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

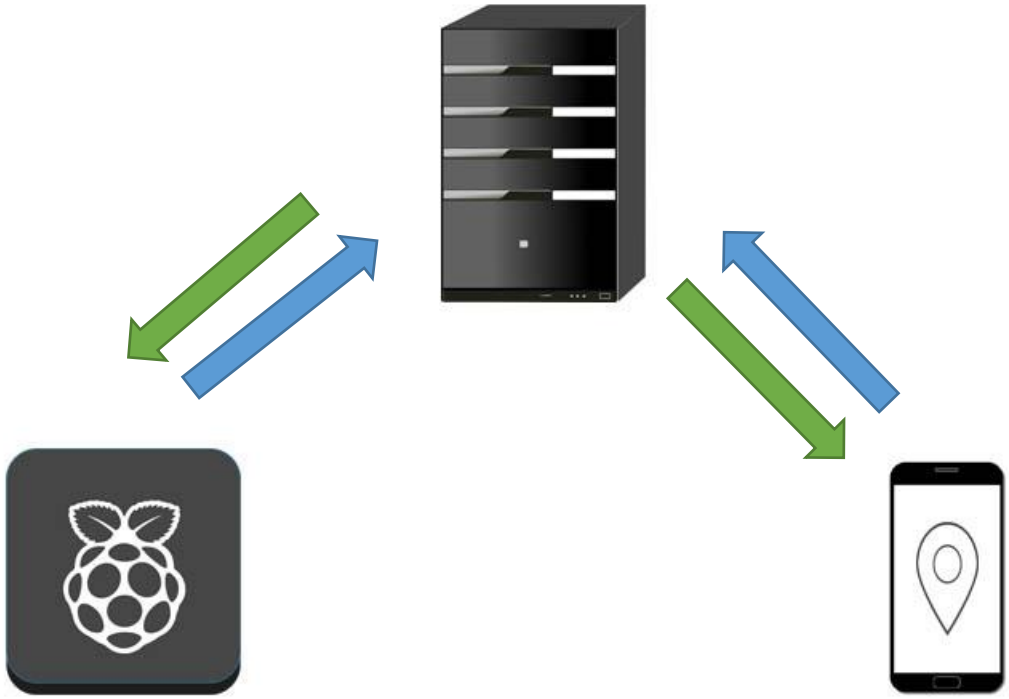
--Work in progress—

--Draft not final version--

Εισαγωγή

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.

Κίνητρο για την διεξαγωγή 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.

Εικόνα 2 Γραφική αναπαράσταση του Internet of Things

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, όπως *NodeJS* και το *front-end framework VueJS,* και την χρήση του μικρο-υπολογιστή Raspberry Pi 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 και στόχοι της εργασίας

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 ένα NodeJS το οποίο συμπεριφέρεται σαν client και αναλαμβάνει να στέλνει και να λάβει πληροφορίες από και προς το Server. Ο Server είναι 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 οποία 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.

Δομή εργασίας

Αυτή η εργασία χωρίστηκε σε τμήματα στοχεύοντας στην ευκολότερη κατανόηση και εκτίμηση της. Έχει δομηθεί σε τρία μέρη, 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 μέρος γίνεται αναφορά στις τεχνολογίες, πρότυπα, γλώσσες προγραμματισμού, προγραμματιστικές διεπαφές και πλαίσια εφαρμογών που χρησιμοποιήθηκαν για την υλοποίηση της πτυχιακής. Για κάθε τεχνολογία παραθέτετέ μια περιγραφή της και ενδεχόμενος κάποιες εγκυκλοπαιδικές πληροφορίες σχετικά με την τεχνολογία αυτή. Στόχος είναι να προσφέρουν στον αναγνώστη πληροφορίες για την κάθε τεχνολογία οι οποίες βοηθούν στην κατανόηση για το πως υλοποιήθηκε το σύστημα.

Στο τρίτο μέρος γίνεται αναλυτική παρουσίαση στο πως δουλεύει το σύστημα εξηγώντας παράλληλα πως έχει υλοποιηθεί. Το σύστημα αποτελείται από πολλά επιμέρους κομμάτια. Για κάθε κομμάτι του συστήματος γίνεται αναφορά στο τι κάνει και πως ακριβώς υλοποιήθηκε αυτή η λειτουργία. Η αναφορά αυτή γίνεται με την παρουσίαση σχεδιαγραμμάτων, στιγμιότυπων οθόνης, φωτογραφιών και με αποσπάσματα από τον κώδικα της εφαρμογής.

Τεχνολογίες και Βασικές Έννοιες

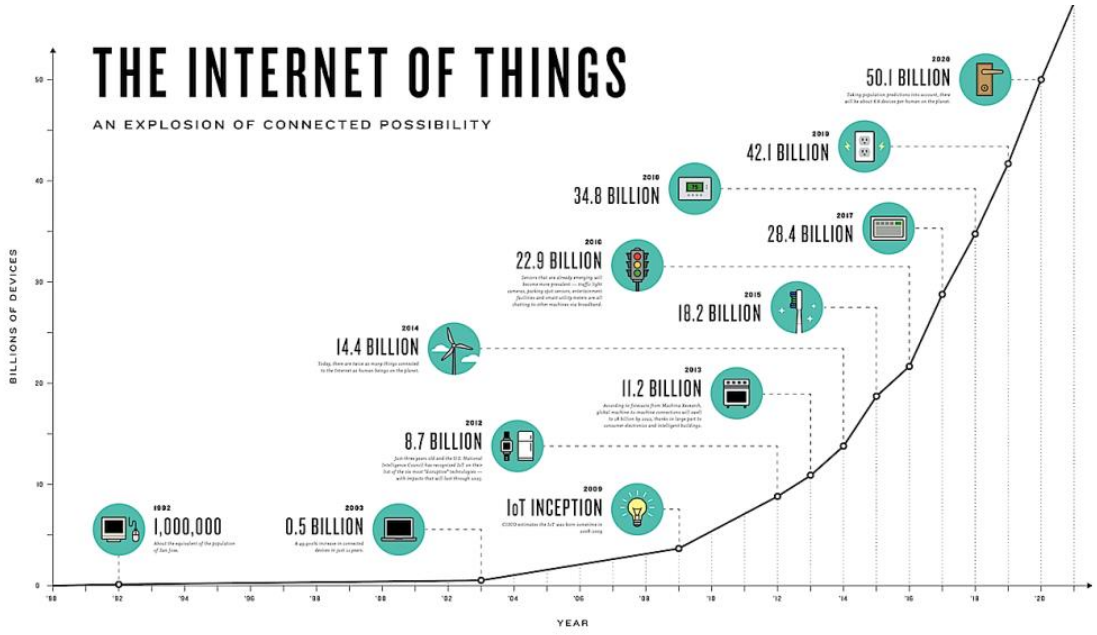
Διαδίκτυο των πραγμάτων

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.



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.

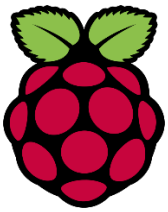
Η εταιρεία ερευνών και συμβούλων Machina διεξήγαγε πρόσφατα μια έρευνα, σύμφωνα με την οποία το 2020 εφαρμογές όπως τα διόδια και η ελεγχόμενη κυκλοφορία θα αντιπροσωπεύουν μια αγορά αξίας $100 δισ., ενώ άλλα $30 δισ. υπολογίζεται ότι θα είναι η αγορά διαχείρισης «έξυπνων» παρκομέτρων που θα «εκπέμπουν»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

Raspberry Pi



Εικόνα 4 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 5 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. 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 προϊών είναι πολύ πιο ικανό από το αρχικό πρότυπο κάνοντας το να ξεπερνάει το όνομα του.

Τεχνολογίες Παγκόσμιου Ιστού

HTML

Εικόνα 6 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.

CSS

Εικόνα 7 Λογότυπο CSS

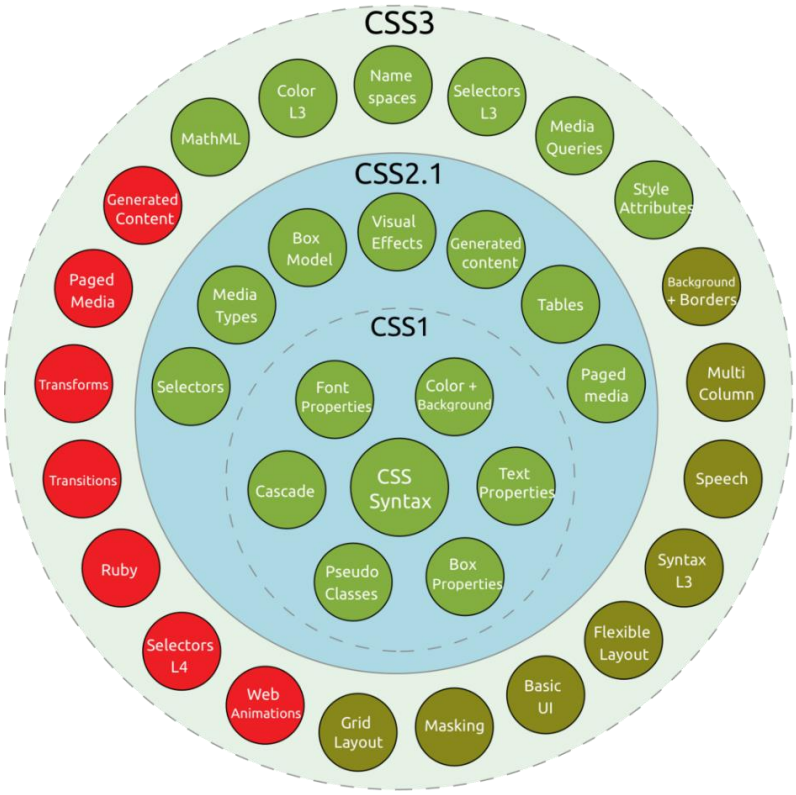
Τα Διαδοχικά Φύλλα Στυλ (CSS, Cascading Style Sheets) είναι μια γλώσσα στυλ (style language) που ορίζει τη διάταξη (layout) των HTML εγγράφων. Για παράδειγμα, τα CSS έχουν να κάνουν με γραμματοσειρές (fonts), με χρώματα (colours), με περιθώρια (margins), με εικόνες φόντου (background images) και με πολλά 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 8 Δείγμα CSS αρχείου

Το CSS έχει διάφορα επίπεδα και προφίλ. Κάθε επίπεδο CSS βασίζεται στο τελευταίο, τυπικά προσθέτοντας νέα χαρακτηριστικά και συνήθως χαρακτηρίζεται ως CSS 1, CSS 2, CSS 3 και CSS 4. Τα προφίλ είναι συνήθως ένα υποσύνολο ενός ή περισσοτέρων επιπέδων CSS που έχουν δημιουργηθεί για μια συγκεκριμένη συσκευή ή περιβάλλον χρήστη .

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 9 Ταξινόμηση και κατάσταση των CSS3 Modules

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.

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.

Document Object Model

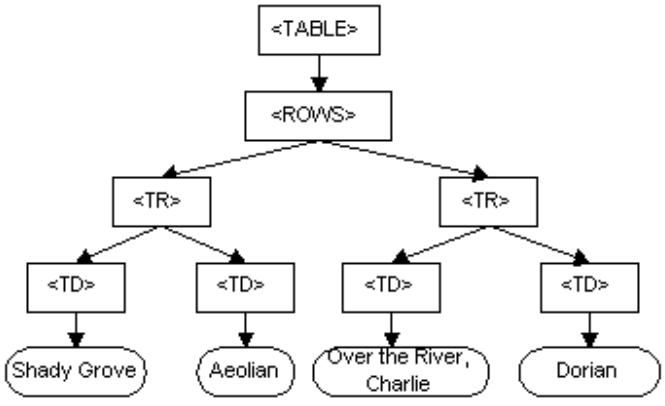
Το Μοντέλο Αντικειμένων Εγγράφων (DOM) είναι μια διεπαφή προγραμματισμού για έγγραφα HTML και XML. Αντιπροσωπεύει τη σελίδα έτσι ώστε τα προγράμματα να μπορούν να αλλάξουν τη δομή, το ύφος και το περιεχόμενο του εγγράφου. Το DOM αντιπροσωπεύει το έγγραφο ως κόμβους και αντικείμενα. Με αυτόν τον τρόπο, 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.

Το μοντέλο αντικειμένου εγγράφου είναι ένα API προγραμματισμού για έγγραφα. Το ίδιο το μοντέλο αντικειμένου μοιάζει πολύ με τη δομή των εγγράφων που μοντελοποιεί. 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>
```

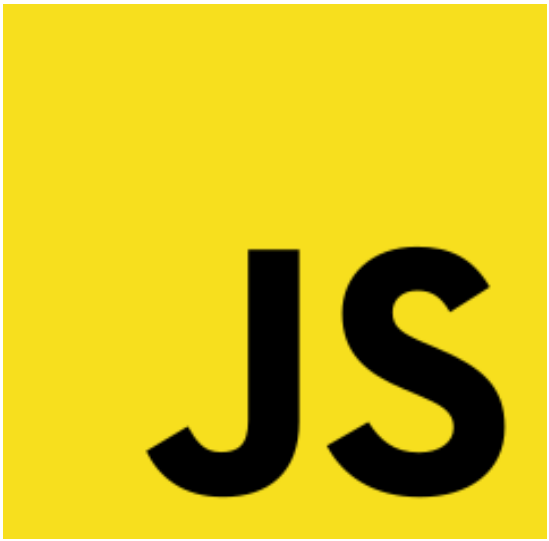
Πίνακας 2 Παράδειγμα html πίνακα

Το DOM αντιπροσωπεύει αυτόν τον πίνακα ως εξής:



Εικόνα 10 DOM αναπαράσταση του html πίνακα

JavaScript / ECMAScript



Εικόνα 11 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());
```

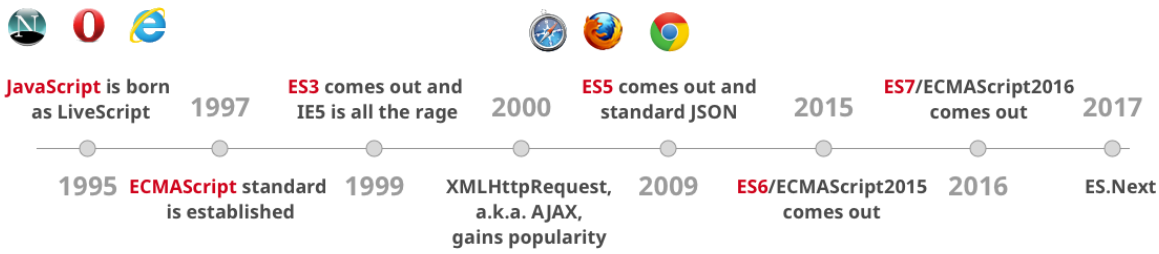
Εικόνα 12 Παράδειγμα κώδικα JavaScript κάνοντας χρήση λειτουργιών και συντακτικού της ECMAScript 2015

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.

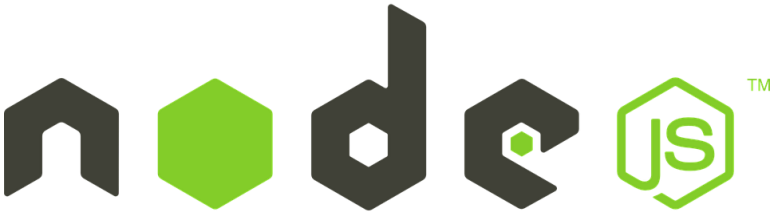


Εικόνα 13 Η ιστορία της JavaScript

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.

NodeJS



Εικόνα 14 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).

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.

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

VueJS

Εικόνα 15 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

Google Maps API

Εικόνα 16 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.

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.

MongoDB

Εικόνα 17 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.

Nginx

Εικόνα 18 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.

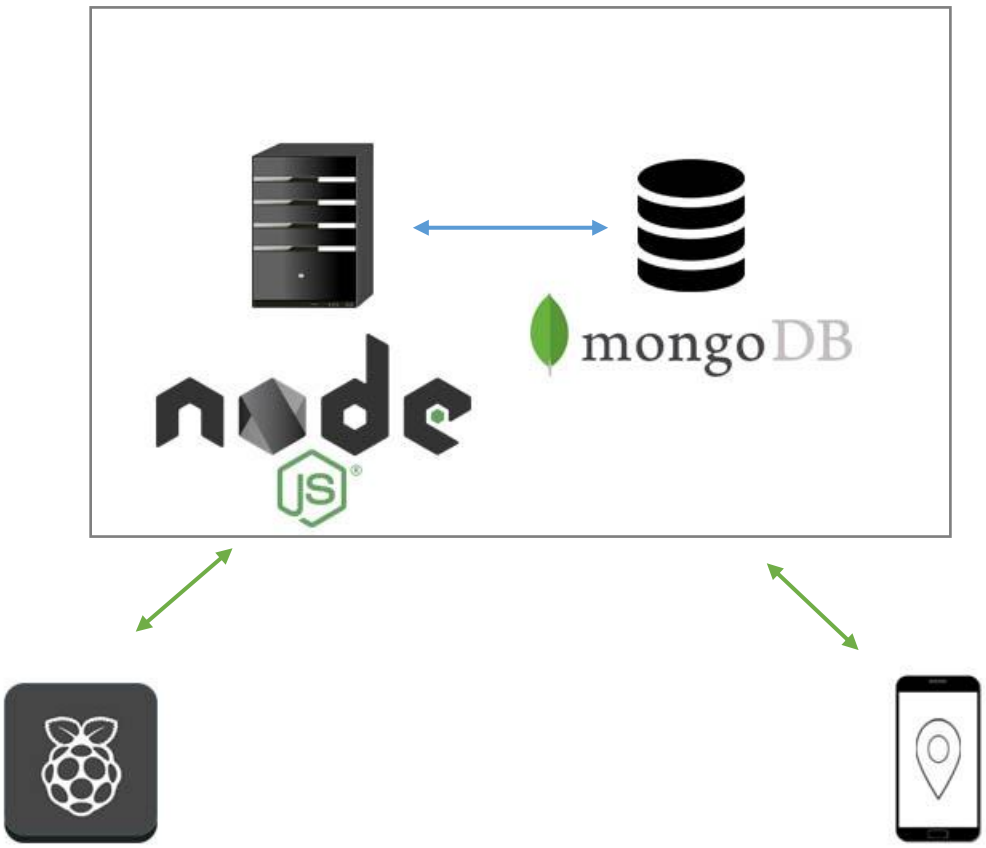
Ο Nginx χρησιμοποιεί μια ασύγχρονη event-driven προσέγγιση για το χειρισμό των αιτημάτων. Η αρθρωτή event-driven αρχιτεκτονική του Nginx μπορεί να προσφέρει πιο προβλέψιμη απόδοση κάτω από υψηλά φορτία.

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.

Δομή και τεχνολογίες του συστήματος

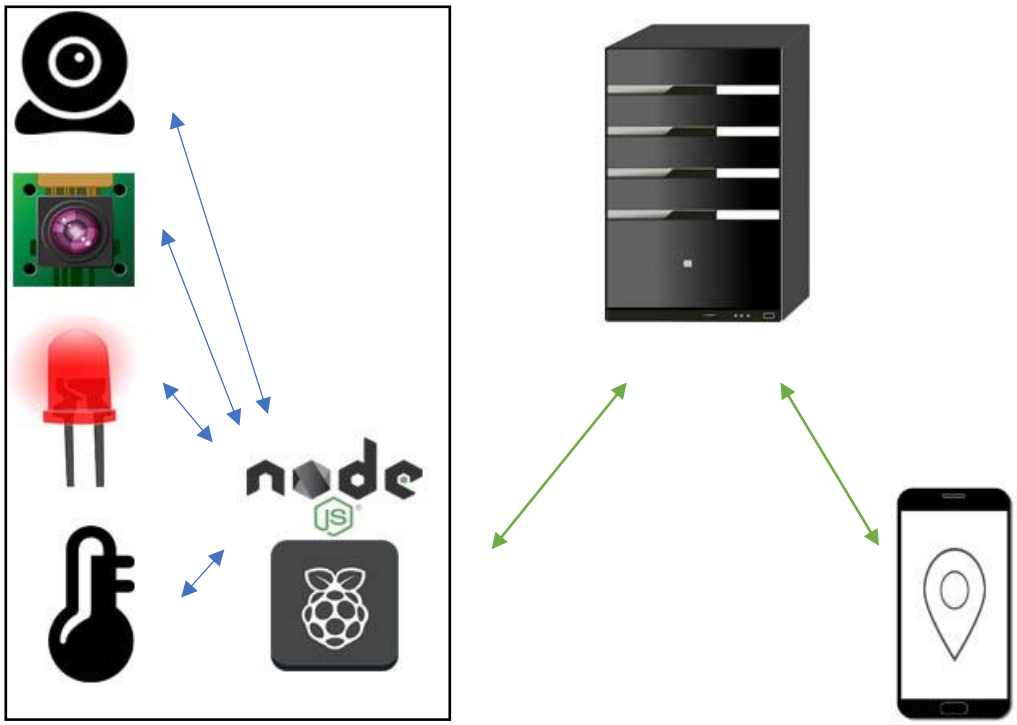
Server



Εικόνα 19 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.

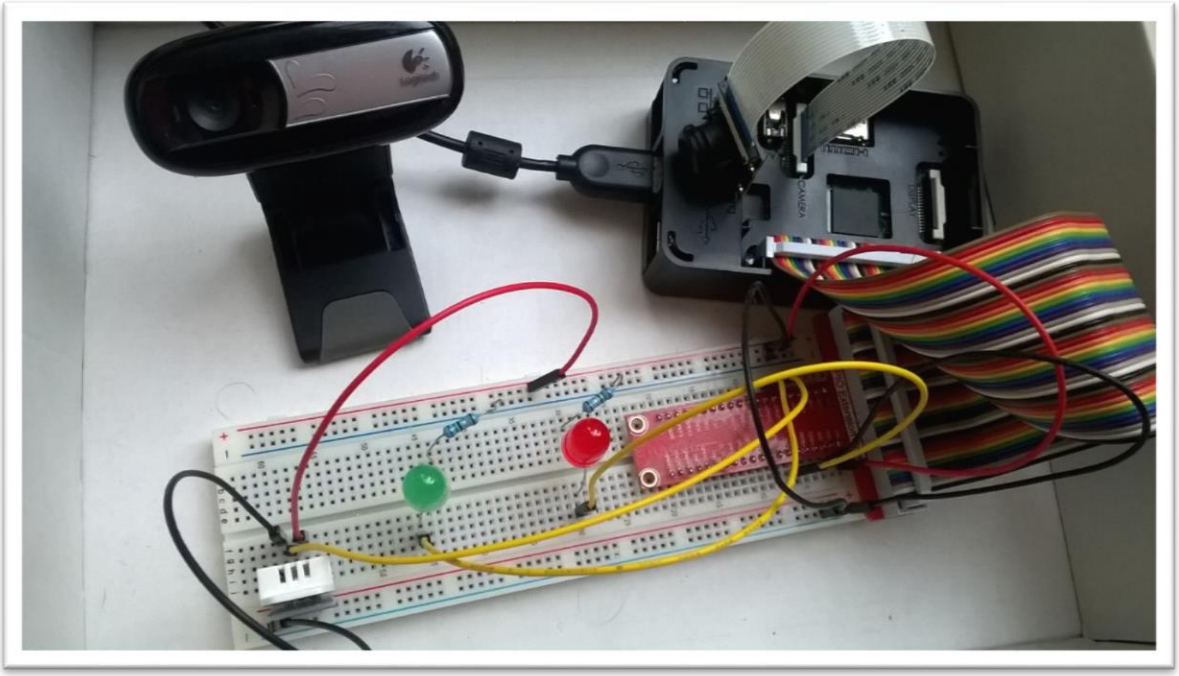
Raspberry Pi



Εικόνα 20 Εικόνα 19 Σχεδιάγραμμα του συστήματος με έμφαση στο κομμάτι του 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.

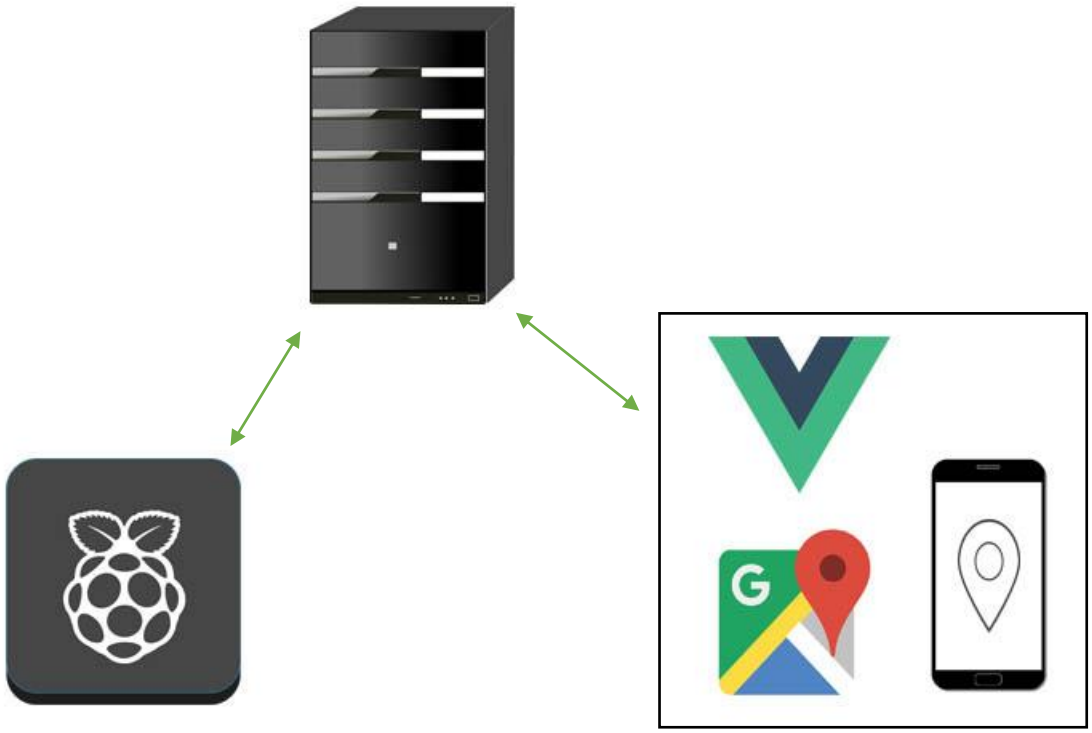
Παρακάτω μια φωτογραφία του Raspberry Pi με όλα τα υλικά συνδεδεμένα:



Εικόνα 21 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.

Web εφαρμογή χρήστη

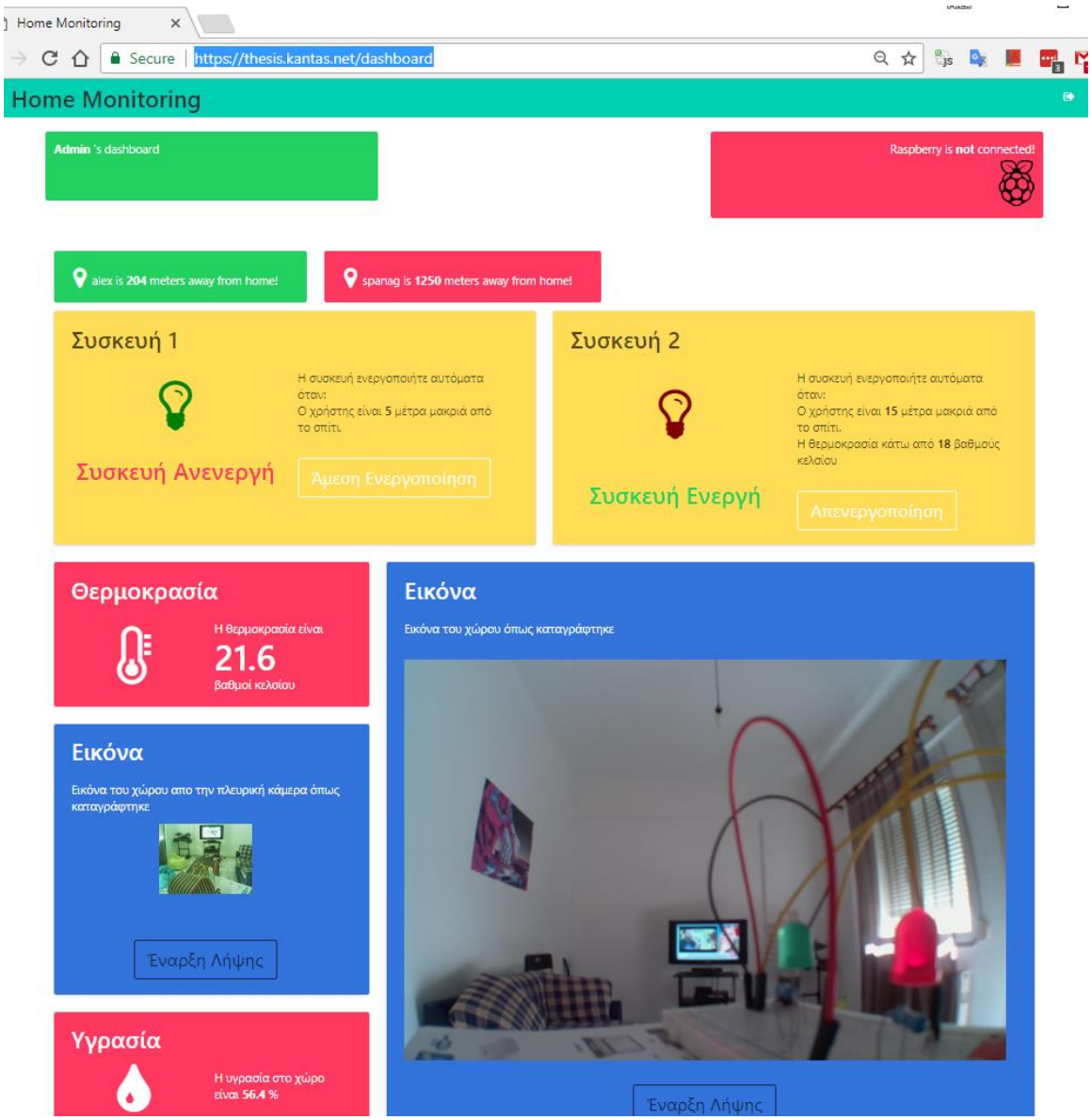


Εικόνα 22 Σχεδιάγραμμα του συστήματος με έμφαση στις τεχνολογίες της Web εφαρμογής του χρήστη

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.

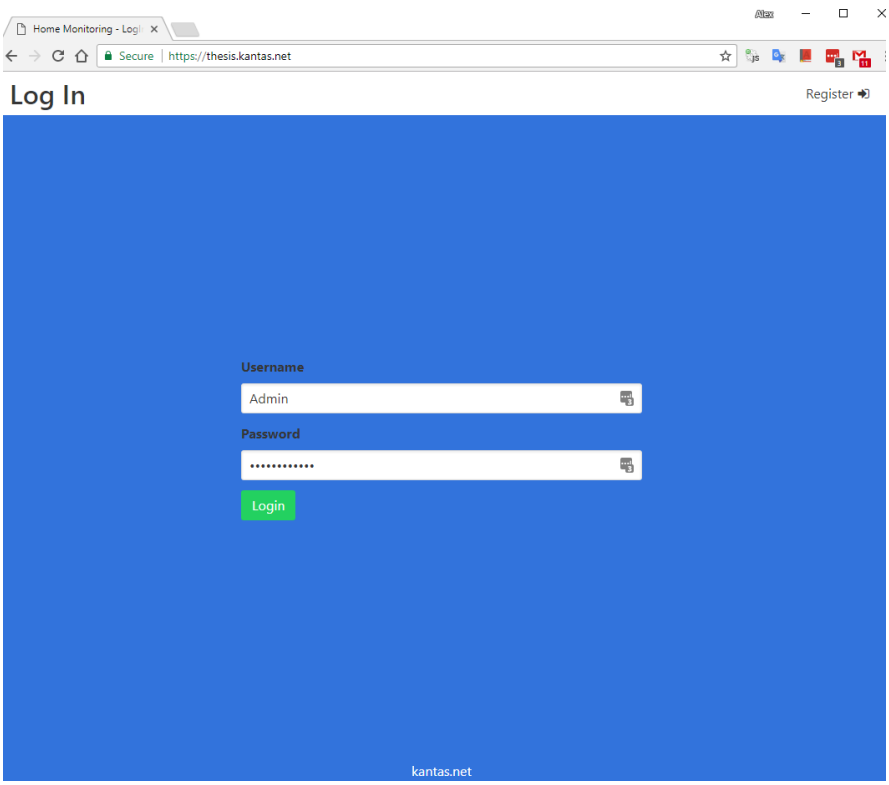
Παρακάτω ένα απόσπασμα οθόνης από την dashboard του χρήστη:



Εικόνα 23 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.

Πιστοποίηση χρηστών



Εικόνα 24 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 3 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 4 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 5 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">
```

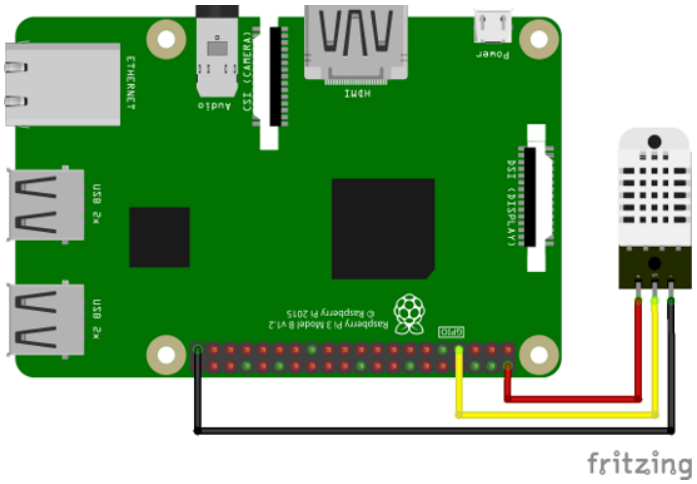
Εκεί στο router του express.js κάνουμε χρήση του passport.js και καλούμε τη μέθοδο authenticate περνώντας σαν πρώτο όρισμα το όνομα της στρατηγικής που θέλουμε να χρησιμοποιήσουμε. Στη προκειμένη περίπτωση την local.

```
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 6 Κώδικας για χρήση της local strategy όταν ο χρήστης υποβάλει τα στοιχειά

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 25 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 7 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 8 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 26 Απόσπασμα οθόνης από την Dashboard στην web εφαρμογή του χρήστη 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 9 Html κώδικας για την εμφάνιση της υγρασίας στο χρήστη

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

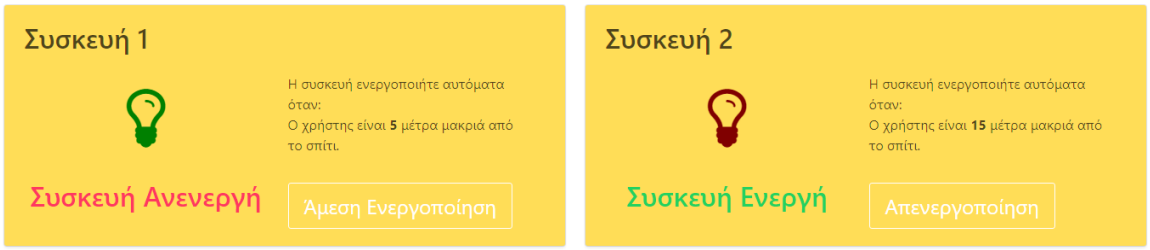
```
const app = new Vue({
    el: '#root',
    data: {
        temperature: 0,
        humidity: 0,
    },
    created() {
        listen.call(this);
    }
});

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

Πίνακας 10 Κώδικας για την ενημέρωση των τιμών της θερμοκρασίας και υγρασίας στην dashboard του χρήστη

Διαχείριση της κατάστασης των LED συσκευών

Η κατάσταση των συσκευών εμφανίζεται στη Dashboard του χρήστη ως εξής



Εικόνα 27 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>
```

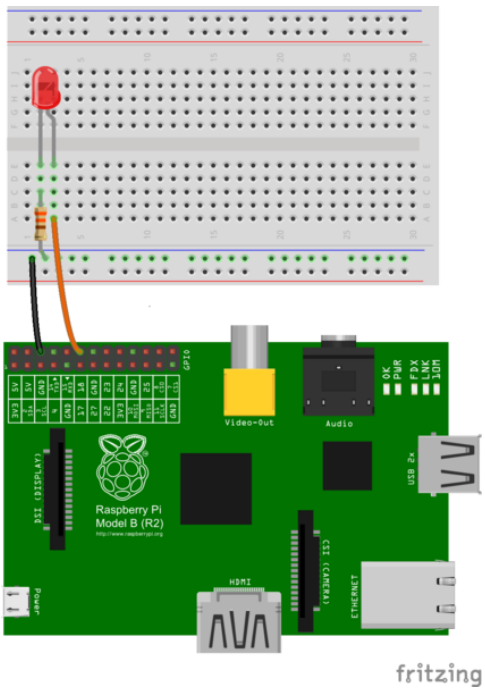
Το οποίο είναι σε θέση να αλλάξει οπτικά στο DOM την κατάσταση της συσκευής με βάσει τα δεδομένα που λαμβάνει από τον Server χάρη στον παρακάτω κώδικα

```
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 11 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 28 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

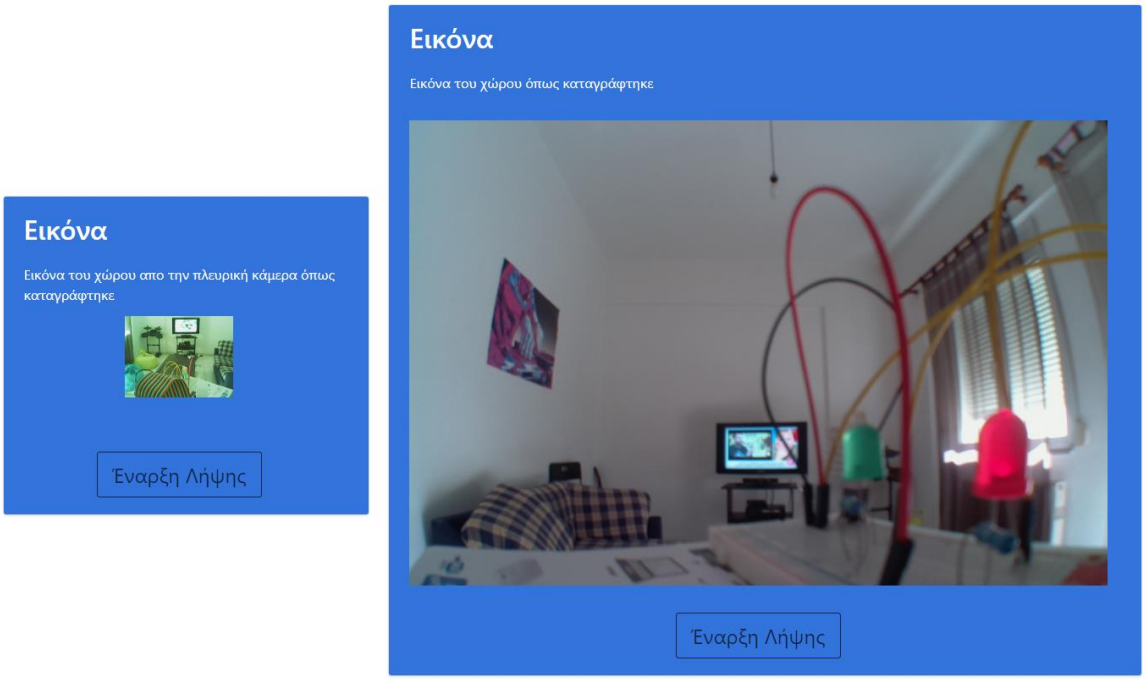
```
const gpio = require('rpi-gpio');
const devices = [
    {   deviceId: 1,
        pin: 13,
        isEnabled: true,
        color: 'red'},
    {   deviceId: 2,
        pin: 12,
        isEnabled: true,
        color: 'green'}]
//Set up pins for write
devices.forEach(device => {
    gpio.setup(device.pin, 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 12 Κώδικας για τροποποίηση των GPIO συσκευών με βάσει τα μηνύματα από το Server

Λήψη εικόνας χώρου από τις κάμερες στο Raspberry Pi

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



Πίνακας 13 Απόσπασμα Οθόνης από την Dashboard του χρήστη με την εικόνα απο 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 14 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 29 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 30 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 15 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 16 Κώδικας για stream της εικόνας από την κάμερα του Raspberry Pi