρολογικές μετρήσεις.

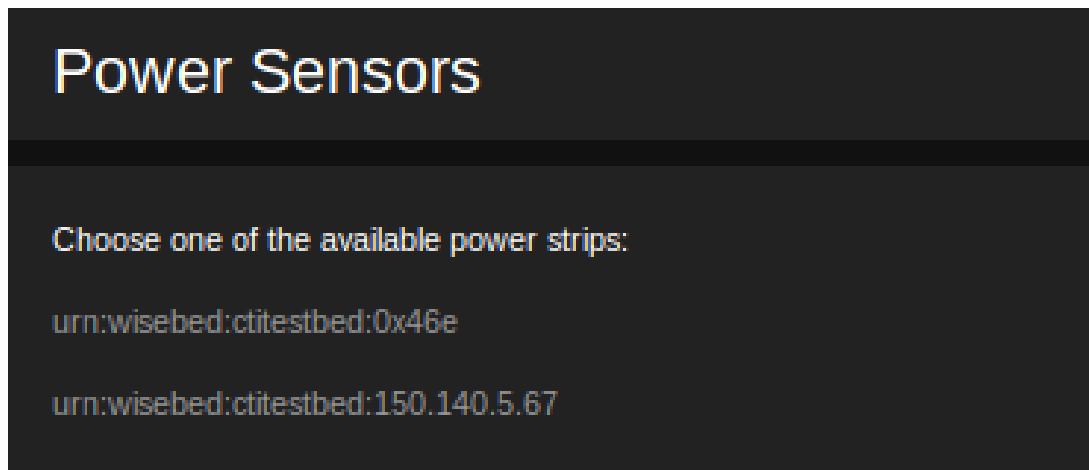
Στη συνέχεια παρουσιάζονται τα Drupal modules που υλοποιήθηκαν.

6.3.2 Power Sensor Module

Το power sensor module υλοποιήθηκε με σκοπό τη διαχείριση ενός μετρητή κατανάλωσης ενέργειας που ελέγχεται απο το σύστημα Uberdust. Στα πλαίσια των δοκιμών που έγιναν για την παρούσα εργασία, ο μετρητής αυτός εγκαταστάθηκε σε ένα πολύπριζο. Μέσω αυτής της συσκευής υπάρχει η δυνατότητα ενεργοποίησης και απενεργοποίησης της πρίζας, η μέτρηση της κατανάλωσής της, καθώς και πληροφορία

για το αν η συσκευή καταλώνει ενέργεια ή όχι. Το παρόν module υλοποιήθηκε με σκοπό να αποτελέσει μια διεπαφή μεταξύ του χρήστη και της συσκευής αυτής.

Ως επέκταση της αρχικής υλοποίησης, ήταν το να καλείται το να εντοπίζονται όλες οι συσκευές μέτρησης καταναλωσης ενέργειας που ελέγχονται απο το Uberdust και όχι μόνο αυτές που έχουν εγκατασταθεί στο πολύπριζο και να παρουσιάζονται στο χρήστη.



Εικόνα 6.1 Διαθέσιμες συσκευές μέτρησης

Για να γίνει αυτό, γίνεται χρήση της διεπαφής του συστήματος και ελέγχεται ποιές συσκευές έχουν 'τύπο' powerstrip ή power sensor:

```
$status_raw=file_get_contents("http://uberdust.cti.gr/rest/testbed/" . $testbed . "/status/raw");
$line = explode("\n", $status_raw);

for($i=0; $i<sizeof($line); $i++){
    $nodetype= explode("\t", $line[$i]);

    if(isset($nodetype[3])&&(($nodetype[3]=="powerstrip")||($nodetype[3]=="powersensor"))){
        $node = explode(":", $nodetype[0]);
        $count = sizeof($node)-1;
    }
}
```

```

urn:wisebed:ctitestbed:0x46e report 1371219621000 0.0
urn:wisebed:ctitestbed:0x46e gateway 1370605196000 urn:wisebed:ctitestbed:0x46e
urn:wisebed:ctitestbed:0x46e urn:wisebed:node:capability:ls 1371219707000 10668.0
urn:wisebed:ctitestbed:0x46e nodetype 1371476882000 powersensor
urn:wisebed:ctitestbed:0x494 urn:wisebed:node:capability:lz3 1373361508000 0.0
urn:wisebed:ctitestbed:0x494 urn:wisebed:node:capability:light 1373361514000 353.0

```

Εικόνα 6.2 Ευρεση μετρητών κατανάλωσης ενέργειας

Υπάρχει πιθανότητα κάποια συσκευή να μην υποστηρίζει όλες τις λειτουργίες που αναφέρθηκαν νωρίτερα ή μπορεί να υποστηρίζονται κι άλλές που δεν χρειάζονται στη συγκεκριμένη εφαρμογή. Για το λόγο αυτό, πριν απο την ανάκτηση των δεδομένων ανιχνεύονται οι «δυνατότητες» του μετρητή και γίνεται η απαραίτητη ενέργεια ώστε να ληφθεί η πληροφορία. Δυνατότητες του μετρητή εννοούνται εννοούνται τα capabilities του μετρητή τα οποία είναι:

S: Η τιμή του ρεύματος σε mA (Sensor value)

I: Δείκτης κατανάλωσης (Indicator)

R: Κατάσταση του ρελέ. (Relay status)

```

urn:wisebed:node:capability:1i
urn:wisebed:node:capability:2i
urn:wisebed:node:capability:3i
urn:wisebed:node:capability:4i
urn:wisebed:node:capability:5i
urn:wisebed:node:capability:6i
urn:wisebed:node:capability:7i
urn:wisebed:node:capability:1r
urn:wisebed:node:capability:1s
urn:wisebed:node:capability:2r
urn:wisebed:node:capability:2s
urn:wisebed:node:capability:3r
urn:wisebed:node:capability:3s
urn:wisebed:node:capability:4r
urn:wisebed:node:capability:4s
urn:wisebed:node:capability:5s
urn:wisebed:node:capability:6r
urn:wisebed:node:capability:6s
urn:wisebed:node:capability:7r
urn:wisebed:node:capability:7s
nodetype
urn:wisebed:node:capability:5r

```

Εικόνα 6.3 Διαθέσιμα capabilities powerstrip

Για την λήψη των δεδομένων:

```
$capabilities = file_get_contents("http://uberdust.cti.gr/rest/testbed/" .
$testbed . "/node/" . $nodeUrn1 . $_GET['ip'] . "/capabilities" );
$line = explode("\n", $capabilities);
$j=0;
$k=0;
$c=0;
for($i=0; $i<sizeof($line); $i++){
    if(substr($line[$i],-1,1)=="i"){
        $c++;
        $capability = substr($line[$i],0,-2 );
        $id = substr($line[$i],-2,1);
        $latestreading= file_get_contents("http://uberdust.cti.gr/rest/testbed/" .
$testbed . "/node/" . $nodeUrn1 . $_GET['ip'] . "/capability/" . $capability . $id .
"r/latestreading");
        $row = explode("\t", $latestreading);
        $indicator[$c]=$row[1];
    }

    if(substr($line[$i],-1,1)=="r"){
        $k++;
        $capability = substr($line[$i],0,-2 );
        $id = substr($line[$i],-2,1);
        $latestreading=
file_get_contents("http://uberdust.cti.gr/rest/testbed/" . $testbed . "/node/" .
$nodeUrn1 . $_GET['ip'] . "/capability/" . $capability . $id . "r/latestreading");
        $row = explode("\t", $latestreading);
        $status[$k]=$row[1];
    }

    if(substr($line[$i],-1,1)=="s"){
        $j++;
        $capability = substr($line[$i],0,-2 );?>
        values[<?php echo $j;?>] = [<?php
        $id = substr($line[$i],-2,1);

        $tabdelimited =
file_get_contents("http://uberdust.cti.gr/rest/testbed/" . $testbed . "/node/" .
$nodeUrn1 . $_GET['ip'] . "/capability/" . $capability . $id . "s/latestreading");

        $lines = explode("\n", $tabdelimited, $maxRows);
        unset($lines[count($lines) - 1]);
        $firstRow = explode("\t", $lines[0]);

        print( $firstRow[1] )
        unset($lines[0]);
```

```

foreach ($lines as $thisLine) {
    $row = explode("\t", $thisLine);
    print(",$row[1]);
}
?>];

```

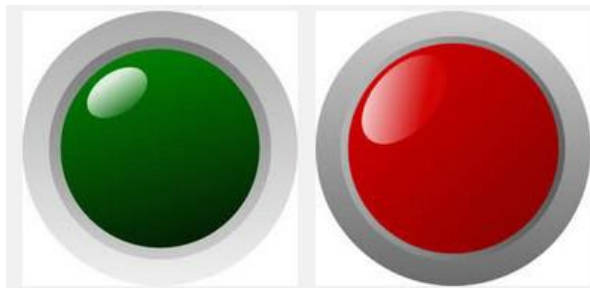
← [ed/1/node/urn:wisebed:ctitestbed:150.140.5.67/capability/urn:wisebed:node:capability:1s/latestreading](#)

1373362278000 36.0 36.0

Εικόνα 6.4 Παράδειγμα τελευταίας μέτρησης για capability 's'

Για κάθε ένα απο τα παραπάνω capabilities (I,S,R) χρησιμοποιείται διαφορετικός τρόπος αναπαράστασης της τιμή τους.

Για την ένδειξη κατανάλωσης χρησιμοποιείται το 'λαμπάκι' που φαίνεται στην παρακάτω εικόνα το οποίο είναι πράσινο αν υπάρχει κατανάλωση και κόκκινο αν όχι.



Εικόνα 6.5 Ενδειξη κατανάλωσης πρίζας

Για την αναπαράσταση της τιμής του ρεύματος που μετρήθηκε χρησιμοποιείται το γράφημα Gauge της βιβλιοθήκης highcharts.js .

Η συνάρτηση που δημιουργεί το γράφημα:

```

$(function drawChart(){

    var chart<?php echo $j;?>;
    chart<?php echo $j;?> = new Highcharts.Chart({

        chart: {
            renderTo: 'chart_div<?php echo $j;?>',
            type: 'gauge',
            plotBackgroundColor: '#222222',
            plotBackgroundImage: null,
            plotBorderWidth: 0,
            plotShadow: false
        },
        title: {
            text: 'Sensor <?php echo $j;?>'
        },
        credits: {
            enabled: false
        },
        pane: {
            startAngle: -150,
            endAngle: 150,
            background: [
                {
                    backgroundColor: {
                        linearGradient: { x1: 0, y1: 0, x2: 0, y2: 1 },
                        stops: [
                            [0, '#FFF'],
                            [1, '#333']
                        ]
                    },
                    borderWidth: 0,
                    outerRadius: '109%'
                },
                {
                    backgroundColor: {
                        linearGradient: { x1: 0, y1: 0, x2: 0, y2: 1 },
                        stops: [
                            [0, '#333'],
                            [1, '#FFF']
                        ]
                    },
                    borderWidth: 1,
                    outerRadius: '107%'
                },
                {
                    backgroundColor: '#FFF',
                    borderWidth: 0,

```

```

outerRadius: '105%',
    innerRadius: '103%'
    }
    ]
},
yAxis: {
    min: 0,
    max: <?php echo $max;?>,
    minorTickInterval: 'auto',
    minorTickWidth: 1,
    minorTickLength: 10,
    minorTickPosition: 'inside',
    minorTickColor : '#666',

    tickPixelInterval: 30,
    tickWidth: 2,
    tickPosition: 'inside',
    tickLength: 10,
    tickColor: '#666',
    labels: {
        step: 2,
        rotation: 'auto'
    },
    title: {
        text: 'mA'
    },
    plotBands: [{
        from: 0,
        to: 70, color: '#A9A9A9'
    }, {
        from: 70,
        to: <?php echo $yellow;?>,
        color: '#55BF3B'
    }, {
        from: <?php echo $yellow;?>,
        to: <?php echo $red;?>,
        color: '#DDDF0D'
    }, {
        from: <?php echo $red;?>,
        to: <?php echo $max;?>,
        color: '#DF5353'
    }]
}, series: [{
    name: 'Current Flow',
    data: [window.current[<?php echo $j;?>]],
    tooltip:
        { valueSuffix: ' mA' }
}],

```

```

    exporting: { enabled: false }
  },
  chart<?php echo $j;?>.setSize(230, 230, false);
});

```

Απο τον κώδικα φαίνεται οτι οι παράμετροι του γραφήματος όπως τα το όνομά του, όρια των χρωμάτων, οι μέγιστη τιμή και φυσικά η τιμή του ρεύματος ορίζονται ξεχωριστά για κάθε μετρητή. Η αρχικοποίηση αυτών των παραμέτρων καθώς και ο ορισμός της συνάρτησης drawChart γίνεται στο παρακάτω for loop :

```

for($j=1; $j<=$count; $j++){
    $nodetab
    =file_get_contents("http://uberdust.cti.gr/rest/testbed/".$testbed."/node/"
    $nodeUrn1 . $_GET['ip'] . "/capability/nodetype/tabdelimited/limit/10");
    $nodetype= explode("\t", $nodetab);
    if(isset($nodetype[1])&&($nodetype[1]=="powerstrip")){$max=1500;$red=1200;
    $yellow=800; }
    else{$max=20000; $red=16000; $yellow=12000;}
    unset($nodetype);
}

```

Ο λόγος που ελέγχεται το nodetype για τον ορισμό των ορίων των χρωμάτων είναι γιατί οι συσκευές που είναι συνδεδεμένες σε μετρητές τυπου powerstrip διαρέονται απο μικρότερης τάξης μεγέθους ρεύμα απ'οτι αυτές που είναι συνδεδεμένες σε powersensor.



Εικόνα 6.3.2.4 Παράδειγμα χρήσης gauge chart

Η επόμενη δυνατότητα που έχει ο μετρητής είναι ο έλεγχος του relay. Για να απεικονιστεί αυτή η δυνατότητα χρησιμοποιείται «διακόπτης» ώστε να θέτουμε την πρίζα και τον αισθητήρα σε λειτουργία ή να τα απενεργοποιούμε.

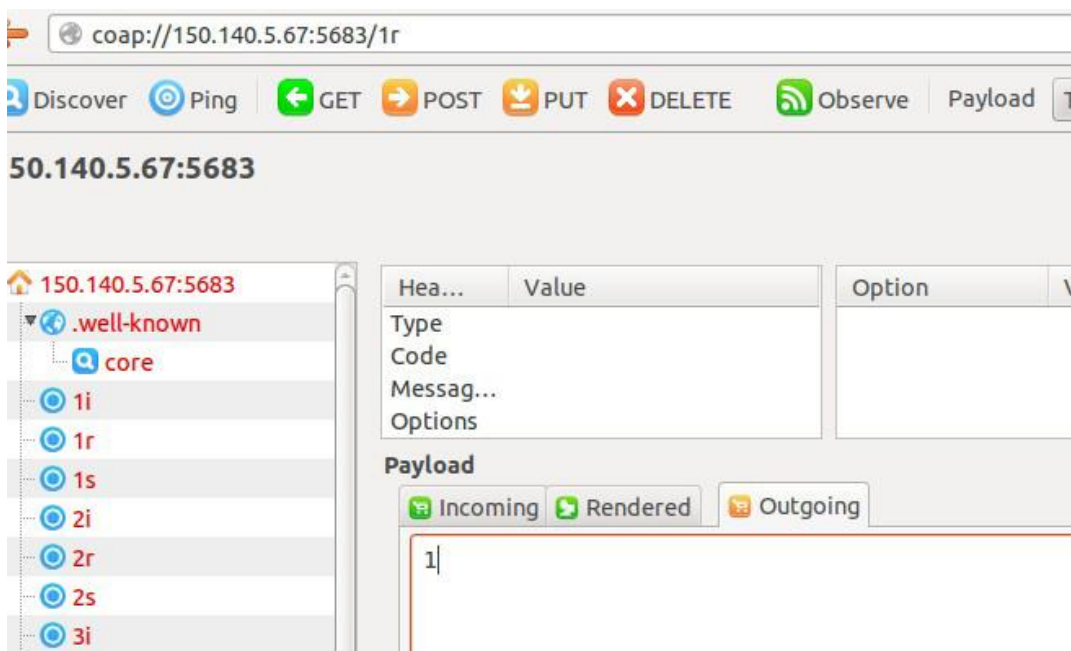


Εικόνα 6.3.2.5 Διακόπτης για τον έλεγχο της πρίζας

Προφανώς, αν ο μετρητής-πρίζα είναι σε λειτουργία, η τιμή του status relay είναι '1' οπότε η εικόνα που εμφανίζεται στο χρήστη είναι αυτή που ο διακόπτης είναι ενεργοποιημένος.

Στην περίπτωση που ο χρήστης κάνει κλικ στην εικόνα (δηλαδή κατεβάσει ή ανεβάσει τον διακόπτη), καλείται η συνάρτηση activate() με τα κατάλληλα ορίσματα η οποία εκτελεί την αντίστοιχη CoAP POST μέθοδο:

```
function activate(zone,status,testbed,nodeUrn,node,capability){
    if (status==1.0){
        window.state[zone]=0.0
        document.getElementById('myswitch'+zone+'.').src="/uberdust/sites/all/images/sensors/newswitch0.png";
        $.post(
            "http://uberdust.cti.gr/rest/testbed/"+testbed+"/node/"+nodeUrn+node+"/capability/"+capability+zone+"r/0/",0);
    }
    else{
        window.state[zone]=1.0;
        document.getElementById('myswitch'+zone+'.').src="/uberdust/sites/all/images/sensors/newswitch1.png";
        $.post(
            "http://uberdust.cti.gr/rest/testbed/"+testbed+"/node/"+nodeUrn+node+"/capability/"+capability+zone+"r/1/",1);
    }
}
```



Εικόνα 6.6 Παράδειγμα εκτέλεσης CoAP μεθόδου με χρήση επέκτασης περιηγητή

Για την συνεχή ενημέρωση της εφαρμογής και κατ'επέκταση του χρήστη, γίνεται χρήση websockets. Κάθε αντικείμενο ανοίγει μία σύνδεση με το websocket χρησιμοποιώντας τη συνάρτηση `connect()`. Η συγκεκριμένη συνάρτηση υλοποιήθηκε με τρόπο τέτοιο ώστε να παίρνει ως όρισμα τη συνάρτηση `onMessage()` η οποία καλείται κάθε φορά που έρχεται ένα νέο μήνυμα (στη συγκεκριμένη περίπτωση, τιμή).

Κλήσεις συναρτήσεων **connect**:

```
<?php
for($j=1;$j<=$count;$j++){
    ?>
    connect("uberdust.cti.gr", "<?php echo $nodeUrn1.$_GET['ip'];?>" , "<?php echo
$capability.$j;?>s",function(value2add){
        var val= new Array();
        var capability=" ";
        val=value2add.split(" ");
        capability=val[1].split(":");
        var count= capability.length;
        var id= capability[count-1].split("");
        window.current[id[0]]=val[8];
        console.log(id[0]+" "+val[8]);
    });
}
```

```

<?php
if(isset($indicator[$j])){ ?>
    connect("uberdust.cti.gr","<?php echo $nodeUrn1.$_GET['ip'];?>" , "<?php
echo $capability.$j;?>i",function(value2add){
    var val= new Array();
    var capability=" ";
    val=value2add.split(" ");
    capability=val[1].split(":");
    var count= capability.length;
    var id= capability[count-1].split("");
    if(val[8]==1.0){

document.getElementById("led"+id[0]).src="/uberdust/sites/all/images/sensors/led-
on-md.png";
    }else{

document.getElementById("led"+id[0]).src="/uberdust/sites/all/images/sensors/led-
off-md.png";
    }
});
<?php }
if(isset($status[$j])){ ?>
    connect("uberdust.cti.gr","<?php echo $nodeUrn1.$_GET['ip'];?>" , "<?php echo
$capability.$j;?>r",function(value2add){
    var val= new Array();
    var capability=" ";
    val=value2add.split(" ");
    capability=val[1].split(":");
    var count= capability.length;
    var id= capability[count-1].split("");
    if(val[8]==1.0){

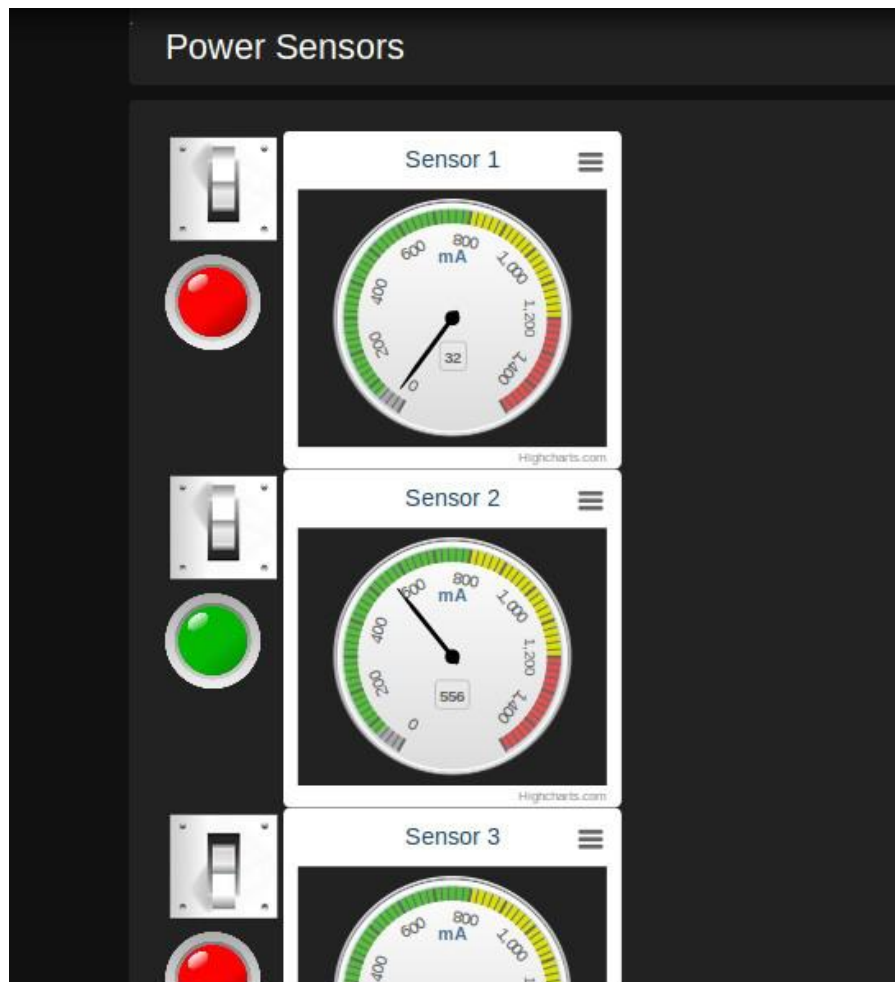
document.getElementById("myswitch"+id[0]).src="/uberdust/sites/all/images/sensor
s/newswitch1.png";
    }else{
document.getElementById("myswitch"+id[0]).src="/uberdust/sites/all/images/sensor
s/newswitch0.png";
    }
})
} } ?>

```

Κάθε φορά που έρχεται ένα νέο μήνυμα, το αντικείμενο που το αφορά ανανεώνει την τιμή του. Στην περίπτωση του δείκτη κατανάλωσης και του διακόπτη ανανεώνεται η αντίστοιχη εικόνα. Στην περίπτωση μίας νέας μέτρησης ρεύματος ανανεώνεται το γράφημα gauge.

Η συνάρτηση που πραγματοποιεί αυτή την ανανέωση είναι η παρακάτω:

```
function (chart<?php echo $j;?>){  
    if (!chart<?php echo $j;?>.renderer.forExport) {  
        setInterval(function () {  
            var point<?php echo $j;?> = chart<?php echo $j;?>.series[0].points[0],  
                newVal<?php echo $j;?>,  
                newVal<?php echo $j;?> = window.current[<?php echo $j; ?>];  
            point<?php echo $j;?>.update(parseFloat(newVal<?php echo $j;?>));  
        }, 3000);  
    }  
});
```



Εικόνα 6.7 Διεπαφή συσκευής μέτρησης κατανάλωσης ενέργειας

Το module power sensor παίρνει ως παράμετρο το αν ο χρήστης είναι administrator της σελίδας ή όχι. Αν είναι, τότε έχει τη δυνατότητα να δει όλα όσα περιγράφηκαν παραπάνω. Αν όχι, τότε δεν έχει τη δυνατότητα ελέγχου της πρίζας οπότε εμφανίζεται μόνο ο δείκτης κατανάλωσης και η τιμή του ρεύματος.

Στο παράδειγμα που ακολουθεί ο χρήστης είναι administrator, η συσκευή είναι συνδεδεμένη σε πολύπριζο και οι τιμές που φαίνονται αφορούν τις τρεις πρώτες πρίζες.

Συγκεκριμένα:

Power Sensor 1 :Είναι σε λειτουργία αλλά δεν καταναλώνεται τίποτα.

Power Sensor 2 : Μετράει 556 mA.

Power Sensor 3: Δεν είναι σε λειτουργία συνεπώς η κατανάλωση είναι μηδενική.

6.3.3 Chart Module

Το chart module χρησιμοποιείται απο τις περισσότερες σελίδες του ιστοτόπου. Είναι το module που καλείται απο κάθε σελίδα παρουσίασης εικονικού δωματίου (virtual room). Σε κάθε τέτοια σελίδα υπάρχουν τα γραφήματα για κάθε τύπου αισθητήρα (θερμοκρασίας, πίεσης κ.α) που υπάρχει εγκατεστημένος στο virtual room. Σ' αυτό το module ορίζονται οι συναρτήσεις που καλούνται για να δημιουργήσουν τα γραφήματα αυτά.

Η συνάρτηση που είναι υπεύθυνη για το σχεδιασμό των γραφημάτων παίρνει ως παράμετρο τον τύπο των δεδομένων που πρόκειται να απεικονίσει καθώς και τη χρονική περίοδο ενδιαφέροντος. Σύμφωνα με αυτά λοιπόν εκτελούνται οι κατάλληλες ενέργειες ώστε να ανακτηθούν απο το Uberdust τα δεδομένα:

```

$timezone=
file_get_contents("http://uberdust.cti.gr/rest/testbed/" . $testbed . "/timezone");
    $offset =
file_get_contents("http://uberdust.cti.gr/rest/testbed/" . $testbed . "/timezone/offset")
;
    $from=intval($from) ;
    $to = intval($to) ;
    $unit =
file_get_contents("http://uberdust.cti.gr/rest/testbed/" . $testbed . "/capability/" . $cap
_prefix.$capability . "/uom/raw");
    $minvalue =
file_get_contents("http://uberdust.cti.gr/rest/testbed/" . $testbed . "/capability/" . $cap
_prefix.$capability . "/minvalue/raw");
    $maxvalue =
file_get_contents("http://uberdust.cti.gr/rest/testbed/" . $testbed . "/capability/" . $cap
_prefix.$capability . "/maxvalue/raw");

```

Έπειτα καλείται η συνάρτηση drawChart() που δημιουργεί το γράφημα, της οποίας η υλοποίηση είναι παρόμοια με της drawChart του sensor module με τη διαφορά ότι αυτή στο πεδίο δεν έχει απλά μια τιμή αλλά ένα σύνολο δεδομένων σε μορφή array ζευγών (time,value). Παρακάτω φαίνεται το τμήμα του κώδικα που ανακτά αυτά τα δεδομένα:

```

data: [ <?
    if(($from==0)&&($to==0))
    {
        $tabdelimited =
file_get_contents("http://uberdust.cti.gr/rest/testbed/" . $testbed . "/node/" . $node_pr
efix.$node . "/capability/" . $cap_prefix.$capability . "/tabdelimited/limit/10");
    }else
    {
        $tabdelimited =
file_get_contents("http://uberdust.cti.gr/rest/testbed/" . $testbed . "/node/" . $node_pr
efix.$node . "/capability/" . $cap_prefix.$capability . "/tabdelimited/from/" . $from . "/to/" .
$to . "/");
    }
    $lines = explode("\n", $tabdelimited);
    unset($lines[count($lines)-1]);
    $firstRow = explode("\t", $lines[0]);
    unset($lines[0]);
    $data="[\".( intval($offset) + intval($firstRow[0])).\" , \".$firstRow[1].\"]";

```

```

foreach ($lines as $thisLine) {
    $row = explode("\t", $thisLine);
    $data="[".(intval($offset)+intval($row[0])).", ".$row[1]."],".$data;
}
print($data); ?>]

```

Επιπλέον, υλοποιήθηκε συνάρτηση μέσω της οποίας ο χρήστης επιλέγει το γράφημα που επιθυμεί επιλέγοντας βήμα-βήμα τον αισθητήρα. Δηλαδή, επιλέγει την πλατφόρμα δοκιμών(testbed), στη συνέχεια προβάλλονται οι διαθέσιμοι κόμβοι(nodes) και αφού επιλέξει κόμβο του εμφανίζονται οι συνδεδεμένοι αισθητήρες(capabilities). Η λειτουργία αυτή είναι υλοποιημένη σε javascript χρησιμοποιώντας AJAX_[13] request (get method). Όταν επιλεγθεί το testbed καλείται η συνάρτηση showNodes(), με παράμετρο το επιλεγμένο testbed, η οποία κάνει get τα nodes:

```

function showNodes(str){
    var
    url='/uber dust/misc/data.php?q=http://uber dust.cti.gr/rest/testbed/'+str+'/node/raw';
    $.get(url,function(data){
        var nodes=data.split("\n");
        if(str=="0"){
            document.capform.nodes.options[0]=new Option("Select Node:", "0");
            document.capform.nodes.options.length=1;
        }else{
            document.capform.nodes.options[0]=new Option("Select Node:", "0");
            for(var i=0;i<nodes.length;i++){
                document.capform.nodes.options[i+1]=new Option(nodes[i],
nodes[i]);
            }
            document.capform.nodes.options.length=nodes.length;
        }
    })
}

```

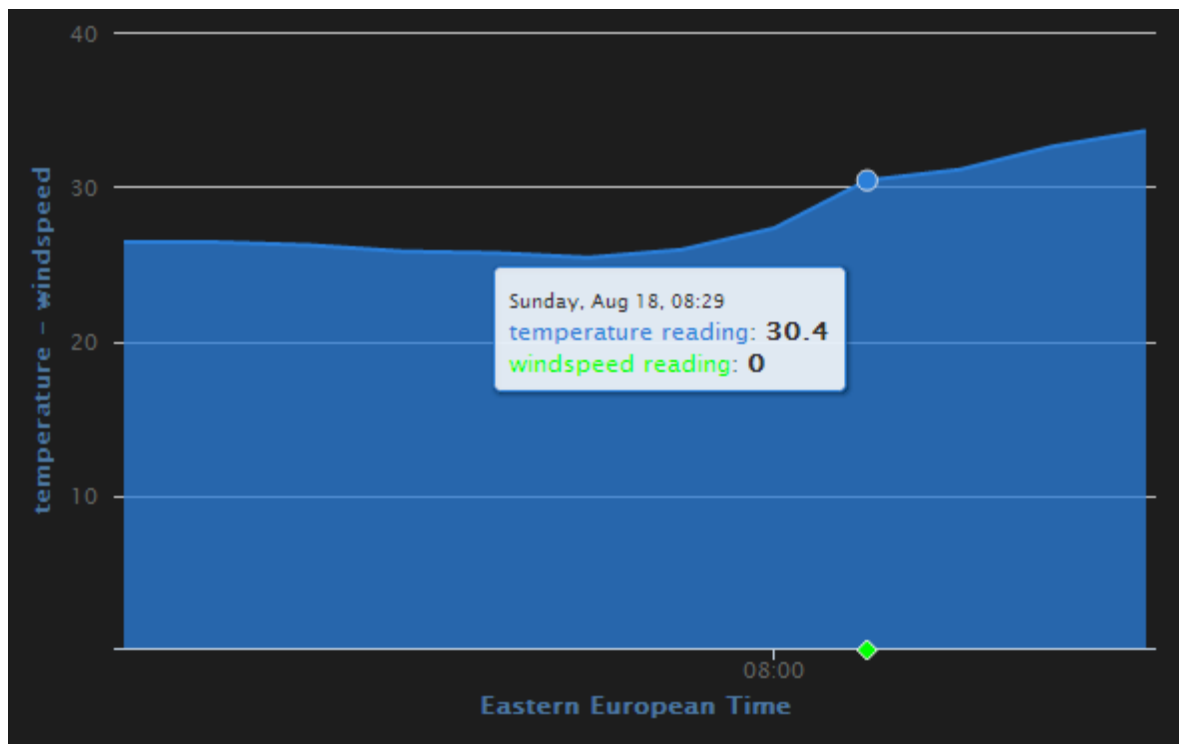
Αντίστοιχη συνάρτηση showCapabilities καλείται και όταν επιλεγθεί το node. Αφού επιλεγθεί και το capability, πρέπει να πατηθεί το κουμπί 'Show Chart' για να κληθεί η συνάρτηση onSubmit η οποία κάνει το τελικό http.get request για να ληφθούν τα δεδομένα:

```

function onSubmit()
{
    nodes=document.getElementById("nodes").value;
    capabilities=document.getElementById("capabilities").value;
    submit=document.getElementById("submit").value;
    from= document.getElementById("from").value;
    to= document.getElementById("to").value;
    var xmlhttp;
    if (window.XMLHttpRequest)
    {
        xmlhttp=new XMLHttpRequest();
    }
    else
    {
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlhttp.onreadystatechange=function()
    {
        if (xmlhttp.readyState==4 && xmlhttp.status==200)
        {
            document.getElementById("container").innerHTML=xmlhttp.responseText;
        }
        xmlhttp.open("GET","chart.php?t="+
Math.random()+"&node="+nodes+"&capabilities="+capabilities+"&from="+from+"&t
o="+to,true);
        xmlhttp.send();
    }
}

```

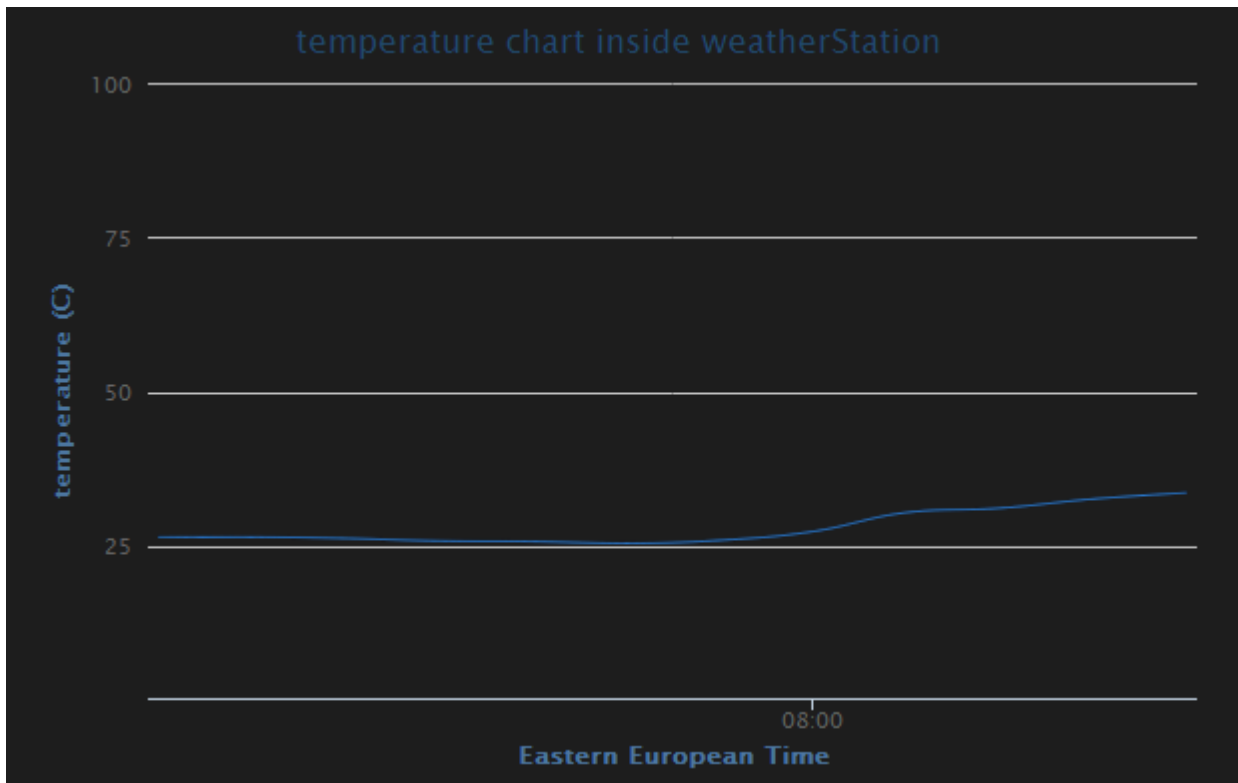
Έναν ακόμη τρόπο παρουσίασης των δεδομένων αποτελεί η προβολή γραφημάτων που παρουσιάζουν δυο γραφικές παραστάσεις ώστε να μπορούν να συγκριθούν οι μετρήσεις (Εικόνα 6.8).



Εικόνα 6.8 Γράφημα θερμοκρασίας-ταχύτητα αέρα

6.3.4 Weather Module

Το weather module υλοποιήθηκε για να χρησιμοποιηθεί από την αρχική σελίδα του ιστοτόπου όπου παρουσιάζονται οι μετρήσεις των αισθητήρων που έχουν τοποθετηθεί στον εξωτερικό χώρο της πλατφόρμα δοκιμών του ΙΤΥΕ Διόφαντος. Οι αισθητήρες που έχουν τοποθετηθεί αφορούν θερμοκρασία, υγρασία, ταχύτητα και διεύθυνση του αέρα. Για κάθε μία από αυτές δημιουργείται γράφημα με ιστορικά δεδομένα και gauge chart με την τρέχουσα τιμή.



Εικόνα 6.9 Ιστορικό θερμοκρασίας

Το μεγαλύτερο μέρος της υλοποίησης των συναρτήσεων του weather module έχουν ήδη περιγραφεί αφού είναι όμοιες με αυτές των sensor και chart modules.

Το διαφορετικό γράφημα που δημιουργείται στο παρόν module είναι αυτό της κατεύθυνσης του ανέμου στο οποίο πρέπει να ορίσουμε διαφορετικά το πεδίο τιμών:

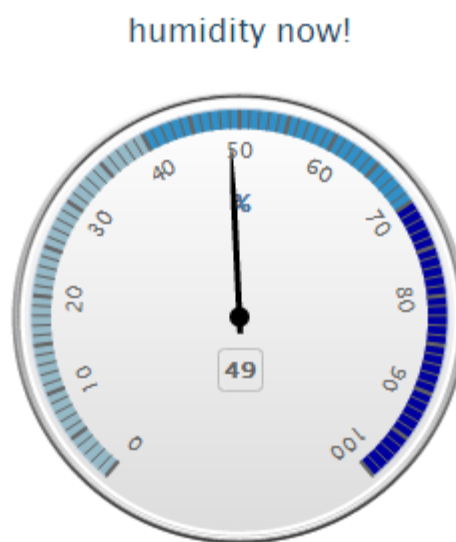
```
yAxis: {
  min: 0,
  max: 360,

  minorTickInterval: 30,
  tickInterval: 90,
  labels: {
    distance: 15,
    formatter: function () {
      return { 0: 'N', 90: 'W', 180: 'S', 270: 'W' }[this.value]; }
    }
  }
}
```

Επίσης παραλείπεται τελείως ο ορισμός χρωμάτων στα πεδία τιμών. Τα γράφημα που δημιουργούνται είναι τα παρακάτω.



Εικόνα 6.10 Κατεύθυνση ανέμου



Εικόνα 6.11 Τρέχουσα τιμή υγρασίας

Μελλοντικές κατευθύνσεις

Κατά τη διάρκεια εκπόνησης της διπλωματικής συζητήθηκαν διάφορες επεκτάσεις που θα μπορούσαν να υλοποιηθούν στηριζόμενες στην παρούσα εργασία.

Αρχικά, στην επέκταση περιηγητή και στο site θα μπορούσαν να υλοποιηθούν μέθοδοι για χρονοπρογραμματισμό των συσκευών, μιας και το σύστημα Uberdust υποστηρίζει κάτι τέτοιο. Παρόμοια λειτουργία θα μπορούσε να υλοποιηθεί και μέσω των On{x} συνταγών. Για παράδειγμα, μία εφαρμογή που είναι εφικτό να αναπτυχθεί είναι η «Όταν πλησιάζω στο σπίτι, άναψε το θερμοσίφωνο για ένα τέταρτο». Γενικά θα μπορούσαν να αναπτυχθούν εφαρμογές που να εκμεταλλεύονται τη δυνατότητα χρονοπρογραμματισμού των συσκευών αυξάνοντας τις δυνατότητες ελέγχου του χρήστη.

Επίσης θα μπορούσε να υλοποιηθούν επεκτάσεις και για άλλους περιηγητές που χρησιμοποιούνται συχνά όπως Firefox κ.α.

Όσον αφορά το Drupal site θα ήταν χρήσιμο να υλοποιηθούν γραφήματα σύγκρισης μετρήσεων μεταξύ χρονικών περιόδων. Έτσι για παράδειγμα, ο χρήστης θα μπορούσε να συγκρίνει την κατανάλωση των συσκευών του με σκοπό την εξοικονόμηση ενέργειας.

Βιβλιογραφία

- [1] Constrained RESTful Environments (CoRE) Working Group.
<https://datatracker.ietf.org/wg/core/charter/>, September 2013
- [2] Amaxilatis, Dimitrios, Georgitzikis, Vasileios, Giannakopoulos, Dimitrios and Chatzigiannakis, Ioannis, Employing Internet of Things Technologies for Building Automation, ETFA 2012, Poland, 2012.
- [3] Hypertext Constraint Protocol(HTTP)
http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- [4] Akribopoulos , Orestis, Amaxilatis, Dimitrios and Chatzigiannakis, Ioannis, Towards integrating IoT devices with the Web, ETFA 2012, Poland, 2012.
- [5] WebSocket Protocol
<http://tools.ietf.org/html/rfc6455>
- [6] O.Akribopoulos, D.Amaxilatis, V.Georgitzikis, M.Logaras, V.Keramidas, K.Kontodimas, E.Lagoudianakis, N.Nikoloutsakos, V.Papoutsakis, I.Prevezanos, G.Pyrgeris, S.Tsampas, V.Voutsas, and I.Chatzigiannakis: Making P-Space Smart: Integrating IoT technologies in a Multi-office environment.
- [7] HTML5
<http://www.w3.org/TR/2011/WD-html5-20110525/>
- [8] Highcharts
<http://www.highcharts.com/>
- [9] Weather Underground
<http://www.wunderground.com/>
- [10] On{x}
<https://www.onx.ms>
- [11] jQuery
<http://en.wikipedia.org/wiki/JQuery>

- [12] Chrome Local Storage
<http://developer.chrome.com/extensions/storage.html>
- [13] AJAX
[http://en.wikipedia.org/wiki/Ajax_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming))
- [14] Drupal
<http://en.wikipedia.org/wiki/Drupal>
- [15] Content Management System
http://en.wikipedia.org/wiki/Content_management_system
- [A] Akribopoulos, Orestis, Amaxilatis, Dimitrios, Dimakopoulos, Dimitrios and Chatzigiannakis, Ioannis, Smart Energy Monitoring and Management in Large Multi-Office Building Environments, in: 17th Panhellenic Conference on Informatics (PCI 2013), IEEE, PCI, Thessaloniki, 2013.
- [B] Akribopoulos, Orestis, Amaxilatis, Dimitrios, Chatzigiannakis, Ioannis, Georgitzikis, Vasileios and Logaras, Marios, Distributed Self-organized Societies in Action, chapter in the book "Distributed Self-organized Societies of Tiny Artefacts: Design & Implementation", Lulu Publishers, ISBN 5800059245538, 2011.