ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ

Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Ανάπτυξη location-based εφαρμογής για απομακρυσμένο έλεγχο συσκευών

Αλέξανδρος Κάντας

ΑΜ 3025

Επιβλέπων Καθηγητής:

Δρ. Σπυρίδων Παναγιωτάκης , Επίκουρος Καθηγητής

Ηράκλειο

2017

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον υπεύθυνο καθηγητή Δρ Σπυρίδωνα Παναγιωτάκη. Με άφησε να δουλέψω με το δικό μου ρυθμό και με τον τρόπο υλοποίησης που εγώ επιθυμούσα ενώ συνεχώς με καθοδηγούσε και με συμβούλευε για το καλύτερο δυνατό αποτέλεσμα. Επίσης θα ήθελα να ευχαριστήσω τους γονείς μου για την στήριξη που μου παρείχαν στις σπουδές μου.

Abstract

Over the past few years the Internet of Things (IoT) is developing at a rapid pace. Location services is a vital dimension of the IoT concept. This thesis, it's dealing with the build and the implementation way of a location-based IoT system. The system consisted of home devices, such cameras and sensors, attached to a Raspberry Pi and a web application. The web application allows connected users to monitor and manipulate the home devices' status. Also, the web application can track a user's location and sent notifications to other connected users when he arrives home. The Raspberry Pi and the web application are communicating through a cloud based server. Inside this thesis is explained in detail how this system works and the way that is implemented. This system is implemented on top of web technologies and programming interfaces such as Node.js, Geolocation API, Google Maps API, Vue.js etc.

Σύνοψη

Τα τελευταία χρόνια το Διαδίκτυο των Πραγμάτων (Internet of Things,IoT) αναπτύσσεται με ταχύ ρυθμό. Οι υπηρεσίες τοποθεσίας αποτελούν μια ζωτικής σημασίας διάσταση της IoT έννοιας. Η παρούσα πτυχιακή διαπραγματεύεται την δημιουργία και το τρόπο υλοποίησης ενός IoT συστήματος με βάση τη τοποθεσία. Το σύστημα αποτελείται από οικιακές συσκευές όπως κάμερες και αισθητήρες συνδεδεμένους σε ένα Raspberry Pi και από μια web εφαρμογή. Η web εφαρμογή επιτρέπει στους συνδεδεμένους χρήστες να ελέγχουν και να αλλάζουν την κατάσταση των οικιακών συσκευών. Επίσης, η web εφαρμογή μπορεί να καταγράψει την τοποθεσία ενός χρήστη και να στείλει ειδοποιήσεις στους άλλους χρήστες όταν αυτός φτάνει σπίτι. Το Raspberry Pi και η web εφαρμογή αλληλεπικοινωνούν μέσω ενός cloud-based Server. Εντός της πτυχιακής εξηγείται με λεπτομέρεια πως το εν λόγω σύστημα λειτουργεί και ο τρόπος με τον οποίο υλοποιήθηκε. Το σύστημα υλοποιήθηκε πάνω σε τεχνολογίες ιστού και προγραμματιστικές διεπαφές όπως Node.js, Geolocation API, Google Maps API, Vue.js κ.α.

Περιεχόμενα

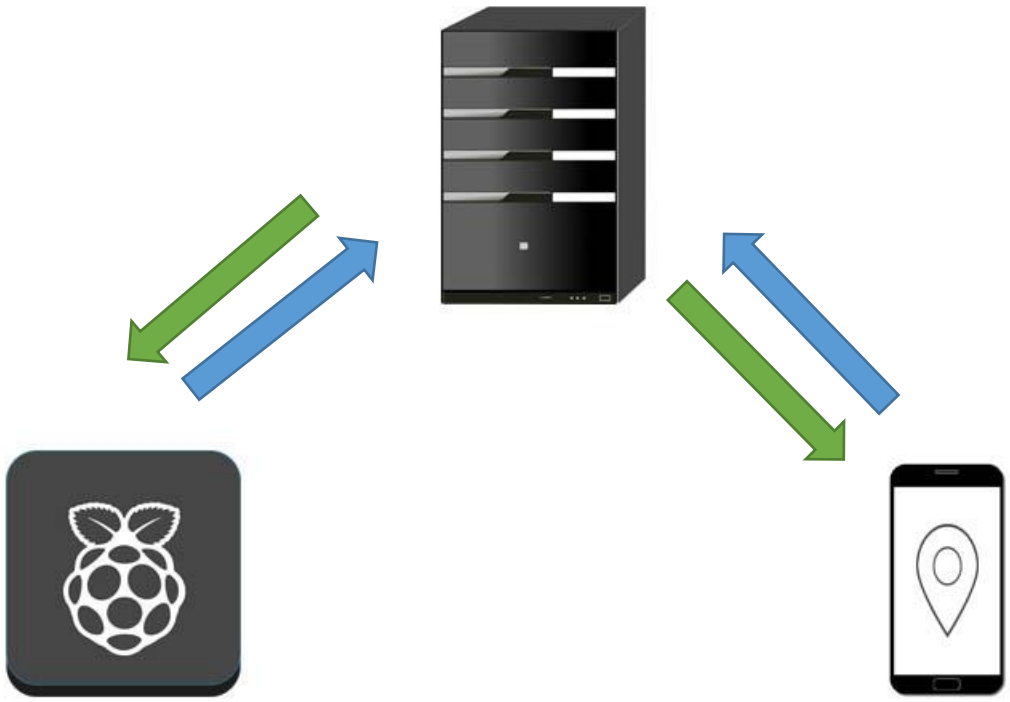
Πίνακας Εικόνων

Πίνακας αποσπασμάτων κώδικα

1. Εισαγωγή

1.1 Περίληψη

Στην παρούσα πτυχιακή εργασία καταγράφονται και αναλύονται όλες οι ενέργειες που έγιναν για την ανάλυση, την σχεδίαση, την υλοποίηση και την λειτουργεία ενός online συστήματος για απομακρυσμένο έλεγχο συσκευών με βάση τα στοιχεία της τοποθεσίας των χρηστών. Το σύστημα αποτελείται από τρία βασικά κομμάτια, τα οποία παρουσιάζονται περιληπτικά παρακάτω και αναλύονται στη συνέχεια της πτυχιακής



Εικόνα 1 Σχεδιάγραμμα που παρουσιάζει τη συνδεσμολογία μεταξύ του Server και των συσκευών.

Το σημαντικότερο κομμάτι του συστήματος είναι ο Server ο οποίος ο οποίος δρα σαν τον ενδιάμεσο κρίκο στον ποιο συνδέονται όλες οι συσκευές. Εκεί βρίσκεται το κύριο κομμάτι της επιχειρησιακής λογικής του συστήματος και αναλαμβάνει την μεταφορά πληροφοριών και αλληλεπίδραση μεταξύ των συσκευών.

Δεύτερο κομμάτι στο σύστημα είναι οι συσκευές που βρίσκονται στο σπίτι. Συγκεκριμένα βρίσκεται ένα Raspberry Pi στο οποίο είναι συνδεδεμένα κάμερες, αισθητήρες και LEDs. Αναλαμβάνει να στείλει πληροφορίες συσκευών στο Server ή να τροποποιήσει την κατάσταση των συσκευών με βάση τα δεδομένα που λαμβάνει από το Server.

Τρίτο κομμάτι του συστήματος είναι οι συσκευές των χρηστών. Πρόκειται για μια διαδικτυακή εφαρμογή η οποία προσφέρει τις κατάλληλες διεπαφές στις οποίες οι χρήστες οι χρήστες μπορούν να ελέγξουν την κατάσταση των οικιακών συσκευών καθώς και να αλληλεπιδράσουν με αυτές. Επίσης η διαδικτυακή εφαρμογή προσφέρει την ικανότητά παρακολούθησης της τοποθεσίας ενός χρήστη ώστε να μπορούν οι υπόλοιποι χρήστες να ενημερώνονται για την θέση του και αν έφτασε σπίτι καθώς επίσης και στον προγραμματισμό αυτοματοποιημένων ενεργειών στις οικιακές συσκευές βάσει της τοποθεσίας αυτής.

1.2 Κίνητρο και στόχοι της εργασίας

Το κίνητρο για τη διεξαγωγή αυτής της εργασίας είναι η δημιουργία ενός συστήματος **Internet of Things** το οποίο θα είναι χτισμένο σε σύγχρονες τεχνολογίες. Με τον όρο «Διαδίκτυο των πραγμάτων», στα αγγλικά «Internet of Things» ή σε συντομογραφία **IoT** αναφερόμαστε στην αλληλοεπικοινωνία μέσω του Internet υπολογιστικών συσκευών ενσωματωμένων σε καθημερινά αντικείμενα, ενεργοποιώντας τες στο να στέλνουν και να λαμβάνουν δεδομένα.

Η ραγδαία εξέλιξη της τεχνολογίας και του διαδικτύου στο πέρασμα των ετών έχει οδηγήσει στο να είναι εφικτό η δημιουργία πολύ σύνθετων συστημάτων και διεπαφών με μεγαλύτερη ευκολία και χαμηλό κόστος. Η δημιουργία ενός τέτοιου συστήματος προγραμματισμένο σε νέες τεχνολογίες αποτελούν μια πρόκληση η οποία συμβάλει στην αναπτυσσόμενη επιστημονική περιοχή έρευνας και εφαρμογών IoT και προσφέρει στον συγγραφέα εμπλουτισμό γνώσεων και εμπειρία στην δημιουργία τέτοιον συστημάτων.

Σκοπός της παρούσης πτυχιακής εργασίας είναι η δημιουργία ενός ολοκληρωμένου IoT συστήματος με ωραίες και χρηστικές διεπαφές αλληλεπίδρασης κάνοντας χρήση των πιο σύγχρονων τεχνολογιών στο τομέα.

Αναλυτικά το σύστημα είναι χτισμένο με ένα Raspberry Pi στο οποίο είναι συνδεδεμένες διάφορες συσκευές όπως κάμερες και LEDs τα οποία ελέγχονται από μια NodeJS εφαρμογή η οποία συμπεριφέρεται σαν client και αναλαμβάνει να στέλνει και να λάβει πληροφορίες από και προς το Server. Ο Server, χτισμένος επίσης με NodeJS, είναι αυτός που παίρνει επεξεργάζεται και στέλνει δεδομένα από το Raspberry Pi και τις συσκευές των χρηστών. Οι χρήστες έχουν πρόσβαση σε μια web εφαρμογή χτισμένη στο JavaScript Framework Vue.js και με components από το CSS Framework Bulma CSS. Η web εφαρμογή προφέρει ένα όμορφο περιβάλλον στο οποίο ένας χρήστης μπορεί να ενημερωθεί για την κατάσταση των συσκευών στο Raspberry Pi και να δει την τοποθεσία των άλλων χρηστών ή και την δικιά του. Η τοποθεσία λαμβάνεται με χρήση των geolocation API που έχουν οι περιηγητές διαδικτύου και εμφανίζεται στην οθόνη σε χάρτη με την χρήση Google Maps API.

Στόχος είναι η δημιουργία ενός ολοκληρωμένου IoT συστήματος με χρήση location based χαρακτηριστικών και συγχρόνων τεχνολογιών web το οποίο μπορεί να βελτιώσει την καθημερινότητα των χρηστών προσφέροντας απομακρυσμένο έλεγχο του σπιτιού και αυτοματισμούς με βάση την τοποθεσία τους.

1.3 Δομή εργασίας

Αυτή η εργασία χωρίστηκε σε τμήματα στοχεύοντας στην ευκολότερη κατανόηση και εκτίμηση της. Έχει δομηθεί σε πέντε κεφάλαια, το πρώτο περιλαμβάνει την εισαγωγή όπου γίνεται μια περιληπτική παρουσίαση του θέματος της πτυχιακής εργασίας. Πιο συγκεκριμένα, παρουσιάζεται μια μικρή περιγραφή του IoT συστήματος της πτυχιακής και οι λόγοι που στάθηκαν ως κίνητρο την δημιουργία του. Επιπλέον, γίνεται αναφορά στους στόχους και τον σκοπό της εργασίας αυτής.

Στο δεύτερο κεφάλαιο γίνεται αναφορά σε τεχνολογίες και βασικές έννοιες που σχετίζονται άμεσα με την εργασία. Για κάθε τεχνολογία παραθέτετέ μια περιγραφή της, εγκυκλοπαιδικές πληροφορίες σχετικά με αυτή και πληροφορίες για τις τρέχουσες εφαρμογές της. Στόχος είναι να προσφέρουν στον αναγνώστη πληροφορίες για την κάθε τεχνολογία οι οποίες βοηθούν στην κατανόηση για το πως υλοποιήθηκε το σύστημα.

Στο τρίτο κεφάλαιο γίνεται μια αναφορά σε υλικό, πρότυπα, γλώσσες προγραμματισμού, προγραμματιστικές διεπαφές και πλαίσια εφαρμογών που χρησιμοποιήθηκαν για την υλοποίηση της πτυχιακής.

Στο τέταρτο γίνεται αναλυτική περιγραφή πως συνδυάστηκαν οι τεχνολογίες του τρίτου κεφαλαίου για την τελική υλοποίηση. Η αναφορά αυτή γίνεται με την παρουσίαση σχεδιαγραμμάτων, στιγμιότυπων οθόνης, φωτογραφιών και με αποσπάσματα από τον κώδικα της εφαρμογής.

Στο πέμπτο κεφάλαιο παρουσιάζονται τα αποτελέσματα του συστήματος που αναπτύχθηκε, τα συμπεράσματα και προτάσεις για μελλοντικές επεκτάσεις της εφαρμογής.

Στο παράρτημα της πτυχιακής περιλαμβάνεται ο κώδικας της εφαρμογής σε πλήρη και όχι αποσπασματική μορφή.

2. Τεχνολογίες και Βασικές Έννοιες

2.1 Διαδίκτυο των πραγμάτων

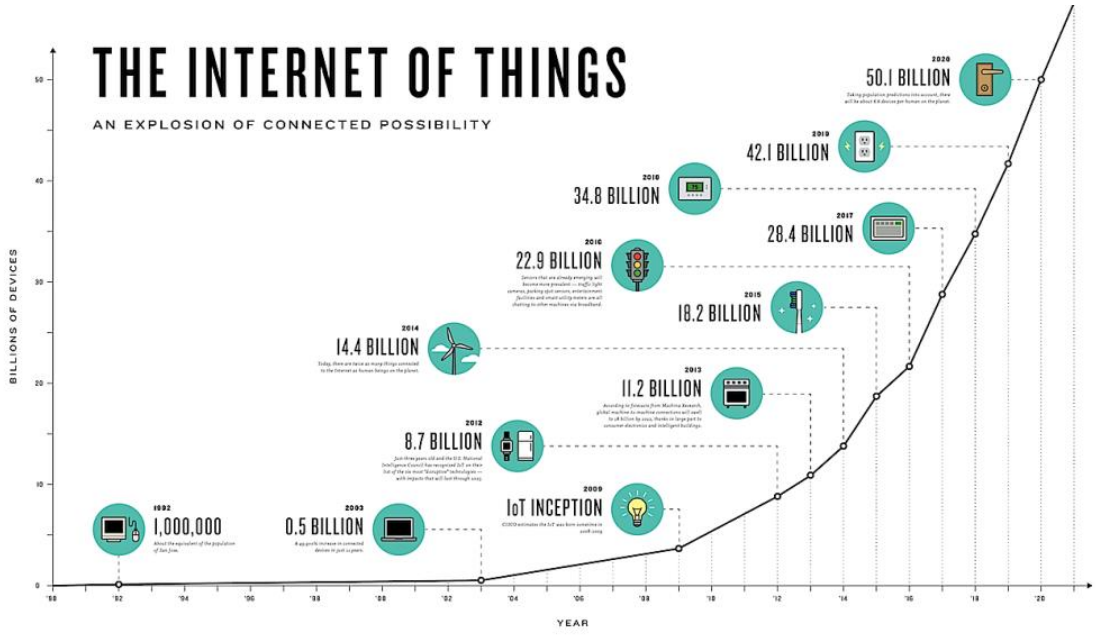
Εικόνα 2 Γραφική αναπαράσταση του Internet of Things

Το Διαδίκτυο των πραγμάτων (IoT) είναι το δίκτυο των φυσικών συσκευών, των οχημάτων, των οικιακών συσκευών και άλλων στοιχείων ενσωματωμένων σε ηλεκτρονικές συσκευές, λογισμικά, αισθητήρες, ενεργοποιητές και ικανότητα σύνδεσης σε δίκτυο που επιτρέπει στα αντικείμενα αυτά να συνδέουν και να ανταλλάσσουν δεδομένα. Κάθε «πράγμα» είναι μοναδικά αναγνωρίσιμο μέσω του ενσωματωμένου υπολογιστικού συστήματος που διαθέτει και είναι σε θέση να αλληλεπιδράσει μέσα στην υπάρχουσα υποδομή του Διαδικτύου.

Το IoT εξελίχθηκε με την γρήγορη διάδοση του ασύρματου Internet και των ενσωματωμένων αισθητήρων και έτσι οι άνθρωποί άρχισαν να αντιλαμβάνονται ότι η τεχνολογία αυτή θα μπορούσε να είναι προσωπικό αλλά και επαγγελματικό εργαλείο. Ο όρος Internet of Things επινοήθηκε στα τέλη της δεκαετίας του 1990 από τον επιχειρηματία Kevin Ashton, ο οποίος ήταν μέρος μιας ομάδας που ανακάλυψε τον τρόπο να συνδέσει τα αντικείμενα με το διαδίκτυο μέσω μιας ετικέτας RFID.

Το διαδίκτυο των πραγμάτων είναι εδώ και γίνεται όλο ένα και πιο ενδιαφέρον τομέας απασχόλησης μεταξύ των γιγάντων της τεχνολογίας και των επιχειρηματικών κοινοτήτων. Η προώθησή του δεν είναι αβάσιμη, καθώς υπάρχουν αρκετά στοιχεία για να υποστηρίξουν την επιτυχία του στα επόμενα χρόνια.

Το 2008 τα «πράγματα» ξεπέρασαν τους ανθρώπους από την άποψη του αριθμού των συνδέσεων. Όπως μιλάμε, υπάρχουν περίπου 5 δισεκατομμύρια συνδεδεμένα πράγματα, από τα οποία τα 1,5 δισεκατομμύρια είναι μόνο smartphones. Η εταιρεία έρευνας και συμβουλευτικής Gartner προβλέπει ότι θα εγκατασταθούν 26 δισεκατομμύρια συσκευές IoT το 2020. Μια από τις πιο εντυπωσιακές προσδοκίες στην ιστορία του Διαδικτύου των πραγμάτων είναι η εξαγορά για 3,2 δισεκατομμύρια δολάρια της Nest Labs - τη γνωστή εταιρία του πολύ δημοφιλούς στις ΗΠΑ έξυπνου θερμοστάτη - από την Google (πλέον Alphabet.Inc). Πέρυσι η αγορά φορητών συσκευών έχει αυξηθεί κατά 223%, με τις πωλήσεις Fitbit και Apple Watch να ανέρχονται σε 4,4 εκατομμύρια και 3,6 εκατομμύρια συναλλαγές. Η αγορά RFID αναμένεται να διπλασιαστεί σχεδόν μέχρι το 2020. Η Κίνα και οι ΗΠΑ αναμένεται να ηγηθούν όσον αφορά την αγορά M2M (1).



Εικόνα 3 Σχεδιάγραμμα που παρουσιάζει την τάση των συνδεδεμένων IoT συσκευών ως προς το έτος

Η εταιρεία κατασκευής εξοπλισμού δικτύων Cisco προβλέπει πως μια από τις πιο κερδοφόρες εφαρμογές του Internet των Πραγμάτων θα είναι διαφημιστικές πινακίδες που θα είναι μόνιμα online. Εξίσου αποδοτικές θα είναι εφαρμογές λειτουργίας και παρακολούθησης της παραγωγής σε «έξυπνα» εργοστάσια που θα «μιλούν» με τις αποθήκες, τα φορτηγά κ.λπ. επιτυγχάνοντας καλύτερη ροή και διάθεση των προϊόντων.

Η εταιρεία ερευνών και συμβούλων Machina διεξήγαγε πρόσφατα μια έρευνα, σύμφωνα με την οποία το 2020 εφαρμογές όπως τα διόδια και η ελεγχόμενη κυκλοφορία θα αντιπροσωπεύουν μια αγορά αξίας $100 δισ., ενώ άλλα $30 δισ. υπολογίζεται ότι θα είναι η αγορά διαχείρισης «έξυπνων» παρκόμετρων που θα «εκπέμπουν» τη θέση τους στο διαδίκτυο.

Στο Σινσινάτι των ΗΠΑ τοποθετήθηκαν πρόσφατα αισθητήρες στους κάδους σκουπιδιών και ανακύκλωσης και μπήκε σε εφαρμογή ένα πρόγραμμα χρέωσης ανάλογα με τον όγκο των απορριμμάτων του κάθε σπιτιού. Το αποτέλεσμα είναι ότι τα απορρίμματα μειώθηκαν κατά 17% και η ανακύκλωση αυξήθηκε κατά 49%, με το οικονομικό και περιβαλλοντικό όφελος να είναι προφανές.

Την ίδια ώρα, πόλεις όπως η Ντόχα του Κατάρ, το Σάο Πάολο στη Βραζιλία και το Πεκίνο στην Κίνα τοποθετούν ήδη αισθητήρες στο σύστημα ύδρευσης για την έγκαιρη ανίχνευση των διαρροών και την άμεση ειδοποίηση σε περίπτωση που κάποιος αγωγός σπάσει ή υποστεί ζημιά. Έχουν πετύχει μείωση 50% του νερού που χάνεται από διαρροές.

Το ίδρυμα ερευνών McKinsey Global Institute έχει υπολογίσει πως με τη χρήση «έξυπνων» δικτύων ηλεκτρικής ενέργειας που θα αυξομειώνουν την τιμή και τη διάθεση ανάλογα με τη ζήτηση, το όφελος θα κυμαίνεται από 200 έως $500 δισ. ετησίως από το 2025 και μετά!

2.1.1 Εφαρμογές του Internet of Things

Το IoT προσφέρει νέες πηγές δεδομένων και νέα επιχειρηματικά μοντέλα που μπορούν να ενισχύσουν την παραγωγικότητα σε διάφορους κλάδους. Το IoT βρίσκει εφαρμογές τόσο σε επίπεδο καταναλωτή όσο και σε βιομηχανικό επίπεδο. Παρακάτω μια λίστα με τομείς που συναντάμε IoT συστήματα:

Υγειονομική Περίθαλψη

Πολλοί άνθρωποι έχουν ήδη υιοθετήσει wearable συσκευές για να παρακολουθούν την φυσική τους κατάσταση, τον ύπνο ή άλλες συνήθειες τους. Αυτά θα μπορούσαν να είναι ένα πολύ μικρό δείγμα του πώς το IoT συνδυάζεται με τον κλάδο της υγείας. Συσκευές παρακολούθησης ασθενών, ηλεκτρονικά αρχεία και άλλα έξυπνα αξεσουάρ μπορούν να σώσουν ζωές.

Βιομηχανική Παραγωγή

Ο έλεγχος του δικτύου και η διαχείριση του εξοπλισμού κατασκευής ή ο έλεγχος των παραγωγικών διαδικασιών φέρνουν το IoT στον τομέα των βιομηχανικών εφαρμογών και των έξυπνων κατασκευών.

Ο όρος βιομηχανικό Διαδίκτυο των πραγμάτων (industrial Internet of things, IIoT) απαντάται συχνά στις μεταποιητικές βιομηχανίες, αναφερόμενος στο βιομηχανικό υποσύνολο του IoT. Η τεχνολογία της κατασκευής θα μπορούσε να παράγει τόσο μεγάλη αξία για την επιχείρηση, που θα οδηγήσει τελικά στην τέταρτη βιομηχανική επανάσταση, και έτσι θα αποκαλείται η Βιομηχανία 4.0 . Εκτιμάται ότι στο μέλλον, οι επιτυχημένες εταιρείες θα μπορούν να αυξήσουν τα έσοδά τους μέσω του διαδικτύου, δημιουργώντας νέα επιχειρηματικά μοντέλα και βελτιώνοντας την παραγωγικότητα, εκμεταλλευόμενοι τα αναλυτικά στοιχεία για την καινοτομία και μεταμορφώνοντας το εργατικό δυναμικό.

Λιανεμπόριο

Τόσο οι καταναλωτές όσο και τα καταστήματα μπορούν να επωφεληθούν από IoT. Τα καταστήματα, για παράδειγμα, θα μπορούσαν να χρησιμοποιήσουν IoT για σκοπούς παρακολούθησης των αποθεμάτων ή της ασφάλειας ενώ οι καταναλωτές μπορούν να έχουν μία περισσότερο εξατομικευμένη εμπειρία αγορών μέσω των δεδομένων που συλλέγονται από τους αισθητήρες ή τις κάμερες.

Γεωργία

Το IoT συμβάλλει σημαντικά στην καινοτομία των γεωργικών μεθόδων. Οι γεωργικές προκλήσεις που προκαλούνται από την αύξηση του πληθυσμού και την αλλαγή του κλίματος έχουν καταστήσει μια από τις πρώτες βιομηχανίες που χρησιμοποιούν το IoT. Η ενσωμάτωση ασύρματων αισθητήρων με εφαρμογές αγροτικών κινητών εφαρμογών και πλατφόρμες σύννεφο βοηθά στη συλλογή ζωτικών πληροφοριών σχετικά με τις περιβαλλοντικές συνθήκες - όπως η θερμοκρασία, η βροχόπτωση, η υγρασία, η ταχύτητα του ανέμου, η μόλυνση από παράσιτα, το χούμο εδάφους ή τα θρεπτικά συστατικά. μπορούν να χρησιμοποιηθούν για τη βελτίωση και την αυτοματοποίηση των γεωργικών τεχνικών, να λάβουν τεκμηριωμένες αποφάσεις για τη βελτίωση της ποιότητας και της ποσότητας και να ελαχιστοποιήσουν τους κινδύνους και τα απόβλητα. Η παρακολούθηση των καλλιεργειών ή των καλλιεργειών με βάση την εφαρμογή μειώνει επίσης τις δυσκολίες διαχείρισης των καλλιεργειών σε πολλαπλές τοποθεσίες. Για παράδειγμα, οι αγρότες μπορούν τώρα να ανιχνεύσουν ποιες χρειάζονται λίπασμα, εάν η γη είναι πολύ ξηρή και να προβλέπουν μελλοντικά συμβάντα.

Ενέργεια

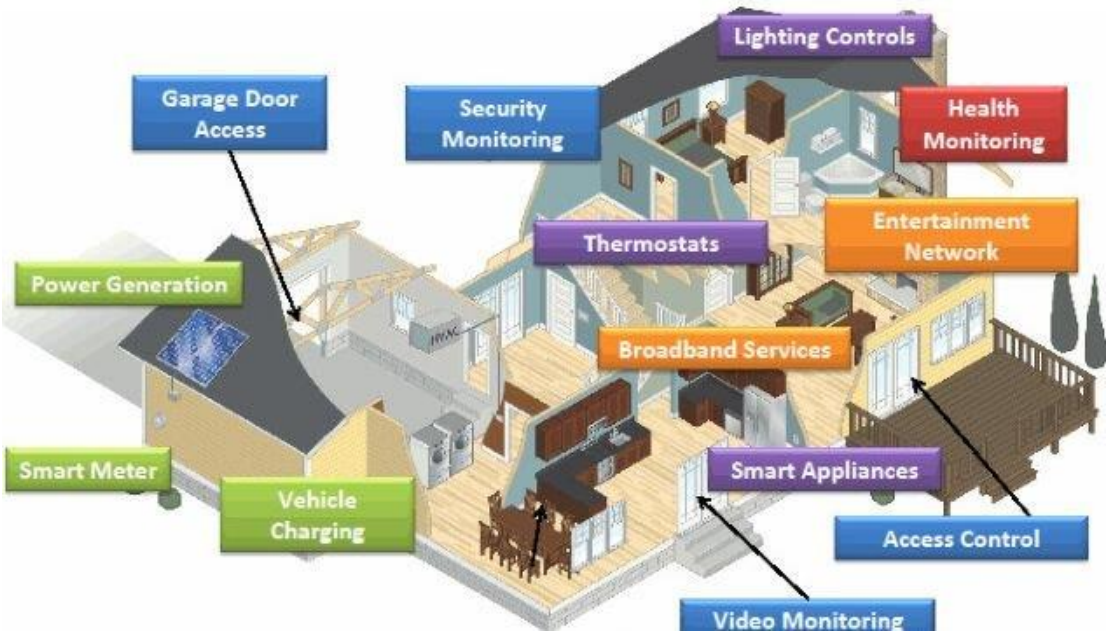
Οι έξυπνοι μετρητές (smart meters), όχι μόνο συλλέγουν δεδομένα αυτόματα, αλλά καθιστούν και δυνατή την εφαρμογή analytics για την παρακολούθηση και τη διαχείριση της χρήσης της ενέργειας. Παρομοίως, αισθητήρες σε συσκευές όπως οι ανεμόμυλοι μπορούν να παρακολουθούν τα δεδομένα και να χρησιμοποιούν προγνωστική μοντελοποίηση ώστε να προγραμματιστεί η διακοπή λειτουργίας για πιο αποδοτική χρήση της ενέργειας.

Αναμένεται ότι οι συσκευές IoT θα ενσωματωθούν σε όλες τις μορφές συσκευών που καταναλώνουν ενέργεια (διακόπτες, ρευματοδότες, βολβοί, τηλεοράσεις κλπ.) Και θα μπορούν να επικοινωνούν με την εταιρεία παροχής ηλεκτρικής ενέργειας , κατανάλωσης ενέργειας. Τέτοιες συσκευές θα παρείχαν επίσης τη δυνατότητα στους χρήστες να ελέγχουν εξ αποστάσεως τις συσκευές τους ή να τις διαχειρίζονται κεντρικά μέσω μιας διασύνδεσης βασισμένης στο cloud και να επιτρέπουν προηγμένες λειτουργίες (όπως π.χ., ενεργοποίηση ή απενεργοποίηση των συστημάτων θέρμανσης, συνθήκες φωτισμού κ.λπ.).

Περιβαλλοντική παρακολούθηση

Οι εφαρμογές περιβαλλοντικής παρακολούθησης του IoT χρησιμοποιούν συνήθως αισθητήρες για να βοηθήσουν στην προστασία του περιβάλλοντος παρακολουθώντας την ποιότητα του αέρα ή των υδάτων ,συνθήκες ατμόσφαιρας ή εδάφους και μπορούν ακόμη να συμπεριλάβουν τομείς όπως η παρακολούθηση των μετακινήσεων της άγριας πανίδας. Η ανάπτυξη τέτοιων συσκευών σημαίνει επίσης ότι άλλες εφαρμογές όπως συστήματα έγκαιρης προειδοποίησης σεισμού ή τσουνάμι τα οποία μπορούν επίσης να χρησιμοποιηθούν από τις υπηρεσίες έκτακτης ανάγκης για την παροχή αποτελεσματικότερης βοήθειας. Οι συσκευές IoT στην εφαρμογή αυτή καλύπτουν συνήθως μια μεγάλη γεωγραφική περιοχή και μπορούν επίσης να είναι κινητές.

2.1.2 Έξυπνα σπίτια και IoT



Εικόνα 4 Γραφική αποτύπωση ενός έξυπνου σπιτιού

Οι συσκευές IoT αποτελούν μέρος της ευρύτερης έννοιας του αυτοματισμού στο σπίτι. Τα μεγάλα έξυπνα οικιακά συστήματα χρησιμοποιούν ένα κύριο διανομέα ή ελεγκτή για να παρέχουν στους χρήστες κεντρικό έλεγχο για όλες τις συσκευές τους. Αυτές οι συσκευές μπορούν να περιλαμβάνουν φωτισμό, θέρμανση και κλιματισμό, μέσα ενημέρωσης και συστήματα ασφαλείας. Η ευκολία χρήσης είναι το πιο άμεσο όφελος για τη σύνδεση αυτών των λειτουργιών. Τα μακροπρόθεσμα οφέλη μπορούν να περιλαμβάνουν τη δυνατότητα δημιουργίας ενός φιλικότερου προς το περιβάλλον σπίτι με την αυτοματοποίηση ορισμένων λειτουργιών, όπως η εξασφάλιση της απενεργοποίησης των φώτων και των ηλεκτρονικών. Ένα από τα σημαντικότερα εμπόδια στην απόκτηση έξυπνης οικιακής τεχνολογίας είναι το υψηλό αρχικό κόστος.

Ένα παράδειγμα εφαρμογής έξυπνης κατοικίας είναι η παροχή βοήθειας σε άτομα με αναπηρίες και ηλικιωμένους. Αυτά τα οικιακά συστήματα χρησιμοποιούν τεχνολογία υποβοήθησης για την κάλυψη συγκεκριμένων αναγκών ενός ιδιοκτήτη. Ο φωνητικός έλεγχος μπορεί να βοηθήσει τους χρήστες με περιορισμούς όρασης και κινητικότητας, ενώ τα συστήματα συναγερμού μπορούν να συνδεθούν απευθείας με κοχλιακά εμφυτεύματα που φοριούνται από χρήστες με προβλήματα ακοής. Μπορούν επίσης να είναι εξοπλισμένα με πρόσθετα χαρακτηριστικά ασφαλείας. Αυτά τα χαρακτηριστικά μπορούν να περιλαμβάνουν αισθητήρες που παρακολουθούν για ιατρικές καταστάσεις έκτακτης ανάγκης όπως πτώσεις ή επιληπτικές κρίσεις. Η τεχνολογία Smart home που εφαρμόζεται με τον τρόπο αυτό μπορεί να προσφέρει στους χρήστες περισσότερη ελευθερία και υψηλότερη ποιότητα ζωής.

Ένα δεύτερο παράδειγμα εφαρμογής έξυπνης κατοικίας ακόμα πιο εξελιγμένη είναι κάποιος μπορεί να καθοδηγήσει τη συνδεδεμένη συσκευή του στο σπίτι ακόμα και από μακριά. Εάν κάποιος, για παράδειγμα, εγκαταλείψει το γραφείο, είναι δυνατό να πει μια συνδεδεμένη συσκευή κλιματιστικού μέσω έξυπνου τηλεφώνου για να κρυώσει το σπίτι σε μια ορισμένη θερμοκρασία. Σε γενικές γραμμές, οι συσκευές Smart Home διευκολύνουν τη ζωή στο σπίτι και μας δίνουν τη δυνατότητα να κάνουμε πολλά πράγματα ταυτόχρονα.

Η ιδέα ενός Smart Home δεν είναι καινούργια. Το 1923, ο Ελβετός αρχιτέκτονας, Le Corbusier χαρακτήρισε ένα σπίτι ως "μια μηχανή για να ζεις μέσα". Από τότε, έχουμε δει πολλές προσπάθειες για να μεταμορφώσουμε αυτό το όραμα στην πραγματικότητα, αλλά στις περισσότερες περιπτώσεις η υιοθεσία περιοριζόταν σε ερασιτεχνικές λύσεις.

Μόνο πρόσφατα έχουμε αρχίσει να βλέπουμε θετικά σημάδια που υποδηλώνουν ότι η έξυπνη εγχώρια αγορά ετοιμάζεται να διασχίσει το χάσμα. Αυτό που είναι διαφορετικό τώρα έχει να κάνει κυρίως ότι οι τεχνολογίες smart-home γίνονται μέρος μιας πολύ μεγαλύτερης εικόνας. Τώρα μπαίνουμε στην εποχή του Διαδικτύου των πραγμάτων, της νέας μετασχηματιστικής προσέγγισης σε έξυπνα συστήματα. Οι μικρές συσκευές ενσωματωμένες με τεχνίτη νοημοσύνη μετακινούνται σε κάθε τομέα της κοινωνίας, ενισχύοντας τα υπάρχοντα πράγματα και δημιουργώντας νέες φυλές συσκευών και υπηρεσιών.

Το πιο σημαντικό είναι ότι το IoT αλλάζει την παραδοσιακή μας προσέγγιση για την κατασκευή συσκευών, συστημάτων και υπηρεσιών. Ο οριζόντιος χαρακτήρας του IoT σπάζει τα εμπόδια των δεκαετιών μεταξύ των τομέων εφαρμογής και μας επιτρέπει να αρχίσουμε να αναπτύσσουμε εφαρμογές και υπηρεσίες που αξιοποιούν τις πληροφορίες που συλλέγονται σε τοπικό επίπεδο σε συνδυασμό με τα δεδομένα που συλλέγονται παντού για να παράγουν νέα και συναρπαστικά αποτελέσματα. Η λειτουργία ενός προγραμματιζόμενου θερμοστάτη στο παρελθόν ήταν περίπου τόσο δύσκολη και επίπονη όσο και ο προγραμματισμός ενός VCR.

Στη συνέχεια άρχισε να πειραματιζόμαστε με συσκευές που μπορούν να ανιχνεύσουν την παρουσία μας και να μάθουν τις καθημερινές μας συνήθειες. Τώρα έχουμε μάθει την ικανότητα να δημιουργούμε συστήματα που μαθαίνουν μόνα τους και γνωρίζουν πράγματα όπως πόσο μακριά βρισκόμαστε από το σπίτι, πόσο ζέστη ή κρύο θα κάνει και αν έχουμε ταΐσει το σκύλο, όλα μέσω μιας απλής εφαρμογής στο έξυπνο τηλέφωνό μας που μπορεί εργάζονται σε συνεννόηση με ένα έξυπνο σπίτι.

Ορισμένες εταιρείες εκκίνησης εφαρμόζουν την ίδια προσέγγιση για τον έλεγχο των συστημάτων καταιωνιστήρων που υγραίνουν τους χλοοτάπητες μας. Αυτά τα συστήματα συγκεντρώνουν τις τοπικές πληροφορίες θερμοκρασίας και υγρασίας και χρησιμοποιούν τις προβλέψεις καιρού για να καθορίσουν εάν το γκαζόν πρέπει να ποτίζεται και πόσο νερό θα πρέπει να χρησιμοποιήσουν. Για να συνοψίσουμε τα έξυπνα σπίτια για πρώτη φορά αρχίζουν να περιλαμβάνουν συστήματα που δεν απαιτούν πολύπλοκη διαμόρφωση και αλληλεπίδραση με τον χρήστη. Αυτά τα συστήματα και συσκευές σχεδιάζονται για να λειτουργούν στο background για να κάνουν τη ζωή των χρηστών πιο άνετη, βελτιώνοντας ταυτόχρονα την αποτελεσματικότητα με την οποία χρησιμοποιούμε ενέργεια, νερό και άλλους σπάνιους πόρους.

2.2 Το Internet of Things σήμερα

Πολλές εφαρμογές και από πολλές μεγάλες εταιρείας έχουν βγάλει προϊόντα για οικιακή και βιομηχανική χρήση τα οποία ήδη χρησιμοποιούνται και προωθούνται ευρέως. Παρακάτω μια λίστα με μερικά από τα πιο δημοφιλή προϊόντα IoT που είναι σήμερα στην αγορά (2).

Έξυπνα Ηχεία

Η Google , η Amazon και η Apple έχουν να προσφέρουν μια λύση στο τομέα των έξυπνων ηχείων. Έξυπνο ηχείο (smart speaker) είναι ένας τύπος ασύρματης συσκευής ομιλίας και φωνητικής εντολής με ενσωματωμένο έναν εικονικό βοηθό (τεχνητή νοημοσύνη) που προσφέρει διαδραστικές ενέργειες και ενεργοποίηση απλώς με τη βοήθεια μια ειδικής λέξης. Την αρχή, από τις μεγάλες εταιρίες, την έκανε η Amazon με το Amazon Echo, ακολούθησε άμεσα η Google με το Google Home και στην πορεία η Apple με το Apple HomePod (3).

Εικόνα 5 Φωτογραφία του Google Home (αριστερά) δίπλα στο Amazon Echo

Έξυπνοι Θερμοστάτες

Ένα από τα πιο ορατά και δημοφιλή κομμάτια της τεχνολογίας Internet of Things οι έξυπνοι θερμοστάτες που συνδέονται με το Διαδίκτυο. Ο θερμοστάτης μαθαίνει τις ρουτίνες της οικογένειάς σας και προσαρμόζει αυτόματα τη θερμοκρασία ανάλογα με το πότε βρίσκετε κάποιος στο σπίτι ή μακριά, την ώρα που κάποιος ξυπνάει ή κοιμάται, όταν κάνει ζέστη ή κρυο, για να γίνει το σπίτι σας πιο ενεργειακά αποδοτικό και να σας βοηθήσει τους χρήστες να εξοικονομήσουν χρήματα για λογαριασμούς θέρμανσης και ψύξης. Η εφαρμογή για κινητά που συνοδεύει τους πιο δημοφιλείς θερμοστάτες επιτρέπει να την επεξεργασία χρονοδιαγραμμάτων, την αλλαγή της θερμοκρασίας όταν ο χρήστης βρίσκεστε μακριά από το σπίτι και ακόμα την λήψη ειδοποιήσεων όταν φαίνεται ότι κάτι έχει πάει στραβά με το σύστημα θέρμανσης ή ψύξης.

Έξυπνες πρίζες

Πρόκειται για πρίζες που συνδέεται σε μια κανονική οικιακή πρίζα και διαθέτουν ένα wifi μηχανισμό. Μια έξυπνη πρίζα δέχεται το καλώδιο τροφοδοσίας από οποιαδήποτε συσκευή και μπορεί να χρησιμοποιηθεί για να την ενεργοποιήσει ή να απενεργοποιήσει σε ένα καθορισμένο πρόγραμμα ή όταν πατηθεί ένα κουμπί στο smartphone του χρήστη. Οι διεπαφές τέτοιων εφαρμογών επιτρέπουν στους χρήστες να δουν πότε είναι ενεργοποιημένα τα βύσματα, πόση ισχύ χρησιμοποιούν και να ρυθμίσουν τα προγράμματα για λειτουργία από την εφαρμογή για κινητά.

Έξυπνες λάμπες

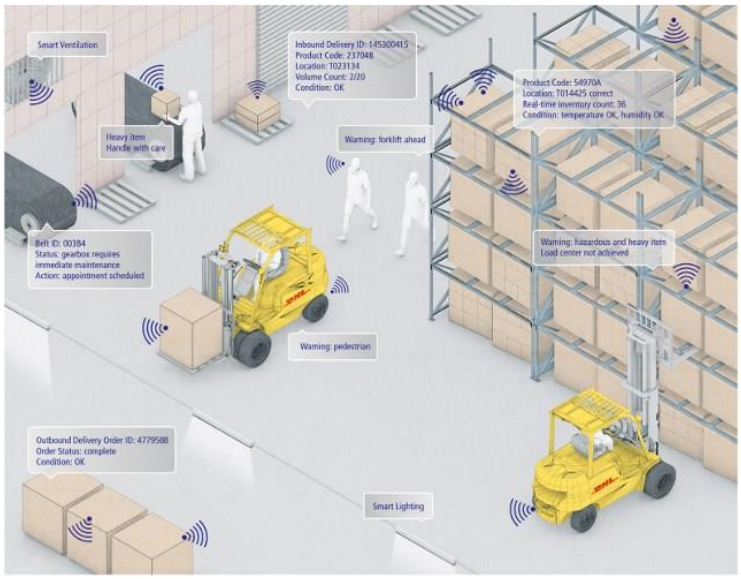
Οι έξυπνοι λαμπτήρες επιτρέπουν στους χρήστες να αλλάζουν τα χρώματα αλλά και την ένταση του φωτός. Πολλές εταιρίες προσφέρουν εφαρμογές για κινητά που επιτρέπουν στους χρήστες να προσαρμόζουν το φωτισμό της λάμπας βάσει μιας φωτογραφίας που ανεβάζουν μέσω της εφαρμογής. Οι λάμπες μπορούν επίσης να ενεργοποιηθούν και να απενεργοποιηθούν με χρονοδιάγραμμα ή από το smartphone . Έχουν την δυνατότητα να συγχρονιστούν και να αλλάζουν χρώμα ανάλογα με τη μουσική.

Εικόνα 6 Φωτογραφία με διάφορες έξυπνες λάμπες

2.2.1 Το IoT σήμερα στις βιομηχανίες

Παρακάτω παρουσιάζονται δυο από τις πιο γνωστές εφαρμογές του IoT στο τομέα της βιομηχανίας

Το σύστημα για καταγραφή και παρακολούθηση της DHL



Εικόνα 7 Γραφική αναπαράσταση του συστήματος καταγραφής της DHL

Σε πολύ μεγάλη κλίμακα, το Διαδίκτυο των πραγμάτων μπορεί να βοηθήσει σημαντικά στα logistics. Για παράδειγμα, η εταιρία διανομών DHL παρέχει υπηρεσίες αποστολής, αποθήκευσης, διανομής και διαχείρισης εφοδιαστικής αλυσίδας σε όλο τον κόσμο και αυτό απαιτεί τεράστιο όγκο επικοινωνίας. Η DHL δημοσίευσε μια έκθεση που περιγράφει μερικές πιθανές χρήσεις της τεχνολογίας IoT που περιλαμβάνει την παρακολούθηση και συντήρηση του οχήματος, την παρακολούθηση πακέτων σε πραγματικό χρόνο, τους περιβαλλοντικούς αισθητήρες σε δοχεία μεταφοράς, τη συλλογή πληροφοριών για τους εργαζόμενους και τα εργαλεία, καθώς και μια σειρά χαρακτηριστικών που βελτιώνουν την ασφάλεια των οχημάτων και των ανθρώπων.

Cisco connected factory

Η Cisco, μεγάλη εταιρία στον τομέα των τεχνολογιών πληροφορικής, ενθαρρύνει τις εταιρείες να καταστήσουν πιο αποτελεσματικές τις διαδικασίες παραγωγής και παραγωγής τους ενσωματώνοντας τις τεχνολογίες IoT στις εγκαταστάσεις τους. Η απομακρυσμένη παρακολούθηση και η πρόσβαση στον εξοπλισμό που χρησιμοποιείται για την κατασκευή θα μπορούσε να βελτιώσει σημαντικά την αποδοτικότητα, να επιτρέψει την ταχύτερη επίλυση των ζητημάτων και, τελικά, να οδηγήσει στην αύξηση της παραγωγής.

2.3 Location based υπηρεσίες

Μια υπηρεσία βάσει τοποθεσίας (Location Based Service, LBS) είναι μια υπηρεσία λογισμικού που χρησιμοποιεί δεδομένα θέσης για τον έλεγχο των λειτουργιών. Ως εκ τούτου, Μια LBS είναι μια υπηρεσία πληροφόρησης και έχει πολλές χρήσεις στην κοινωνική δικτύωση σήμερα για σκοπούς πληροφόρησης, ψυχαγωγίας ή ασφάλειας, που είναι προσβάσιμη με κινητές συσκευές μέσω του δικτύου κινητής τηλεφωνίας και η οποία χρησιμοποιεί πληροφορίες σχετικά με τη γεωγραφική θέση της κινητής συσκευής.

Μια LBS μπορεί να χρησιμοποιηθεί σε διάφορα πλαίσια, όπως η υγεία, η αναζήτηση οικιακών αντικειμένων, η διασκέδαση, η εργασία, η προσωπική ζωή κλπ.

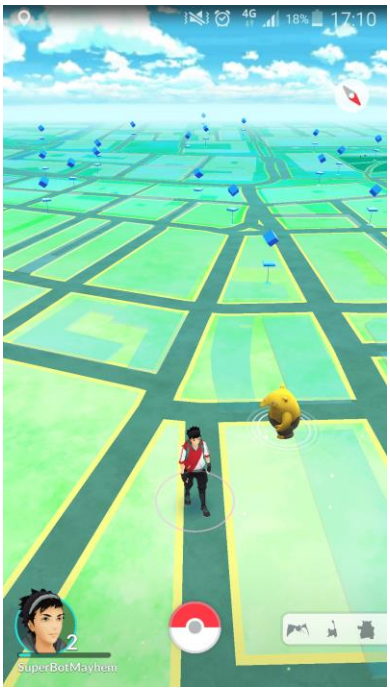
Μια LBS είναι κρίσιμη για πολλές επιχειρήσεις καθώς και κυβερνητικές οργανώσεις για να οδηγήσουν στην πραγματική διορατικότητα από τα δεδομένα που συνδέονται με μια συγκεκριμένη τοποθεσία όπου διεξάγονται οι δραστηριότητες.

Μια LBS περιλαμβάνει υπηρεσίες για τον εντοπισμό μιας θέσης ενός προσώπου ή ενός αντικειμένου, όπως η ανακάλυψη της πλησιέστερης τραπεζικής μηχανής (ATM) ή η τοποθεσία ενός φίλου ή υπαλλήλου. Μια LBS περιλαμβάνει υπηρεσίες παρακολούθησης αποστολών και υπηρεσίες παρακολούθησης οχημάτων. Μια LBS μπορεί να περιλαμβάνει το κινητό εμπόριο όταν λαμβάνει τη μορφή κουπονιών ή διαφημίσεων που απευθύνονται σε πελάτες με βάση την τρέχουσα τοποθεσία τους. Περιλαμβάνουν εξατομικευμένες υπηρεσίες καιρού και ακόμη και παιχνίδια βάσει τοποθεσίας.

2.3.1 Εφαρμογές των Location Based υπηρεσιών

Οι υπηρεσίες βάσει τοποθεσίας μπορούν να χρησιμοποιηθούν σε διάφορες εφαρμογές, όπως:

- Να προτείνουν κοινωνικές εκδηλώσεις σε μια πόλη
- Να επιτρέπουν στους χρήστες να βρίσκουν την πλησιέστερη επιχείρηση ή υπηρεσία, όπως ATM, εστιατόριο ή κατάστημα λιανικής πώλησης
- Να προσφέρουν αναλυτικές οδηγίες πλοήγησης βοηθώντας το χρήστη να κατευθυνθεί σε οποιαδήποτε διεύθυνση επιθυμεί
- Να προσφέρουν βοηθητικά συστήματα υγειονομικής περίθαλψης
- Να εντοπίζουν άτομα σε χάρτη που εμφανίζεται στο κινητό τηλέφωνο
- Να προσφέρουν ειδοποιήσεις, όπως ενημέρωση της τιμής της βενζίνης ανά βενζινάδικο ή προειδοποιούν για κυκλοφοριακή συμφόρηση
- Να εμφανίζουν διαφημίσεις μέσω κινητού τηλεφώνου βάσει τοποθεσίας
- Παιχνίδια όπου η θέση σας είναι μέρος του παιχνιδιού, όπως για παράδειγμα οι κινήσεις του χρήστη κατά τη διάρκεια της ημέρας, κάνουν το avatar του να μετακινείται στο παιχνίδι ή η θέση να ξεκλειδώνει κάποιο περιεχόμενο.
- Παρακολούθηση δορυφόρων

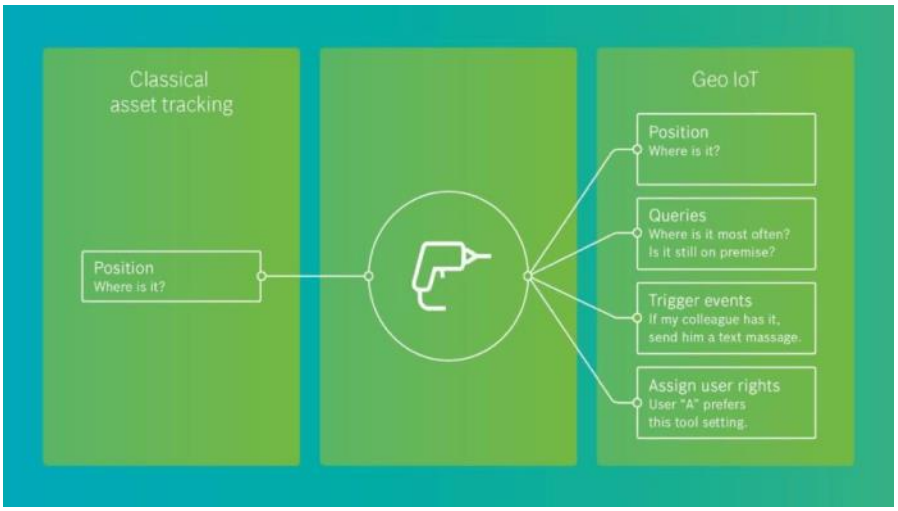


Εικόνα 8 Απόσπασμα οθόνης από το Location Based παιχνίδι Pokemon Go

2.4 Location based υπηρεσίες και Internet Of Things

Ένας νέος όρος που συνδυάζει IoT και LBS είναι ο Geo IoT (4). Στην κλασική παρακολούθηση ενός αντικειμένου, κάποιος χρήστης παρακολουθεί την τοποθεσία αυτού του αντικειμένου. Με το Geo IoT, μπορούμε να παρακολουθήσουμε όχι μόνο την τοποθεσία του αντικειμένου αλλά και την κατάσταση του αντικειμένου (ή του ατόμου).

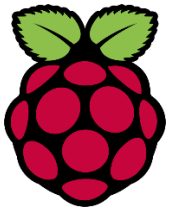
Για παράδειγμα έστω ένα υπόστεγο αεροσκαφών όπου τα αεροπλάνα συντηρούνται και επισκευάζονται. Η λάθος το τοποθέτηση ενός ηλεκτρικού εργαλείου σε ένα τέτοιο μέρος θα κοστίσει αρκετά λεπτά παραγωγικότητας και ενδεχόμενος το αεροπλάνο να μην είναι έτοιμο για την προγραμματισμένη πτήση του. Μια μικρή προσθήκη μιας συσκευής εντοπισμού σε ένα τέτοιο εργαλείο βοηθάει στην άντληση πληροφοριών όπως πόσο συχνά χρησιμοποιείται το εργαλείο, από ποιον κτλ.



Εικόνα 9 Κλασική καταγραφή ενός αντικειμένου σε σύγκριση με το Geo IoT

3. Τεχνολογίες για την υλοποίηση του συστήματος

3.1 Raspberry Pi



Εικόνα 10 Λογότυπο Raspberry Pi

Το Raspberry Pi είναι ένας πλήρης υπολογιστής σε μέγεθος πιστωτικής κάρτας. Αναπτύχθηκε από ομάδα ερευνητών στο Πανεπιστήμιο του Cambridge, για να βοηθήσει μαθητές και φοιτητές στην εκμάθηση προγραμματισμού. Το αρχικό μοντέλο έγινε πολύ πιο δημοφιλές από τις αρχικές εκτιμήσεις, καταφέρνοντας να πωλήσει εκτός της στοχευόμενης αγοράς για χρήση σε τομείς όπως η ρομποτική.

Τα Raspberry Pi αναπτύσσονται και κατασκευάζονται από το Raspberry Pi Foundation. Η κατασκευή τους γίνεται σε ένα εργοστάσιο της Sony στο Pencoed της Ουαλίας.

Αρκετές γενιές Raspberry Pis έχουν κυκλοφορήσει. Η πρώτη γενιά (Raspberry Pi 1 Model B) κυκλοφόρησε τον Φεβρουάριο του 2012. Το Ίδρυμα Raspberry Pi προσφέρει το λειτουργικό σύστημα Raspbian, μια βασισμένη στο Debian διανομή linux για χρήση στο Raspberry Pi.

Τα αρχικά Raspberry Pi βασίζονται στο BCM2835 System On a Chip (SOC) της Broadcom, που διαθέτει επεξεργαστή ARM1176JZF-S 700MHz, μονάδα επεξεργασίας γραφικών VideoCore IV (GPU) , και αρχικά διανέμονταν με 256 Megabytes μνήμη , όπου αργότερα αναβαθμίστηκε σε 512 Megabytes στα μοντέλα B και B+.

Εικόνα 11 Εικόνα ενός Raspberry Pi

Το όνομα Raspberry Pi έχει να κάνει με την παράδοση να αποδίδονται ονόματα φρούτων σε μικροϋπολογιστές. Πολλές εταιρείες υπολογιστών πήραν το όνομά τους από φρούτα. Το Pi είναι επειδή αρχικά επρόκειτο να δημιουργηθεί ένας υπολογιστής που θα μπορούσε να τρέξει Python. Έτσι, το Pi είναι για την Python. Είναι εφικτή η χρήση Python στο Raspberry Pi αλλά το τελικό προϊόν είναι πολύ πιο ικανό από το αρχικό πρότυπο κάνοντας το να ξεπερνάει το όνομα του (5).

3.2 Τεχνολογίες Παγκόσμιου Ιστού

3.2.1 HTML

Εικόνα 12 Λογότυπο HTML

Η HTML είναι η γλώσσα με την οποία κατασκευάζουμε ιστοσελίδες. Τα αρχικά HTML σημαίνουν Hypertext Markup Language (ελλ. Γλώσσα Σήμανσης Υπερκειμένου). Οι ιστοσελίδες που επισκεπτόμαστε στο Internet δεν είναι τίποτε άλλο παρά αρχεία τα οποία περιέχουν κώδικα γραμμένο στην γλώσσα HTML. Από το 1996 και μετά, οι προδιαγραφές της HTML τηρούνται, μαζί με ανάδραση από τους δημιουργούς λογισμικού, από το World Wide Web Consortium (W3C).Τα έγγραφα HTML αποτελούνται από στοιχεία HTML τα οποία στην πιο γενική μορφή τους έχουν τρία συστατικά: ένα ζεύγος από ετικέτες, την «ετικέτα εκκίνησης» και την «ετικέτα τερματισμού», μερικές ιδιότητες μέσα στην ετικέτα εκκίνησης, και τέλος το κείμενο ή το γραφικό περιεχόμενο μεταξύ των ετικετών, το οποίο μπορεί να περιλαμβάνει και άλλα στοιχεία εμφωλευμένα μέσα του. Τα έγγραφα HTML πρέπει να αρχίζουν με μια Δήλωση τύπου εγγράφου (Document Type Declaration, ανεπίσημα λέγεται και «doctype»). Αυτή η δήλωση βοηθά τους browser να καταλάβουν πώς πρέπει να διαβάσουν το περιεχόμενο του εγγράφου και πώς να το παρουσιάσουν μετά, και ιδιαίτερα όταν χρησιμοποιείται το quirks mode, το οποίο αποτελεί μια τεχνική για λόγους συμβατότητας των ιστοσελίδων που είχαν σχεδιαστεί για παλιούς browsers.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <p>Hello world</p>
  </body>
</html>
```

Πίνακας 1 Δείγμα δομής κώδικα HTML

η γενική μορφή ενός στοιχείου HTML είναι: *<ετικέτα ιδιότητα1="τιμή1" ιδιότητα2="τιμή2">περιεχόμενο</ετικέτα>*. Μερικά στοιχεία HTML περιγράφονται ως άδεια στοιχεία, έχουν τη μορφή *<ετικέτα ιδιότητα1="τιμή1" ιδιότητα2="τιμή2">*, και δεν έχουν καθόλου περιεχόμενο. Ο πιο κοινός τύπος αρχείου για έγγραφα HTML είναι *.html*, όμως έχει επιβιώσει και η συντόμευση *.htm*, από μερικά παλαιότερα λειτουργικά συστήματα που δεν αναγνώριζαν επεκτάσεις αρχείων με περισσότερα από τρία γράμματα.

3.2.2 CSS

Εικόνα 13 Λογότυπο CSS

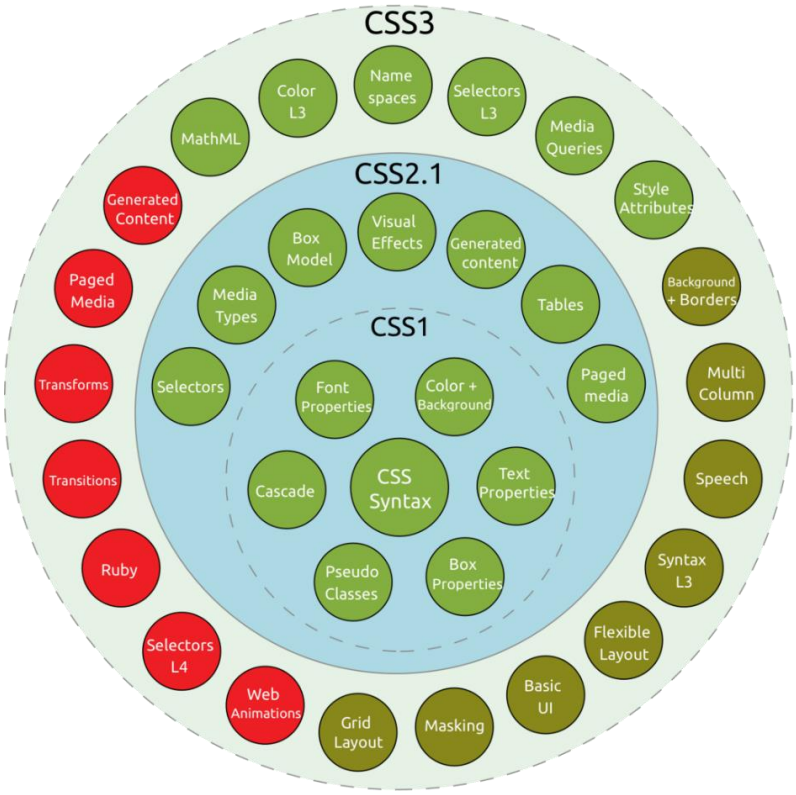
Τα Διαδοχικά Φύλλα Στυλ (CSS, Cascading Style Sheets) είναι μια γλώσσα στυλ (style language) που ορίζει τη διάταξη (layout) των HTML εγγράφων. Για παράδειγμα, τα CSS έχουν να κάνουν με γραμματοσειρές (fonts), με χρώματα (colours), με περιθώρια (margins), με εικόνες φόντου (background images) και με πολλά άλλα. Με την HTML θα δυσκολευτούμε να αλλάξουμε τη διάταξη των ιστοσελίδων μας, αλλά τα CSS προσφέρουν πολλές επιλογές και είναι πολύ πιο συγκεκριμένα στις λεπτομέρειες. Επιπλέον, υποστηρίζονται απ' όλους τους φυλλομετρητές.

Εικόνα 14 Δείγμα CSS αρχείου

Το CSS έχει διάφορα επίπεδα και προφίλ. Κάθε επίπεδο CSS βασίζεται στο τελευταίο, τυπικά προσθέτοντας νέα χαρακτηριστικά και συνήθως χαρακτηρίζεται ως CSS1, CSS2, CSS3 και CSS4. Τα προφίλ είναι συνήθως ένα υποσύνολο ενός ή περισσοτέρων επιπέδων CSS που έχουν δημιουργηθεί για μια συγκεκριμένη συσκευή ή περιβάλλον χρήστη .

3.2.3 CSS 3

Το CSS 3 χωρίζεται από το W3C σε διάφορα ξεχωριστά έγγραφα που ονομάζονται "modules"(ενότητες). Κάθε ενότητα προσθέτει νέες δυνατότητες ή επεκτείνει τις λειτουργίες που ορίζονται στο CSS 2, διατηρώντας την συμβατότητα προς τα πίσω. Οι εργασίες στο επίπεδο 3 του CSS ξεκίνησαν γύρω από τη δημοσίευση της αρχικής σύστασης CSS 2. Τα πρώτα σχέδια του CSS 3 δημοσιεύθηκαν τον Ιούνιο του 1999.



Εικόνα 15 Ταξινόμηση και κατάσταση των CSS3 Modules

3.2.4 CSS Frameworks

Τα CSS Frameworks (πλαίσια εφαρμογών CSS) είναι έτοιμες βιβλιοθήκες που προορίζονται να επιτρέπουν ευκολότερη και συμβατότερη με τα πρότυπα σχεδίαση ιστοσελίδων με τη χρήση της γλώσσας Cascading Style Sheets. Στα πιο δημοφιλή CSS Frameworks περιλαμβάνονται το Bootstrap, το Fountation, το Materialalize, το Semantic-UI και το Bulma. Όπως οι βιβλιοθήκες στις γλώσσες προγραμματισμού, τα CSS Frameworks συνήθως ενσωματώνονται ως εξωτερικά φύλλα .css και εισάγονται με αναφορά στο <head> του HTML. Παρέχουν μια σειρά από έτοιμες επιλογές για το σχεδιασμό και τη διαμόρφωση της ιστοσελίδας.

3.3 WEB API

WebAPI είναι ένας όρος που χρησιμοποιείται για να αναφερθεί σε μια σειρά από APIs (προγραμματιστικές διεπαφές εφαρμογής) που επιτρέπουν σε εφαρμογές Web να έχουν πρόσβαση στο υλικό συσκευών (όπως η κατάσταση της μπαταρίας ή τη δόνησης της συσκευής), καθώς και η πρόσβαση στα δεδομένα που είναι αποθηκευμένα στη συσκευή (όπως ημερολόγιο ή λίστα επαφών).

Παραδείγματα τέτοιων APIs είναι το Geolocation API το οποίο επιτρέπει στη web εφαρμογή να έχει πρόσβαση στην τοποθεσία του γεωγραφική τοποθεσία του χρήστη και το Notification API για την αποστολή ειδοποιήσεων από τη συσκευή.

3.4 Document Object Model

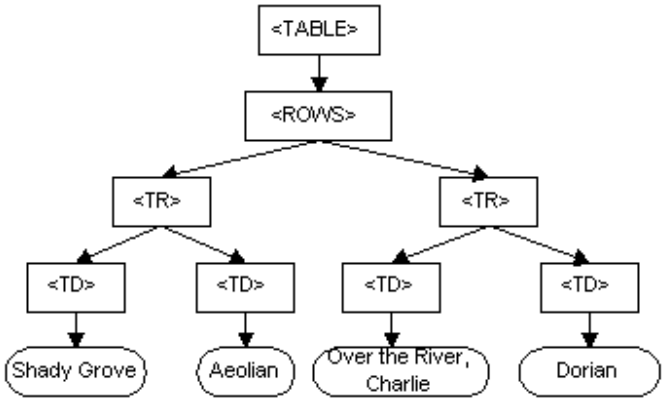
Το Μοντέλο Αντικειμένων Εγγράφων (DOM) είναι μια διεπαφή προγραμματισμού για έγγραφα HTML και XML. Αντιπροσωπεύει τη σελίδα έτσι ώστε τα προγράμματα να μπορούν να αλλάξουν τη δομή, το ύφος και το περιεχόμενο του εγγράφου. Το DOM αντιπροσωπεύει το έγγραφο ως κόμβους και αντικείμενα. Με αυτόν τον τρόπο, οι γλώσσες προγραμματισμού μπορούν να συνδεθούν στη σελίδα. Μια ιστοσελίδα είναι ένα έγγραφο. Αυτό το έγγραφο μπορεί είτε να εμφανιστεί στο παράθυρο του προγράμματος περιήγησης είτε ως κώδικας HTML. Αλλά είναι το ίδιο έγγραφο και στις δύο περιπτώσεις. Το Μοντέλο Αντικειμένων Εγγράφων (DOM) αντιπροσωπεύει το ίδιο έγγραφο ώστε να μπορεί να γίνει διαχειρίσιμο. Το DOM είναι μια αντικειμενοστρεφής αναπαράσταση της ιστοσελίδας, η οποία μπορεί να τροποποιηθεί με μια γλώσσα προγραμματισμού σεναρίων (scripting language) όπως η JavaScript (6).

Το μοντέλο αντικειμένου εγγράφου είναι ένα API προγραμματισμού για έγγραφα. Το ίδιο το μοντέλο αντικειμένου μοιάζει πολύ με τη δομή των εγγράφων που μοντελοποιεί (7). Για παράδειγμα, έστω ότι έχουμε το παρακάτω πίνακα (html table) , που προέρχεται από ένα έγγραφο HTML:

```
<TABLE>
<ROWS>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Dorian</TD>
</TR>
</ROWS>
</TABLE>
```

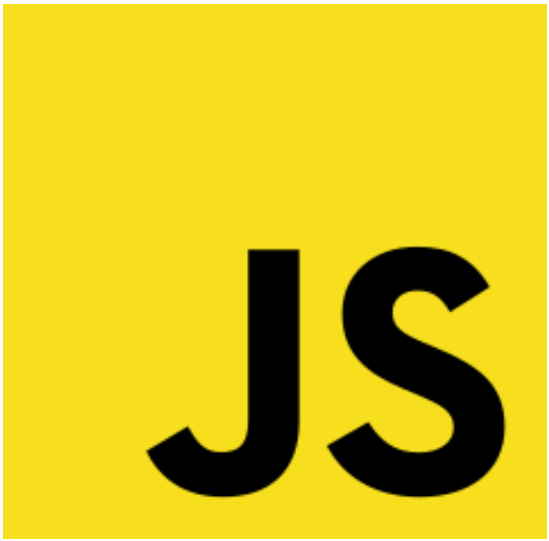
Πίνακας 2 Παράδειγμα html πίνακα

Το DOM αντιπροσωπεύει αυτόν τον πίνακα ως εξής:



Εικόνα 16 DOM αναπαράσταση του html πίνακα

3.5 JavaScript / ECMAScript



Εικόνα 17 Λογότυπο JavaScript

Η JavaScript (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές. Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη C. Είναι γλώσσα βασισμένη σε διαφορετικά προγραμματιστικά παραδείγματα (multi-paradigm), υποστηρίζοντας αντικειμενοστρεφές, προστακτικό και συναρτησιακό στυλ προγραμματισμού.

Η JavaScript χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων — τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers) και οι μικρές εφαρμογές της επιφάνειας εργασίας (desktop widgets). Οι νεότερες εικονικές μηχανές και πλαίσια ανάπτυξης για JavaScript (όπως το Node.js) έχουν επίσης κάνει τη JavaScript πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού στην πλευρά του διακομιστή (server-side).

Το πρότυπο της γλώσσας κατά τον οργανισμό τυποποίησης ECMA ονομάζεται ECMAScript.

Η JavaScript έχει γίνει μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού ηλεκτρονικών υπολογιστών στον Παγκόσμιο Ιστό (Web). Αρχικά, όμως, πολλοί επαγγελματίες προγραμματιστές υποτίμησαν τη γλώσσα διότι το κοινό της ήταν ερασιτέχνες συγγραφείς ιστοσελίδων και όχι επαγγελματίες προγραμματιστές (και μεταξύ άλλων λόγων).Με με τη χρήση της τεχνολογίας Ajax, η JavaScript γλώσσα επέστρεψε στο προσκήνιο και έφερε πιο επαγγελματική προσοχή προγραμματισμού. Το αποτέλεσμα ήταν ένα καινοτόμο αντίκτυπο στην εξάπλωση των πλαισίων και των βιβλιοθηκών, τη βελτίωση προγραμματισμού με JavaScript, καθώς και αυξημένη χρήση της JavaScript έξω από τα προγράμματα περιήγησης στο Web.

Τον Ιανουάριο του 2009, το έργο CommonJS ιδρύθηκε με στόχο τον καθορισμό ενός κοινού προτύπου βιβλιοθήκης κυρίως για την ανάπτυξη της JavaScript έξω από το πρόγραμμα περιήγησης και μέσα σε άλλες τεχνολογίες (π.χ. server-side).

Η αρχική έκδοση της Javascript βασίστηκε στη σύνταξη στη γλώσσα προγραμματισμού C, αν και έχει εξελιχθεί, ενσωματώνοντας πια χαρακτηριστικά από νεότερες γλώσσες.

Αρχικά χρησιμοποιήθηκε για προγραμματισμό από την πλευρά του πελάτη (client), που ήταν ο φυλλομετρητής (browser) του χρήστη, και χαρακτηρίστηκε σαν client-side γλώσσα προγραμματισμού. Αυτό σημαίνει ότι η επεξεργασία του κώδικα Javascript και η παραγωγή του τελικού περιεχομένου HTML δεν πραγματοποιείται στο διακομιστή, αλλά στο πρόγραμμα περιήγησης των επισκεπτών, ενώ μπορεί να ενσωματωθεί σε στατικές σελίδες HTML. Αντίθετα, άλλες γλώσσες όπως η PHP εκτελούνται στο διακομιστή (server-side γλώσσες προγραμματισμού).

Παρά την ευρεία χρήση της Javascript για συγγραφή προγραμμάτων σε περιβάλλον φυλλομετρητή, από την αρχή χρησιμοποιήθηκε και για τη συγγραφή κώδικα από την πλευρά του διακομιστή, από την ίδια τη Netscape στο προϊόν LiveWire, με μικρή επιτυχία. Η χρήση της Javascript στο διακομιστή εμφανίζεται πάλι σήμερα, με τη διάδοση του Node.js, ενός μοντέλου προγραμματισμού βασισμένο στα γεγονότα (events)

test.js

```
1 class A {}
2 class B extends A {}
3
4 const f = () => {
5     let world = "world";
6     return `Hello, ${world}`;
7 };
8
9 console.log(f());
```

Εικόνα 18 Παράδειγμα κώδικα JavaScript κάνοντας χρήση λειτουργιών και συντακτικού της ECMAScript 2015

3.5.1 Ιστορία της JavaScript

Η γλώσσα προγραμματισμού JavaScript δημιουργήθηκε αρχικά από τον Brendan Eich της εταιρείας Netscape με την επωνυμία Mocha. Αργότερα, η Mocha μετονομάστηκε σε LiveScript, και τελικά σε JavaScript, κυρίως επειδή η ανάπτυξή της επηρεάστηκε περισσότερο από τη γλώσσα προγραμματισμού Java. LiveScript ήταν το επίσημο όνομα της γλώσσας όταν για πρώτη φορά κυκλοφόρησε στην αγορά σε βήτα (beta) εκδόσεις με το πρόγραμμα περιήγησης στο Web, Netscape Navigator εκδοχή 2.0 τον Σεπτέμβριο του 1995. Η LiveScript μετονομάστηκε σε JavaScript σε μια κοινή ανακοίνωση με την εταιρεία Sun Microsystems στις 4 Δεκεμβρίου, 1995, όταν επεκτάθηκε στην έκδοση του προγράμματος περιήγησης στο Web, Netscape.

Η JavaScript απέκτησε μεγάλη επιτυχία ως γλώσσα στην πλευρά του πελάτη (client-side) για εκτέλεση κώδικα σε ιστοσελίδες, και περιλήφθηκε σε διάφορα προγράμματα περιήγησης στο Web. Η εταιρεία Microsoft ονόμασε την εφαρμογή της JScript για να αποφύγει θέματα εμπορικών σημάτων. Η JScript πρόσθεσε νέους μεθόδους για να διορθώσει τα Y2K-προβλήματα στην JavaScript, οι οποίοι βασίστηκαν στην java.util.Date τάξη της Java. Η JScript περιλήφθηκε στο πρόγραμμα Internet Explorer εκδοχή 3.0, το οποίο κυκλοφόρησε τον Αύγουστο του 1996.

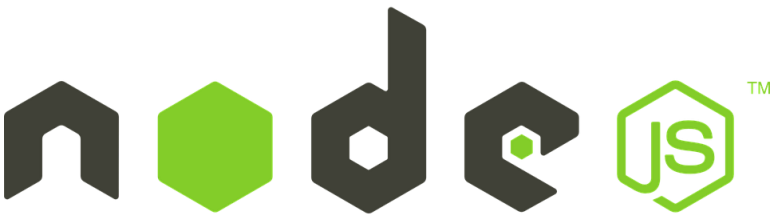
Τον Νοέμβριο του 1996, η Netscape ανακοίνωσε ότι είχε υποβάλει τη γλώσσα JavaScript στο Ecma International (μια οργάνωση της τυποποίησης των γλωσσών προγραμματισμού) για εξέταση ως βιομηχανικό πρότυπο, και στη συνέχεια το έργο είχε ως αποτέλεσμα την τυποποιημένη μορφή που ονομάζεται ECMAScript.

Εικόνα 19 Η ιστορία της JavaScript

3.5.2 Εκδόσεις ECMAScript

Η πρώτη έκδοση της ECMAScript εγκρίθηκε από τη Γενική Συνέλευση της Ecma τον Ιούνιο του 1997. Από τότε δημοσιεύθηκαν αρκετές εκδόσεις του προτύπου γλώσσας. Η τέταρτη έκδοση της ECMAScript δεν δημοσιεύτηκε ποτέ λόγω διαφωνιών σχετικά με την πολυπλοκότητα της γλώσσας και μετά από πολυετή καθυστέρηση δημοσιεύτηκε το Δεκέμβριο του 2009 η πέμπτη έκδοση της ECMAScript. Σημαντικό ορόσημο στην ιστορία της ECMAScript είναι έκτη έκδοση της, με το όνομα ECMAScript2015, κοινώς γνωστή ως ECMAScript6 ή ES6, η οποία πρόσθεσε νέο συντακτικό όπως κλάσεις και modules και νέα στοιχεία όπως arrow functions και promises. Η ECMAScript2015 δημοσιεύτηκε τον Ιούνιο του 2015. Η Ecma International αποφάσισε να επισπεύσει το ρυθμό δημοσίευσης των νέων εκδόσεων έτσι από το 2015 και μετά κυκλοφορεί κάθε χρόνο μια νέα έκδοση ECMAScript (8).

3.6 NodeJS



Εικόνα 20 Λογότυπο NodeJs

Το Node.js είναι μια πλατφόρμα ανάπτυξης λογισμικού (κυρίως διακομιστών) χτισμένη σε περιβάλλον Javascript. Στόχος του Node είναι να παρέχει ένα εύκολο τρόπο δημιουργίας κλιμακωτών διαδικτυακών εφαρμογών. Σε αντίθεση από τα περισσότερα σύγχρονα περιβάλλοντα ανάπτυξης εφαρμογών δικτύων μία διεργασία node δεν στηρίζεται στην πολυνηματικότητα αλλά σε ένα μοντέλο ασύγχρονης επικοινωνίας εισόδου/εξόδου.

Το Node.js δημιουργήθηκε από τον Ryan Dahl το 2009. Η δημιουργία και η συντήρηση του έργου χορηγήθηκε από την εταιρία Joyent. Η ιδέα για την ανάπτυξη του node προήλθε από την ανάγκη του Ryan Dahl να βρει τον πιο αποδοτικό τρόπο να ενημερώνει τον χρήστη σε πραγματικό χρόνο για την κατάσταση ενός αρχείου που ανέβαζε στο διαδίκτυο. Επίσης επηρεάστηκε από το Mongrel του Zed Shaw. Επιπροσθέτως μετά από αποτυχημένα έργα σε C, Lua, Haskell η κυκλοφορία της μηχανής V8 (V8 JavaScript Engine) της Google τον ώθησε να ασχοληθεί με την Javascript.

Η κοινότητα έχει δημιουργήσει ένα ολόκληρο οικοσύστημα από βιβλιοθήκες που προορίζονται ή είναι συμβατές με το node. Ανάμεσά τους εργαλεία που ξεχώρισαν όπως το Mongodb, το MVC framework Express και τη βιβλιοθήκη για επικοινωνία σε πραγματικό χρόνο Socket.IO παίζουν σημαντικό ρόλο υποστηρίζοντας την ασύγχρονη διάδραση με τις παραδοσιακές και NoSQL μεθόδους βάσεων δεδομένων. Αυτό επιτυγχάνεται με την χρήση του node package manager το οποίο επιτρέπει την εγκατάσταση των παραπάνω βιβλιοθηκών. Χρησιμοποιείται συνήθως σε εφαρμογές Chat, Proxy, Http Server καθώς και για παρακολούθηση εφαρμογών και του συστήματος (monitoring).

3.7 Front-End JavaScript Framework

Τα Front-End JavaScript Framework αποτελούν πλαίσια εφαρμογής γραμμένα σε JavaScript. Προσφέρουν διευκολύνσεις στη δημιουργία και συντήρηση πολύ σύνθετων διαδικτυακών χωρίς να χρειάζεται να γίνει refresh στη σελίδα. Συνήθως υλοποιούν κάποιο Model View design pattern όπως Model View Controller ή Model View ViewModel. Ο όρος framework δεν έχει προταθεί ή προτυποποιηθεί από κάποιον επίσημο οργανισμό προτυποποίησης οπότε είναι στο χέρι του δημιουργού να ονομάσει το πρόγραμμα του framework ή όχι με βάσει τον ορισμό που επέλεξε να υιοθετήσει. Λόγου χάρη το Facebook αποκαλεί την εφαρμογή του ReactJS βιβλιοθήκη και όχι framework (9).

Τα πιο δημοφιλή frameworks – βιβλιοθήκες είναι το ReactJS, το Angular και το VueJS.

3.7.1 VueJS

Εικόνα 21 Λογότυπο VueJS

Το Vue.js δημιουργήθηκε για να οργανώσει και να απλοποιήσει την ανάπτυξη εφαρμογών ιστού. Το έργο επικεντρώνεται στο να γίνουν πιο προσιτά οι ιδέες για την ανάπτυξη του UI στο διαδίκτυο (components, declarative UI, hot-reloading, time-travel debugging, κλπ). Προσπαθεί να είναι λιγότερο δογματικό και έτσι πιο εύκολο για τους προγραμματιστές να το υιοθετήσουν. Διαθέτει μια προοδευτικά υιοθετήσιμη αρχιτεκτονική. Η κεντρική βιβλιοθήκη επικεντρώνεται στη δηλωτική απόδοση και τη σύνθεση συστατικών και μπορεί να ενσωματωθεί σε υπάρχουσες σελίδες. Οι προηγμένες λειτουργίες που απαιτούνται για σύνθετες εφαρμογές όπως η δρομολόγηση, η διαχείριση κατάστασης της σελίδας και η κατασκευή εργαλείων προσφέρονται μέσω επίσημα διατηρούμενων βιβλιοθηκών και πακέτων υποστήριξης

3.8 Google Maps API

Εικόνα 22 Λογότυπο Google Maps

Η υπηρεσία Google Maps είναι μια διαδικτυακή υπηρεσία χαρτογράφησης που παρέχεται από την Google και τρέχει σε Desktop και Smartphones. Η υπηρεσία επίσης προσφέρει ένα εύρη φάσμα εφαρμογών όπως δορυφορικές εικόνες, οδικούς χάρτες , φωτογραφίες οδών καθώς και λειτουργίες όπως σχεδιασμό διαδρομής , πλοήγησης με το αυτοκίνητο, τα πόδια και το ποδήλατο. Το Google Maps API υποστηρίζει τη χρήση χαρτών από ιστοσελίδες τρίτων και πολλά άλλα όπως εγγραφή τοποθεσίας επιχείρησης και οργανισμών στους χάρτες της. Η υπηρεσία/βάση δεδομένων δεν ενημερώνεται σε πραγματικό χρόνο αλλά γίνετε συνεχείς αναβάθμιση των εικόνων και καμία εικόνα δεν είναι παλαιότερη των τριών χρόνων.

Η Google ξεκίνησε το Google Maps API, τον Ιούνιο του 2005 για να επιτρέπει στους προγραμματιστές να ενσωματώσουν το Google Maps στις ιστοσελίδες τους. Πρόκειται για μια δωρεάν υπηρεσία. Με τη χρήση του Google Maps API, είναι δυνατόν να ενσωματώσετε το Google Maps σε μια ιστοσελίδα τρίτου, επί του οποίου σε συγκεκριμένο χώρο συγκεκριμένα δεδομένα μπορούν να επικαλύπτονται. Τα Google Maps APIs περιέχουν μια σειρά από υπηρεσίες όπως, μια υπηρεσία για την ανάκτηση στατικών εικόνων χάρτη και διαδικτυακών υπηρεσιών για την εκτέλεση γεωκωδικοποίησης, δημιουργία οδηγιών από ένα σημείο σε ένα άλλο και την απόκτηση σημείων ανύψωσης. Πάνω από 1,000,000 ιστοσελίδες χρησιμοποιούν το Google Maps API, καθιστώντας αυτή ως την πιο ευρέως χρησιμοποιημένη διαδικτυακή εφαρμογή ανάπτυξης API. Το Google Maps API είναι δωρεάν για εμπορική χρήση, με κάποιες προϋποθέσεις που έχουν να κάνουν κυρίως με το αριθμό των request που δέχεται.

3.9 NoSQL

Μια βάση δεδομένων NoSQL (αρχικά αναφερόμενη ως "μη SQL" ή "μη σχεσιακή") παρέχει έναν μηχανισμό αποθήκευσης και ανάκτησης δεδομένων που μοντελοποιείται με μέσα διαφορετικά από τις σχέσεις που χρησιμοποιούνται σε σχεσιακές βάσεις δεδομένων.

Τέτοιες βάσεις δεδομένων υπήρχαν από τα τέλη της δεκαετίας του 1960, αλλά δεν έλαβαν τον όρο "NoSQL" μέχρι την ραγδαία αύξηση της δημοτικότητας τους στις αρχές του 21ου αιώνα, που προκλήθηκαν από τις ανάγκες των μεγάλων Web 2.0 εταιρειών όπως το Facebook, το Google και το Amazon.com.

3.9.1 MongoDB

Εικόνα 23 Λογότυπο MongoDB

Μια από της πιο δημοφιλής NoSQL βάσεις είναι η MongoDB, η MongoDB χρησιμοποιεί έγγραφα σε στυλ τύπου JSON για την δημιουργία του σχήματος της βάσης. Η MongoDB αναπτύσσεται από την MongoDB Inc.

3.10 Nginx

Εικόνα 24 Λογότυπο Nginx

Το Nginx είναι ένας διακομιστή ιστού ο οποίος μπορεί επίσης να χρησιμοποιηθεί ως αντίστροφος διακομιστής μεσολάβησης, εξισορρόπιστης φορτίου και κρυφή μνήμη HTTP. Το λογισμικό δημιουργήθηκε από τον Igor Sysoev και δημοσιεύθηκε για πρώτη φορά το 2004. Μια εταιρεία του ιδίου ονόματος ιδρύθηκε το 2011 για να παράσχει υποστήριξ. Ο Nginx είναι δωρεάν λογισμικό ανοιχτού κώδικα, το οποίο κυκλοφορεί υπό όρους άδειας τύπου BSD. Ένα μεγάλο μέρος των εξυπηρετητών ιστού χρησιμοποιεί το NGINX, συχνά ως load balancer.

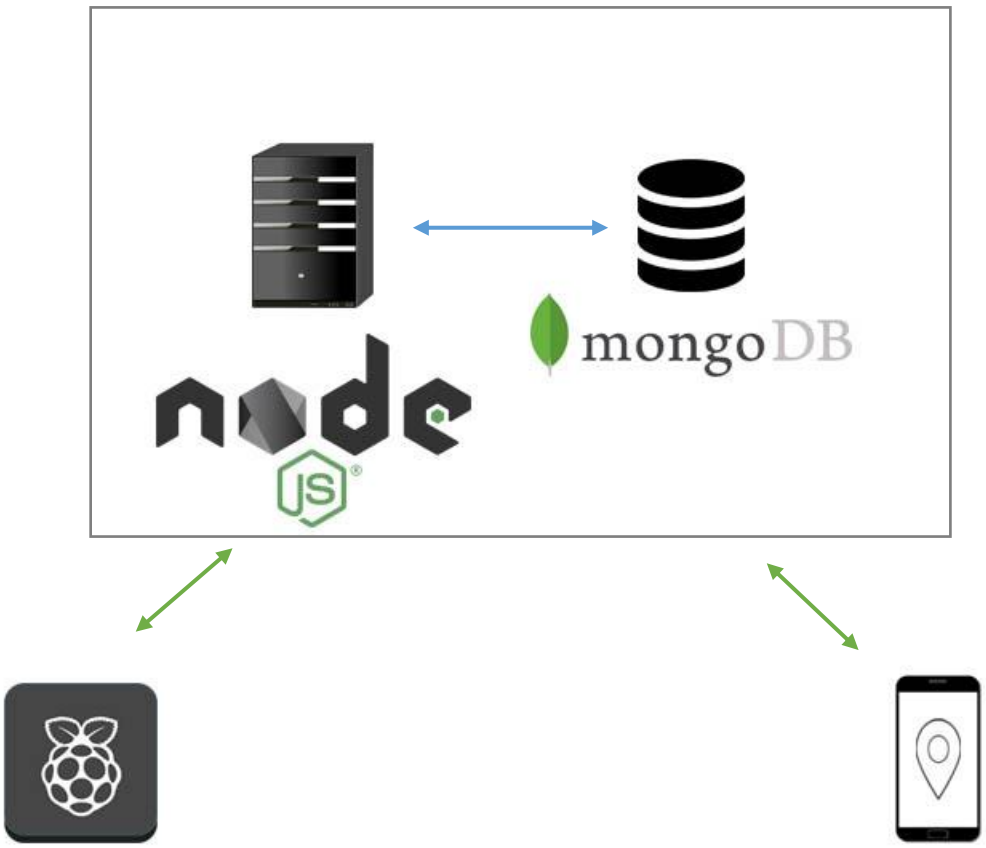
Ο Nginx χρησιμοποιεί μια ασύγχρονη event-driven προσέγγιση για το χειρισμό των αιτημάτων. Η αρθρωτή event-driven αρχιτεκτονική του Nginx μπορεί να προσφέρει πιο προβλέψιμη απόδοση κάτω από υψηλά φορτία.

4. Υλοποίηση του συστήματος

Σε αυτό το κεφάλαιο γίνεται αναλυτική παρουσίαση του συστήματος που παρουσιάστηκε στην εισαγωγή. Γίνεται λεπτομερείς παρουσίαση όλων των τεχνολογιών που χρησιμοποιήθηκαν με αναλυτικές περιγραφές και βοήθεια σχεδιαγραμμάτων. Το κεφάλαιο έχει δομηθεί με τέτοιο τρόπο ώστε να παρουσιάζεται η κάθε λειτουργία ξεχωριστά. Η παρουσίαση γίνεται παράλληλα με την υλοποίηση δείχνοντας αποσπάσματα κώδικα ή εικόνες από συνδεσμολογία υλικού. Τα αποσπάσματά κώδικα έχουν μειωθεί και τροποποιηθεί έτσι ώστε να εμφανίζονται μόνο οι γραμμές κώδικα που χρειάζονται για την εκάστοτε λειτουργία. Ο πλήρης κώδικας αναφέρεται στο παράρτημα της πτυχιακής. Όλη η διαδρομή που ακολουθήθηκε μεταξύ raspberry, Server και χρήστη παρουσιάζεται με λεπτομέρεια και με τη βοήθεια σχεδιαγραμμάτων όπου χρειάζεται.

4.1 Δομή και τεχνολογίες του συστήματος

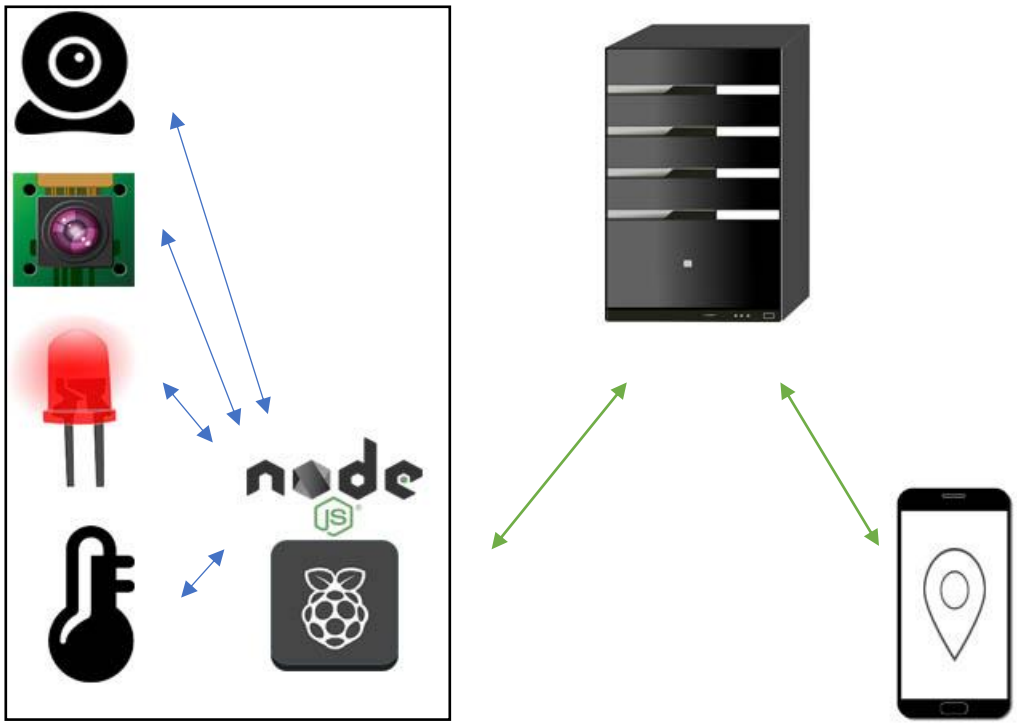
4.1.1 Server



Εικόνα 25 Σχεδιάγραμμα του συστήματος με έμφαση στις τεχνολογίες του Server

Ο Server αποτελεί το βασικό κομμάτι του συστήματος. Εκεί βρίσκεται ο κορμός της επιχειρησιακής λογικής του συστήματος. Δρα σαν ενδιάμεσος κρίκος μεταξύ του Raspberry Pi και του χρήστη. Επίσης είναι ο ίδιος υπεύθυνος για να μεταφέρει την web εφαρμογή στο χρήστη. Είναι υλοποιημένος σε NodeJS με τη χρήση του framework Express.js. Για την μόνιμη αποθήκευση των δεδομένων για τους χρήστες γίνεται χρήση της μη σχεσιακής βάσης δεδομένων MongoDB. Ο NodeJS Server είναι διαθέσιμος στο internet πίσω από ένα Nginx Server ο οποίος ακούει στη θύρα 80 και κάνει reverse proxy στη θύρα που τρέχει ο NodeJS Server.

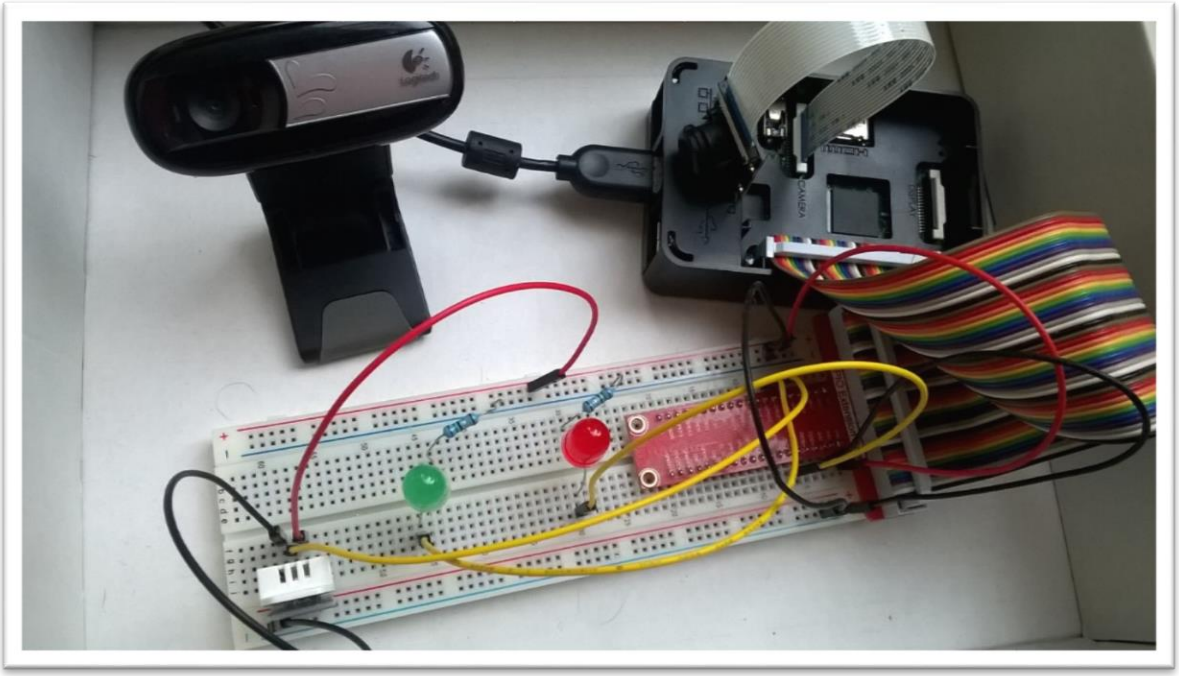
4.1.2 Raspberry Pi



Εικόνα 26 Σχεδιάγραμμα του συστήματος με έμφαση στο Raspberry Pi και των συνδεδεμένων συσκευών σε αυτό

Για το σύστημα της πτυχιακής χρησιμοποιήθηκε ένα **Raspberry Pi 3 Model B**. Πάνω σε αυτό τοποθετήθηκαν, μια κάμερα στην ειδική υποδοχή που διαθέτει, ένας αισθητήρας θερμοκρασίας-υγρασίας, δυο LEDs και άλλη μια web κάμερα συνδεδεμένη σε μια USB θύρα του Raspberry Pi. Για την πιο άνετη χρήση των GPIO υλικών (αισθητήρας και LEDs) χρησιμοποιήθηκε ένα εξωτερικό Breadboard με χρήση μιας καλωδιοταινίας και ενός T Type GPIO extension board.

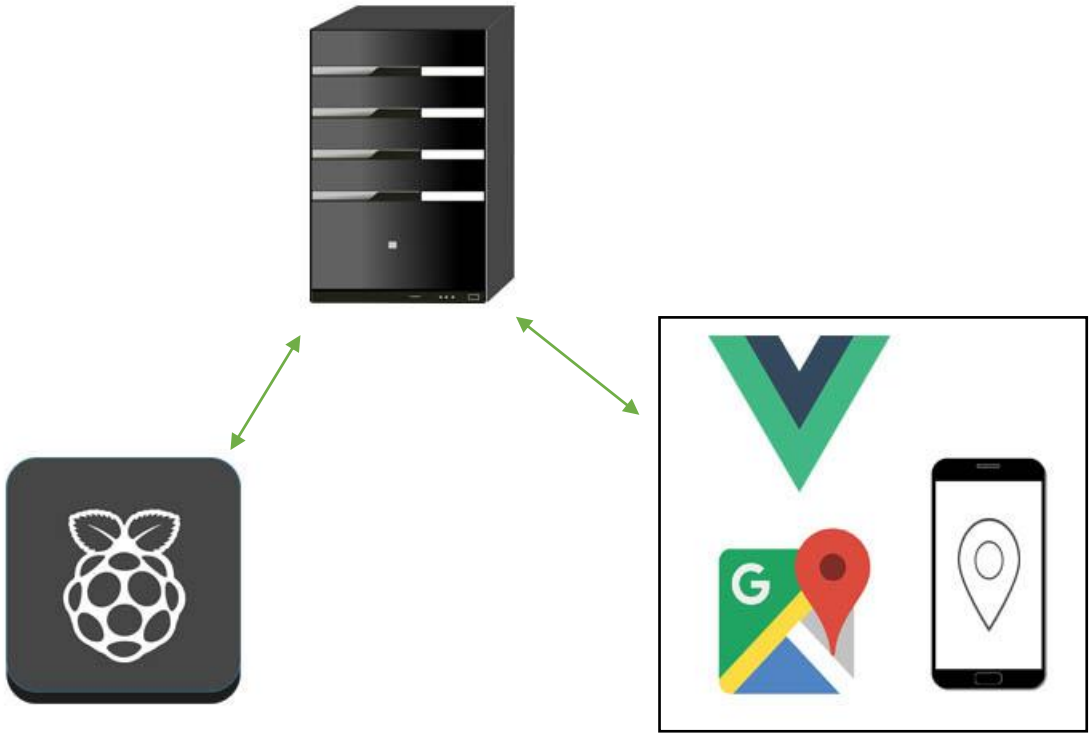
Παρακάτω μια φωτογραφία του Raspberry Pi με όλα τα υλικά συνδεδεμένα:



Εικόνα 27 Φωτογραφία του Raspberry Pi με όλες τις GPIO συσκευές και κάμερες συνδεδεμένα

Οι τιμές από τις GPIO συσκευές και η εικόνα από τις κάμερες λαμβάνεται με χρήση Node.js και των κατάλληλων npm πακέτων. Ο Node.js server πάνω στο Raspberry Pi δρα σαν client και μεταφέρει τις πληροφορίες στον Server μέσω Socket.io.

4.1.3 Web εφαρμογή χρήστη

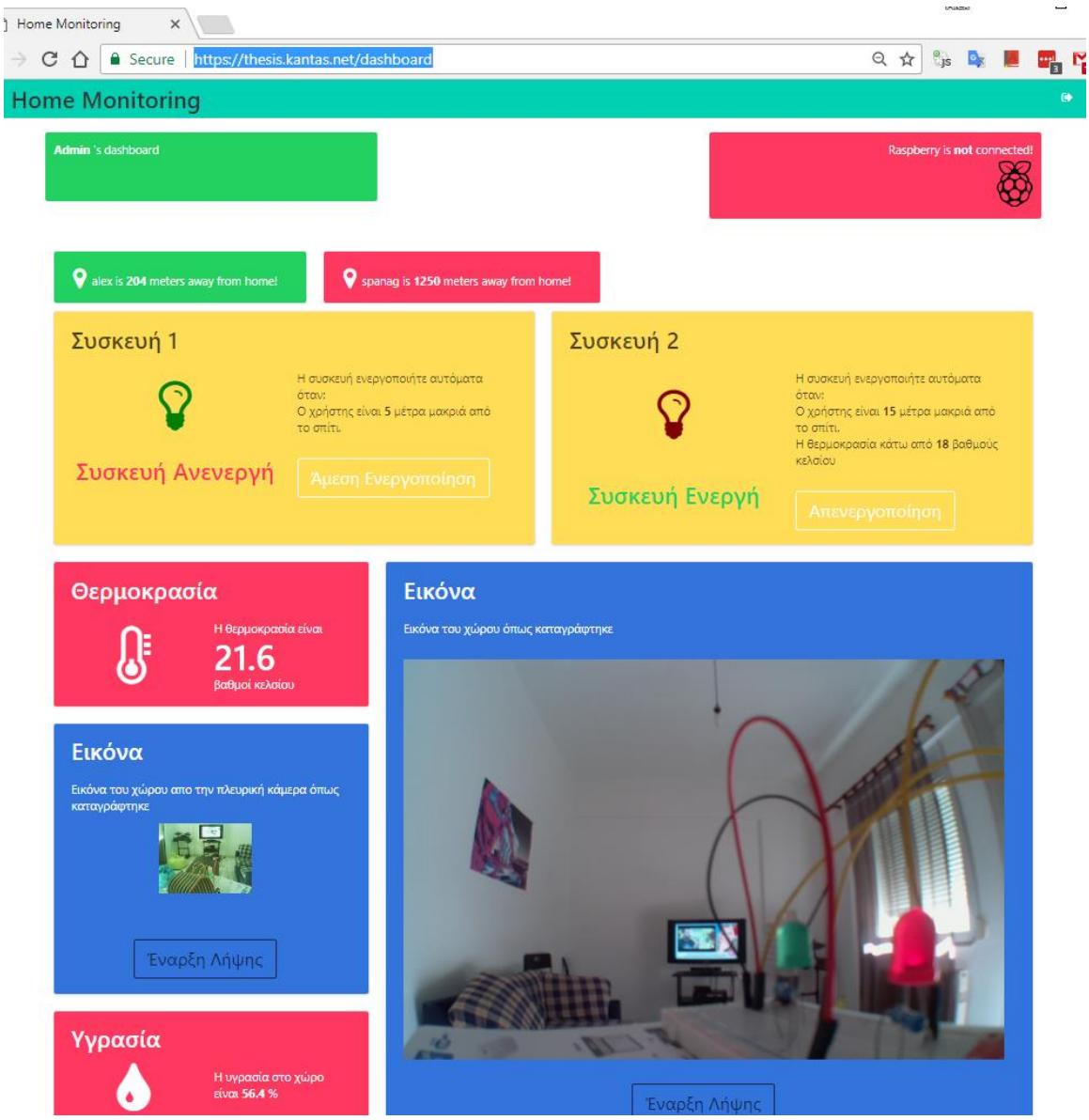


Εικόνα 28 Σχεδιάγραμμα του συστήματος με έμφαση στις τεχνολογίες της Web εφαρμογής του χρήστη

Ο χρήστης έχει πρόσβαση σε μια web εφαρμογή η οποία του προσφέρει πρόσβαση σε ένα πίνακα οργάνων (Dashboard) όπου μπορεί να δει και να τροποποιήσει τις τιμές των συσκευών στο Raspberry Pi.

Όντας web εφαρμογή είναι γραμμένη σε HTML, CSS και JavaScript. Η εφαρμογή χρησιμοποιεί UI στοιχεία από το CSS Framework Bulma. Οι κατάσταση των συσκευών λαμβάνεται σε πραγματικό χρόνο μέσω Socket.IO. Οι τιμές ενημερώνονται στο DOM μέσω του Vue.js.

Παρακάτω ένα απόσπασμα οθόνης από την dashboard του χρήστη:



Εικόνα 29 Απόσπασμα οθόνης από τη dashboard του χρήστη

Ο χρήστης λαμβάνει ενημέρωση αν η Raspberry Pi συσκευή είναι συνδεδεμένη. Αν είναι μπορεί να αποκτήσει πρόσβαση στις τιμές υγρασίας, θερμοκρασίας και να δει την κατάσταση των συσκευών. Πατώντας τα κουμπιά πάνω στις συσκευές μπορεί να αλλάξει την κατάσταση τους. Επίσης μπορεί να δει ζωντανά την εικόνα στο χώρο από τις κάμερες. Επίσης πατώντας στο όνομα κάποιου συνδεδεμένου χρήστη μπορεί να δει ζωντανά τη θέση του στο χάρτη. Οι ενέργειες αυτές περιγράφονται με μεγαλύτερη λεπτομέρεια στη συνέχεια.

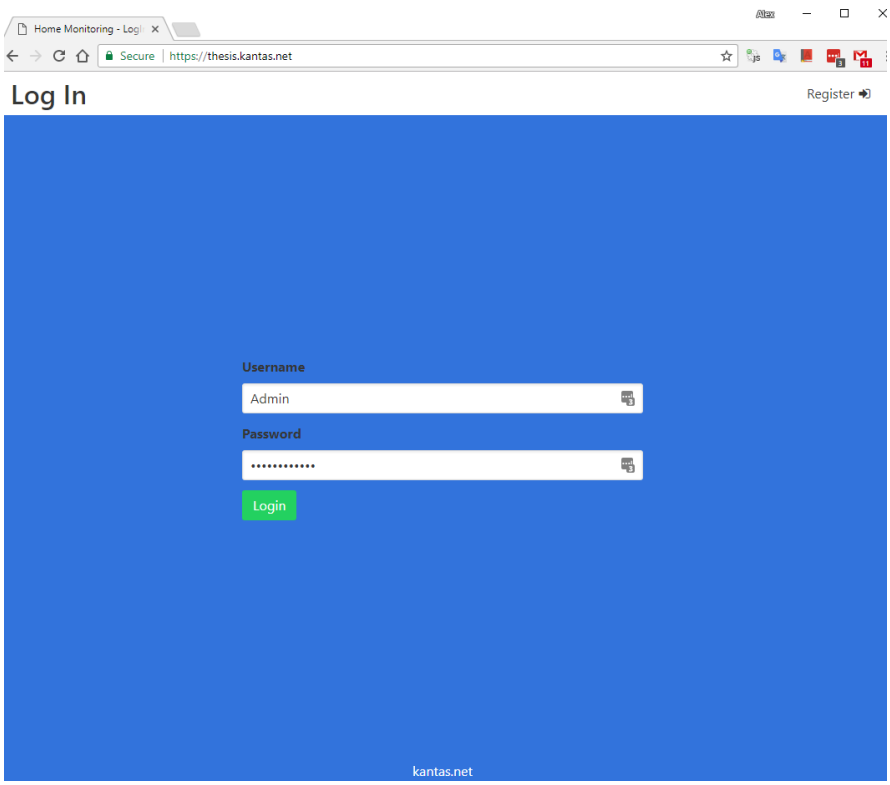
4.2 Ρύθμιση αντίστροφου διακομιστή μεσολάβησης

Για λόγους ασφάλειας και διαχείρισης πολλών NodeJS εφαρμογών γίνεται χρήση ενός αντίστροφου διακομιστή μεσολάβησης. Στο σύστημα της πτυχιακής χρησιμοποιούμε έναν NGINX server ο οποίος ακούει στην, προεπιλεγμένη για το http πρωτόκολλο, θύρα 80 και όταν ο χρήστης επισκέπτεται την διεύθυνση *thesis.kantas.net* κάνει reverse proxy στην τοπική θύρα 5000 που τρέχει η NodeJS web εφαρμογή. Επίσης ανακατευθύνει τα http αιτήματα στην ασφαλή https έκδοση. Ο κώδικας του αρχείου διαμόρφωσης του NGINX παρουσιάζεται παρακάτω:

```
server {
    listen 80 ;
    listen [::]:80 ;
    listen 443 ssl http2;
    listen [::]:443 ssl ;

    #
    #SSL Certificates
    #

    server_name thesis.kantas.net;

    location / {
        proxy_pass http://localhost:5000;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}

#Redirect http to https
server {
    listen 80;
    server_name thesis.kantas.net;

    return 301 https://$server_name$request_uri;
}
```

Πίνακας 3 Αρχείο διαμόρφωσης του NGINX

4.3 Πιστοποίηση χρηστών



Εικόνα 30 Απόσπασμα οθόνης για την είσοδο χρήστη

Κάθε χρήστης για να χρησιμοποιήσει το σύστημα θα πρέπει να έχει δημιουργήσει ένα νέο λογαριασμό. Οι λογαριασμοί χρηστών αποθηκεύονται σε μια MongoDB βάση δεδομένων. Για την διαχείριση της βάσης γίνεται χρήση του Object Document Mapper *mongoose*. Το σχήμα της βάσης για το κάθε χρήστη περιγράφεται από το παρακάτω απόσπασμα κώδικα:

```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
    username: { type: String, index: { unique: true } },
    password: String
});

module.exports = mongoose.model('User', userSchema);
```

Πίνακας 4 Κώδικας για την περιγραφή του Χρήστη στη MongoDB με χρήση mongoose

Μπορούμε να διακρίνουμε ότι κάθε χρήστης έχει δυο πεδία τα username και password με το username να πρέπει να είναι μοναδικό. Με τη μέθοδο model το mongoose δημιουργεί μια συλλογή (collection) με το όνομα Users (γίνεται μετατροπή στο πληθυντικό αυτόματα) με έγγραφα (documents) όπως περιγράφονται στο userSchema.

Για την πιστοποίηση των χρηστών χρησιμοποιούμε το passport.js. Το passport.js είναι middleware πιστοποίησης για το express.js. Για να χρησιμοποιήσουμε το passport.js πρέπει να ορίσουμε τη στρατηγική που θα χρησιμοποιηθεί ανάλογα το τρόπο σύνδεσης (όπως πχ σύνδεση μέσω Facebook, Google ή απλώς τοπικά username – password). Εφόσον ο χρήστης κάνει είσοδο μέσω username και password θα χρειαστούμε μια local-strategy. Ο κώδικας της δείχνει ως εξής:

```
const passport = require('passport');
const LocalStategy = require('passport-local').Strategy;
const User = require("../../models/user.model.js");
const mongoose = require('mongoose');
const mongoUrl = 'mongodb://localhost/thesis';
mongoose.connect(mongoUrl, { useMongoClient: true });
mongoose.Promise = global.Promise;


const userStategy = function () {
    passport.use(
        new LocalStategy({ usernameField: 'username', passwordField:
'password' },
            (username, password, done) => {
                User.findOne({ username, password })
                    .then(user => {
                        if (user) {
                            done(null, user);
                            return;
                        }
                        done(null, false, { message: 'Wrong creadantials'
});
                    })
                    .catch(err => console.log('Error' + err));
            }
        ));
}

module.exports = userStategy;
```

Πίνακας 5 Κώδικας local strategy για το passport.js

Εφόσον η επαλήθευση των στοιχείων του χρήστη στη βάση είναι σωστή καλούμε την συνάρτηση done περνώντας σαν δεύτερο όρισμα το χρήστη. Αυτό έχει σαν αποτέλεσμα το request object του express.js να έχει ένα πεδίο με το χρήστη. Με βάση αυτό μπορούμε να δημιουργήσουμε ένα middleware πιστοποίησης στο router του express.js.

Με τον παρακάτω κώδικα ελέγχουμε αν υπάρχει το user πεδίο στο request object. Αν υπάρχει το αφήνουμε το router να πάει στην επόμενη διαδρομή (route) καλώντας την συνάρτηση next(). Διαφορετικά αν δεν βρεθεί ο χρήστης ανακατευθύνουμε το χρήστη στην αρχική σελίδα για να κάνει είσοδο. Το middleware για την παραπάνω λειτουργία είναι το εξής

```
const authMildware = (req, res, next) => {
    if (req.user) return next();
    return res.redirect('/');
}
const mainRouter = express.Router();
mainRouter.use(authMildware);
```

Πίνακας 6 Κώδικας για το middleware πιστοποίησης χρήστη

Έτσι αν κάποιος μη συνδεδεμένος χρήστης δοκιμάσει για παράδειγμα να συνδεθεί στο https://thesis.kantas.net/dashboard θα μεταφερθεί αυτόματα στην αρχική σελίδα https://thesis.kantas.net/

Για να γίνει η πιστοποίηση του χρήστη την πρώτη φορά που θα εισαχθεί χρησιμοποιείται ο παρακάτω κώδικας. Η φόρμα για την είσοδο κατευθύνει το χρήστη στη αρχική διαδρομή / με post μέθοδο.

```
<form class="loginform" action="/" method="post">
```

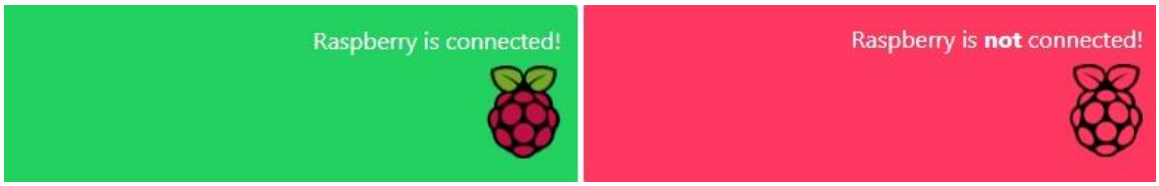
Εκεί στο router του express.js κάνουμε χρήση του passport.js και καλούμε τη μέθοδο authenticate περνώντας σαν πρώτο όρισμα το όνομα της στρατηγικής που θέλουμε να χρησιμοποιήσουμε. Στη προκειμένη περίπτωση την local.

```
const bodyParser = require('body-parser');
const urlencodedParser = bodyParser.urlencoded({ extended: false });
const passport = require('passport');
const authenticate = [ urlencodedParser,
        (req, res, next) => {
            passport.authenticate('local', (err, user, info) => {
                if (err) { console.log(err); return }
                if (!user) { res.render('login', { title: 'Home Monitoring
- LogIn', wrongInfo: true }); return; }
                req.logIn(user, err => {
                if (err) { console.log(err); return }
                return res.redirect('/dashboard');})(req, res, next); }]
const mainRouter = express.Router();
mainRouter.route('/').post(authenticate);
```

Πίνακας 7 Κώδικας για χρήση της local strategy όταν ο χρήστης υποβάλει τα στοιχειά

4.4 Έλεγχος της κατάστασης του Raspberry Pi

Στη dashboard της web εφαρμογής εμφανίζεται το κατάλληλο πλακίδιο για το αν είναι συνδεδεμένο ή όχι το Raspberry Pi.



Εικόνα 31 Πλακίδιο της Dashboard του χρήστη με την κατάσταση του Raspberry Pi

Ο Html / Vue.js κώδικας για την οπτική απεικόνιση των πλακιδίων φαίνεται παρακάτω

```
γ<div class="column  is-4 is-offset-4 notification"
:class="{'isdanger':!raspberryConnected,'is-success':raspberryConnected}">
<p class="has-text-right">Raspberry is
<strong v-show="!raspberryConnected">not</strong> connected!</p>
<p class="has-text-right littleMargin">
<img src="/public/img/raspConn.png" alt="raspberry-connected"
class="shadowfilter raspImage" v-if="raspberryConnected">
<img src="/public/img/raspNotConn.png" alt="raspberry-not-connected"
class="shadowfilter raspImage" v-else>
                                </p></div>
```

Πίνακας 8 Html κώδικας για την εμφάνιση της κατάστασης του Raspberry Pi

Η κατάσταση του Raspberry Pi ορίζεται από την μεταβλητή *raspberryConnected* και οι custom συναρτήσεις *listen* και *askReport* που αναλύονται παρακάτω περιέχουν κώδικα για την ενημέρωση της μεταβλητής.

```
const app = new Vue({el: '#root',
    data: {
        raspberryConnected: false,
        temperature: 0,
        humidity: 0,
        currentUser: '',
        loading: true
    },
    created() {listen.call(this); },
    mounted(){askReport();this.loading = false; }});
```

Πίνακας 9 Javascript κώδικας για την ενημέρωση της κατάστασης του Rasberry Pi

Η συνάρτηση *listen* ακούει στο raspberrystatus event του socket.io και ενημερώνει κατάλληλα την *raspberryConnected* μεταβλητή ενώ η askreport κάνει εκπέμπει ένα event στο server για να στείλει την ενημερωμένη κατάσταση.

```
function listen() {
      socket.on('raspberryStatus', data => {
      this.raspberryConnected = data.connected;
})}

function askReport() {
    socket.emit('raspberryStatus');
}
```

Πίνακας 10 Κώδικας για την ενημέρωση της κατάστασης του Raspberry Pi

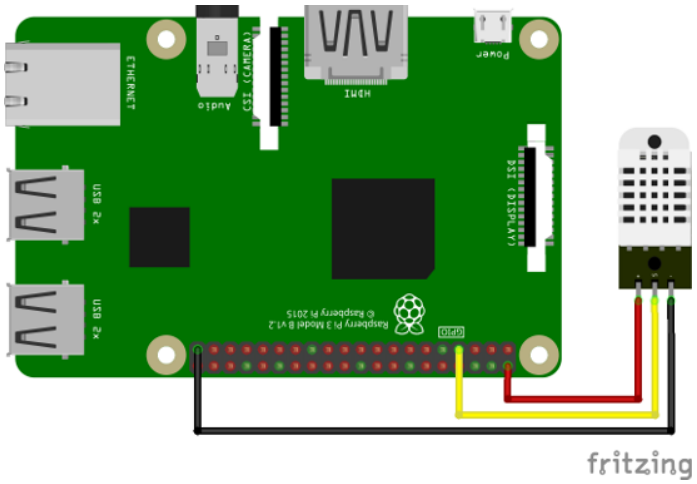
Στο κώδικα του server όταν γίνει αίτημα από την web εφαρμογή αποστέλλεται η κατάσταση του Raspberry Pi. Όταν το Raspberry Pi συνδεθεί για πρώτη φορά εκπέμπεται ένα *welcomeRaspberry* event. Ο server αποθηκεύει την κατάσταση σε true στη μεταβλητή raspberryConnected, την οποία χρησιμοποιεί για να απαντάει στα αιτήματα του client. Επίσης καταγράφεται το socket δίνοντας του μια ιδιότητα name με την τιμή raspberry. Έτσι στο disconnect event του Socket.io όταν αποσυνδεθεί το Raspberry Pi αποστέλλεται η ενημερωμένη τιμή στη web εφαρμογή.

```
    const io = require('socket.io').listen(server);
 // Application variables
    let raspberryConnected = false;
    io.on('connection', socket => {
        //User -> Server
        socket.on('raspberryStatus', data => {
            io.emit('raspberryStatus', { connected: raspberryConnected });
        });
        //Raspberry -> Server
        socket.on('welcomeRaspberry', data => {
            socket.name = 'raspberry';
            raspberryConnected = true;
            io.emit('raspberryStatus', { connected: raspberryConnected });
        });
       //Common
        socket.on('disconnect', () => {
            if (socket.name != 'raspberry') return;
            raspberryConnected = false;
            io.emit('raspberryStatus', { connected: raspberryConnected });
        })});}
```

Πίνακας 11 Κώδικας για την ενημέρωση κατάστασης του Raspberry Pi από την μεριά του Server

4.5 Λήψη θερμοκρασίας και υγρασίας

Για την λήψη τιμών θερμοκρασίας και υγρασίας θα χρησιμοποιήσουμε τον αισθητήρα DHT22 το οποίο συνδέουμε στο Raspberry Pi όπως δείχνει η παρακάτω εικόνα



Εικόνα 32 Σύνδεση του DHT22 στο Raspberry Pi

Ο αριστερός ακροδέκτης βρίσκεται στα 5V, ο δεξιός στη γείωση και ο μεσαίος στο GPIO Pin που θέλουμε να κάνουμε την ανάγνωση της τιμής. Για να έχουμε πρόσβαση στις τιμές του αισθητήρα χρησιμοποιούμε *NodeJS* και το *node-dht-sensor* πακέτο που κατεβάζουμε από το *npm*. Για να στείλουμε τις τιμές στο server χρησιμοποιούμε *Socket.IO*.

```
const url = 'https://thesis.kantas.net'; // Server URL
const socket = require('socket.io-client')(url);
const sensor = require('node-dht-sensor'); // Temperature - Humidity Sensor
const sensorPin = 17;
let weatherIvl = setInterval(updateWeatherData, 2500);
function updateWeatherData() {
    sensor.read(22, sensorPin, (error, temperature, humidity) => {
        if (error) { socket.emit('errorMessage', { error }); return; };
        socket.emit('setWeatherData', { temperature, humidity});
    });
}
```

Πίνακας 12 Κώδικας για την λήψη θερμοκρασίας από τον αισθητήρα και αποστολή στο Server

Με το παραπάνω κώδικα διαβάζουμε την θερμοκρασία και την υγρασία από τον αισθητήρα και τη στέλνουμε στο url του server. Ο server με τη σειρά του δρα σαν ενδιάμεσος κρίκος λαμβάνει τις τιμές θερμοκρασίας και υγρασίας και τις στέλνει στο χρήστη.

```
const app = require('express')();
const server = app.listen(80)// Listening
const io = require('socket.io').listen(server);
let temperature , humidity ;
io.on('connection', socket =>{
    socket.on('setWeatherData', data => {
        ({ temperature, humidity } = data);
        io.emit('weatherData', { temperature, humidity });
    })
})
```

Πίνακας 13 Κώδικας του Server για λήψη των τιμών θερμοκρασίας από το Raspberry Pi και προώθηση τους στην Web εφαρμογή του χρήστη

Ο χρήστης μπορεί να δει τις τιμές της θερμοκρασίας μέσω της web εφαρμογής

Εικόνα 33 Απόσπασμα οθόνης από την Dashboard που εμφανίζει τις τιμές υγρασίας και θερμοκρασίας

Ο κώδικας είναι γραμμένος με Vue.js για τις διαδραστικές λειτουργείες και με components του Bulma CSS για το αισθητικό κομμάτι. Ο html κώδικας για την υγρασία είναι ο εξής:

```
<div class="tile is-child box notification is-danger">
    <p class="title">Υγρασία</p>
    <div class="columns">
        <div class="column centeredElements">
            <span class="icon">i class="fa fa-tint fa-5x"></i></span>
        </div>
        <div class="column">
           <p>Η υγρασία στο χώρο είναι <strong>{{humidity}}</strong> %</p>
        </div>
    </div>
</div>
```

Πίνακας 14 Html κώδικας για την εμφάνιση της υγρασίας στο χρήστη

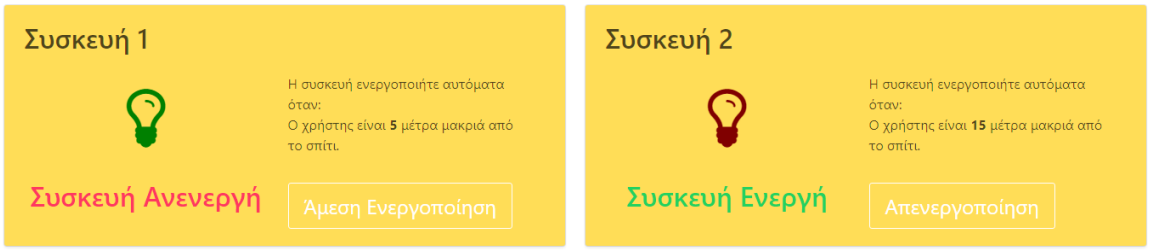
Το Vue.JS ενημερώνει τις τιμές στο DOM σε πραγματικό χρόνο με το παρακάτω κώδικα

```
const app = new Vue({
    el: '#root',
    data: {
        temperature: 0,
        humidity: 0,
    },
    created() {
        listen.call(this);
    }
});

function listen() {
    socket.on('weatherData', data => {
        this.temperature = data.temperature;
        this.humidity = data.humidity;
    })
}
```

Πίνακας 15 Κώδικας για την ενημέρωση των τιμών της θερμοκρασίας και υγρασίας στην dashboard του χρήστη

4.6 Διαχείριση της κατάστασης των LED συσκευών

Η κατάσταση των συσκευών εμφανίζεται στη Dashboard του χρήστη ως εξής



Εικόνα 34 Απόσπασμα Οθόνης από την Dashboard του χρήστη με την κατάσταση της κάθε συσκευής

Για να εμφανιστεί αυτό έχουμε δημιουργήσει ένα Vue.js component που λέγεται device

```
<device icon-class="fa-lightbulb-o greenBulp" :device-id='1' > Συσκευή 1</device>
```

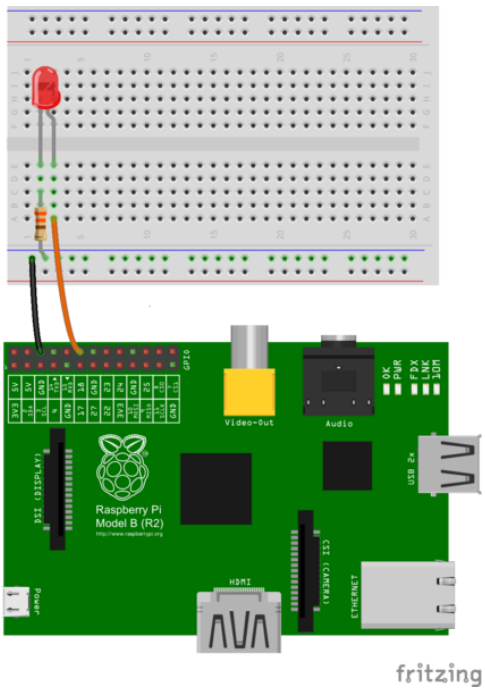
Το οποίο είναι σε θέση να αλλάξει οπτικά στο DOM την κατάσταση της συσκευής με βάσει τα δεδομένα που λαμβάνει από τον Server χάρη στον παρακάτω κώδικα

```
Vue.component('device', {
    props: {deviceId: { type: Number, required: true },
    data() { return {
            isDeviceEnabled: false,
            isLoading: false
        }},
    mounted() {socket.on('deviceStatus', (data) => {
            if (data.deviceId == this.deviceId) {
                this.isDeviceEnabled = data.isEnabled;
                this.isLoading = false;
//
```

Πίνακας 16 Κώδικας για την ενημέρωση της κατάστασης των συσκευών στην dashboard του χρήστη

Όταν γίνεται κλικ το κουμπί αποστέλλεται εντολή για την αλλαγή κατάστασης της συσκευής στο Server ο οποίος με τη σειρά του τη προωθεί στο Raspberry Pi.

Για το σύστημα της πτυχιακής οι συσκευές που χρησιμοποιούμε είναι απλά LEDs, ωστόσο η λογική θα ήταν παρόμοια αν αντί για led υπήρχε ένα relay συνδεδεμένο με κάποια οικιακή συσκευή. Για να συνδέσουμε τα LED στο Raspberry Pi τοποθετούμε το μικρό ακροδέκτη στη γείωση και το μεγάλο στο GPIO Pin όπως φαίνεται στο παρακάτω σχήμα



Εικόνα 35 Σύνδεση LED στο Raspberry Pi

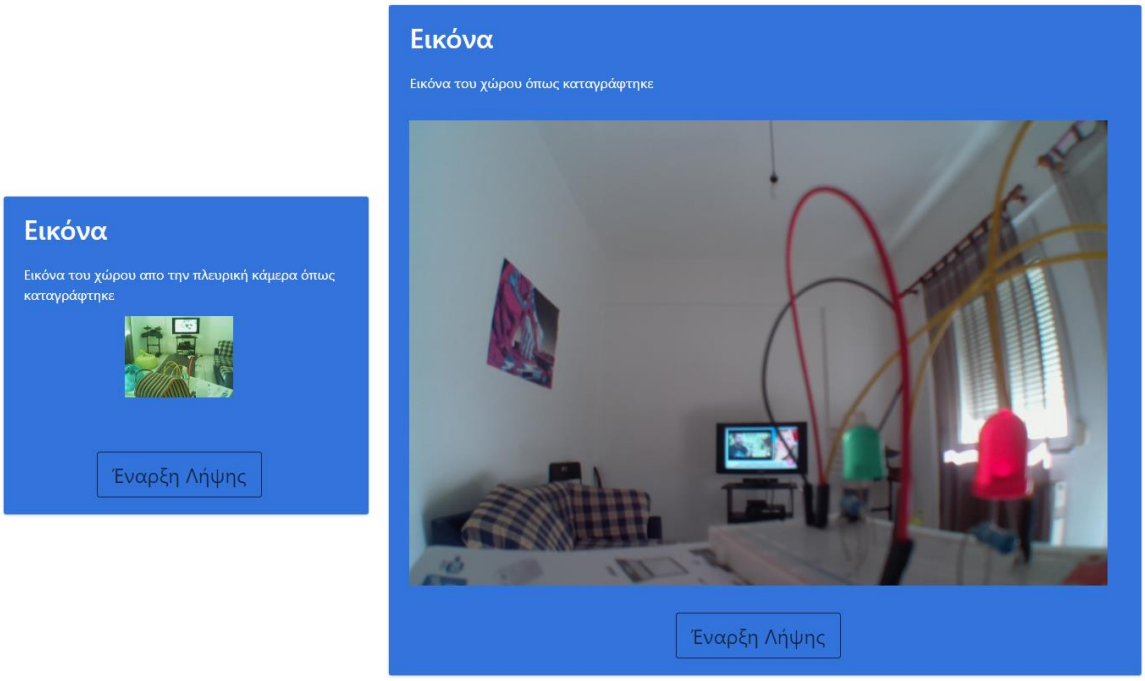
Για να τροποποιήσουμε της τιμές των GPIO συσκευών κάνουμε χρήση του *rpi-gpio* πακέτου από το npm. Τροποποίηση των τιμών και επικοινωνία με το Server γίνεται με την χρήση του παρακάτω κώδικα

```
const gpio = require('rpi-gpio');
const devices = [
    {   deviceId: 1,
        pin: 13,
        isEnabled: true,
        color: 'red'},
    {   deviceId: 2,
        pin: 12,
        isEnabled: true,
        color: 'green'}]
//Set up pins for write
devices.forEach(device => {
    gpio.setup(device.pin, gpio.DIR_OUT, () => gpio.write(device.pin,
device.isEnabled));
});

socket.on('setDeviceStatus', data => {
    const devIndex = devices.findIndex(d => data.deviceId == d.deviceId);
    if (devIndex === -1) { return socket.emit('errorMessage', { error: 'noDeviceFound' }) };
    if (!data.justReport) devices[devIndex].isEnabled = data.isEnabled;
    const { deviceId, pin, isEnabled } = devices[devIndex];
    gpio.write(pin, isEnabled, error => {
        if (error) { socket.emit('errorMessage', { error }); return; };
        socket.emit('deviceStatus', { password, deviceId, isEnabled });
    })
});
```

Πίνακας 17 Κώδικας για τροποποίηση των GPIO συσκευών με βάσει τα μηνύματα από το Server

4.7 Λήψη εικόνας χώρου από τις κάμερες στο Raspberry Pi

Στην εφαρμογή του χρήστη η εικόνα από τις κάμερες εμφανίζεται στον χρήστη ως εξής:



Πίνακας 18 Απόσπασμα Οθόνης από την dashboard του χρήστη με την εικόνα από τις κάμερες

Για να γίνει αυτό έχουμε δημιουργήσει στο Vue.js ένα component που λέγεται video-area

```
<video-area css-classes="image is-128x128 sideCam" cam-id="2">
Εικόνα του χώρου απο την πλευρική κάμερα όπως καταγράφτηκε
</video-area>
```

Ο κώδικας για την ανανέωσης εικόνας στην dashboard όταν αυτή λαμβάνεται από τον Server συνοψίζεται ως εξής

```
Vue.component('video-area', {
    props: ['cssClasses', 'imageUrl', 'camId'],
    data() {return {
            dateStamp: 0,
            image: ''
                    }},
    template,
    mounted() {
        this.image = this.imageUrl;
        socket.on('imageStream', (data) => {
            if (data.camId == this.camId) {
                this.dateStamp = parseInt(data.dateStamp);
                this.image=data.image;}}); },});
```

Πίνακας 19 Κώδικας για την ενημέρωση της εικόνας του χώρου στην dashboard του χρήστη

Στο Raspberry Pi έχουμε συνδέσει δυο κάμερες. Η πρώτη αποτελεί ένα module ευρυγώνιας κάμερας η οποία συνδέεται στη συσκευή μέσω καλωδιοταινίας.

Εικόνα 36 Φωτογραφία του Raspberry Pi Wide Angle Camera Module συνδεδεμένο στη συσκευή

Η δεύτερη κάμερα είναι μια τυπική web camera, συγκεκριμένα η Logitech Webcam c170, η οποία συνδέεται στο Raspberry Pi μέσω USB.

Εικόνα 37 Φωτογραφία της κάμερας που χρησιμοποιήθηκε στο σύστημα Logitech Webcam c170

Αποκτάμε πρόσβαση στην εικόνα από τις κάμερες με την χρήση των npm πακέτων raspicam και node-web. Για να στείλουμε την εικόνα στον Server την μετατρέπουμε σε base64 String και την μεταδίδουμε μέσω Socket.IO

Με τον παρακάτω κώδικα κάνουμε συνεχώς stream μια νέα εικόνα από την webcam στο Server

```
const webcamOptions = { width: 256, height: 256, output: "jpeg",
callbackReturn: 'base64' };
const Webcam = require("node-webcam").create(webcamOptions);

function webCamShot() {
  Webcam.capture("picture", (err, image) => {
   if (err) { return err; }
   socket.emit('imageStream', { camId: 2, dateStamp: Date.now(), image });
   SetTimeout(webCamShot, 100);
   });
}

webCamShot();
```

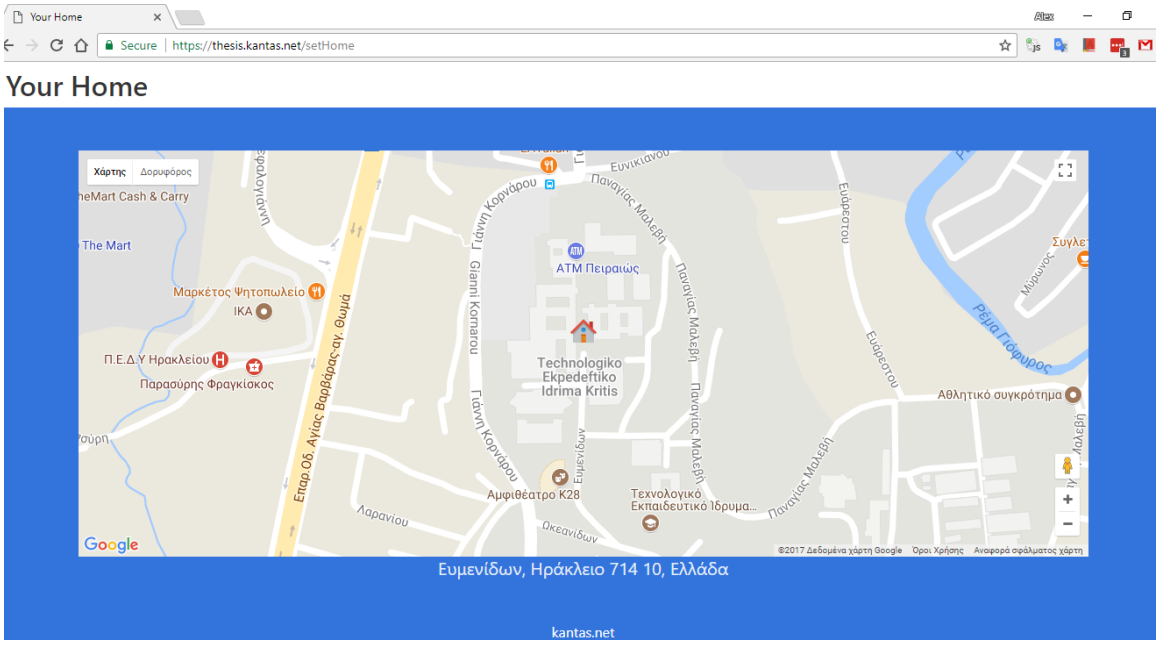
Πίνακας 20 Κώδικας για stream της εικόνας από την webcam

Και με τον παρακάτω κώδικα κάνουμε stream εικόνες στο Server από την κάμερα στο Raspberry Pi που είναι συνδεδεμένη με καλωδιοταινία

```
const base64Img = require('base64-img');
const raspiCamOptinos = {
    mode: 'timelapse',
    timeout: 120000,
    timelapse: 1000,
    output: __dirname + '/photo.jpg',
    vf: true,
    w: 600,
    h: 400
}
const camera = new require("raspicam")(raspiCamOptinos);

camera.on("read", (err, dateStamp, file) => {
    base64Img.base64(file, (err, image) => {
        socket.emit('imageStream', { camId: 1, dateStamp, image });
    });
});

camera.start();
```

Πίνακας 21 Κώδικας για stream της εικόνας από την κάμερα του Raspberry Pi

4.8 Ορισμός τοποθεσίας σπιτιού



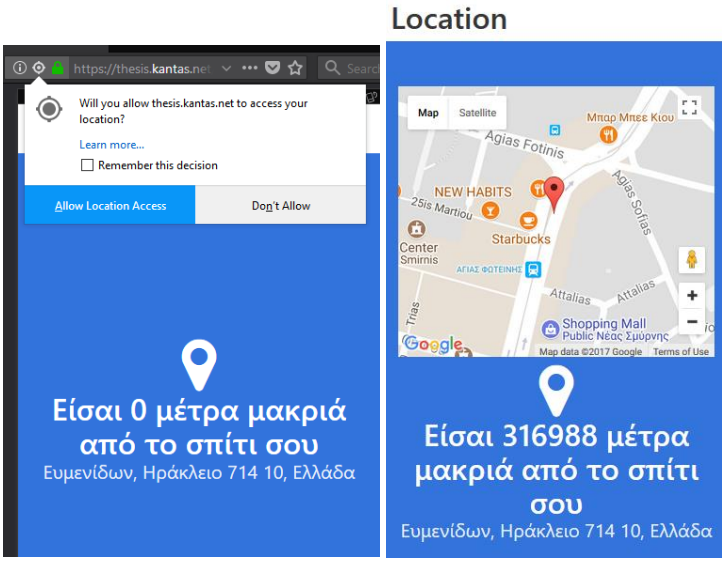
Εικόνα 38 Στιγμιότυπο οθόνης από την επιλογή σπιτιού στην web εφαρμογή

Ο κάθε χρήστης μπορεί να επιλέγει την τοποθεσία του σπιτιού μέσω της web εφαρμογής. Με βάση την τοποθεσία αυτή γίνεται ο υπολογισμός της απόστασης. Η web εφαρμογή επιτρέπει στο χρήστη να κάνει κλικ σε μια τοποθεσία στο χάρτη και παίρνει άμεσα feedback με την ακριβή διεύθυνση του σπιτιού. Όλα αυτά χάρη στο API ανάστροφης γεωκωδικοποίησης των Google Maps. Η τοποθεσία αποστέλνεται στο Server ασύγχρονα μέσω fetch API. Ο Server αποθηκεύει την τοποθεσία και πλέον όλοι οι υπολογισμοί απόστασης γίνονται βάσει αυτής.

```
function placeMarkerAndPanTo(location, map) {
    marker.setPosition(location);
    map.panTo(location);
    geocoder.geocode({ location }, (results, status) => {
        if (status != 'OK') { console.log("Error: " + status); return }
        if (!results[0]) { console.log("No results"); return }
        app.homeAddr = results[0].formatted_address;
        app.lat = location.lat();
        app.lng = location.lng();
    });
}
```

Πίνακας 22 Κώδικας για την ανάστοφη γεωκωδικοίηση μέσω των Google Maps APIs

4.9 Καταγραφή τοποθεσίας χρήστη



Εικόνα 39 Οθόνη καταγραφής τοποθεσίας

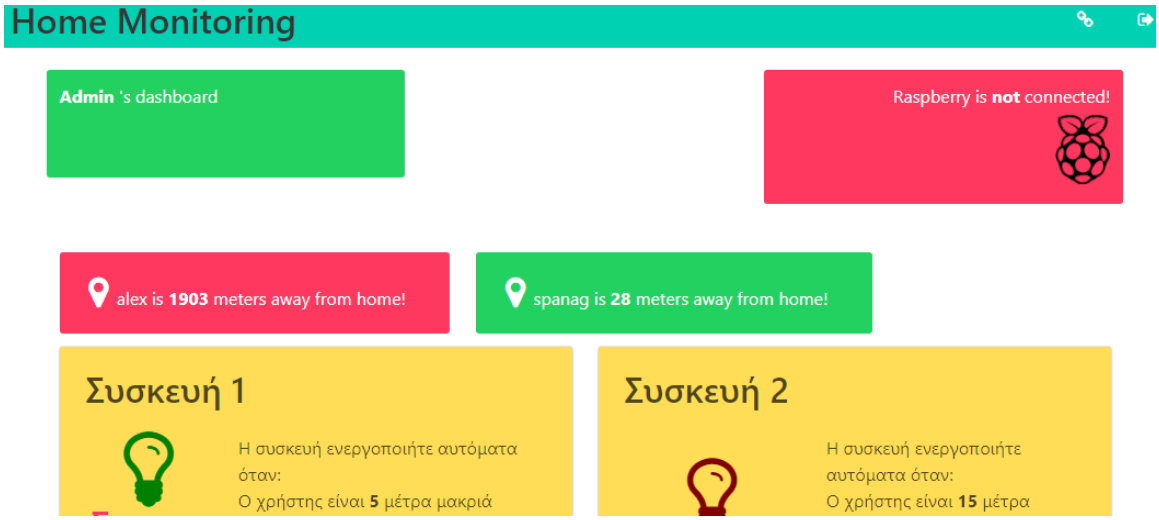
Ο χρήστης επισκεπτόμενος τις */track* και */track/map* ενεργοποιεί την καταγραφή της τοποθεσίας του. Η καταγραφή γίνεται με χρήση του geolocation web API. Η εν λόγω διεπαφή προσφέρει την μέθοδο *navigator.geolocation.watchPosition* η οποία μας δίνει τις πληροφορίες για την θέση του χρήστη όπως δίνεται από το υλικό της συσκευής που χρησιμοποιεί. Αν χρησιμοποιεί GPS το στίγμα θα είναι απόλυτα ακριβές.

```
function startTracking() {
    if (!navigator.geolocation) return alert("Your device doesn't support
geolocation!");
    const options = { enableHighAccuracy: true, maximumAge: 2000 };
    navigator.geolocation.watchPosition(watchLocation, displayError,
options)
}
function watchLocation(position) {
 const { latitude, longitude } = position.coords;
 app.distance = geolib.getDistance({ latitude, longitude },
homePostition);
 socket.emit('setLocation', {distance:app.distance,location:{latitude,
longitude },username});
}
```

Πίνακας 23 Κώδικας για την λήψη και αποστολή της τοποθεσίας

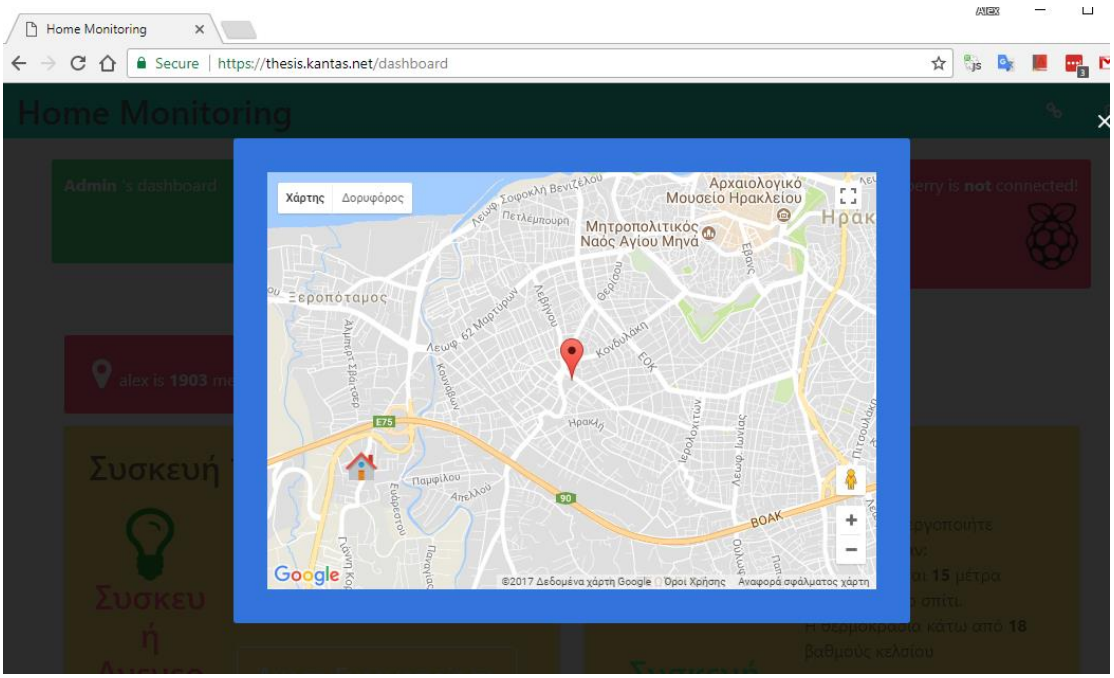
Ο Server λαμβάνει το στίγμα του χρήστη και βάσει κάποιον προκαθορισμένων κανόνων ενεργοποιεί απομακρυσμένα τις συσκευές.

Εφόσον ένας χρήστης βρίσκεται στη σελίδα καταγραφής τοποθεσίας οι υπόλοιποι χρήστες μπορούν να έχουν πρόσβαση σε αυτή μέσω της dashboard. Στη Dashboard εμφανίζεται ένα τετράγωνο με το όνομα του κάθε χρήστη και την απόσταση αυτού από το σπίτι.



Εικόνα 40 Στιγμιότυπο από τη Dashboard με πληροφορίες για την απόσταση των χρηστών

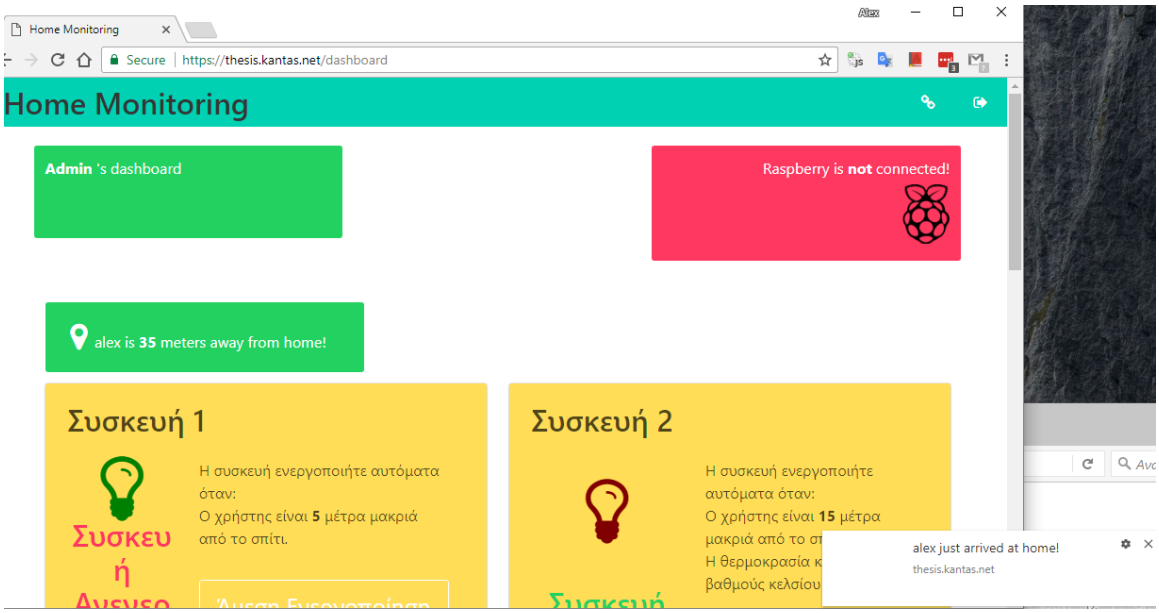
Κάνοντας κλικ πάνω στο κάθε τετράγωνο ανοίγει μες στη σελίδα ένα παράθυρο με την ακριβή θέση του χρήστη. Η θέση αυτή ενημερώνεται σε πραγματικό χρόνο μέσω Socket.IO.



Εικόνα 41 Το στίγμα της θέσης ενός χρήστη από την dashboard κάποιου άλλου χρήστη

4.10 Ειδοποίηση όταν ο χρήστης φτάνει σπίτι

Η web εφαρμογή φροντίζει να ενημερώνει τους χρήστες όταν κάποιος άλλος χρήστης έφτασε στο σπίτι. Η ειδοποίηση γίνεται όταν ένας χρήστης βρίσκεται εντός ενός προκαθορισμένου εύρους που το έχουμε ορίσει στα 100 μέτρα. Η ειδοποίηση εμφανίζεται στο χρήστη μέσω του Notification Web API.



Εικόνα 42 Παράδειγμα ειδοποίησης στον ερχομό ενός χρήστη στο σπίτι

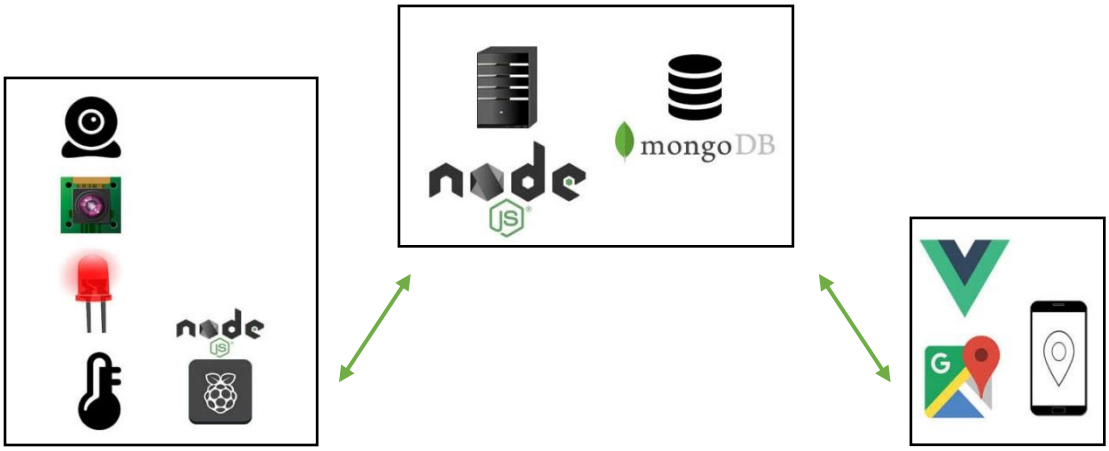
Ο τρόπος εμφάνισης της ειδοποίησης διαφέρει ελαφρός από περιηγητή σε περιηγητή και με το αν πρόκειται για επιτραπέζιο υπολογιστή ή κινητό. Για όσες συσκευές δεν υποστηρίζουν το Notification API η ειδοποίηση γίνεται απλώς μέσα στη σελίδα με χρήση CSS και JavaScript.

Για να επιτευχθεί αυτό προγραμματιστικά γίνεται χρήση ενός Vue component με το όνομα *userbox* το οποίο κρατά περιέχει μια μεταβλητή *distance* για την απόσταση του χρήστη. Δίνεται μέριμνα ώστε να μην εμφανίζεται η ειδοποίηση επαναλαμβανόμενα παρά μόνο όταν ο χρήστης όντως εισέρχεται στο σπίτι.

```
Vue.component('userbox', {
    data() {
        return {
            userDistance: 1000,
        }
    },
    mounted() {
        socket.on('userLocation', (data) => {
            if (data.username != this.username) return;
            const distanceLimit = 100;
            const userJustArrived = data.distance < distanceLimit &&
!(this.userDistance > 0 && this.userDistance < distanceLimit) &&
data.distance < this.userDistance // True if user not home, new distance
less than distanceLimit and smaller tha previous
            if (userJustArrived) {
                let title = `${this.username} just arrived at home!`;
                new Notification(title);
            }
            this.userDistance = data.distance;
        });
    }
});
```

Πίνακας 24 Κώδικας για τον έλεγχο αν ο χρήστης μόλις έφτασε στο σπίτι και εμφάνιση ειδοποίησης

5. Αποτελέσματα και Συμπεράσματα



Εικόνα 43 Αναλυτικό σχεδιάγραμμα του συστήματος

Στη παρούσα πτυχιακή είδαμε βήμα – βήμα την δημιουργία και υλοποίηση ενός σύγχρονου IoT συστήματος. Το σύστημα συνδυάζει μια πληθώρα συσκευών και την αλληλοεπικοινωνία μεταξύ τους. Οι χρήστες μπορούν να έχουν απομακρυσμένη επίβλεψη και έλεγχο του σπιτιού. Πράγμα πολύ χρήσιμό καθώς μπορούν να αποφευχθεί λόγου χάρη άσκοπη κατανάλωση ενέργειας και πολύ βασικό ότι μπορεί να υπάρχει άμεση αντίδραση σε περίπτωση που κάποιός άγνωστος παραβίασε το σπίτι. Επίσης περιλαμβάνει μια απόλυτα λειτουργική εφαρμογή καταγραφής τοποθεσίας. Μια τέτοια εφαρμογή μπορεί να βρει πολλές εφαρμογές στην καθημερινότητα των χρηστών. Λόγου χάρη μπορεί να χρησιμοποιηθεί από γονείς για την επίβλεψη των παιδιών. Οι αυτοματισμοί με βάση την τοποθεσία μπορεί στην παρούσα πτυχιακή να ανοιγοκλείνουν αυτόματα μόνο τα LED αλλά με ίδια λογική στο Server και την εφαρμογή του χρήστη μπορούν να ελεγχθούν όλες οι οικιακές συσκευές. Έτσι μπορεί ο κάθε χρήστης μπορεί όταν φτάνει σπίτι του να βρίσκει τα φώτα ανοιχτά το σπίτι στη σωστή θερμοκρασία και ζεστό νερό στο μπάνιο. Όλα αυτά με μικρές τροποποιήσεις στο τρόπο λειτουργίας των οικιακών συσκευών. Το τελικό αποτέλεσμα της παρούσας πτυχιακής εμπλουτίζει την IoT κοινότητα παρουσιάζοντας μια νέα εφαρμογή και έναν σύγχρονο τρόπο υλοποίησης της. Η πτυχιακή κατάφερε να συνδυάσει πολλές τεχνολογίες και παρουσιάζει ένα τρόπο υλοποίησης ενός Location-based IoT συστήματος που μπορεί να χρησιμοποιηθεί άμεσα από τον καθένα.

5.1 Μελλοντικές επεκτάσεις

Το σύστημα της παρούσας πτυχιακής έχει υλοποιηθεί με τέτοιο τρόπο ώστε να είναι συντηρήσιμο και να μπορεί να δεχτεί εύκολα αλλαγές και βελτιώσεις. Έχει χτιστεί πάνω στο Model View Controller design pattern πράγμα που προσφέρει μεγάλη ευελιξία σε μελλοντικές αλλαγές.

Μια πρώτη επέκταση που μπορεί να υπάρξει στην εφαρμογή είναι η αντικατάσταση των LEDs με relays τα οποία τροφοδοτούν μια ηλεκτρική συσκευή. Στο παρόν σύστημα δίνεται ένας λογικός άσσος (3 volt) όταν θέλουμε το LED να είναι ενεργό και ένα λογικό μηδέν (0 volt) όταν είναι ανενεργό. Η ακριβώς ίδια λογική θα υπήρχε και με τα relays. Η μόνη σημαντική αλλαγή που απαιτείται είναι η αλλαγή στην GPIO συνδεσμολογία και σύνδεση μιας ηλεκτρικής συσκευής με το relay. Μια τέτοια αλλαγή θέλει πολύ μικρές έως μηδενικές αλλαγές στο μέχρι τώρα κώδικα της εφαρμογής.

Επίσης το παρών σύστημα για να μεταδώσει ζωντανή εικόνα στο χώρο στέλνει συνεχώς εικόνες σε base64 String μορφή. Πράγμα που σημαίνει ότι το κάθε frame δεν έχει γνώσει του προηγούμενου οπότε δεν υπάρχει ένα ομαλό αποτέλεσμα στο βίντεο που λαμβάνει ο χρήστης. Μια λύση και πρόταση για μελλοντική επέκταση θα ήταν χρήση άλλου τρόπου για την μετάδοση βίντεο όπως για παράδειγμά το ανοιχτό format αρχείων πολυμεσικού περιεχομένου WebM.

Τα διαδικτυακά APIs είναι ακόμα σε πρώιμο στάδιο. Η εφαρμογή για την καταγραφή θέσης του χρήστη, αν και λειτουργεί απροβλημάτιστα λόγω του περιορισμού των Web APIs δεν μπορεί να εκτελεστεί στο υπόβαθρο. Αυτό επηρεάζει την συνολική εμπειρία του χρήστη. Μια λύση που θα μπορούσε να υπάρξει είναι η χρήση μιας native εφαρμογής. Η MVC αρχιτεκτονική της εφαρμογής επιτρέπει την αλλαγή του front end κομματιού χωρίς να επηρεάζεται καθόλου το back-end. Στη περίπτωση που παραδείγματος χάρη που προστεθούν νέα web APIs αλλά και στην περίπτωση της πλήρους αλλαγής της εφαρμογής του χρήστη με native εφαρμογές ανά πλατφόρμα (Android , IOS, κτλ).

Βιβλιογραφία

1. **Marcel Mauer.** ***An Internet of Narrative, Making Sense of the IoT.*** **[Ηλεκτρονικό] Ιούνιος 2016. http://www.marcelmauer.eu/an-internet-of-narrative-making-sense-of-the-iot/.**

2. Beebom.com. ***15 Examples of Internet of Things Technology in Use Today.*** **[Ηλεκτρονικό] Φεβρουάριος 2017. https://beebom.com/examples-of-internet-of-things-technology/.**

3. Techradar.com. ***Amazon Echo vs Apple HomePod vs Google Home: the battle of the smart speakers.*** **[Ηλεκτρονικό] Οκτώβριος 2015. http://www.techradar.com/news/amazon-echo-vs-homepod-vs-google-home-the-battle-of-the-smart-speakers.**

4. Geoawesomeness.com. ***Inside Geo IoT: Intelligent geolocation of assets and people.*** **[Ηλεκτρονικό] Μάιος 2017. http://geoawesomeness.com/inside-geo-iot-intelligent-geolocation-assets-people/.**

5. Techspot.com. ***Interview with Raspberry's Founder Eben Upton.*** **[Ηλεκτρονικό] Μάιος 2012. https://www.techspot.com/article/531-eben-upton-interview/.**

6. Developer.mozilla.org. ***Introduction to the DOM.*** **[Ηλεκτρονικό] https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction.**

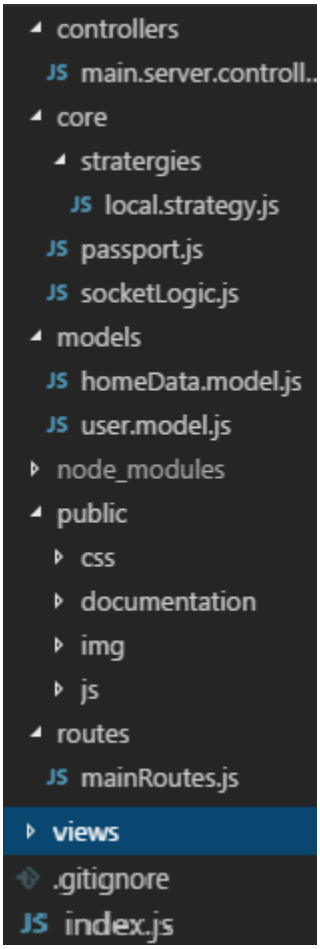
7. Jonathan Robie, Texcel Research. w3.org. ***What is the Document Object Model?*** **[Ηλεκτρονικό] https://www.w3.org/TR/WD-DOM/introduction.html.**

8. Roland Guijt. Pluralsight.com. ***Getting started with ES2017/ES8.*** **[Ηλεκτρονικό] Ιούλιος 2017. https://www.pluralsight.com/courses/es2017-es8-getting-started.**

9. Ben Jaffe. Udacity.com. ***JavaScript Design Patterns.*** **[Ηλεκτρονικό] https://www.udacity.com/course/javascript-design-patterns--ud989.**

Παράρτημα

Παρακάτω απεικονίζεται η βασική δομή των φακέλων και αρχείων για το server και τη web εφαρμογή του συστήματος.



Backend

Το αρχείο *index.js* είναι το entry point της εφαρμογής και ο κώδικας του παρουσιάζεται παρακάτω:

/index.js

```
// Libraries
const express = require('express');
const bodyParser = require('body-parser');
const cookieParser = require('cookie-parser');
const session = require('express-session');
const assert = require('assert');
const cons = require('consolidate');

// Basic Variables
const app = express();
const port = 5000;
```

```
// Router
const mainRouter = require('./routes/mainRoutes')();

// Set up view engine and views extension
app.use('/public', express.static(__dirname + '/public')); //file in
public directory served in public route
app.engine('html', cons.ejs); // assign the ejs engine to .html files
app.set('view engine', 'html'); // set .html as the default extension
app.set('views', __dirname + '/views');

// Seesions and Passport setup
app.use(session({ secret: 'th3s1s', saveUninitialized: true, resave: true
}));
require('./core/passport')(app);

// Socket IO logic
require('./core/socketLogic')(app, port);

// Assign routes
app.use('/', mainRouter);
```

Το αρχείο */routes/mainRoutes.js* παρουσιάζει τις πιθανές διαδρομές της εφαρμογής και το */controllers/main.server.controller.js* τον κώδικα που εκτελείτε όταν κάποιος πλοηγείτε στις διαδρομές αυτές.

/routes/mainRoutes.js

```
// Libraries
const express = require('express');
const mainController =
require('../controllers/main.server.controller.js');


//
const mainRouter = express.Router();

//Router object
const Router = function () {

    mainRouter.route('/').get(mainController.login);
    mainRouter.route('/').post(mainController.authenticate);
    mainRouter.route('/links').get(mainController.links);
    mainRouter.route('/documentation').get(mainController.documentation);
    mainRouter.route('/logIn').get(mainController.loginForm);
    mainRouter.route('/logOut').get(mainController.logout);
```

```
    mainRouter.route('/register').get(mainController.registerForm);
    mainRouter.route('/register').post(mainController.register);

mainRouter.route('/upLoadImage/camera').post(mainController.camImageStore)
;
    mainRouter.use(mainController.authMildware);
    mainRouter.route('/dashboard').get(mainController.home);
    mainRouter.route('/dashboard/simple').get(mainController.homeSimple);
    mainRouter.route('/setHome').get(mainController.setHome);
    mainRouter.route('/setHome').post(mainController.updateHomeData);
    mainRouter.route('/track').get(mainController.trackUser);
    mainRouter.route('/track/map').get(mainController.trackUserMap);

mainRouter.route('/track/map/simulate').get(mainController.trackUserSim);
    mainRouter.route('/canvas').get(mainController.canvas);

    return mainRouter;
}
```

/controllers/main.server.controller.js

```
// Libraries
const multer = require('multer');
const upload = multer({ dest: 'tempFiles' });
const fs = require('fs');
const bodyParser = require('body-parser');
const jsonParser = bodyParser.json()
const urlencodedParser = bodyParser.urlencoded({ extended: false });
const passport = require('passport');
const User = require("../models/user.model.js");

//
let homeInfo = { lat: 35.32098178540996, lng: 25.10274052619934 };
const url = 'https://thesis.kantas.net/';

const mainController = function () {


    const dataPesistanceMildware = (req, res, next) => {
        req.homeInfo = homeInfo;
        next();
    }

    const login = (req, res) => {
        if (req.user) return res.redirect('/dashboard');
```

```
        res.render('login', {
            title: 'Home Monitoring - LogIn',
            wrongInfo: false
        })
    }

    const loginForm = (req, res) => {
        res.render('login', {
            title: 'Home Monitoring - LogIn',
            wrongInfo: false
        })
    }

    const links = (req, res) => {
        res.render('links', {
            title: 'Home Monitoring - links'
        })
    }

    const documentation = (req, res) => {
        res.redirect('/public/documentation/kantasThesis.pdf')
    }

    const authenticate = [
        urlencodedParser,
        (req, res, next) => {
            passport.authenticate('local', (err, user, info) => {
                if (err) { console.log(err); return }
                if (!user) { res.render('login', { title: 'Home Monitoring
- LogIn', wrongInfo: true }); return; }
                req.logIn(user, err => {
                    if (err) { console.log(err); return }
                    return res.redirect('/dashboard');
                });
            })(req, res, next);
        }]

    const logout = (req, res) => {
        req.logout();
        res.redirect('/');
    }

    const registerForm = (req, res) => {
        res.render('register', {
            title: 'Home Monitoring - Register',
```

```
            wrongInfo: false
        })
    }

    const register = [
        urlencodedParser,
        (req, res) => {
            const { username, password } = req.body;
            const user = new User({
                username,
                password
            });
            user.save()
                .then(r => {
                    res.send('You are registered successfully! <a
href="/">You can log-in now!</a>');
                })
                .catch(err => {
                    res.end('Error during registration' + err);
                })
        }];

    const authMildware = (req, res, next) => {
        if (req.user) {
            next();
            return;
        }
        return res.redirect('/');
    }

    const home = (req, res) => {
        res.render('index', {
            title: 'Home Monitoring',
            username: req.user.username,
            url
        })
    }

    const homeSimple = (req, res) => {
        res.render('indexSimple', {
            title: 'Home Monitoring',
            username: req.user.username,
            url
        })
    }
```

```
    const camImageStore = [
        upload.single('camimage'),
        (req, res) => {
            console.log('Image 4', req.body);
            if (!req.file) {
                res.status(500).json({ error: 'noFile' });
                req.io.emit('errorMessage', { error });
                return;
            }
            fs.rename(req.file.path, `public/img/${req.body.camName}.jpg`,
(error) => {
                if (error) {
                    req.io.emit('errorMessage', { error });
                    console.log(error);
                    return;
                }
                req.io.emit('newImage', { dateStamp: req.body.dateStamp,
camId: req.body.camId });
                res.json(req.file);
            })
        }];

    const setHome = (req, res) => {
        res.render('homeLocation', {
            title: "Your Home",
            homeAddr: `Please select your Home`,
            lat: homeInfo.lat,
            lng: homeInfo.lng
        });
    }

    const updateHomeData = [jsonParser, (req, res) => {
        const { lat, lng, geocodedAddress } = req.body;
        console.log(req.body);
        if (!req.body.lat) { res.json({ error: "Couldn't update home" });
return; }
        homeInfo = { lat, lng, geocodedAddress };
        res.json({
            homeAddr: `Address successfuly set to:
${homeInfo.geocodedAddress}`,
            lat: homeInfo.lat,
            lng: homeInfo.lng
        });
    }]
```

```
    const trackUser = (req, res) => {
        console.log(homeInfo);
        res.render('location', {
            title: "Location",
            home: homeInfo,
            username: req.user.username,
            url
        });
    };

    const trackUserMap = (req, res) => {
        console.log(homeInfo);
        res.render('locationMap', {
            title: "Location",
            home: homeInfo,
            username: req.user.username,
            url
        });
    };

    const trackUserSim = (req, res) => {
        console.log(homeInfo);
        res.render('locationMapSim', {
            title: "Location",
            home: homeInfo,
            username: req.user.username,
            url
        });
    };

    const canvas = (req, res) => {
        res.render('canvas', {
            url
        });
    };

    return { documentation, register, links, registerForm, homeSimple,
logout, trackUserSim, loginForm, trackUserMap, login, authMildware,
authenticate, dataPesistanceMildware, home, camImageStore, setHome,
updateHomeData, trackUser, canvas }
}

module.exports = mainController();
```

Η λογική στο πως ο Server διαχειρίζεται τα sockets παρουσιάζεται στο αρχείο */core/socketLogic.js*.

/core/socketLogic.js

```
function socketLogic(app, port) {
    // Socket IO init
    const server = app.listen(port, () => { console.log(`Server listen on
port ${port}`) }); // Listening
    const io = require('socket.io').listen(server);

    // Application variables
    const trackUsers = new Set();
    const raspberryPassword = 'superSecretCode';
    const isNotRaspberry = data => (data.password != raspberryPassword) ?
true : false;
    let raspberryConnected = false;
    let temperature = 20;
    let humidity = 70;

    // Make io accessible to router
    app.use((req, res, next) => {
        req.io = io;
        next();
    });

    io.on('connection', socket => {
        //User -> Server
        socket.on('raspberryStatus', data => {
            console.log(`rasp is ${raspberryConnected}`);
            io.emit('raspberryStatus', { connected: raspberryConnected });
        });

        socket.on('weatherData', data => {
            io.emit('weatherData', { temperature, humidity });
        })

        socket.on('setDeviceStatus', data => {
            console.log('2', data);
            socket.broadcast.to('raspberry').emit('setDeviceStatus',
data);
        })

        socket.on('updateImage', data => {
            console.log('Image 2', data);
            socket.broadcast.to('raspberry').emit('updateImage', data);
```

```
    })

    socket.on('setLocation', locationData => {
        trackUsers.add(locationData.username);
        io.emit('userLocation', locationData);
        rules(locationData);
    })

    socket.on('startCapture', data => {
        console.log('video', 2);
        io.emit('startCapture', data);
    })

    socket.on('getCaptureStatus', data => {
        io.emit('getCaptureStatus', data);
    })


    //Raspberry -> Server
    socket.on('welcomeRaspberry', data => {
        if (isNotRaspberry(data)) return;
        socket.name = 'raspberry';
        socket.join('raspberry');
        raspberryConnected = true;
        io.emit('raspberryStatus', { connected: raspberryConnected });
    });

    socket.on('errorMessage', error => {
        socket.name = 'raspberry';
        io.emit('errorMessage', { error });
    });

    socket.on('setWeatherData', data => {
        if (isNotRaspberry(data)) return;
        ({ temperature, humidity } = data);
        io.emit('weatherData', { temperature, humidity });
    })

    socket.on('deviceStatus', data => {
        if (isNotRaspberry(data)) return;
        console.log('4', data);
        const { deviceId, isEnabled } = data;
        io.emit('deviceStatus', { deviceId, isEnabled });
    })
```

```
        socket.on('imageStream', data => {
            console.log('video', 4);
            io.emit('imageStream', data);
        })

        socket.on('captrureStatus', data => {
            console.log('sent status', 2);
            io.emit('captrureStatus', data);
        })

        //Common
        socket.on('disconnect', () => {
            if (socket.name != 'raspberry') return;
            raspberryConnected = false;
            io.emit('raspberryStatus', { connected: raspberryConnected });
        })
    });

    function rules(locationData) {
        if (locationData.distance > 5) {
            socket.broadcast.to('raspberry').emit('setDeviceStatus', {
deviceId: 1, isEnabled: true });
        }

        if (locationData.distance > 5 && temperature < 18) {
            socket.broadcast.to('raspberry').emit('setDeviceStatus', {
deviceId: 2, isEnabled: true });
        }
    }
}

module.exports = socketLogic;
```

Τα αρχεία */core/passport.js* και */core/strategies/local.strategy.js* περιλαμβάνουν κώδικα για την πιστοποίηση του χρήστη.

/core/passport.js

```
const passport = require('passport');

const passCng = function (app) {

    app.use(passport.initialize());
    app.use(passport.session());
    passport.serializeUser((user, done) => { done(null, user) });
    passport.deserializeUser((user, done) => { done(null, user) });
```

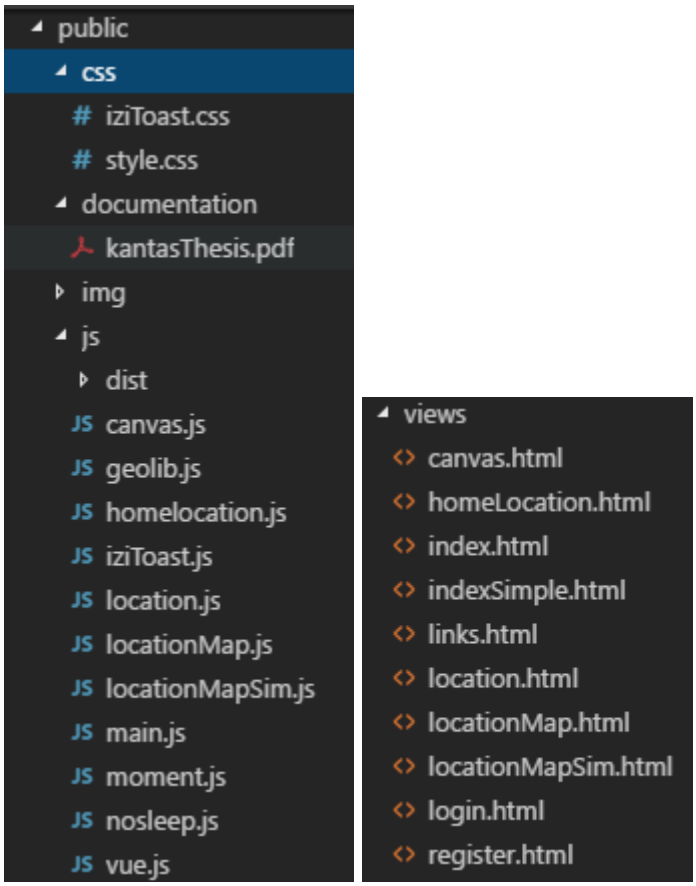
```
    require('./stratergies/local.strategy.js')();
}

module.exports = passCng;
```

/core/strategies/local.strategy.js

```
const passport = require('passport');
const LocalStategy = require('passport-local').Strategy;
const User = require("../../models/user.model.js");
const mongoose = require('mongoose');
const mongoUrl = 'mongodb://localhost/thesis';
const mongoConnect = mongoose.connect(mongoUrl, { useMongoClient: true });
mongoose.Promise = global.Promise;


const userStategy = function () {
    passport.use(
        new LocalStategy({ usernameField: 'username', passwordField:
'password' },
            (username, password, done) => {
                User.findOne({ username, password })
                    .then(user => {
                        if (user) {
                            done(null, user);
                            return;
                        }
                        done(null, false, { message: 'Wrong creadantials'
});
                    })
                    .catch(err => console.log('Error' + err));
            }
        ));
}

module.exports = userStategy;
```

Front-end

Ο κώδικας για το front end περιλαμβάνει CSS και JavaScript αρχεία που βρίσκονται στο public directory και html αρχεία από το views directory. Η δομή των φακέλων αυτών είναι η εξής:



Ο κώδικας για να εμφανιστεί η dashboard του χρήστη περιλαμβάνεται στα index.html και main.js αρχεία ενώ το για το styling χρησιμοποιούνται εκτός από την προεπιλεγμένες css κλάσεις του bulma και custom styles από το αρχείο style.css

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>
        <%= title %>
    </title>
    <meta name="author" content="Alexandros Kantas">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
```

```
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/bulma/0.5.3/css/bulma.css">
    <link rel="stylesheet" href="/public/css/iziToast.css">
    <link rel="stylesheet" href="/public/css/style.css">
</head>

<body>
    <header class="is-primary navbar header">
        <div class="navbar-brand">
            <a class="navbar-item" href="#">
                <h1 class="title ">
                    <%= title %>
                </h1>
            </a>
        </div>
        <div class="navbar-menu">
            <div class="navbar-end">
                <a href="/links" class="navbar-item">
                    <span class="icon">
                        <i class="fa fa-link" aria-hidden="true"></i>
                    </span>
                </a>
                <a href="/logOut" class="navbar-item">
                    <span class="icon">
                        <i class="fa fa-sign-out" aria-hidden="true"></i>
                    </span>
                </a>
            </div>
        </div>
    </header>

    <main id="root" class="container">
        <h4 class="title" v-show="loading">
            <em>Παρακαλώ περιμένετε γίνεται ακόμα φόρτωση της
σελίδας...</em>
        </h4>
        <section class="columns">
            <div class="column is-4 notification is-success">
                <p class="">
                    <strong>
                        <%=username%>
                    </strong>'s dashboard</p>
                </span>
            </div>
```

```
            <div class="column  is-4 is-offset-4 notification"
:class="{'is-danger':!raspberryConnected,'is-
success':raspberryConnected}">
                <p class="has-text-right">Raspberry is
                    <strong v-show="!raspberryConnected">not</strong>
connected!</p>
                <p class="has-text-right littleMargin">
                    <img src="/public/img/pi1.png" alt="raspberry-
connected" class="shadowfilter raspImage" v-if="raspberryConnected">
                    <img src="/public/img/pi3.png" alt="raspberry-not-
connected" class="shadowfilter raspImage" v-else>
                </p>
            </div>
        </section>

        <usersarea></usersarea>

        <section>
            <div class="tile is-ancestor">
                <div class="tile is-parent">
                    <device icon-class="fa-lightbulb-o greenBulp" :device-
id='1' :meters="5" :enabled="false">Συσκευή 1</device>
                </div>

                <div class="tile is-parent">
                    <device icon-class="fa-lightbulb-o redBulp" :device-
id='2' :meters="15" :temp="18" :enabled="true">Συσκευή 2</device>
                </div>
            </div>

            <div class="tile is-ancestor">
                <div class="tile is-parent is-vertical is-4 ">
                    <div class="tile is-child box notification is-danger">
                        <p class="title">Θερμοκρασία</p>
                        <div class="columns">
                            <div class="column centeredElements">
                                <span class="icon">
                                    <i class="fa fa-thermometer-quarter
fa-5x"></i>
                                </span>
                            </div>
                            <div class="column">
                                <p>Η θερμοκρασία είναι</p>
                                <h1 class="title is-1
temperature">{{temperature}}</h1>
```

```
                            <p>βαθμοί κελσίου</p>
                        </div>
                    </div>
                </div>

                <video-area css-classes="image is-128x128 sideCam"
image-url="/public/img/webcamera.jpg" cam-id="2">Εικόνα του χώρου απο την
πλευρική κάμερα όπως καταγράφτηκε </video-area>

                <div class="tile is-child box notification is-danger">
                    <p class="title">Υγρασία</p>
                    <div class="columns">
                        <div class="column centeredElements">
                            <span class="icon">
                                <i class="fa fa-tint fa-5x"></i>
                            </span>
                        </div>
                        <div class="column">
                            <p>Η υγρασία στο χώρο είναι
                                <strong>{{humidity}}</strong> %</p>
                        </div>
                    </div>
                </div>

            </div>

            <div class="tile is-parent">
                <video-area css-classes="image is-3by2 topMargin"
image-url="/public/img/camera.jpg" cam-id="1">Εικόνα του χώρου όπως
καταγράφτηκε </video-area>
            </div>
        </div>

    </section>

</main>
<script src="/socket.io/socket.io.js"></script>
<script src="/public/js/moment.js"></script>
<script src="/public/js/iziToast.js"></script>
<script>
    var socket = io('<%=url%>');
    moment.locale('el');
</script>
<script src="/public/js/vue.js"></script>
```

```
    <script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBLEzuHdS3Y6eLXyXRLG
IGIoVDBoYMVjBo"></script>
    <script src="/public/js/main.js"></script>
</body>

</html>
```

Πρόκειται για ejs αρχείο και όπου υπάρχουν τα σύμβολα **<% %>** πρόκειται για server side rendering. Client side rendering γίνεται με τη χρήση του Vue.js και του κώδικα από το main.js αρχείο.

Main.js

```
let map;
let homeMarker;
let userMarker;
window.Event = new Vue();

Vue.component('mapmodal', {
    data() {
        return {
            homelocation: {},
            userlocation: {}
        }
    },
    mounted() {
        initMap()
        Event.$on('reloadMap', data => {
            setTimeout(() => {
                google.maps.event.trigger(map, 'resize');
                if (data && data.location) map.panTo(data.location);
            }, 100);
        })
    },
    template: `<div class="modal is-active">
    <div class="modal-background" @click="$emit('close')"></div>
    <div class="modal-content">
    <div class="modalMap notification is-info">
    <section id="map" class="map">Loading map...</section>
    </div></div>
    <button class="modal-close is-large" aria-label="close"
@click="$emit('close')"></button>
    </div>`
})
```

```
Vue.component('userbox', {
    props: ['username', 'index', 'distance'],
    data() {
        return {
            userDistance: 1000,
        }
    },
    computed: {
        isUserNearby() {
            const distanceLimit = 100;
            return this.userDistance > 0 && this.userDistance <
distanceLimit;
        }
    },
    mounted() {
        this.userDistance = this.distance;
        socket.on('userLocation', (data) => {
            if (data.username != this.username) return;
            const distanceLimit = 100;
            const userJustArrived = data.distance < distanceLimit &&
!(this.userDistance > 0 && this.userDistance < distanceLimit) &&
data.distance < this.userDistance // True if user not home, new distance
less than distanceLimit and smaller tha previous
            if (userJustArrived) {
                let title = `${this.username} just arrived at home!`;
                iziToast.success({
                    title,
                    timeout: 10000
                });
                new Notification(title);
            }
            this.userDistance = data.distance;
            console.log(data.location);
            window.Event.$emit('updateMarker', { username: this.username,
index: this.index, position: { lat: data.location.latitude, lng:
data.location.longitude } });
        });
    },
    template: `<div class="column is-narrow">
    <div :class="{'is-danger':!isUserNearby,'is-success':isUserNearby}"
class="notification">
    <span class="icon littleMargin">
    <i class="fa  fa-map-marker fa-2x"></i>
    </span>
```

```
    {{username}} is <strong>{{userDistance}}</strong> meters away from
home!
    </div></div>`
});

Vue.component('usersarea', {
    data() {
        return {
            users: [],
            mapModal: false,
        }
    },
    methods: {
        showMap(username, i) {
            this.mapModal = true;
            this.currentUser = username;
            userMarker.setPosition(this.users[i].position);
            Event.$emit('reloadMap', { location: this.users[i].position
});
        },
        hideModal() {
            this.mapModal = false;
        }
    },
    mounted() {
        Event.$on('updateMarker', data => {
            this.users[data.index].position = data.position;
            if (this.currentUser === data.username) {
                userMarker.setPosition(data.position);
                map.panTo(data.position);
            }
            Event.$emit('reloadMap');
        });
        socket.on('userLocation', (data) => {
            console.log('Dater are ' + this.users.findIndex(u =>
u.username === data.username));
            if (this.users.findIndex(u => u.username === data.username)
=== -1) {
                this.users.push({ username: data.username, distance:
data.distance, position: { lat: data.location.latitude, lng:
data.location.longitude } });
            }
        })
    },
    template: `<div>
```

```
    <mapmodal v-show="mapModal" @close="hideModal"></mapmodal>
    <section class="columns usersArea">
        <template  v-for="(user,index) of users">
            <userbox :username="user.username" :index="index"
:distance="user.distance"
@click.native="showMap(user.username,index)"></userbox>
        </template>
    </section>
    </div>`
})

Vue.component('device', {
    props: {
        deviceId: { type: Number, required: true },
        iconClass: String,
        meters: { type: Number, default: 0 },
        temp: { type: Number, default: 0 },
        enabled: { type: Boolean, default: false },
    },
    data() {
        return {
            isDeviceEnabled: this.enabled,
            isLoading: false
        }
    },
    computed: {
        statusText() {
            return this.isDeviceEnabled ? 'Συσκευή Ενεργή' : 'Συσκευή
Ανενεργή';
        },
        buttonText() {
            return this.isDeviceEnabled ? 'Απενεργοποίηση' : 'Άμεση
Ενεργοποίηση';
        },
        statusClass() {
            return this.isDeviceEnabled ? 'has-text-success' : 'has-text-
danger';
        }
    },
    methods: {
        changeDeviceStatus() {
            this.isLoading = true;
            socket.emit('setDeviceStatus', { deviceId: this.deviceId,
isEnabled: !this.isDeviceEnabled });
        }
```

```
    },
    created() {
        console.log('1');
        socket.emit('setDeviceStatus', { deviceId: this.deviceId,
justReport: true });
    },
    mounted() {
        socket.on('deviceStatus', (data) => {
            console.log('5', data);
            if (data.deviceId == this.deviceId) {
                this.isDeviceEnabled = data.isEnabled;
                this.isLoading = false;
            }
        });
    },
    template: `<div class="tile is-child box notification is-warning">
    <p class="title"><slot></slot></p>
    <div class="columns">
        <div class="column centeredElements">
            <span class="icon">
            <i class="fa fa-5x" :class="iconClass"></i>
            </span>
            <p class="title has-text-centered"
:class="statusClass">{{statusText}}</p>
        </div>
        <div class="column">
            <p>Η συσκευή ενεργοποιήτε αυτόματα όταν:</p>
            <p v-if="meters > 0">Ο χρήστης είναι
<strong>{{meters}}</strong> μέτρα μακριά από το σπίτι.</p>
            <p v-if="temp > 0">Η θερμοκρασία κάτω από
<strong>{{temp}}</strong> βαθμούς κελσίου</p>
            <p class="topMargin"><button class="button is-large is-success
is-outlined is-inverted" :class="{'is-loading':isLoading}"
@click="changeDeviceStatus">{{buttonText}}</button></p>
        </div>
    </div>
</div>`
});

Vue.component('image-area', {
    props: ['cssClasses', 'imageUrl', 'camId'],
    data() {
        return {
            dateStamp: 0,
            isLoading: false
```

```
        }
    },
    template: `<div class="tile is-child box notification is-info">
    <p class="title">Εικόνα</p>
    <p><slot></slot><strong>{{imageDate}}</strong></p>
    <figure :class="cssObj">
        <img :src="getImageURL">
    </figure>
    <p class="topMargin has-text-centered"><button class="button is-large
is-warning is-outlined is-inverted" :class="{'is-loading':isLoading}"
@click="updateImage">Ανανέωση εικόνας</button></p>
</div>`,
    mounted() {
        console.log(this.cssClasses, this.imageUrl, this.camId);
        socket.on('newImage', (data) => {
            console.log('Image 5', data);
            if (data.camId == this.camId) {
                this.dateStamp = parseInt(data.dateStamp);
                this.isLoading = false;
            }
        });
    },
    methods: {
        updateImage() {
            console.log('Image 1');
            this.isLoading = true;
            socket.emit('updateImage', { camId: this.camId });
        }
    },
    computed: {
        cssObj() {
            return this.cssClasses.split(' ').reduce((obj, val) =>
(obj[val] = true, obj), {});
        },
        getImageURL() {
            return `${this.imageUrl}?d=${this.dateStamp}`;
        },
        imageDate() {
            return moment(this.dateStamp).calendar();
        }
    }
});

Vue.component('video-area', {
    props: ['cssClasses', 'imageUrl', 'camId'],
```

```
    data() {
        return {
            dateStamp: 0,
            activeCapture: false,
            image: ''
        }
    },
    template: `<div class="tile is-child box notification is-info">
    <p class="title">Εικόνα</p>
    <p><slot></slot><strong>{{imageDate}}</strong></p>
    <figure :class="cssObj">
        <img :src="image">
    </figure>
    <p class="topMargin has-text-centered"><button class="button is-large
is-warning is-outlined is-inverted" @click="startCapture"
:disabled="activeCapture">{{buttonText}}</button></p>
</div>`,
    mounted() {
        this.image = this.imageUrl;
        console.log(this.cssClasses, this.imageUrl, this.camId);
        socket.emit('getCaptureStatus',{camId:this.camId})
        socket.on('imageStream', (data) => {
            console.log('video',5);
            if (data.camId == this.camId) {
                this.dateStamp = parseInt(data.dateStamp);
                this.image=data.image;
            }
        });
        socket.on('captrureStatus',data => {
            if (data.camId != this.camId) return;
            this.activeCapture = data.captureStatus
        });
    },
    methods: {
        startCapture() {
            console.log('video',1);
            this.activeCapture = true;
            socket.emit('startCapture', { camId: this.camId });
        }
    },
    computed: {
        cssObj() {
            return this.cssClasses.split(' ').reduce((obj, val) =>
(obj[val] = true, obj), {});
        },
```

```
        getImageURL() {
            return `${this.imageUrl}?d=${this.dateStamp}`;
        },
        imageDate() {
            return this.dateStamp > 0 ? moment(this.dateStamp).calendar()
: '';
        },
        buttonText(){
            return this.activeCapture ? 'Ζωντανή Εικόνα' : 'Έναρξη Λήψης'
;
        }
    }
});

function askReport() {
    socket.emit('raspberryStatus');
    socket.emit('weatherData');
}

function listen() {

    socket.on('raspberryStatus', data => {
        console.log("Ok the status!!", data);
        this.raspberryConnected = data.connected;
    })

    socket.on('weatherData', data => {
        console.log("Ok the weather!!", data);
        this.temperature = data.temperature;
        this.humidity = data.humidity;
    })

    socket.on('errorMessage', err => {
        console.log(err);
    });
}

function initMap() {
    let position = { lat: 35.32098178540996, lng: 25.10274052619934 };
    let icon = '/public/img/house-icon.png';

    map = new google.maps.Map(document.getElementById('map'), {
        center: position,
        zoom: 14
    });
```

```
    homeMarker = new google.maps.Marker({
        map,
        position,
        icon
    });

    userMarker = new google.maps.Marker({
        map,
        title: 'Your position'
    });

    console.log('Map i showing!!');
}

const app = new Vue({
    el: '#root',
    data: {
        raspberryConnected: false,
        temperature: 0,
        humidity: 0,
        currentUser: '',
        loading: true
    },
    created() {
        listen.call(this);
    },
    mounted() {
        askReport();
        this.loading = false;

        if (window.Notification && Notification.permission !== "denied" &&
Notification.permission !== "granted") {
            Notification.requestPermission().then(response => {
                if (response === 'denied') {
                    iziToast.warning({
                        title: 'Notifications',
                        message: `It's Ok you still can watch these
notifications and can enable the browser's web notifications later by
click the page options left to addrees bar`,
                        timeout: 15000,
                        position: 'topLeft'
                    });
                }
            })}}});
```

Raspberry Pi

Ο κώδικας για το Raspberry Pi περιγράφεται στο παρακάτω αρχείο

raspberryPi.js

```
// Server URL
const url = 'https://thesis.kantas.net';
// Auth Data
const auth = {
    'user': 'alex',
    'pass': 'alexinio22'
}
// Socket IO
const socket = require('socket.io-client')(url);
// Request
const request = require('request');
const fs = require('fs');
// WebCam
const NodeWebcam = require("node-webcam");
const Webcam = NodeWebcam.create({ width: 256, height: 256, output:
"jpeg", callbackReturn: 'base64' });
// Raspicam
const RaspiCam = require("raspicam");
const raspiCamOptinos = {
    mode: 'timelapse',
    timeout: 120000,
    timelapse: 1000,
    output: __dirname + '/photo.jpg',
    vf: true,
    w: 600,
    h: 400
}
const camera = new RaspiCam(raspiCamOptinos);
const raspiCamStaticOptios = { mode: 'photo', output: __dirname +
'/cam.jpg', vf: true, w: 720, h: 480 }
const scamera = new RaspiCam(raspiCamStaticOptios);
// Gpio
const gpio = require('rpi-gpio');
// Temperature - Humidity Sensor
const sensor = require('node-dht-sensor');
//Base64 Img
const base64Img = require('base64-img');

// Application variables
const sensorPin = 17;
```

```
const password = 'superSecretCode';
const devices = [
    {
        deviceId: 1,
        pin: 13,
        isEnabled: true,
        color: 'red'
    },
    {
        deviceId: 2,
        pin: 12,
        isEnabled: true,
        color: 'green'
    }
]

//Set up pins for write
devices.forEach(device => {
    gpio.setup(device.pin, gpio.DIR_OUT, () => gpio.write(device.pin,
device.isEnabled));
});

const cameras = [
    {
        camId: 1,
        captureActive: false,
        name: 'camera'
    },
    {
        camId: 2,
        captureActive: false,
        name: 'webcamera'
    },
]

console.log('Raspberry running ...');

//Sent Data
socket.emit('welcomeRaspberry', { password });


//Listening
socket.on('setDeviceStatus', data => {
    console.log('3', data);
    const devIndex = devices.findIndex(d => data.deviceId == d.deviceId);
```

```
    if (devIndex === -1) { return socket.emit('errorMessage', { error:
'noDeviceFound' }) };
    if (!data.justReport) devices[devIndex].isEnabled = data.isEnabled;
    const { deviceId, pin, isEnabled } = devices[devIndex];
    gpio.write(pin, isEnabled, error => {
        if (error) { socket.emit('errorMessage', { error }); return; };
        console.log('Value changed!');
        socket.emit('deviceStatus', { password, deviceId, isEnabled });
    })
});

socket.on('updateImage', data => {
    console.log('Image 3', data);
    const camIndex = cameras.findIndex(c => data.camId == c.camId);
    if (camIndex === -1) return;
    if (cameras[camIndex].name === 'camera') {
raspCamShot(cameras[camIndex].camId, cameras[camIndex].name); return };
    if (cameras[camIndex].name === 'webcamera') {
webCamShotStatic(cameras[camIndex].camId, cameras[camIndex].name); return
};
});
// Sent Updated Weather Data
let weatherIvl = setInterval(updateWeatherData, 2500);
socket.on('weatherData', data => {
    clearInterval(weatherIvl);
    weatherIvl = setInterval(updateWeatherData, 1500);
    setTimeout(() => { clearInterval(weatherIvl) }, 30000)
})
// Enable RaspiCam
camera.on("read", function (err, dateStamp, file) {
    console.log('video', 3.52);
    base64Img.base64(file, function (err, image) {
        socket.emit('imageStream', { camId: 1, dateStamp, image });
    });
});
// Functions
function updateWeatherData() {
    sensor.read(22, sensorPin, function (error, temperature, humidity) {
        if (error) { socket.emit('errorMessage', { error }); return; };
        socket.emit('setWeatherData', { password, temperature:
temperature.toFixed(1), humidity: humidity.toFixed(1) });
    });
}
function webCamShot() {
    console.log('video', 3.52);
```

```
    Webcam.capture("picture", (err, image) => {
        if (err) { return err; }
        socket.emit('imageStream', { camId: 2, dateStamp: Date.now(),
image });
        if (cameras[1].captureActive) setTimeout(webCamShot, 550);
    });
}
socket.on('startCapture', data => {
    console.log('video', 2.5);
    if (cameras[1].camId == data.camId) startWebCam()
    else if (cameras[0].camId == data.camId) startCam();
});
socket.on('getCaptureStatus', data => {
    const i = cameras.findIndex(c => c.camId == data.camId);
    socket.emit('captrureStatus', { camId: data.camId, captureStatus:
cameras[i].captureActive });
});
function startWebCam() {
    console.log('video', 3.2);
    if (cameras[1].captureActive) return;
    cameras[1].captureActive = true;
    webCamShot();
    setTimeout(() => {
        cameras[1].captureActive = false;
        console.log('sent status', 1);
        socket.emit('captrureStatus', { camId: 2, captureStatus: false })
    }, 60000);
}
function startCam() {
    console.log('video', 3.1);
    if (cameras[0].captureActive) return;
    cameras[0].captureActive = true;
    camera.start();
    setTimeout(() => {
        camera.stop();
        cameras[0].captureActive = false;
        console.log('sent status', 1);
        socket.emit('captrureStatus', { camId: 1, captureStatus: false })
    }, 60000);
}
// Enable static RaspiCam
scamera.on("read", sentCamData);
//Static Img functions
function webCamShotStatic(camId, camName) {
    Webcam.capture("webcam", (error, data) => {
```

```
        if (error) { socket.emit('errorMessage', { error }); return }
        const formData = { camimage: fs.createReadStream(__dirname +
'/webcam.jpg'), dateStamp: Date.now(), camId, camName };
        request.post({ url: `${url}/upLoadImage/camera`, formData, auth },
(error, httpRes, body) => {
            if (error) {
                socket.emit('errorMessage', { error });
                console.error('upload failed:', error);
                return;
            }
            console.log('Upload successful!  Server responded with:',
body);
        });
    });
}
function raspCamShot(camId, camName) {
    sentCamData.camId = camId;
    sentCamData.camName = camName;
    scamera.start();
}
function sentCamData(error, dateStamp, filename) {
    if (error) { socket.emit('errorMessage', { error }); return }
    const { camId, camName } = sentCamData;
    const formData = { camimage: fs.createReadStream(__dirname +
'/cam.jpg'), dateStamp, camId, camName };
    console.log(formData);
    request.post({ url: `${url}/upLoadImage/camera`, formData, auth },
(error, httpRes, body) => {
        if (error) {
            socket.emit('errorMessage', { error });
            console.error('upload failed:', error);
            return;
        }
        console.log('Upload successful!  Server responded with:', body);
    });
}
```