ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εφαρμογή Web για αμφίδρομη επικοινωνία με σύστημα ασφαλείας

του

Παπαχρήστου Γεώργιου

Επιβλέπων: Στεφανιδάκης Μιχαήλ
Επίκουρος Καθηγητής

Κέρκυρα, Ιούλιος, 2019

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εφαρμογή Web για αμφίδρομη επικοινωνία με σύστημα ασφαλείας

του

Παπαχρήστου Γεώργιου

Επιβλέπων: Στεφανιδάκης Μιχαήλ
Επίκουρος Καθηγητής

Κέρκυρα, Ιούλιος, 2019

Εφαρμογή Web για αμφίδρομη επικοινωνία με σύστημα ασφαλείας

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Παπαχρήστος Γεώργιος

Υπογραφή

Πρόλογος

Το πόνημα τούτο αποτελεί τη πτυχιακή εργασία του Παπαχρήστου Γεώργιου στο Τμήμα Πληροφορικής του Ιόνιου Πανεπιστημίου.

Θα ήθελα να εκφράσω την ευγνωμοσύνη μου προς το οικείο περιβάλλον μου και κυρίως στην οικογένεια μου για την συμπαράσταση τους.

Κέρκυρα 2 - 7 - 2019

Παπαχρήστος Γεώργιος

Περίληψη

Στη εργασία αυτή έχει υλοποιηθεί ένα σύστημα αμφίδρομης επικοινωνίας μεταξύ διακομιστή και πελάτη μέσω χρήσης της τεχνολογίας Socket.io που αποτελεί βιβλιοθήκη για υλοποίηση WebSockets με Node.js. Υλοποιήθηκε αμφίδρομη επικοινωνία, μεταξύ ενός εξουσιοδοτημένου χρήστη που επικοινωνεί μέσω του περιηγητή ιστού (browser) του με τον διακομιστή και του Raspberry Pi που επικοινωνεί με εξουσιοδότηση με τον ίδιο διακομιστή. Με αυτό τον τρόπο διασφαλίζεται ότι το Raspberry μπορεί να τοποθετηθεί οπουδήποτε αρκεί να διαθέτει πρόσβαση στο διαδίκτυο για να επικοινωνήσει με τον server. Μέσω των ακροδεκτών GPIO του Raspberry Pi, όπου έχουν τοποθετηθεί αισθητήρες, ο χρήστης μπορεί να διαβάσει τις τιμές τους σε πραγματικό χρόνο, όπως και να πραγματοποιήσει μια αλλαγή σε μια έξοδο του για να ενεργοποιήσει κάποιο σύστημα ασφαλείας, συναγερμού ή κάτι άλλο.

Abstract

In this work, a two-way communication between server and client has been implemented through the use of Socket.io technology that is a library for WebSockets implemented with Node.js. Two-way communication has been implemented between an authorized user who communicates through its web browser to the server and Raspberry Pi communicating with the same server. This ensures that Raspberry can be placed anywhere at any time as long as it has access to the internet to communicate with the server. Via Raspberry Pi's GPIO ports, the user can check their values in real time as well as make a change to a port and activate a security system or something else.

Περιεχόμενα

Κεφάλαιο 1ο

Εισαγωγή

1.1 **Συνεισφορά της εργασίας**

Ο κύριος στόχος αυτής της εργασίας είναι η κατασκευή ενός συστήματος ασφαλείας που υλοποιείται με Raspberry έχοντας αμφίδρομη επικοινωνία πραγματικού χρόνου με ένα απομακρυσμένο χρήστη μέσω διαδικτύου χρησιμοποιώντας WebSockets.

Συγκεκριμένα, οι στόχοι της εργασίας είναι:

Ο χρήστης να μπορεί να παρακολουθεί τους αισθητήρες του Raspberry μέσω του διαδικτύου από τον browser του.

Ο χρήστης να μπορεί να ενεργοποιήσει ακροδέκτες εξόδου του Raspberry μέσω του διαδικτύου από τον browser του, όπως ρελέ, σειρήνα, led και άλλα.

Οι χρήστες να χρησιμοποιούν την ιστοσελίδα πρόσβασης και αυτοί που τοποθετούν και χρησιμοποιούν το Raspberry είναι εξουσιοδοτημένοι. Το κάθε Raspberry να μπορεί να τοποθετηθεί οπουδήποτε υπάρχει πρόσβαση στο διαδίκτυο, αρκεί να είναι δηλωμένο στο Server στον αντίστοιχο συνδεδεμένο χρήστη. Η αμφίδρομη επικοινωνία πραγματικού χρόνου πραγματοποιήθηκε με την βοήθεια της βιβλιοθήκης Socket.io που χρησιμοποιεί WebSockets υλοποιημένα στο περιβάλλον Node.js.

1.2 **Δομή της εργασίας**

Στο δεύτερο κεφάλαιο της εργασίας παρουσιάζεται μια μικρή εισαγωγή των WebSockets. Στο τρίτο κεφάλαιο περιγράφεται το περιβάλλον Node.js και με ποιο τρόπο μπορούν να υλοποιηθούν τα WebSockets σε αυτό το περιβάλλον χρησιμοποιώντας τη βιβλιοθήκη Socket.io. Στο τέταρτο κεφάλαιο αναλύεται το Raspberry Pi 3 και οι ακροδέκτες του. Στο πέμπτο τμήμα περιγράφεται το σύστημα ασφαλείας και οι αισθητήρες που χρησιμοποιούνται και μπορούν να χρησιμοποιηθούν. Στο έκτο και κύριο κεφάλαιο αναλύεται η εφαρμογή Web για αμφίδρομη επικοινωνία με περιγραφή διαγραμμάτων και κώδικα για όλα τα μέρη του συστήματος, Raspberry, πελάτη και διακομιστή.

Στο τέλος παρουσιάζονται τα συμπεράσματα της εργασίας και θέματα για μελλοντική έρευνα.

Κεφάλαιο 2ο

Εισαγωγή στα Web sockets

2.1 **Τι είναι τα Web Sockets**

Το WebSocket [1] είναι ένα πρωτόκολλο επικοινωνίας υπολογιστών το οποίο παρέχει κανάλια πλήρους αμφίδρομης επικοινωνίας μέσω μίας TCP σύνδεσης. Το πρωτόκολλο WebSocket έγινε πρότυπο από τον οργανισμό IETF με τον κωδικό RFC 6455 το 2011 [2]. Σήμερα υποστηρίζεται σχεδόν σε όλους τους κύριους φυλλομετρητές, όπως οι Google Chrome, Microsoft Edge, Internet Explorer, Firefox, Safari και Opera.

Το WebSocket διαφέρει από το HTTP [3]. Και τα δύο πρωτόκολλα βρίσκονται στο 7ο επίπεδο του Μοντέλου Αναφοράς OSI. Παρόλο που είναι διαφορετικά, το RFC 6455 ορίζει ότι το WebSocket είναι σχεδιασμένο να λειτουργεί στις πόρτες 80 και

443 του HTTP αλλά και να υποστηρίζει HTTP proxies. Έτσι είναι συμβατό με το πρωτόκολλο HTTP.

Σε αντίθεση με το HTTP, το WebSocket παρέχει επικοινωνία πλήρους αμφίδρομης επικοινωνίας. Επιπρόσθετα, το WebSocket επιτρέπει μεταφορά μηνυμάτων πάνω από το TCP.

Πριν από το WebSocket, η επικοινωνία πλήρους αμφίδρομης θύρας 80 ήταν εφικτή χρησιμοποιώντας κανάλια Comet[4]. Ωστόσο, η εφαρμογή Comet είναι αναποτελεσματική για μικρά μηνύματα. Το πρωτόκολλο WebSocket στοχεύει στην επίλυση αυτών των προβλημάτων χωρίς να μειώνει τον βαθμό ασφαλείας.

Οι προδιαγραφές του πρωτοκόλλου WebSocket ορίζουν ws (WebSocket) και wss (WebSocket Secure) ως δύο νέα αναγνωριστικά πόρων τα οποία χρησιμοποιούνται για μη κρυπτογραφημένες και κρυπτογραφημένες συνδέσεις αντίστοιχα.

2.1.1 Ιστορικά

Το WebSocket αναφέρθηκε για πρώτη φορά ως TCP επικοινωνία στην HTML5, ως ένα API socket υποδοχής TCP. Στα μέσα του 2008 προέκυψε η πρώτη έκδοση του πρωτοκόλλου γνωστού ως WebSocket.

Το Δεκέμβριο 2009, το Google Chrome 4 ήταν ο πρώτος περιηγητής με πλήρη υποστήριξη του προτύπου, με το WebSocket ενεργοποιημένο από προεπιλογή. Η ανάπτυξη του πρωτοκόλλου

WebSocket έγινε στη συνέχεια από την ομάδα W3C [6] και WHATWG [7] τον Φεβρουάριο 2010.

2.1.2 Σύνδεση WebSocket

Για να δημιουργήσει μια σύνδεση το WebSocket, ο υπολογιστής-πελάτης στέλνει ένα αίτημα χειραψίας WebSocket, για το οποίο ο server-διακομιστής επιστρέφει μια απάντηση χειραψίας WebSocket, όπως φαίνεται στο παρακάτω παράδειγμα, όπως αναφέρεται στην [8]. Παράδειγμα αιτήματος πελάτη (ακριβώς όπως στο HTTP, κάθε γραμμή τελειώνει με \ r \ n και στο τέλος πρέπει να υπάρχει μια επιπλέον κενή γραμμή):

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

Server response:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
```

```
Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=
Sec-WebSocket-Protocol: chat
```

Η χειραψία ξεκινάει με αίτημα / απόκριση HTTP, επιτρέποντας στους Server να χειρίζονται συνδέσεις HTTP καθώς και συνδέσεις WebSocket στην ίδια θύρα. Μόλις δημιουργηθεί η σύνδεση, η επικοινωνία μεταβαίνει σε αμφίδρομο πρωτόκολλο το οποίο δεν συμφωνεί με το πρωτόκολλο HTTP.

Εκτός από τις κεφαλίδες Upgrade, ο υπολογιστής-πελάτης αποστέλλει μια κεφαλίδα Sec-WebSocket-Key που περιέχει τυχαία ψηφιοποιημένα base64 [9] και ο διακομιστής απαντά με ένα hash του κλειδιού στην κεφαλίδα Sec-WebSocket-Accept.

Μόλις δημιουργηθεί η σύνδεση, ο υπολογιστής-πελάτης και ο διακομιστής-server μπορούν να στέλνουν δεδομένα WebSocket ή πλαίσια κειμένου εμπρός και πίσω σε λειτουργία πλήρους αμφίδρομης λειτουργίας. Τα δεδομένα είναι ελάχιστα πλαισιωμένα, με μια μικρή κεφαλίδα που ακολουθείται από τα δεδομένα μας. Οι μεταδόσεις WebSocket περιγράφονται ως "μηνύματα", όπου ένα μοναδικό μήνυμα μπορεί προαιρετικά να χωριστεί σε διάφορα πλαίσια δεδομένων. Αυτό μπορεί να επιτρέψει την αποστολή μηνυμάτων όπου τα αρχικά δεδομένα είναι διαθέσιμα, αλλά το πλήρες μήκος του μηνύματος είναι άγνωστο (στέλνει ένα πλαίσιο δεδομένων μετά το άλλο μέχρι να

φτάσει το τέλος και να επισημανθεί με το bit FIN). Με τις επεκτάσεις του πρωτοκόλλου, αυτό μπορεί επίσης να χρησιμοποιηθεί για την πολυπλεξία αρκετών ροών-συνδέσεων ταυτόχρονα.

2.2 **Εξυπηρετητές για τα Web Sockets**

Υπάρχουν πολλοί τρόποι να δημιουργήσουμε έναν server που να υποστηρίζει Web Sockets σε διάφορες γλώσσες. Παρακάτω παρουσιάζονται οι σημαντικότερες.

Server με C#

Η C# έχει την κλάση TcpListener που ανήκει στο System.Net.Sockets namespace ώστε να υποστηρίξει και να κάνει εύκολη την χρήση των Websockets.[10]

Server C++

Για την C++ υπάρχει ο διαδεδομένος open source κώδικας Simple-WebSocket-Server

Είναι απλή, γρήγορη, multithreaded, ανεξάρτητη λειτουργικού συστήματος WebSocket (WS) και WebSocket Secure (WSS) server και client βιβλιοθήκη υλοποιημένη με C++ 11. [11]

Server PHP

Για την PHP υπάρχει η βιβλιοθήκη Ratchet η οποία παρέχει στους προγραμματιστές εργαλεία για να δημιουργήσουν αμφίδρομης επικοινωνίας εφαρμογές μεταξύ πελάτη και server με WebSockets. [12]

Server Python

Για την Python υπάρχουν πολλές εφαρμογές και open sources έργα. Ένα από αυτά είναι το Pithikos/python-websocket-server όπου ο προγραμματιστής μπορεί να χρησιμοποιήσει WebSockets χωρίς επιπλέον βιβλιοθήκες.[13]

Server Java

Το JSR 356 ή το Java API για WebSocket, είναι ένα API όπου οι Java προγραμματιστές μπορούν να χρησιμοποιήσουν για ενσωματώσουν WebSockets στις εφαρμογές τους.

Αυτό το Java API παρέχει εργαλεία-components για server και client.[14]

Για Server: μέσα από το πακέτο javax.websocket.server package.

Για Client: μέσα από το πακέτο javax.websocket package.

Server Node.js

Στο επόμενα υπό-κεφάλαια παρουσιάζονται το framework Node.js [15] ως γενικό εργαλείο και υλοποιεί τα web sockets και συγκεκριμένα πως τα υλοποιεί μέσω της τεχνολογίας socket.

Κεφάλαιο 3ο

Node.js Και Socket.io

3.1 **Εισαγωγή στο περιβάλλον Node.js**

Το Node.js [16] είναι ένα περιβάλλον ανοιχτού κώδικα, που εκτελεί κώδικα JavaScript εκτός του προγράμματος περιήγησης. Συνήθως, το JavaScript χρησιμοποιείται κυρίως για scripting από την πλευρά του πελάτη, στο οποίο προγράμματα είναι γραμμένα σε JavaScript και ενσωματώνονται στην HTML της ιστοσελίδας και εκτελούνται από την πλευρά του πελάτη από μια μηχανή JavaScript στον περιηγητή ιστού του χρήστη.

Σχήμα 3.1. Node Logo

Το Node.js επιτρέπει στους προγραμματιστές να χρησιμοποιούν JavaScript για να γράψουν εργαλεία στη γραμμή εντολών και για server-side scripting-running scripts και για να παράγουν δυναμικό περιεχόμενο ιστοσελίδας πριν η σελίδα φορτωθεί στο πρόγραμμα περιήγησης του χρήστη. Συνεπώς, το Node.js αντιπροσωπεύει ένα παράδειγμα "JavaScript παντού", ενοποιώντας την ανάπτυξη web εφαρμογών γύρω από μια ενιαία γλώσσα προγραμματισμού, αντί για διαφορετικές γλώσσες για τις σελίδες διακομιστή και τις σελίδες πελάτη.

- Δημιουργός: Ryan Dahl
- Αρχικό release May 27, 2009
- Σταθερό release 11.2.0 / November 15, 2018
- Repository: github.com/nodejs/node
- Γραμμένο σε: C, C++, JavaScript

- Λειτουργικό Σύστημα:
- Linux, macOS, Microsoft Windows, SmartOS, FreeBSD
- Άδεια: MIT license[4][5]
- Ιστοσελίδα: nodejs.org

Παρόλο που η .js είναι η συμβατική επέκταση αρχείου για κώδικα JavaScript, το όνομα "Node.js" δεν αναφέρεται σε ένα συγκεκριμένο αρχείο και είναι απλώς το όνομα του προϊόντος. Το Node.js έχει μια αρχιτεκτονική που βασίζεται σε events-συμβάντα και είναι ικανή για ασύγχρονα I/O (input-output). Αυτές οι επιλογές σχεδιασμού στοχεύουν στη βελτιστοποίηση της απόδοσης και χρήσης σε εφαρμογές ιστού με πολλές λειτουργίες εισόδου / εξόδου, καθώς και για εφαρμογές Web σε πραγματικό χρόνο (π.χ. προγράμματα επικοινωνίας σε πραγματικό χρόνο και παιχνίδια με προγράμματα περιήγησης.

Το έργο Node.js, το οποίο διανέμεται από το Node.js Foundation, υποστηρίζεται από πρόγραμμα συνεργασίας με το Linux Foundation.

3.1.1 Ιστορία

Το Node.js γράφτηκε αρχικά από τον Ryan Dahl το 2009. Η αρχική έκδοση υποστηρίζει μόνο το Linux και το Mac OS X. Η

ανάπτυξη και η συντήρησή του καθοδηγούνται από τον Dahl και αργότερα από τον Joyent.

Ο Dahl εμπνεύστηκε για να δημιουργήσει το Node.js αφού είδε μια μπάρα προόδου upload file στο Flickr. Το πρόγραμμα περιήγησης δεν γνώριζε πόσο από το αρχείο είχε ανεβεί και έπρεπε να ερωτήσει τον διακομιστή Web. Ο Ντάλ επιθυμούσε έναν ευκολότερο τρόπο. [18]

Ο Dahl επέκρινε τις περιορισμένες δυνατότητες του πιο δημοφιλέστερου διακομιστή ιστού το 2009, του Apache HTTP Server, για να χειριστεί πολλές ταυτόχρονες συνδέσεις (έως και 10.000) και τον πιο συνηθισμένο τρόπο δημιουργίας κώδικα (διαδοχικός προγραμματισμός).

Το Node.js συνδυάζει τη μηχανή JavaScript V8 της Google, έναν βρόχο συμβάντων και ένα χαμηλού επιπέδου I/O API.

Μετά την πρώτη παρουσίαση το 2009, τον Ιανουάριο του 2010 έβαλε έναν διαχειριστή πακέτων για το περιβάλλον Node.js που ονομάζεται npm [20]. Ο διαχειριστής πακέτων διευκολύνει τους προγραμματιστές να δημοσιεύουν και να μοιράζονται τον πηγαίο κώδικα των βιβλιοθηκών Node.js και έχει σχεδιαστεί για να απλοποιεί την εγκατάσταση, την ενημέρωση και την απεγκατάσταση των βιβλιοθηκών.

Η πρώτη έκδοση του Node.js που υποστηρίζει τα Windows κυκλοφόρησε τον Ιούλιο του 2011.

Τον Ιανουάριο του 2012, ο Dahl παραιτήθηκε, προωθώντας τον συνεργάτη και τον δημιουργό του Isaac Schlueter να διαχειριστεί το έργο. Τον Ιανουάριο του 2014, ο Schlueter ανακοίνωσε ότι ο Timothy J. Fontaine θα ηγηθεί του σχεδίου.

Τον Φεβρουάριο του 2015 ανακοινώθηκε η πρόθεση να σχηματιστεί ένα ουδέτερο Node.js Foundation. Μέχρι τον Ιούνιο του 2015, οι κοινότητες Node.js και io.js, μια παραλλαγή του, ψήφισαν να συνεργαστούν στο πλαίσιο του Node.js Foundation.

Τον Σεπτέμβριο του 2015, τα Node.js v0.12 και io.js v3.3 συγχωνεύθηκαν ξανά μαζί στο Node v4.0. Αυτό έφερε τις λειτουργίες V8 ES6 στο Node.js και ένα μακροπρόθεσμο κύκλο έκδοσης υποστήριξης. Από το 2016, ο ιστότοπος io.js συνιστά να επιστρέψουν οι προγραμματιστές στο Node.js και να μην προγραμματιστούν περαιτέρω κυκλοφορίες io.js λόγω της συγχώνευσης.

3.1.2 Αρχιτεκτονική

Το Node.js υποστηρίζει προγραμματισμό που βασίζεται σε γεγονότα-events σε διακομιστές ιστού - servers, επιτρέποντας την ανάπτυξη διακομιστών γρήγορου web με JavaScript. Οι

προγραμματιστές μπορούν να δημιουργήσουν εξαιρετικά κλιμακούμενους διακομιστές χρησιμοποιώντας απλοποιημένο μοντέλο προγραμματισμού που βασίζεται σε συμβάντα και χρησιμοποιεί ανακλήσεις-callbacks για να σηματοδοτήσει την ολοκλήρωση μιας εργασίας.

Το Node.js χτίστηκε στη μηχανή JavaScript Google V8 αφού ήταν ανοιχτό με βάση την άδεια BSD. Είναι εξαιρετικά γρήγορη και ικανή με βασικά στοιχεία του διαδικτύου όπως HTTP, DNS, TCP. Επίσης, η JavaScript είναι μια πολύ γνωστή γλώσσα, καθιστώντας το Node.js άμεσα προσβάσιμο σε ολόκληρη την κοινότητα ανάπτυξης ιστού.

3.1.3 Υποστήριξη

Υπάρχουν χιλιάδες βιβλιοθήκες ανοιχτού κώδικα για το Node.js, οι περισσότεροι από τους οποίους φιλοξενούνται στον ιστότοπο npm.

Το Node.js είναι ένα περιβάλλον διακομιστή ανοιχτού κώδικα

Το Node.js είναι δωρεάν

Το Node.js εκτελείται σε διάφορες πλατφόρμες (Windows, Linux, Unix, Mac OS X κ.λπ.)

Το Node.js χρησιμοποιεί το JavaScript στον διακομιστή

Το Node.js χρησιμοποιεί ασύγχρονο προγραμματισμό

Παράδειγμα:

Μια κοινή εργασία για έναν διακομιστή-server ιστού μπορεί να είναι να ανοίξει ένα αρχείο στο διακομιστή-server και να επιστρέψει το περιεχόμενο στον πελάτη.

- Ακολουθεί το πώς η PHP χειρίζονται ένα αίτημα αρχείου:

1. Στέλνει την εργασία στο σύστημα αρχείων του υπολογιστή.
2. Περιμένετε ενώ ανοίγει το σύστημα αρχείων και διαβάζει το αρχείο.
3. Επιστρέφει το περιεχόμενο στον πελάτη.
4. Έτοιμο να χειριστεί το επόμενο αίτημα.

- Εδώ είναι πώς το Node.js χειρίζεται ένα αίτημα αρχείου:

1. Στέλνει την εργασία στο σύστημα αρχείων του υπολογιστή.
2. Έτοιμο να χειριστεί το επόμενο αίτημα.

3. Όταν το σύστημα αρχείων έχει ανοίξει και να διαβάσει το αρχείο, ο διακομιστής επιστρέφει το περιεχόμενο στον πελάτη.

4. Το Node.js εξαλείφει την αναμονή και απλώς συνεχίζει με το επόμενο αίτημα.

5. Το Node.js εκτελεί ένα μη αποκλειστικό, ασύγχρονο προγραμματισμό, ο οποίος είναι πολύ αποδοτικός στη μνήμη.

Επιπρόσθετα το Node.js μπορεί να δημιουργήσει δυναμικό περιεχόμενο σελίδας. Το Node.js μπορεί να δημιουργήσει, να ανοίξει, να διαβάσει, να γράψει, να διαγράψει και να κλείσει αρχεία στο διακομιστή. Το Node.js μπορεί να συλλέξει δεδομένα φόρμας

3.1.4 Πρώτη επαφή - Οδηγίες Χρήσης

Κατεβάζουμε το Node.js [20] και κάνουμε εγκατάσταση.

Δημιουργούμε ένα νέο αρχείο "myfirst.js"και γράφουμε το παρακάτω κομμάτι κώδικα:

```
var http = require('http');
http.createServer(function (req, res) {
res.writeHead(200, {'Content Type':'text/html'});
```

```
    res.end('Hello World!');
}).listen(8080);
```

Ανοίγουμε ένα CMD ή ένα command line και γράφουμε

node myfirst.js

Μόλις φτιάξαμε έναν server σε τοπικό επίπεδο localhost στην θύρα 8080.

Ανοίγουμε ένα browser και γράφουμε το

http://localhost:8080

Αυτό που βλέπουμε είναι το `Hello World! Που` μας δίνει ο server.

3.2 **Socket.io**

Το Socket.IO είναι μια βιβλιοθήκη JavaScript για εφαρμογές web σε πραγματικό χρόνο. Επιτρέπει την αμφίδρομη επικοινωνία σε πραγματικό χρόνο μεταξύ των web clients και των διακομιστών. Έχει δύο μέρη: μια βιβλιοθήκη πελάτη στην πλευρά που τρέχει στο πρόγραμμα περιήγησης, και μια βιβλιοθήκη διακομιστή στην πλευρά για Node.js. Όπως και το Node.js, οδηγείται από γεγονότα. [22]

- Δημιουργός: Guillermo Rauch

- Σταθερό release: 2.1.1 / May 23, 2018[1]
- Repository: github.com/Automattic/socket.io
- Γραμμένο σε: JavaScript
- Λειτουργικό Σύστημα: Cross-platform
- Τύπος: Event-driven networking
- Άδεια: MIT License
- Ιστοσελίδα: socket.io

Το Socket.IO χρησιμοποιεί πρωτίστως το πρωτόκολλο WebSocket ενώ παρέχει την ίδια διεπαφή. Παρόλο που μπορεί να χρησιμοποιηθεί απλώς ως περιτύλιγμα για το WebSocket, παρέχει πολλά ακόμα χαρακτηριστικά, συμπεριλαμβανομένης της μετάδοσης σε πολλαπλές υποδοχές-sockets, αποθήκευσης δεδομένων που σχετίζονται με κάθε πελάτη και ασύγχρονης εισόδου / εξόδου. Μπορεί να εγκατασταθεί με το εργαλείο npm.

Το Socket.IO παρέχει τη δυνατότητα μεταφοράς δεδομένων σε πραγματικό χρόνο, αμφίδρομης ροής, άμεσων μηνυμάτων και συνεργασίας με έγγραφα.

Το Socket.IO χειρίζεται τη σύνδεση με διαφάνεια. Μπορεί να γίνει αυτόματη αναβάθμιση στο WebSocket αν είναι δυνατόν. Αυτό απαιτεί από τον προγραμματιστή να έχει μόνο γνώση Socket.IO.

Ένας διακομιστής υλοποίησης Socket.IO δεν μπορεί να συνδεθεί με έναν πελάτη WebSocket του nonSocket.IO. Ένας πελάτης υλοποίησης Socket.IO δεν μπορεί να μιλήσει σε διακομιστή WebSocket. Το Socket.IO απαιτεί τη χρήση των βιβλιοθηκών Socket.IO από την πλευρά του πελάτη και του διακομιστή.
Από την έκδοση 2.0, το Socket.IO χρησιμοποιεί τα μWebSockets ως την υποκείμενη βιβλιοθήκη WebSocket.

Το Socket.IO επιτρέπει επικοινωνία σε πραγματικό χρόνο, αμφίδρομη και βασισμένη σε γεγονότα.
Λειτουργεί σε κάθε πλατφόρμα, πρόγραμμα περιήγησης ή συσκευή, εστιάζοντας εξίσου στην αξιοπιστία και την ταχύτητα.

Κεφάλαιο 4ο

Εισαγωγή στο Raspberry Pi 3

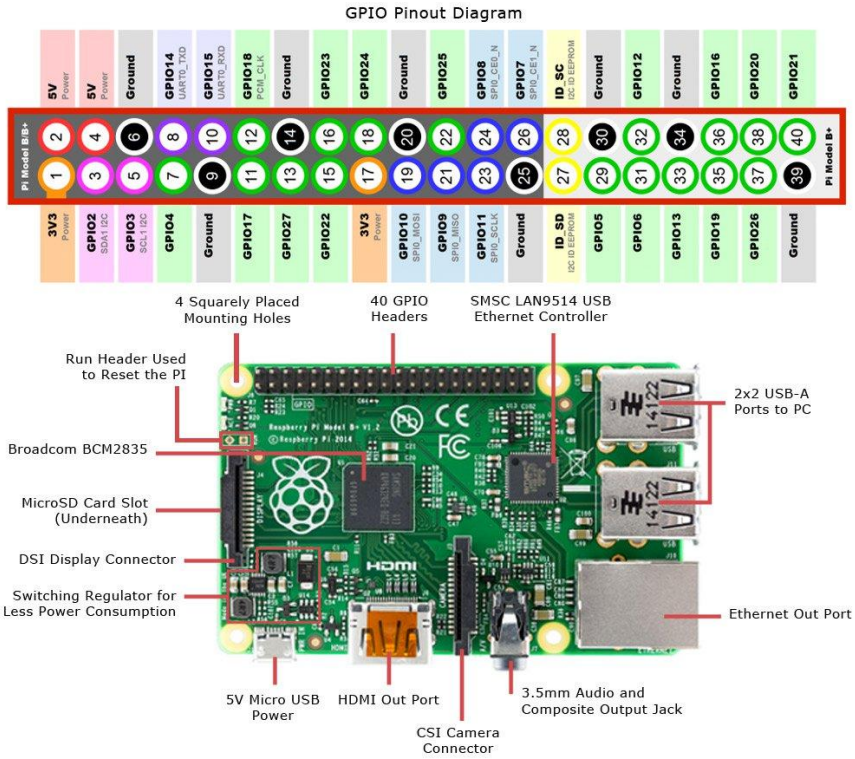
4.1 **Τι είναι το Raspberry Pi 3**

Το Raspberry Pi 3 [23][24] είναι ένας πλήρης υπολογιστής σε μέγεθος πιστωτικής κάρτας.

Παρά τον μικρό όγκο του, το Raspberry Pi στη τελευταία του έκδοση διαθέτει έναν τετραπύρηνο επεξεργαστή 1200MHz, μια διπύρηνη κάρτα γραφικών με GPU, 1GB RAM, τέσσερις θύρες USB, έχει έξοδο HDMI, τροφοδοτείται μέσω Micro USB και έχει 40 pins γενικής χρήσης για σύνδεση με άλλες ηλεκτρονικές συσκευές και περιφερειακά. Το συνδέεις σε μια οθόνη και με ένα πληκτρολόγιο/ποντίκι έχουμε έναν υπολογιστή με λειτουργικό Linux, συνήθως.

Τα βασικά χαρακτηριστικά του Raspberry Pi 3

SoC: Broadcom BCM2837
CPU: 4× ARM Cortex-A53, 1.2GHz
GPU: Broadcom VideoCore IV
RAM: 1GB LPDDR2 (900 MHz)
Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
Storage: microSD
GPIO: 40-pin header, populated
Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)



Σχήμα 4.1. Πλακέτα Raspberry Pi 3 με τους ακροδέκτες

Η έκδοση Raspberry Pi 3 με τις παραπάνω προδιαγραφές κοστίζει γύρω στα 40 ευρώ στην Ελλάδα, ενώ μπορούμε να το αγοράσουμε και σαν μέρος ενός πλήρους Starter kit.

4.1.1 Ιστορία

Το 2006, στο Πανεπιστήμιο του Cambridge, μια ομάδα στο τμήμα Computer Laboratory παρατήρησε το μειωμένο ενδιαφέρον των φοιτητών στην πληροφορική. [23]

Στην ομάδα αυτή ανήκαν οι Eben Upton, Rob Mullins, Jack Lang, και Alan Mycroft και σκέφτηκαν ότι η λύση θα ήταν ένας μικρός και προσιτός υπολογιστής.

Με ένα τέτοιο σύστημα, θα ήταν δυνατόν να διδάσκεται πιο εύκολα η πληροφορική στα σχολεία ενώ παράλληλα να έχει μεγαλύτερο ενδιαφέρον για τους μαθητές/φοιτητές.

Η ομάδα σχεδίασε αρκετά αρχικά πρωτότυπα του Raspberry και το κόστος στην αρχή ήταν πολύ ακριβό.

Μετά την κυκλοφορία του πρώτου iPhone το 2007 και στην συνέχεια των smartphones, το κόστος της τεχνολογίας άρχισε να μειώνεται αρκετά και έτσι έγινε βιώσιμη η υλοποίηση του Raspberry Pi.

Το 2008 κυκλοφόρησε το πρώτο Raspberry Pi, με τα Model A, Model A+ και Model B.

Τα μοντέλα αυτά διέθεταν επεξεργαστή ARMv6k στα 700 MHz, 256MB RAM, κάρτα γραφικών Broadcom VideoCore IV, και κατανάλωση από 1 έως 3.5 watt, ενώ η αποθήκευση των δεδομένων γινόταν σε κάρτες SD, SDHC και MicroSD.

Μέσα σε δύο χρόνια το Raspberry Pi Model B πούλησε πάνω από δύο εκατομμύρια κομμάτια.

Στη συνέχεια κυκλοφόρησαν οι εκδόσεις Model B rev 2 και Model B+ με 512MB RAM. Τον Οκτώβριο του 2014, οι συνολικές πωλήσεις άγγιζαν τα 4 εκατομμύρια.

Το Φεβρουάριο του 2015 κυκλοφόρησε το Raspberry Pi Generation 2 Model B, με RAM 1GB και 6 φορές μεγαλύτερη ταχύτητα επεξεργαστή. Χρησιμοποιούσε τον τετραπύρηνο Cortex-A7 (ARMv7) και διπύρηνη κάρτα γραφικών Broadcom VideoCoreIV.

Το 2015 κυκλοφόρησε το Raspberry Pi Zero, το μικρό του Raspberry Pi, 512MB RAM, και επεξεργαστή ARM1176JZF-S στα 1GHz.

Το δικό μας Raspberry Pi Generation 3 Model B κυκλοφόρησε το Φεβρουάριο του 2016. Ακόμα ταχύτερος επεξεργαστής ARM Cortex-A53 στα 1.2GHz, 1GB RAM και κάρτα γραφικών Broadcom VideoCore IV χρονισμένη στα 250MHz, συχνότητα υψηλότερη από κάθε προηγούμενη έκδοση.

Αξίζει να σημειωθεί ότι το Raspberry Pi έχει κατανάλωση μόλις 4W.

4.1.2 Διανομές

Όπως αναφέρθηκε συνδέοντάς το σε μια οθόνη και προσθέτοντας πληκτρολόγιο και ποντίκι, έχουμε έναν πλήρη υπολογιστή, ο οποίος υποστηρίζει συγκεκριμένες διανομές Linux.

Αυτή τη στιγμή, οι διανομές που υποστηρίζονται στην τελευταία έκδοση του Raspberry Pi είναι οι εξής:[25]

- Raspbian
- Kali Linux
- Pidora
- Windows 10 IoT Core
- Ubuntu Core
- RISC OS
- Arch Linux ARM
- FreeBSD
- RetroPie

4.1.3 Ακροδέκτες Εισόδου-Εξόδου - General purpose input-output (GPIO) connector

Τα Raspberry Pi 1 Models A+ and B+, Pi 2 Model B, Pi 3 Model B and B+, and Pi Zero and Zero W GPIO J8 έχουν 40-pin ακροδέκτες.

Στον Πίνακα 4.1 παρουσιάζονται όλοι οι ακροδέκτες που πρέπει να γνωρίσουμε κατά την σύνδεση.

Πίνακας 4.1. Ακροδέκτες Εισόδου-Εξόδου για Raspberry 3

GPIO#	2nd func.	Pin#		Pin#	2nd func.	GPIO#
	+3.3 V	1		2	+5 V	
2	SDA1 (I²C)	3		4	+5 V	
3	SCL1 (I²C)	5		6	GND	
4	GCLK	7		8	TXD0 (UART)	14
	GND	9		10	RXD0 (UART)	15
17	GEN0	11		12	GEN1	18
27	GEN2	13		14	GND	

22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK (SPI)	23	24	CE0_N (SPI)	8
	GND	25	26	CE1_N (SPI)	7
(Pi 1 Models A and B stop here)					
EEPROM	ID_SD	27	28	ID_SC	EEPROM
5	N/A	29	30	GND	
6	N/A	31	32		12

13	N/A	33	34	GND	
19	N/A	35	36	N/A	16
26	N/A	37	38	Digital IN	20
	GND	39	40	Digital OUT	21

Στο Σχήμα 4.2 η αντιστοιχία των GPIO αριθμών με τα αντίστοιχα πινάκια που βρίσκονται πάνω στην πλακέτα.

Σχήμα 4.2. Πλακέτα Raspberry Pi 3 με τους ακροδέκτες

4.2 **Τρόποι επικοινωνίας του Raspberry Pi 3 με το περιβάλλον - αισθητήρες**

Το Raspberry έχει δύο συγκεκριμένες λειτουργίες μέσω των ακροδεκτών του, να μπορεί να γίνει έξοδος ή είσοδος. Χρησιμοποιώντας αυτούς τους ακροδέκτες μπορούμε να χρησιμοποιήσουμε κάθε ψηφιακό αισθητήρα αλλά όταν θέλουμε να χρησιμοποιήσουμε αναλογικό αισθητήρα δεν μπορούμε να το συνδέσουμε απευθείας στο Raspberry αλλά μόνο με την χρήση ειδικών μετατροπέων αναλογικού σε ψηφιακό σήμα.

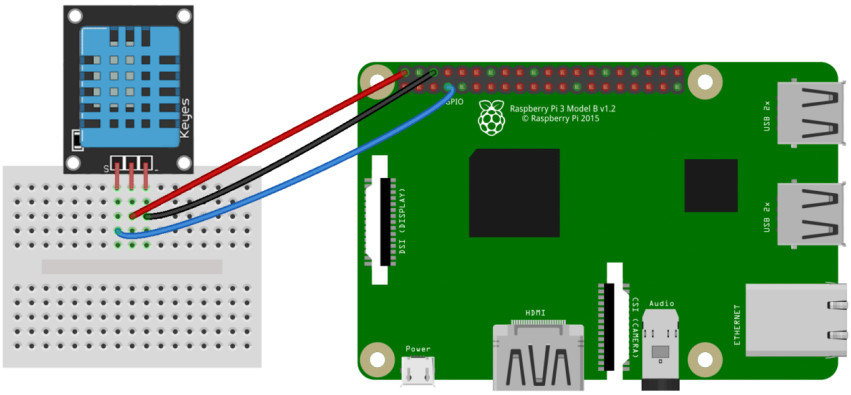
Ενώ μπορούν να διαβάσουν 1 ή 0 και να γράψουν 1 ή 0 στην έξοδο μπορούν να εκτελέσουν πολλές λειτουργίες γιατί το πρόγραμμα τους γράφεται σε γλώσσα προγραμματισμού όπως η Python προσομοιώνοντας πολλές λειτουργίες μικροελεγκτών, όπως SPI, I2C, serial, usb κτλ.

Μερικοί αισθητήρες που μπορούν να χρησιμοποιηθούν με το Raspberry παρουσιάζονται στην [26] και στο επόμενο κεφάλαιο.

Από τους πιο σημαντικούς αισθητήρες είναι το Temperature / Humidity DHT11 που χρησιμοποιείται στην εργασία.

Είναι ο πιο οικονομικός αισθητήρας Υγρασίας και Θερμοκρασίας. Χρησιμοποιείται για την μέτρηση της υγρασίας και της θερμοκρασίας του χώρου με ψηφιακή έξοδο. Το εύρος υγρασίας που μετράει είναι 20-80% και το εύρος της θερμοκρασίας είναι 0-

50 °C. Η τροφοδοσία κυμαίνεται μεταξύ 3.0V έως 5.0V DC. Είναι συμβατός με τις περισσότερες αναπτυξιακές πλακέτες, όπως Arduino[27], Raspberry, ESP32[28].



Σχήμα 4.3. Σύνδεση Raspberry με το DHT11 [29]

Ο κώδικας και με ποιον τρόπο επικοινωνεί το Raspberry με τους ακροδέκτες με το DTH11 παρουσιάζεται στο επόμενο κεφάλαιο.

Κεφάλαιο 5ο

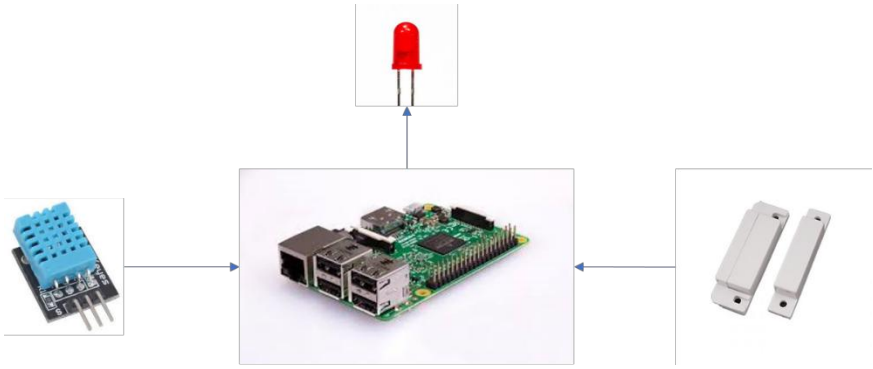
Σύστημα ασφαλείας με χρήση δικτύου αισθητήρων

5.1 **Το σύστημα ασφαλείας του χώρου**

Σκοπός του συστήματος ασφαλείας είναι να μπορεί να ελέγχει τον χώρο που είναι εγκατεστημένο σε πραγματικό χρόνο και να επικοινωνεί με τον χρήστη αμφίδρομα μέσω μιας ιστοσελίδας.

Στο σύστημα τοποθετήθηκαν ενδεικτικά ένας αισθητήρας θερμοκρασίας/υγρασίας και ένα led εξόδου το οποίο μπορεί να είναι μια σειρήνα ή ένα ρελέ που να ενεργοποιεί μια συσκευή ή κάτι άλλο. Επίσης, μπορεί να τοποθετηθεί και μια μαγνητική παγίδα η οποία δεν συμπεριλαμβάνεται αρχικά στο σύστημα.

Στο παρακάτω σχήμα φαίνεται ένα ενδεικτικό πλάνο-σχέδιο του συστήματος.



Σχήμα 5.1. Τοπολογία σύνδεσης Raspberry τα περιφερειακά

Μπορώ να ελέγξω τις θύρες εισόδου/ εξόδου (I/O) του Raspberry πολύ εύκολα με χρήση Python η οποία είναι και η κύρια γλώσσα και με την μεγαλύτερη πληθώρα παραδειγμάτων στο διαδίκτυο για το Raspberry.

Ένα παράδειγμα κώδικά με την Python είναι

```
import RPi.GPIO as GPIO
import dht11
import time
import datetime

# initialize GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
```

```
GPIO.cleanup()

# read data using pin 4
instance = dht11.DHT11(pin=4)

while True:
    result = instance.read()
    if result.is_valid():
        print("Last valid input: " +
str(datetime.datetime.now()))
        print("Temperature: %d C" %
result.temperature)
        print("Humidity: %d %%" % result.humidity)

    time.sleep(1)
```

Όπως φαίνεται στον κώδικα χρησιμοποιούμε έτοιμες βιβλιοθήκες για υποστήριξη και χρήση των ακροδεκτών I/O και για την ανάγνωση τιμών από το DHT11.

Χρησιμοποιούμε μια επαναληπτική διαδικασία για να μετράμε συνεχώς τον αισθητήρα.

Στην εργασία αυτή δεν μπορούμε να χρησιμοποιήσουμε Python ή άλλη γλώσσα γιατί χρειάζεται να χρησιμοποιήσουμε WebSockets. Βεβαίως μπορούμε να χρησιμοποιήσουμε WebSockets και με την Python αλλά δεν μπορούμε να συνδεθούμε με το node.js server ιδίως με την socket.io βιβλιοθήκη.

Οπότε για την χρήση ακροδεκτών με node.js χρησιμοποιούμε την βιβλιοθήκη 'onoff'

```
var Gpio = require('onoff').Gpio;
```

και ορίζουμε ποιους ακροδέκτες θα κάνουμε Έξοδο και Είσοδο.

```
var LED = new Gpio(4, 'out');
```

και με μία εντολή μπορούμε να κάνουμε την έξοδο 0 ή 1

```
LED.writeSync(1);
```

Για την χρήση του αισθητήρα θα χρησιμοποιήσουμε την βιβλιοθήκη 'node-dht-sensor'

```
var sensor = require('node-dht-sensor');
```

Όπου με λίγες γραμμές κώδικα μπορούμε να διαβάσουμε τις τιμές θερμοκρασίας και υγρασίας.

```
--- Ανάγνωση θερμοκρασίας και υγρασίας από τον αισθητήρα DHT11

sensor.read(11, 4, function(err, temp, hum) {

--- Αν ο αισθητήρας δουλεύει σωστά τότε μπορώ να αποθηκεύσω τις τιμές

    if (!err) {
```

```
temperature = temp.toFixed(1);

humidity = hum.toFixed(1);

console.log('temp: ' + temperature + '°C, ' +

'humidity: ' + humidity + '%');

}

});
```

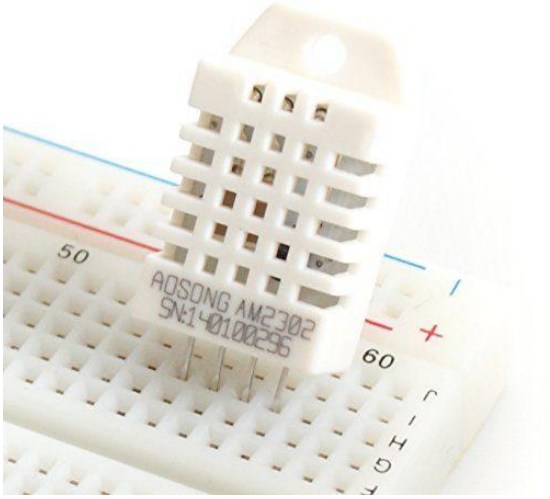
5.2 **Αισθητήρες**

Στην ενότητα αυτή παρουσιάζονται οι πιο σημαντικοί αισθητήρες που μπορούν να χρησιμοποιηθούν με το Raspberry [26].

Θερμοκρασία / Υγρασία

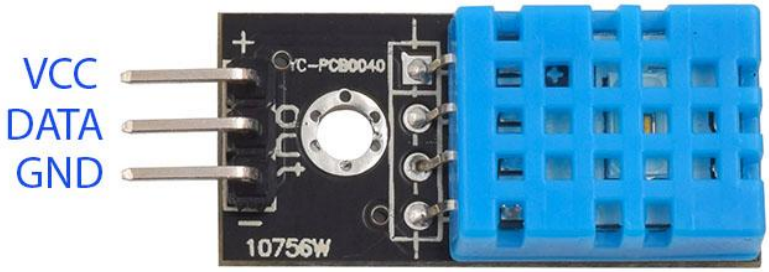
Οι αισθητήρες DHT11 και DHT22 μπορούν να μετρήσουν την υγρασία καθώς και τη θερμοκρασία. Χρησιμοποιείται μόνο ένα GPIO.

Η διαφορά μεταξύ των δύο είναι κυρίως η περιοχή μέτρησης και η ακρίβεια.



Σχήμα 5.2. Αισθητήρας DHT22

Το λευκό DHT22 μπορεί να μετρήσει όλα τα επίπεδα υγρασίας από 0-100% με ακρίβεια 2%. Συγκριτικά, το DHT11 (μπλε) μπορεί να μετρήσει μόνο περιοχές με υγρασία 20-90% και πάνω από 'όλα, η ακρίβεια είναι σημαντικά χειρότερη με 5%.

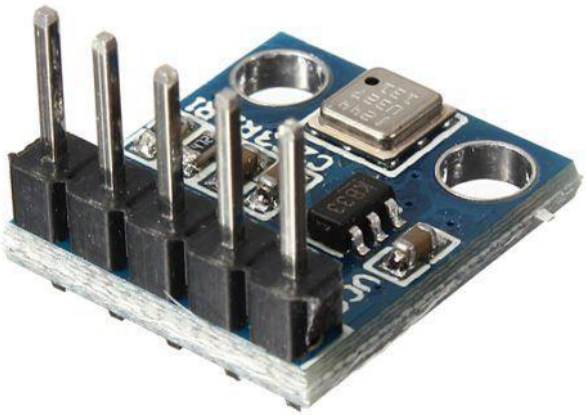


Σχήμα 5.3. Αισθητήρας DHT11

Ο αισθητήρας DHT11 έχει ένα μικρό πλεονέκτημα τιμής (περίπου ένα ευρώ).

Πίεση αέρα-Βαρόμετρο BMP180

Ο προσδιορισμός της πίεσης του αέρα μπορεί να έχει νόημα στους μετεωρολογικούς σταθμούς και σε παρόμοια έργα.



Σχήμα 5.4. Βαρόμετρο BMP180

Αυτό γίνεται καλύτερα χρησιμοποιώντας το BMP180, το οποίο ελέγχεται μέσω του I2C στο Raspberry Pi.

Αισθητήρας αερίου MQ-2

Οι αισθητήρες αερίου MQ μπορούν να ανιχνεύσουν διαφορετικά αέρια σε θερμοκρασία δωματίου. Ανάλογα με το μοντέλο, υποστηρίζονται και άλλα αέρια.

Σχήμα 5.5. Αισθητήρας αερίου MQ-2

Το MQ-2 μπορεί να αναγνωρίσει μεθάνιο, βουτάνιο, LPG και καπνό, το MQ3 ανιχνεύει, για παράδειγμα, το αλκοόλ, την αιθανόλη και τον καπνό κλπ.

Αισθητήρας κίνησης PIR

Ο αισθητήρας κίνησης PIR έχει κάποια πλεονεκτήματα έναντι άλλων παρόμοιων προϊόντων: Εκτός από τη χαμηλή τιμή, αποστέλλεται σήμα μόνο αν κινείται κάτι.

Σχήμα 5.6. Αισθητήρας κίνησης PIR

Αυτό σας επιτρέπει απλά να περιμένετε το σήμα χρησιμοποιώντας το GPIO.

HC-SR04 αισθητήρας υπερήχων

Ο αισθητήρας HC-SR04 δεν είναι ανιχνευτής απόστασης / κίνησης, αλλά αισθητήρας υπερήχων. Μέσω ενός μικρού τέχνασμα είναι δυνατό να μετρηθεί η απόσταση.

Σχήμα 5.7. HC-SR04 αισθητήρας υπερήχων

Με τη μέτρηση του χρόνου που παρήλθε μεταξύ της μετάδοσης και της λήψης ενός σήματος υπερήχων, μπορείτε να μετρήσετε την απόσταση καθώς είναι γνωστή η ταχύτητα του ήχου στον αέρα.

Μαγνητικός διακόπτης / παγίδα

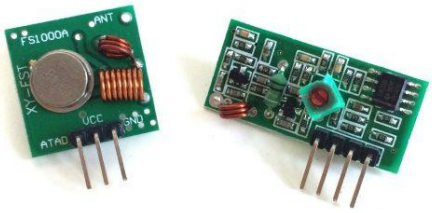
Μέσω μαγνητικών αισθητήρων / ηλεκτρονόμων, μπορείτε να ελέγξετε για δυαδικές καταστάσεις.

Σχήμα 5.8. Μαγνητικός διακόπτης / παγίδα

Ο μαγνητικός ηλεκτρονόμος ανοίγει μόλις ο μαγνήτης βρίσκεται κοντά. Διαφορετικά, η πρόσβαση είναι κλειστή. Επομένως, εάν περάσει η τάση, μπορείτε να ελέγξετε την κατάσταση.

RF 433 MHz

Μία από τις πιο απλές μεθόδους για τη μετάδοση σημάτων μέσω RF είναι ο πομπός και ο δέκτης 433 MHz.



Σχήμα 5.9. RF 433 MHz

Ραδιοελεγχόμενες πρίζες / Υποδοχείς ρεύματος

Στον τομέα του οικιακού αυτοματισμού, οι ασύρματες υποδοχές είναι σχεδόν στάνταρ. Η συντριπτική πλειονότητα αυτών των συσκευών λειτουργούν με ραδιοσήματα 433 MHz.

Σχήμα 5.10. Ραδιοελεγχόμενες πρίζες / Υποδοχείς ρεύματος

Με την ανάγνωση των κωδικών του τηλεχειριστηρίου με ένα δέκτη στο Raspberry Pi, μπορεί κανείς να αλλάξει αυτές τις ραδιοφωνικές υποδοχές ξεχωριστά.

GSM module

Το Raspberry Pi χρησιμοποιείται σε πολλά υπαίθρια έργα, π.χ. ως μετεωρολογικός σταθμός ή για την παρακολούθηση ορισμένων παραμέτρων. Ωστόσο, ακόμη και αν δεν υπάρχει διαθέσιμο σήμα (ή μόνο ένα αδύναμο) WiFi, πολλές λειτουργίες είναι περιορισμένες.

Σχήμα 5.11. GSM Module

Εάν εξακολουθείτε να θέλετε να έχετε πρόσβαση στο Pi και επίσης να λάβετε τα δεδομένα ενός τέτοιου εξωτερικού έργου, απαιτείται σύνδεση στο Internet. Με μια κάρτα SIM μπορείτε να έχετε πρόσβαση στο Pi , παρέχοντας του internet.

Κεφάλαιο 6ο

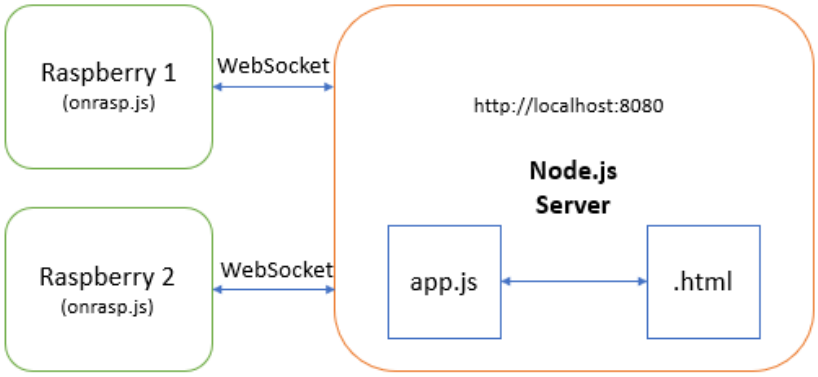
Εφαρμογή Web για αμφίδρομη επικοινωνία

6.1 **Περιγραφή και ανάλυση του έργου**

Στο κεφάλαιο αυτό θα περιγραφεί ο τρόπος και οι μέθοδοι που χρησιμοποιήθηκαν για να υλοποιηθεί το σύστημα με αμφίδρομη επικοινωνία πραγματικού χρόνου μεταξύ μιας συσκευής Raspberry που βρίσκεται σε ένα χώρο, όπως το σπίτι, με έναν πελάτη που βρίσκεται οπουδήποτε. Η επικοινωνία του πελάτη πραγματοποιείται μέσω ενός browser, όπως το chrome και Firefox, αφού ο χρήστης συνδεθεί μέσω ενός απλού συστήματος σύνδεσης – πιστοποίησης. Το σύστημα πετυχαίνει, εκτός από την αμφίδρομη επικοινωνία, να διασφαλίσει ασφαλή επικοινωνία και μοναδικά sockets.

Στο Σχήμα 6.1 φαίνεται πως στον server, ο οποίος μπορεί να βρίσκεται οπουδήποτε στον κόσμο, βρίσκονται το κεντρικό αρχείο app.js το οποίο είναι γραμμένο στην τεχνολογία node.js και οι ιστοσελίδες που μπορεί να μπει ο χρήστης-πελάτης.

Επίσης, στο σχήμα φαίνεται ότι μπορούμε να συνδέσουμε διάφορα Raspberry στο σύστημα αρκεί ο κάθε χρήστης να έχει καταχωρημένο το μοναδικό id του Raspberry στον Server.

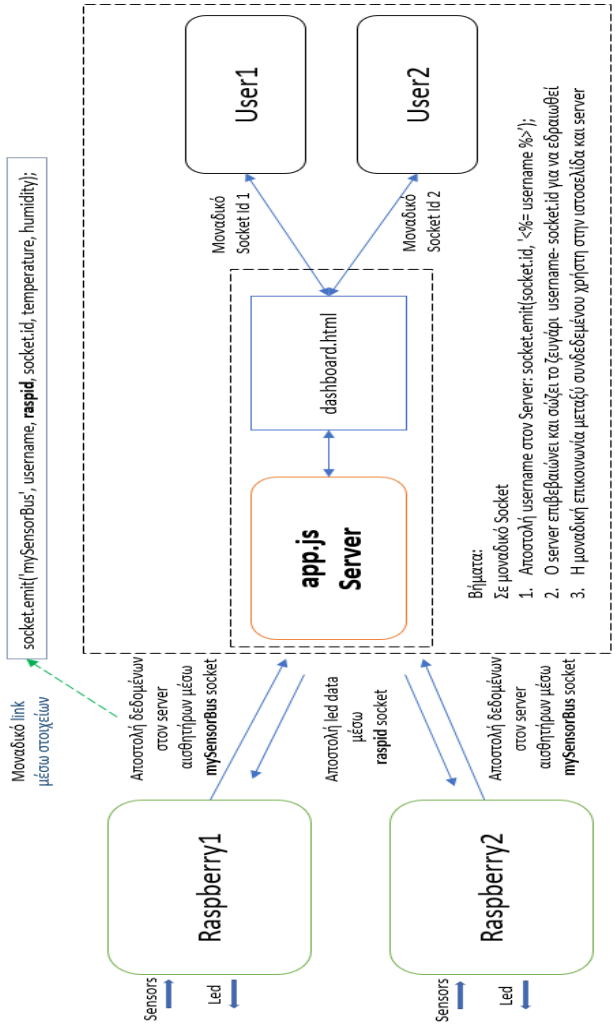


Σχήμα 6.1. Σχέδιο συστήματος

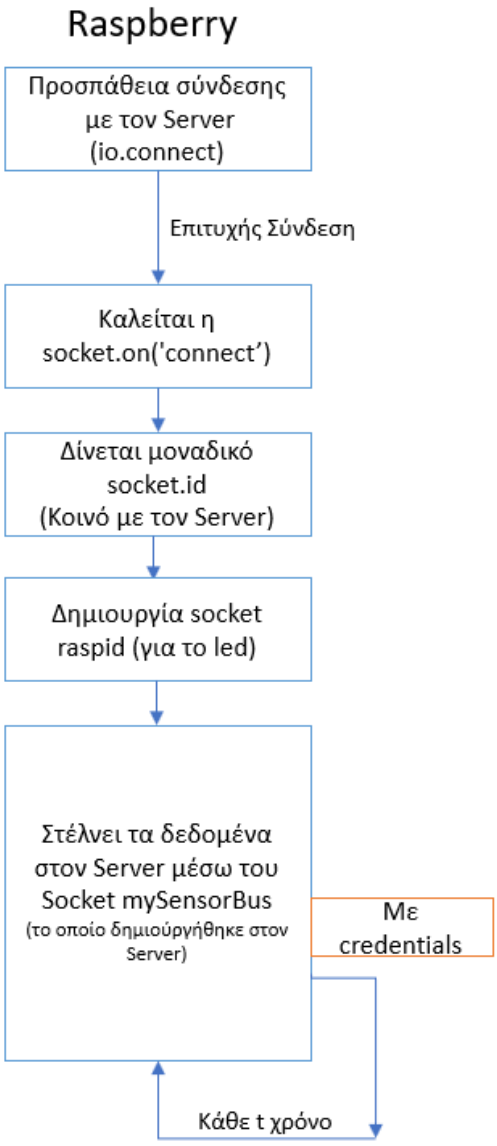
Στο Σχήμα 6.2 φαίνεται η διασύνδεση των sockets μεταξύ των τριών μονάδων.

Οι servers χρησιμοποιούν διαφορετικά Session αλλά μεταξύ τους να έχουν τις ίδιες τιμές. Αυτό συμβαίνει γιατί ο κώδικας στο app.js και ο κώδικας που χρησιμοποιεί το socket.io χρησιμοποιούν διαφορετικό τρόπο που ορίζουν-χρησιμοποιούν τα sessions.

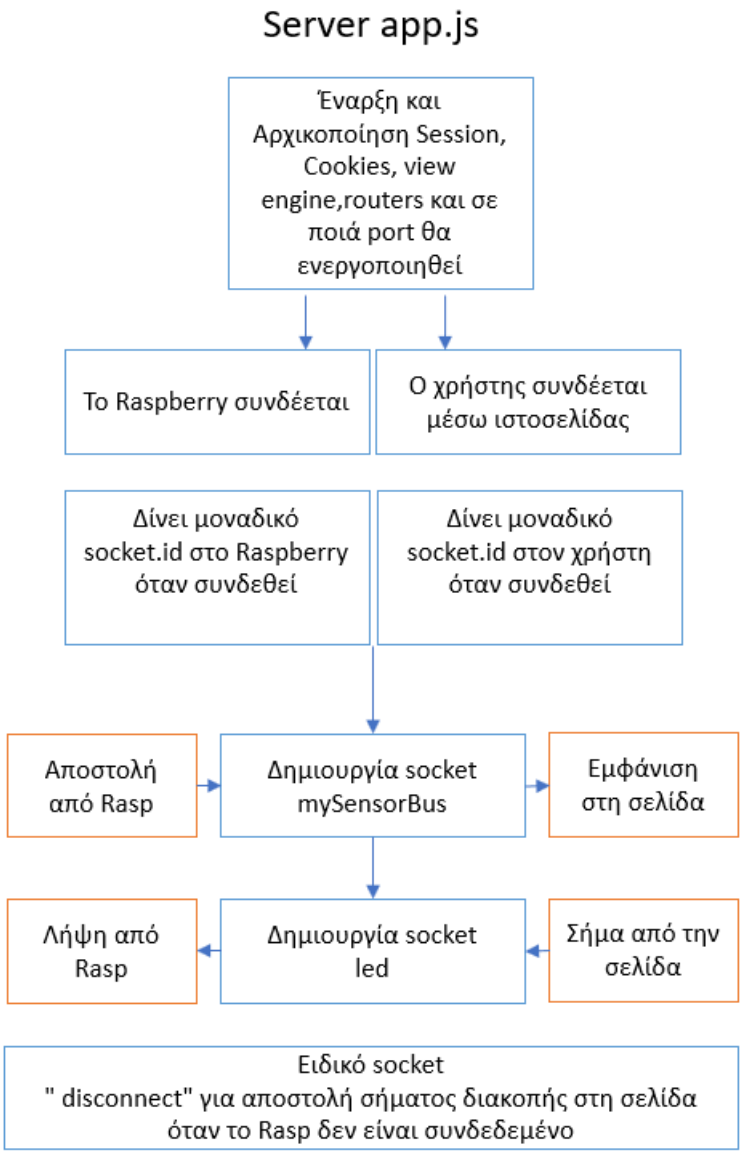
Στο σχήμα 6.2 φαίνονται όλα τα Sockets που χρησιμοποιούνται από την κάθε πλευρά. Όλα τα sockets είναι αμφίδρομα αλλά στο σχήμα διευκινίζεται με ποια κατεύθυνση χρησιμοποιούνται.



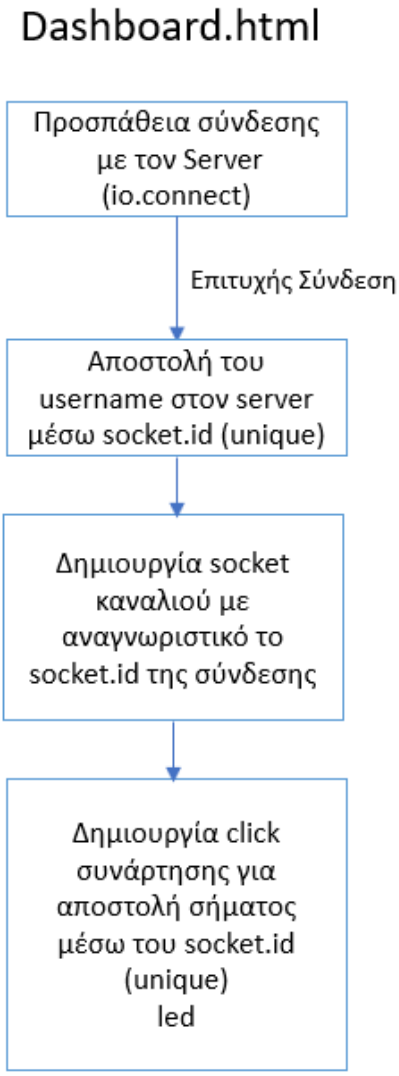
Σχήμα 6.2. Σχέδιο συστήματος



Σχήμα 6.3. Διάγραμμα Ροής για τον κώδικα στο Raspberry

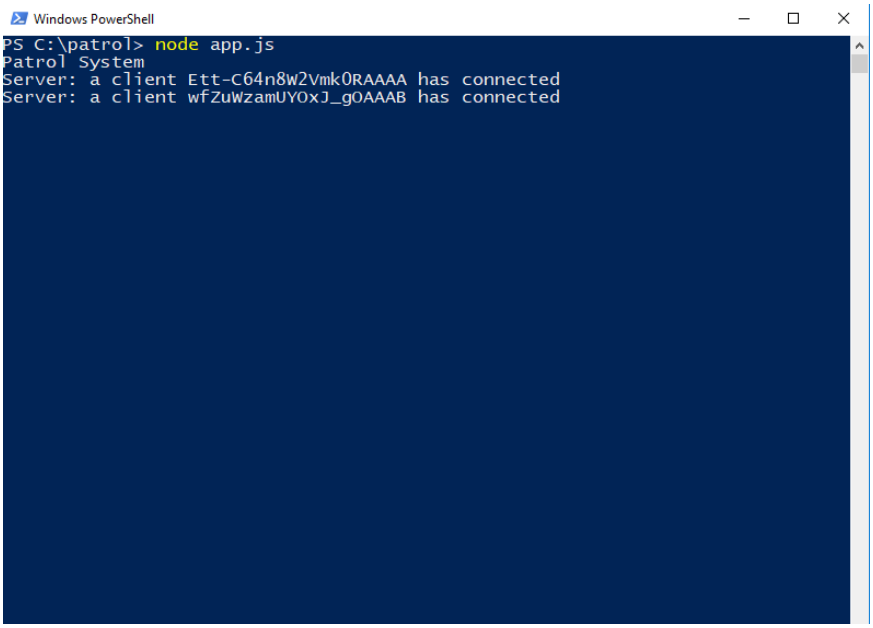


Σχήμα 6.4. Διάγραμμα Ροής για τον κώδικα node.js στο Server



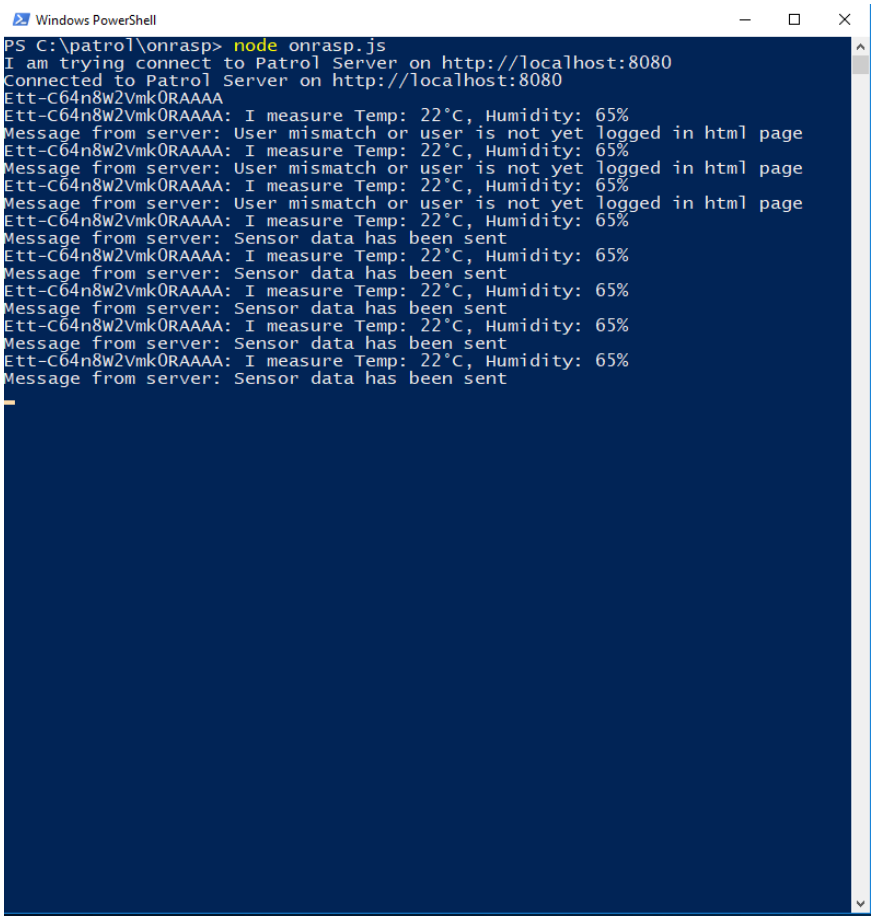
Σχήμα 6.5. Διάγραμμα Ροής για τον κώδικα html για την κεντρική σελίδα ελέγχου στο Server

Στα Σχήματα 6.3-6.5 παρουσιάζονται τα Διαγράμματα Ροής για κάθε κομμάτι κώδικα που χρησιμοποιήθηκε στα τρία τμήματα.



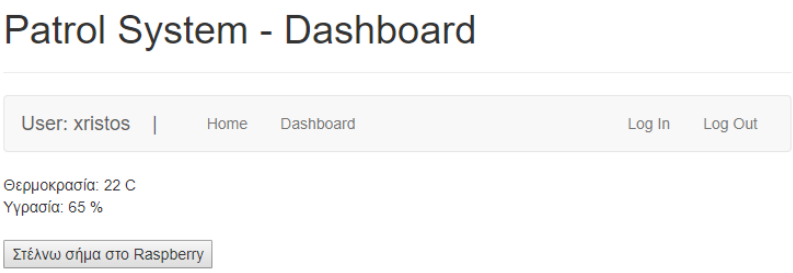
Σχήμα 6.6. Εκτέλεση του app.js

Στο Σχήμα 6.6 φαίνεται η εκτέλεση του app.js που βρίσκεται στον Server και αποτελεί τον πυρήνα ή αλλιώς ο server του συστήματος.
Φαίνεται ότι έχουν συνδεθεί δύο χρήστες. Στην πραγματικότητα ο ένας είναι ο χρήστης Raspberry και ο άλλος είναι ο χρήστης που έχει συνδεθεί (Log in) μέσω της ιστοσελίδας. Αυτοί οι χρήστες είναι ίδιοι και χρησιμοποιούμε μόνο το πρώτο id για την σύνδεση Raspberry.



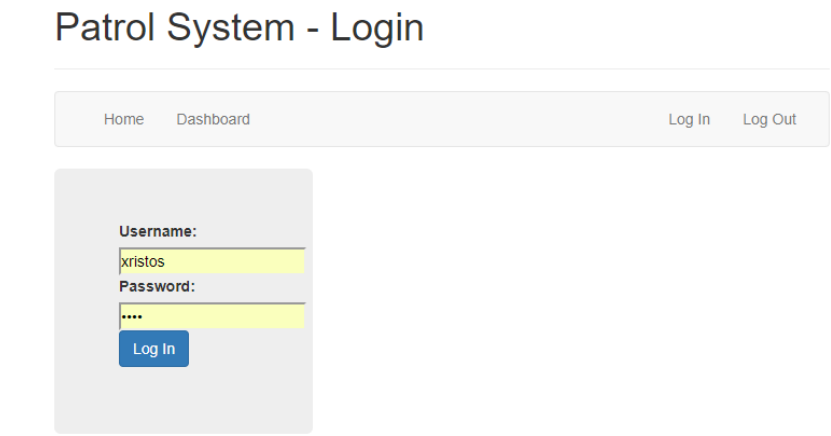
Σχήμα 6.7. Εκτέλεση του onrasp.js στο Raspberry

Στο Σχήμα 6.7 φαίνεται η εκτέλεση του onrasp.js που βρίσκεται στον Raspberry. Όταν συνδέεται επιτυχώς στον Server τότε αποκτάει μοναδικό socket.id και μπορεί να επικοινωνήσει αμφίδρομα με τον Server και μέσω ιστοσελίδας με τον πελάτη-χρήστη.

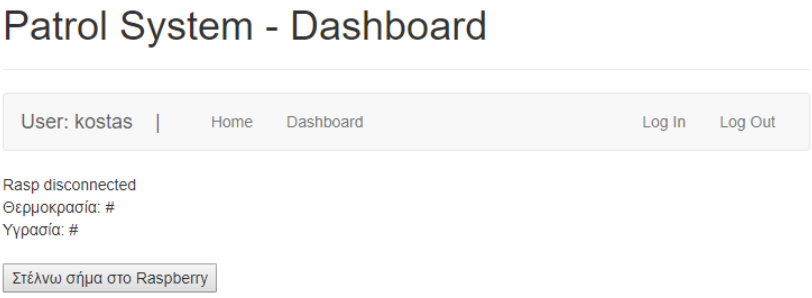


Σχήμα 6.8. Η κεντρική ιστοσελίδα για τον χειρισμό από τον χρήστη

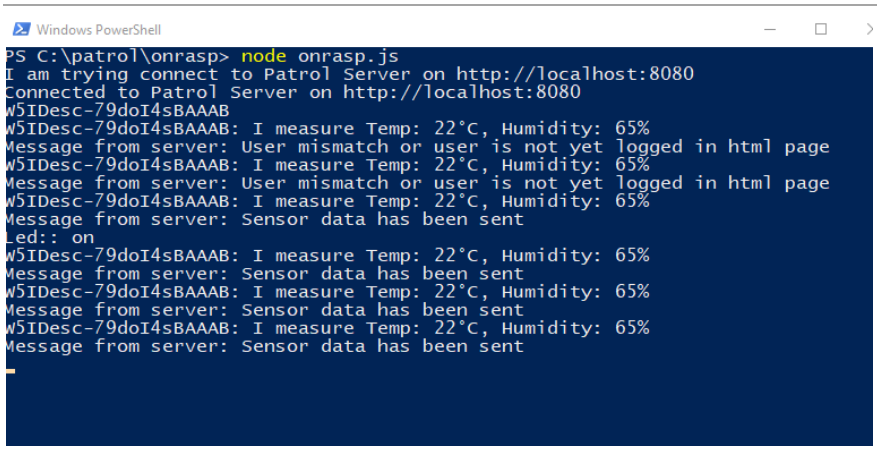
Στο Σχήμα 6.8 παρουσιάζεται η κεντρική ιστοσελίδα για τον χειρισμό από τον χρήστη για την επίβλεψη των τιμών θερμοκρασίας-υγρασίας και την αποστολή σήματος στο Raspberry.



Σχήμα 6.9. Η ιστοσελίδα για την σύνδεση του χρήστη στο σύστημα



Σχήμα 6.10. Η ιστοσελίδα με αποτυχημένη ή κλειστή σύνδεση με το Raspberry



Σχήμα 6.11. Επίβλεψη λήψης του σήματος από τον Server (Led:: on)

Στο Σχήμα 6.11 φαίνεται το σήμα που έλαβε οτο Raspberry από τον Server που ο χρήστης πάτησε στην ιστοσελίδα. Εδώ φαίνεται μια από τις καλύτερες δυνατότητες που έχει η χρήση του WebSocket, να υπάρχει αμφίδρομη επικοινωνία πραγματικού χρόνου μεταξύ του κάθε πελάτη με τον Server χωρίς να χρειάζεται να γίνονται συνέχεια Requests.

6.1.1 Επεξήγηση λειτουργίας Server μέσω του κώδικα App.js

Στην αρχή περιλαμβάνουμε τις βιβλιοθήκες που χρειάζεται ο server:

```
var bodyParser = require('body-parser');
var cookieParser = require('cookie-parser');


var app = require('express')(),
```

```
http  = require("http").createServer(app),
io = require("socket.io")(http),
```

Χρησιμοποιούμε την τεχνολογία 'express' και ‘socket.io’ για την δημιουργία και χρήση WebSockets στο node.js server.

Χρησιμοποιούμε την βιβλιοθήκη "express-session" για χρήση sessions και cookies σε όλο το εύρος του κώδικα. Δίνουμε ένα προσωπικό κλειδί και κωδικό για να είναι μοναδική η κωδικοποίηση.

```
session = require("express-session")({
    key: 'user_sid',
  secret: 'EeBJ60JQzyCyRFfQaVUqTyRGfQvDUt',
  resave: false,
  saveUninitialized: false
});
```

Αρχικοποίηση body-parser για να περνάνε οι παράμετροι από την είσοδο στο req.body

```
app.use(bodyParser.urlencoded({ extended: true }));
```

Αρχικοποίηση του cookie-parser για να επιτρέψει την πρόσβαση στα cookies που είναι αποθηκευμένα στο πρόγραμμα περιήγησης.

```
app.use(cookieParser());
```

Χρήση-ενεργοποίηση session

```
app.use(session);
```

Θέτουμε την μηχανή για view στο ejs

```
app.set('view engine', 'ejs');
```

Οι χρήστες με μοναδικό rasp id ο καθένας

```
var users = [
        {id:20, username:'xristos', pass:'1234', raspid:'EeBJ60JQzy'},
        {id:21, username:'kostas', pass:'1234', raspid: 'CyRFfQaVUq'},
        {id:22, username:'maria', pass:'1234', raspid:  'TyRGfQvDUt'}
        ];
```

Αυτό το middleware - μεσαίο λογισμικό θα ελέγξει εάν το cookie του χρήστη είναι ακόμα αποθηκευμένο στο πρόγραμμα περιήγησης και ο χρήστης δεν έχει οριστεί ακόμα. Τότε αυτόματα τον αποσυνδέει.

Αυτό συμβαίνει συνήθως όταν σταματήσει ο server μετά την είσοδο-login του χρήστη, τότε το cookie παραμένει αποθηκευμένο στο πρόγραμμα περιήγησης.

```
app.use((req, res, next) => {
    if (req.cookies.user_sid && !req.session.username) {
        res.clearCookie('user_sid');
    }
    next();
});
```

Αυτή η middleware συνάρτηση ελέγχει για συνδεδεμένους χρήστες

```
var sessionChecker = (req, res, next) => {
    if (req.session.username && req.cookies.user_sid) {
        res.redirect('/dashboard');
    } else {
        next();
    }
};
```

Route για Home-Page

```
app.get('/', sessionChecker, (req, res) => {
```

```
    res.redirect('/login');
});
```

Route για user Login

```
app.route('/login')
    .get(sessionChecker, (req, res) => {
        res.sendFile(__dirname + '/login.html');
    })
    .post((req, res) => {
```

Αυτα τα λαμβάνουμε απο την Login.html page

```
        var username = req.body.username,
            password = req.body.password;
```

Ψάχνουμε αν τα credentials είναι σωστά συγκρίνοντας με ένα πίνακα που έχουμε κάποια στοιχεία.

Θα μπορούσε να συνδεθεί με μια βάση δεδομένων

```
        loggeduser = users.find(x => x.username === username && x.pass === password)
```

Αν δεν βρεθεί ο χρήστης στον πίνακα τότε επιστρέφει στην Login σελίδα

```
if(loggeduser === undefined)
{
        res.redirect('/login');
}
else
{
```

Αν βρεθεί ο χρήστης στον πίνακα τότε επιστρέφει στην dashboard σελίδα και σώζουμε σε Session(μοναδικό για κάθε χρήστη που συνδέεται το id και το username του χρήστη)

```
//Session για να χρησιμοποιηθούν τα δεδομένα στα Cookies και
  σελίδες html
req.session.userid = loggeduser.id;
req.session.username = loggeduser.username;

res.redirect('/dashboard');
}
});
```

Route για το dashboard

```
app.get('/dashboard', (req, res) => {
```

Κάθε φορά που μπαίνει κάποιος χρήστης σε αυτή την σελίδα πρέπει να ελέξουμε αν ο χρήστης είναι logged.

Οπότε αν υπάρχει το Session του user και το cookie του user_id μόνο τότε θα μπορεί να μπαίνει σε κάθε σελίδα. Αλλιώς θα πηγαίνει στη σελίδα login.

```
    if (req.session.username && req.cookies.user_sid) {
  //Στέλνουμε το μοναδικό username στην ιστοσελίδα dashboard
  //για να χρησιμοποιηθεί στην αποστολή του socket.id της σύνδεσης στον server και να γίνει μοναδική αντιστοίχηση του με το username του user socket.emit(socket.id, '<%= username %>');
        res.render('dashboard',{username:req.session.username});
    } else {
        res.redirect('/login');
    }
  });
```

Route για user logout

```
app.get('/logout', (req, res) => {
```

Αν υπάρχει το Session του user και το cookie του user_id τότε στο logout θα καθαρίσουμε το cookie και με αυτό καθαρίζει και το αντίστοιχο session.

```
    if (req.session.username && req.cookies.user_sid) {
        res.clearCookie('user_sid');
        res.redirect('/');
    } else {
```

Αν δεν ήταν logged τότε απλά θα πάει στη σελίδα του login

```
        res.redirect('/login');
    }
});
```

Route για τον χειρισμό των 404 requests(unavailable routes)

```
app.use(function (req, res, next) {
  res.status(404).send("Δεν υπάρχει αυτό το path")
});
```

Ο Server ενεργοποιείται στην port 8080

```
    http.listen(8080, function(){
    console.log('Patrol System');
```

```
});
```

Όταν κάποιος client (σελίδα ή rasp) συνδεθεί στον server τότε καλείται αυτή η συνάρτηση

```
io.on('connection', function(socket){
```

Όταν κάποιος client (σελίδα ή rasp) συνδεθεί στον server αποκτάει Μοναδικό id, το socket.id

```
console.log('Server: a client '+ socket.id +' has connected')
```

Για κάθε σύνδεση με το Raspberry ενεργοποιείται ένας δίαυλος επικοινωνίας με όνομα mySensorBus

Το raspberry στέλνει τις εξής μεταβλητές:

- username: του χρήστη το αναγνωριστικό-μοναδικό
- raspid: του raspberry το αναγνωριστικό-μοναδικό που ανήκει σε κάποιον χρήστη
- socketid: το id του websocket μεταξύ raspberry και server
- sensor1: η τιμή του αισθητήρα θερμοκρασίας
- sensor2: η τιμή του αισθητήρα υγρασίας

```
socket.on('mySensorBus',function(username,raspid,socketid,sensor1,sensor2)
```

Για μήνυμα στην σελίδα για ενημέρωση χρήστη

```
io.emit('info', 'Rasp connected');
```

Authorization - Εξουσιοδότηση για να προχωρήσουμε στον έλεγχο του συστήματος.

Μπορεί κάποιος κακόβουλος να έχει στην διάθεση του τον κώδικα του client onrasp.js τότε μπορεί να ανοίξει με ευκολία αυτό το socket και να στείλει δεδομένα.

Εμείς όμως θα ελέξουμε μέσω των credentials που θα στείλει (username,raspid,socketid) ότι είναι ο ίδιος, ότι η επικοινωνία raspberry και server είναι μοναδική και ασφαλής

```
user = users.find(x => x.username === username && x.raspid === raspid)
        if(user === undefined)
        {
                console.log('Forbidden');
  //Δεν βρέθηκε ο χρήστης στον πίνακα που να του ανήκει αυτό το rasp
                io.emit(socketid, 'Rasp authorization failed');
        }
        else
        {
```

Οταν ο χρήστης συνδέεται στην ιστοσελίδα (dashboard) καλείται η *socket.emit(socket.id, '<%= username %>');*

και στέλνει στο κανάλι socket.id το username του χρήστη.

Ετσι θα υπάρχει μοναδική αντιστοίχηση user και socket.id μεταξύ site-server.

Βέβαια για κάθε επιτυχημένη σύνδεση μεταξύ rasp-server και site-server υπάρχει αυτή η συνάρτηση *[socket.on(socket.id),...]* που εξυπηρετεί τα μοναδικά κανάλια των sockets.

Εδώ όμως χρειαζόμαστε την μοναδική σύνδεση site-server ώστε να μας στείλει το username του χρήστη, να τον βρούμε στον πίνακα users και να βάλουμε στην μεταβλητή socketidpage το μοναδικό socket.id της σύνδεσης.

Με αυτό τον τρόπο μπορούμε πλέον από οποιαδήποτε γραμμή του κώδικα να στέλνουμε μεσω μιας αντίστοιχης *io.emit(socketidforpage, 'temp', sensor1 + ' C');* τιμές στην ιστοσελίδα.

```
  io.emit(socketidforpage, 'temp', sensor1 + ' C');
        //τιμές στην ιστοσελίδα
     socket.on(socket.id, function (username) {
       users.find(x => x.username === username).socketidpage =
socket.id;
     });
```

Έλεγχος Αν ο χρήστης είναι συνδεδεμένος στην ιστοσελίδα τότε μέσω της μεθόδου

socket.on(socket.id, function (username) {...

η μεταβλητή socketidpage του user έχει τροποποιηθεί

```
if(user.socketidpage != '')
{
//Ειδικό format για την αποστολή δεδομένων στην ιστοσελίδα
let socketidforpage = 'k'+user.socketidpage;
//Το πρόσθετο 'k' προστίθεται μπροστά στο id της σύνδεσης για να μην υπάρχει
//μπέρδεμα με τα κανάλια sockets μεταξύ rasp-server

//Αποστολή των τιμών των αισθητήρων στην ιστοσελίδα dashboard
io.emit(socketidforpage, 'temp', sensor1 + ' C');
io.emit(socketidforpage, 'hum', sensor2 + ' %');
}
else
{
//Το raspberry είναι συνδεδεμένο με τον server αλλά ο χρήστης δεν είναι συνδεδεμένος
//μέσω της ιστοσελίδας ώστε να επικοινωνήσει με το rasp
}
```

Με αυτό το socket ο server επικοινωνεί με τον χρήστη που είναι συνδεδεμένος μέσω της ιστοσελίδας και στέλνει δεδομένα μέσω του Socket raspid στο rasp.

```
    socket.on('led', function (raspidfrompage) {
  //Ο χρήστης που είναι συνδεδεμένος στην ιστοσελίδα στέλνει εδώ στον server το μοναδικό
  //socket.id(raspidfrompage) της μεταξύ τους σύνδεσης.
  //Ψάχνουμε στον πίνακα users αν κάποιος χρήστης έχει αυτό το socket.id(socketidpage)
  //Αν ναι, τότε παίρνουμε το raspid που είναι το μοναδικό socket.id που μας έχει δώσει το rasp του user όταν συνδέθηκε με τον server
  let raspid = users.find(x => x.socketidpage === raspidfrompage).raspid;

  //Και στέλνουμε μια τιμή στο rasp του χρήστη (μοναδικά)
                if(raspid != undefined)
                  io.emit(raspid, 'ON');
    });
```

Οταν το rasp διακόψει την επικοινωνία με τον server τότε καλείται αυτή η συνάρτηση η οποία στέλνει το μήνυμα της αποσύνδεσης.

```
    socket.on('disconnect', function(msg){
            console.log('Disconnected');
```

```
        });
    });
```

Στη συνέχεια παρουσιάζεται η πρώτη σελίδα που βλέπει ο χρήστης όταν δεν είναι συνδεδεμένος.

6.1.2 Επεξήγηση λειτουργίας Login.html

Στο πεδίο <head> </head> φαίνεται ότι χρησιμοποιείται το bootstrap.css για να κάνει τα γραφικά πιο όμορφα και responsive.

```
<html>
    <head>
        <title>Patrol System - Login</title>
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4
Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
    </head>
```

Εδώ φαίνεται η navigation bar που τοθετούνται τα links για την καθοδήγηση του χρήστη.

```
        <nav class="navbar navbar-default">
            <div class="container-fluid">
```

```
            <div class="collapse navbar-collapse" id="bs-example-navbar1">
                <ul class="nav navbar-nav">
                    <li><a href="/">Home</a></li>
                    <li><a href="/dashboard">Dashboard</a></li>
                </ul>

                <ul class="nav navbar-nav navbar-right">
                    <li><a href="/login">Log In</a></li>
                    <li><a href="/logout">Log Out</a></li>
                </ul>
            </div><!-- /.navbar-collapse -->
        </div><!-- /.container-fluid -->
    </nav>
```

Σε αυτό το σημείο παρουσιάζεται το κομμάτι του κώδικα με τα πεδία που πρέπει να συμπληρώσει ο χρήστης για να συνδεθέι στο server.

```
<form action="/login" method="post">
<div>
<label>Username:</label>
<input type="text" name="username"/>
</div>
<div>
<label>Password:</label>
<input type="password" name="password"/>
</div>
<div>
<input class="btn btn-primary" type="submit" value="Log In"/>
```

```
</div>
</form>
```

6.1.3 Επεξήγηση λειτουργίας Dashboard.html

Στο πεδίο <head> </head> φαίνεται ότι χρησιμοποιείται το bootstrap.css για να κάνει τα γραφικά πιο όμορφα και responsive. Χρησιμοποιούνται οι βιβλιοθήκες jquery για βελτιωμένη χρήση της javascript.
Το πιο σημαντικό κομμάτι είναι η βιβλιοθήκη socket.io.js ώστε η ιστοσελίδα να μπορεί να χρησιμοποιήσει τα WebSockets που παρέχει το socket.io

```
<html>
<head>
<title>Patrol System - Dashboard</title>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68v
bdEjh4u" crossorigin="anonymous">

<script src="https://code.jquery.com/jquery-1.11.2.min.js"></script>
<script src="https://code.jquery.com/jquery-migrate-1.2.1.min.js"></script>
```

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.0.3/socket.io.js"></script
>
</head>
```

Στο παρακάτω κομμάτι φαίνονται τα πεδία που εμφανίζονται οι πληροφορίες, η θερμοκρασία και η υγρασία.

```
<div class="container row">
        <span id="info"> </span>
<div class="sss">
        <span>Θερμοκρασία: </span> <span id="temp"> </span>
</div>
<div class="sss">
        <span>Υγρασία: </span> <span id="hum"> </span>
</div>
</br>
        <button id="led">Στέλνω σήμα στο Raspberry</button>
</div>
```

Στο τμήμα αυτό χρησιμοποιείται η javascript-jquery για να δημιουργηθεί μια socket σύνδεση με τον server στην θύρα 8080. Όπως φαίνεται χρησιμοποιούμε μια το localhost και μια το server που μας φιλοξενεί.

Στο s1 φαίνεται με ποιο τρόπο στέλνει η ιστοσελίδα ένα σήμα-δεδομένο στον Server μέσω της εντολής socket.emit('led', socket.id);

Στο επόμενο κομμάτι s2 χρησιμοποιείται μια συνάρτηση που δημιουργεί καθυστέρηση ενός δευτερολέπτου για να προλάβει το socket.io μέσω του server να δημιουργήσει μοναδικό socket.id της σύνδεσης τους. Αυτό χρησιμοποιεί για να στείλει στον Server το username για να γίνει μοναδική η σύνδεση μεταξύ user και socket.id.

Στο κομμάτι κώδικα s3 βρίσκεται το μοναδικό socket ανα user και server για να λαμβάνει η σελίδα-συνδεδεμένος user τις τιμές των αισθητήρων. Στη συνέχεια τα αναπαριστούνε σε html μέσω jquery.

```
<script>
$(function () {
//var socket = io('http://papaxristos.eparko.com:8080');
var socket = io('http://localhost:8080');

//s1
//Για να στείλουμε στον server μέσω του μοναδικού καναλιού socket.id
$( "#led" ).click(function() {
socket.emit('led', 'on');
});

//s2
setTimeout(func, 1000);
function func() {
//Οταν ο χρήστης συνδέεται στην ιστοσελίδα (dashboard) καλείται η
```

```
//και στέλνει στο κανάλι socket.id το username του χρήστη
//έτσι θα υπάρχει μοναδική αντιστοίχιση user και socket.id
//μεταξύ site-server
socket.emit(socket.id, '<%= username %>');


//s3
//Για να λάβουμε από τον Server τιμές
socket.on('k'+socket.id, function(type,msg){
        if(type == 'temp') $('#temp').text(msg);
        if(type == 'hum') $('#hum').text(msg);
        if(type == 'info') $('#info').text(msg);
      });
                }
});
</script>
```

6.1.4 Επεξήγηση λειτουργίας Raspberry μέσω του κώδικα Onrasps.js

Το αρχείο αυτό βρίσκεται σε κάθε raspberry. Το μόνο που πρέπει να αλλάξει ο χρήστης είναι το username και το μοναδικά ορισμένο αναγνωριστικό του Raspberry με στοιχεία που βρίσκονται στον Server.

username = XXXX

raspid = ABABABABA;

Η πιο σημαντική βιβλιοθήκη για να επικοινωνήσει ένας κώδικας γραμμένος σε node.js με τον server είναι η 'socket.io-client'.

```
var io = require('socket.io-client');
```

Η Βιβλιοθήκη για να πραγματοποιηθεί η ανάγνωση των τιμών από τον αισθητήρα θερμοκρασίας DHT11.

```
var sensor = require('node-dht-sensor');
```

Η Βιβλιοθήκη για να πραγματοποιηθεί το read/write στους ακροδέκτες του Raspberry.

```
var Gpio = require('onoff').Gpio; //include onoff to interact with the GPIO
var LED = new Gpio(4, 'out'); //use GPIO pin 4, and specify that it is output
```

Όπως φαίνεται χρησιμοποιούμε μια το localhost και μια το server που τον φιλοξενεί.

```
var serverurl = "http://localhost:8080";
//var serverurl = "http://papaxristos.eparko.com:8080";
```

Συνδέεται με τον Server

```
var socket = io.connect(serverurl);
```

Όταν η σύνδεση με τον Server είναι επιτυχής τότε καλείται αυτή η συνάρτηση.
Ολόκληρος ο κώδικας βρίσκεται μέσα σε αυτή την συνάρτηση και όλες οι συμπεριλαμβανόμενες συναρτήσεις-μέθοδοι εκτελούνται μόνο αν η σύνδεση είναι επιτυχής.

```
socket.on('connect', function () {
    console.log('Connected to Patrol Server on ' + serverurl);
```

Εγώ τρέχω σε ένα raspberry και πρέπει να έχω μοναδικό username και raspid

```
username = 'xristos'
raspid = 'EeBJ60JQzy';
```

Αυτό είναι το μοναδικό socket id όταν συνδέομαι με τον Server
Αυτό το ξέρει και ο client-raspberry και ο server

```
console.log(socket.id);
```

Η παρακάτω συνάρτηση 'ακούει' από τον server μόνο στο κανάλι raspid που είναι μοναδικό για κάθε χρήστη μέσω του πίνακα users. Με αυτό το Socket επικοινωνεί το Raspberry με τον Server για να δεκτεί ένα σήμα ώστε να ενεργοποιήσει έναν ακροδέκτη εξόδου

και για παράδειγμα να ανάψει ένα Led ή να ενεργοποιήσει ένα ρελέ.

```
socket.on(raspid, function(msg){
    LED.writeSync(1); //set pin state to 1 (turn LED on)
    console.log('Led::', msg);
  });
```

Με αυτήν την μέθοδο-συνάρτηση τρέχουμε ένα συγκεκριμένο κομμάτι κώδικα επαναληπτικά σε χρονικό διάστημα που ορίζουμε εμείς.
Με αυτόν τον τρόπο διαβάζουμε τις τιμές των σισθητήρων περιοδικά.

```
var myInt = setInterval(function () {
```

Θέτω αρχικές τιμές ίσες με 1000
που δεν μπορούν ισχύουν ώστε να καταλάβει ο server ότι
δεν δουλεύει σωστά το αισθητήριο

```
temperature =1000;
humidity =1000;
```

Ανάγνωση θερμοκρασίας και υγρασίας από τον αισθητήρα DHT11
Διαβάζει τον αισθητήρα DTH11 από τον ακροδέκτη 4

```
sensor.read(11, 4, function(err, temp, hum) {
```

Αν ο αισθητήρας δουλεύει σωστά τότε μπορώ να αποθηκεύσω τις τιμές

```
  if (!err) {
          temperature = temp.toFixed(1);
          humidity = hum.toFixed(1);
     console.log('temp: ' + temperature + 'ºC, ' +
        'humidity: ' + humidity + '%'
     );
  }
});

console.log(socket.id + ': I measure Temp: ' + temperature + 'ºC, ' + 'Humidity: '
+ humidity + '%');
```

Στέλνει τα δεδομένα στον Server μέσω του Socket με όνομα 'mySensorBus'

```
socket.emit('mySensorBus', username, raspid, socket.id, temperature, humidity);
}, 5000);  //Κάθε 5 δευτερόλεπτα
});
```

Κεφάλαιο 7ο

Συμπεράσματα – Θέματα για μελλοντική έρευνα

Στη εργασία αυτή υλοποιήθηκε ένα σύστημα αμφίδρομης επικοινωνίας μεταξύ διακομιστή και πελάτη μέσω χρήσης της τεχνολογίας Socket.io. Χρησιμοποιήθηκε αυτή η βιβλιοθήκη για την υλοποίηση WebSockets γιατί είναι εύκολη στην χρήση σε html και κώδικα σε script.

Υλοποιήθηκε αμφίδρομη επικοινωνία, μεταξύ ενός εξουσιοδοτημένου χρήστη που επικοινωνεί μέσω του περιηγητή ιστού (browser) του με τον διακομιστή και του Raspberry Pi που επικοινωνεί με εξουσιοδότηση με τον ίδιο διακομιστή. Είναι πολύ σημαντικό το θέμα ασφάλειας και αυτό επιτεύχθηκε με δύο τρόπους, ο ένας είναι μέσω εξουσιοδοτημένης πρόσβασης και ο άλλος μέσω μοναδικών sockets που παρέχει το sockets.io κατά την σύνδεση του χρήστη.

Ένα θέμα για μελλοντική έρευνα πάνω σε θέματα ασφάλειας είναι να υλοποιηθεί σε Server που είναι SSL secure. Για αυτό χρειάζεται να αλλαχθεί η port σε 8443 και να αγοραστούν certifications.

Αν ο Server είχε τοποθετηθεί στο Raspberry τότε θα χρειαζόταν μόνο δύο modules, το app.js του server και τα αρχεία .html για την πλοήγηση του πελάτη. Αλλά ο χρήστης έπρεπε να γνωρίζει πάντα την ip που βρίσκεται το Raspberry. Επίσης, δεν θα μπορούσε να υπάρχει κοινή βάση με τους χρήστες και τα id των Raspberry ενώ δεν θα μπορούσαμε να παρακολουθήσουμε τόσο εύκολα πολλά Raspberry ταυτόχρονα.

Το Raspberry μπορεί να τοποθετηθεί οπουδήποτε αρκεί να διαθέτει πρόσβαση στο διαδίκτυο για να επικοινωνήσει με τον server. Οι αισθητήρες που τοποθετήθηκαν είναι ενδεικτικά και ο αναγνώστης που θέλει να επεκτείνει το έργο μπορεί να τοποθετήσει οτιδήποτε είναι προσαρμόσιμο στους ακροδέκτες του Raspberry μέσω της πληθώρας από έργα που είναι δημοσιευμένα στο διαδίκτυο.

Μελλοντικά θα μπορούσε να δημιουργηθεί μια ιστοσελίδα που να αναγνωρίζει αυτόματα τα πολλά Raspberry που συνδέονται ταυτόχρονα στον διακομιστή και ανήκουν σε ένα χρήστη (συνδεδεμένο). Επίσης θα μπορούσε να βελτιωθεί η ιστοσελίδα και να γίνει καλύτερη σε θέματα responsive για να μπορεί να είναι εύκολα προσβάσιμη από σχεδόν όλες τις κινητές συσκευές,

Βιβλιογραφία

[1] https://en.wikipedia.org/wiki/WebSocket

[2] https://www.rfc-editor.org/info/rfc6455

[3] https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.3.0/com.ibm.cics.ts.internet.doc/topics/dfhtl_conintro.html

[4] https://en.wikipedia.org/wiki/Comet_(programming)

[5] https://www.w3schools.com/html/html5_intro.asp

[6] https://www.w3.org/

[7] https://en.wikipedia.org/wiki/WHATWG

[8] Ian Fette; Alexey Melnikov (December 2011). "Protocol Overview". RFC 6455 The WebSocket Protocol

[9] https://en.wikipedia.org/wiki/Base64

[10] https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_server

[11] https://github.com/eidheim/Simple-WebSocket-Server

[12] http://socketo.me/

[13] https://github.com/Pithikos/python-websocket-server

[14] https://www.baeldung.com/java-websockets

[15] https://en.wikipedia.org/wiki/Node.js

[16] "Node.js Foundation - Node.js". Retrieved 4 July 2015.

[17] "Linux Foundation Collaborative Projects". Retrieved 4 July 2015.

[18] Harris, Amber (April 1, 2012). "The Birth of Node: Where Did it Come From? Creator Ryan Dahl Shares the History". Devops Angle. Retrieved 26 October 2013

[19] https://www.npmjs.com/

[20] https://nodejs.org

[21] https://socket.io/

[22] https://en.wikipedia.org/wiki/Socket.IO

[23] https://en.wikipedia.org/wiki/Raspberry_Pi

[24] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[25] https://www.raspberrypi.org/downloads/

[26] https://tutorials-raspberrypi.com/raspberry-pi-sensors-overview-50-important-components/

[27] https://en.wikipedia.org/wiki/Arduino

[28] https://www.espressif.com/en/products/hardware/esp32/overview

[29] http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-the-raspberry-pi/

Παράρτημα Α

Στο παράρτημα αυτό παρουσιάζονται οδηγίες εγκατάστασης και ρύθμισης των εργαλείων που χρησιμοποιήθηκαν.

Εγκατάσταση του nodejs

https://nodejs.org/en/

Test το node

1. Δημιούργησε έναν φάκελο και δημιούργησε το αρχείο hello.js

2. Άνοιξε το με τον κειμενογράφο ή με το πολύ χρήσιμο εργαλείο επεξεργασίας κειμένου σχετικού με γλώσσες προγραμματισμού, το notepad++ (https://notepad-plus-plus.org/download/v7.5.9.html)

3. Αντέγραψε τον παρακάτω κώδικα στο αρχείο

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(2019, "127.0.0.1");
console.log('Server running at http://127.0.0.1:2019/');
```

4. Αποθήκευσε το αρχείο

5. Αφού είσαι μέσα στον φάκελο κράτα πατημένο το shift και πάτα το δεξί κλίκ του ποντικιού. Τότε σε εμφανίζει να

ανοίξεις το CMD, αλλιώς terminal, αλλιώς Powershell (Windows 10.

6. Γράφεις

   node hello.js

7. Έχει εκκινηθεί ο server για το localhost στην port:2019 για το node.

8. Άνοιξε το chrome και πάτα το
   http://127.0.0.1:2019/
   μπορείς να δεις τα αποτελέσματα.

9. Για να σταματήσεις τον server πάτα CTRL+C
   Ή κλείσε το terminal window

10. Αν θέλετε να το τοποθετήσε το node application σας σε έναν shared hosting server θα πρέπει να μεταφέρετε όλα τα αρχεία σας σε αυτόν και να ονομάσετε το application app.js (επίσης να μην είναι ενεργοποιημένο το proxy mode)

Παράρτημα Β

Κώδικες

App.js

```
var bodyParser = require('body-parser');
var cookieParser = require('cookie-parser');

var app = require('express')(),
    http = require("http").createServer(app),
    io = require("socket.io")(http),
    session = require("express-session")({
        key: 'user_sid',
        secret: 'EeBJ60JQzyCyRFfQaVUqTyRGfQvDUt',
        resave: false,
        saveUninitialized: false
    });

//Αρχικοποίηση body-parser για να περνάνε οι παράμετροι από
την είσοδο στο req.body
app.use(bodyParser.urlencoded({extended: true}));

//Αρχικοποίηση του cookie-parser για να επιτρέψει την
πρόσβαση στα cookies που είναι αποθηκευμένα στο πρόγραμμα
περιήγησης.
app.use(cookieParser());

// Χρήση session
app.use(session);

//θέτουμε την μηχανή για view στο ejs
```

```
app.set('view engine', 'ejs');
//============================================================
==================

//Οι χρήστες με μοναδικό rasp id ο καθένας
let users = [
    {id: 20, username: 'xristos', pass: '1234', raspid:
'EeBJ60JQzy', socketidpage: ''},
    {id: 21, username: 'kostas', pass: '1234', raspid:
'CyRFfQaVUq', socketidpage: ''},
    {id: 22, username: 'maria', pass: '1234', raspid:
'TyRGfQvDUt', socketidpage: ''}
];

// Αυτό το middleware - μεσαίο λογισμικό θα ελέγξει εάν το
cookie του χρήστη είναι ακόμα αποθηκευμένο στο πρόγραμμα
περιήγησης και ο χρήστης δεν έχει οριστεί ακόμα. Τότε
αυτόματα τον αποσυνδέει.
// Αυτό συμβαίνει συνήθως όταν σταματήσει ο server μετά την
είσοδο-login του χρήστη, τότε το cookie παραμένει
αποθηκευμένο στο πρόγραμμα περιήγησης.
app.use((req, res, next) => {
    if (req.cookies.user_sid && !req.session.username) {
        res.clearCookie('user_sid');
    }
    next();
});

// Αυτή η middleware συνάρτηση ελέγχει για συνδεδεμένους
χρήστες
var sessionChecker = (req, res, next) => {
    if (req.session.username && req.cookies.user_sid) {
        res.redirect('/dashboard');
    } else {
        next();
    }
};

// route για Home-Page
app.get('/', sessionChecker, (req, res) => {
    res.redirect('/login');
});

// route για user Login
app.route('/login')
    .get(sessionChecker, (req, res) => {
        res.sendFile(__dirname + '/login.html');
    })
```

```
    .post((req, res) => {

        //Αυτα τα λαμβάνουμε απο την Login.html page
        var username = req.body.username,
            password = req.body.password;

        //Ψάχνουμε αν τα credentials είναι σωστά
συγκρίνοντας με ένα πίνακα που έχουμε κάποια στοιχεία.
        //Θα μπορούσε να συνδεθεί με μια βάση δεδομένων
        loggeduser = users.find(x => x.username === username
&& x.pass === password)

        //Αν δεν βρεθεί ο χρήστης στον πίνακα τότε
επιστρέφει στην Login σελίδα
        if (loggeduser === undefined) {
            res.redirect('/login');
        }
        else {
            //Αν βρεθεί ο χρήστης στον πίνακα τότε
επιστρέφει στην dashboard σελίδα
            //και σώζουμε σε Session(μοναδικό για κάθε
χρήστη που συνδέεται το id και το username του χρήστη)

            //Session για να χρησιμοποιηθούν τα δεδομένα στα
Cookies και σελίδες html
            req.session.userid = loggeduser.id;
            req.session.username = loggeduser.username;

            res.redirect('/dashboard');
        }
    });

// route για το dashboard
app.get('/dashboard', (req, res) => {
    //Κάθε φορά που μπαίνει κάποιος χρήστης σε αυτή την
σελίδα πρέπει να ελέξουμε αν ο χρήστης είναι logged.
    //Οπότε αν υπάρχει το Session του user και το cookie του
user_id μόνο τότε θα
    //μπορεί να μπαίνει σε κάθε σελίδα.
    //αλλιώς θα πηγαίνει στη σελίδα login
    if (req.session.username && req.cookies.user_sid) {

        //Στέλνουμε το μοναδικό username στην ιστοσελίδα
dashboard
        //για να χρησιμοποιηθεί στην αποστολή του socket.id
της σύνδεσης στον server
        //και να γίνει μοναδική αντιστοίχηση του με το
username του user
```

```
        //socket.emit(socket.id, '<%= username %>');
        res.render('dashboard', {username:
req.session.username});
    } else {
        res.redirect('/login');
    }
});

// route για user logout
app.get('/logout', (req, res) => {

    //Αν υπάρχει το Session του user και το cookie του
user_id
    //τότε στο logout θα καθαρίσουμε το cookie και με αυτό
καθαρίζει και το αντίστοιχο session
    if (req.session.username && req.cookies.user_sid) {
        res.clearCookie('user_sid');
        res.redirect('/');
    } else {
        //Αν δεν ήταν logged τότε απλά θα πάει στη σελίδα
του login
        res.redirect('/login');
    }
});

// route για τον χειρισμό των 404 requests(unavailable
routes)
app.use(function (req, res, next) {
    res.status(404).send("Δεν υπάρχει αυτό το path")
});
//============================================================
==================
//Ο Server ενεργοποιείται στην port 8080
http.listen(8080, function () {
    console.log('Patrol System');
});

//Όταν κάποιος client (σελίδα ή rasp) συνδεθεί στον server
τότε καλείται αυτή η συνάρτηση
io.on('connection', function (socket) {

    //Όταν κάποιος client (σελίδα ή rasp) συνδεθεί στον
server
    //αποκτάει Μοναδικό id, το socket.id
    console.log('Server: a client ' + socket.id + ' has
connected');

    //Για κάθε σύνδεση με το Raspberry ενεργοποιείται ένας
```

```
δίαυλος επικοινωνίας με όνομα mySensorBus
    //Το raspberry στέλνει τις εξής μεταβλητές:
    //username: του χρήστη το αναγνωριστικό-μοναδικό
    //raspid: του raspberry το αναγνωριστικό-μοναδικό που
ανήκει σε κάποιον χρήστη
    //socketid: το id του websocket μεταξύ raspberry και
server
    //sensor1: η τιμή του αισθητήρα θερμοκρασίας
    //sensor2: η τιμή του αισθητήρα υγρασίας
    socket.on('mySensorBus', function (username, raspid,
socketid, sensor1, sensor2) {

        //Authorization - Εξουσιοδότηση για να προχωρήσουμε
στον έλεγχο του συστήματος
        //Μπορεί κάποιος κακόβουλος να έχει στην διάθεση του
το κώδικα του client onrasp.js
        //τότε μπορεί να ανοίξει με ευκολία αυτό το socket
και να στείλει δεδομένα
        //
        //Εμείς όμως θα ελέξουμε μέσω των credentials που θα
στείλει (username,raspid,socketid)
        //ότι είναι ο ίδιος
        //-->>>
        user = users.find(x => x.username === username &&
x.raspid === raspid);
        if (user === undefined) {
            //Δεν βρέθηκε ο χρήστης στον πίνακα που να του
ανήκει αυτό το rasp
            io.emit(socketid, 'Rasp authorization failed');
        }
        else {
            //Αν ο χρήστης είναι συνδεδεμένος στην
ιστοσελίδα τότε μέσω της μεθόδου
            //socket.on(socket.id, function (username) {
            //η μεταβλητή socketidpage του user έχει
τροποποιηθεί
            if (user.socketidpage != '') {
                //Ειδικό format για την αποστολή δεδομένων
στην ιστοσελίδα
                let socketidforpage = 'k' +
user.socketidpage;
                //Το πρόσθετο 'k' προστίθεται μπροστά στο id
της σύνδεσης για να μην υπάρχει
                //μπέρδεμα με τα κανάλια sockets μεταξύ
rasp-server

                //Αποστολή των τιμών των αισθητήρων στην
ιστοσελίδα dashboard
```

```
                io.emit(socketidforpage, 'temp', sensor1 + '
C');
                io.emit(socketidforpage, 'hum', sensor2 + '
%');
            }
            else {
                //Το raspberry είναι συνδεδεμένο με τον
server αλλά ο χρήστης δεν είναι συνδεδεμένος
                //μέσω της ιστοσελίδας ώστε να επικοινωνήσει
με το rasp
            }
        }
    });

    //Με αυτό το socket ο server επικοινωνεί με τον χρήστη
που είναι συνδεδεμένος μέσω της
    //της ιστοσελίδας. Και στέλνει δεδομένα μέσω του Socket
raspid στο rasp.
    socket.on('led', function (raspidfrompage) {
        //Ο χρήστης που είναι συνδεδεμένος στην ιστοσελίδα
στέλνει εδώ στον server το μοναδικό
        //socket.id(raspidfrompage) της μεταξύ τους
σύνδεσης.
        //Ψάχνουμε στον πίνακα users αν κάποιος χρήστης έχει
αυτό το socket.id(socketidpage)
        //Αν ναι, τότε παίρνουμε το raspid που είναι το
μοναδικό socket.id που μας έχει δώσει το rasp
        //του user όταν συνδέθηκε με τον server
        let raspid = users.find(x => x.socketidpage ===
raspidfrompage).raspid;

        //Και στέλνουμε μια τιμή στο rasp του χρήστη
(μοναδικά)
        if (raspid != undefined)
            io.emit(raspid, 'ON');
    });

    //Οταν το rasp διακόψει την επικοινωνία με τον server
τότε καλείται αυτή η συνάρτηση
    socket.on('disconnect', function (msg) {
        console.log('Disconnected');
    });

    //Οταν ο χρήστης συνδέεται στην ιστοσελίδα (dashboard)
καλείται η
    //socket.emit(socket.id, '<%= username %>');
    //και στέλνει στο κανάλι socket.id το username του
χρήστη.
```

```
    //Ετσι θα υπάρχει μοναδική αντιστοίχηση user και
socket.id μεταξύ site-server.
    //Βέβαια για κάθε επιτυχημένη σύνδεση μεταξύ rasp-server
και site-server υπάρχει
    //αυτή η συνάρτηση [socket.on(socket.id),...] που
εξυπηρετεί τα μοναδικά κανάλια των sockets.
    //Εδώ όμως χρειαζόμαστε την μοναδική σύνδεση site-server
ώστε να μας στείλει
    //το username του χρήστη, να τον βρούμε στον πίνακα
users και να βάλουμε στην μεταβλητή socketidpage
    //το μοναδικό socket.id της σύνδεσης.
    //Με αυτό τον τρόπο μπορούμε πλέον από οποιαδήποτε
γραμμή του κώδικα να στέλνουμε
    //μεσω  μιας αντίστοιχης io.emit(socketidforpage,
'temp', sensor1 + ' C');
    //τιμές στην ιστοσελίδα
    socket.on(socket.id, function (username) {
        users.find(x => x.username ===
username).socketidpage = socket.id;
    });

});
```

Onrasp.js

```
//
var io = require('socket.io-client');

//var sensor = require('node-dht-sensor');

//var Gpio = require('onoff').Gpio; //include onoff to
interact with the GPIO
//var LED = new Gpio(4, 'out'); //use GPIO pin 4, and
specify that it is output

//
var serverurl = "http://localhost:8080";
//var serverurl = "http://papaxristos.eparko.com:8080";

//
//var socket = io.connect("http://rasp.eparko.com:8080",
{//reconnect: true , secure: true});
var socket = io.connect(serverurl);

console.log('I am trying connect to Patrol Server on ' +
serverurl);
```

```
//
socket.on('connect', function () {
    console.log('Connected to Patrol Server on ' +
serverurl);

    //Για πειραματικά δεδομένα
    //{id:20, username:'xristos', pass:'1234',
raspid:'EeBJ60JQzy'},
    //{id:21, username:'kostas', pass:'1234', raspid:
'CyRFfQaVUq'}
    //{id:22, username:'maria', pass:'1234', raspid:
'TyRGfQvDUt'}

    //Εγώ τρέχω σε ένα raspberry και πρέπει να έχω μοναδικό
username και raspid
    username = 'xristos'
    raspid = 'EeBJ60JQzy';

    //Αυτό είναι το μοναδικό socket id όταν συνδέομαι με τον
Server
    //Αυτό το ξέρει και ο client-raspberry και ο server
    //console.log(socket.id);


    //Αυτή η συνάρτηση 'ακούει' από τον server μόνο στο
κανάλι raspid που είναι μοναδικό
    //για κάθε χρήστη μέσω του πίνακα users.
    socket.on(raspid, function (msg) {

        //LED.writeSync(1); //set pin state to 1 (turn LED
on)

        console.log('Led::', msg);
    });

    var myInt = setInterval(function () {

        //Τυπικές τιμές σε περίπτωση που θέλουμε να
στείλουμε όταν δεν υπάρχει raspberry
        temperature = 22;
        humidity = 65;

        //Ανάγνωση θερμοκρασίας και υγρασίας από τον
αισθητήρα DHT11
        /*
         sensor.read(11, 4, function(err, temp, hum) {
```

```
        //Αν ο αισθητήρας δουλεύει σωστά τότε μπορώ να
αποθηκεύσω τις τιμές
        if (!err) {
        temperature = temp.toFixed(1);
        humidity = hum.toFixed(1);
        console.log('temp: ' + temperature + '°C, ' +
        'humidity: ' + humidity + '%'
        );
        }
        });
        */
        console.log(socket.id + ': I measure Temp: ' +
temperature + '°C, ' + 'Humidity: ' + humidity + '%');

        socket.emit('mySensorBus', username, raspid,
socket.id, temperature, humidity);

        //socket.emit('mybusHum', Math.random());
    }, 5000);
});
```

Login.html

```
<html>
<head>
    <title>Patrol System - Login</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bo
otstrap.min.css"
          integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdE
jh4u" crossorigin="anonymous">
</head>
<body class="container">
<div class="page-header">
    <h1>Patrol System - Login</h1>
</div>

<nav class="navbar navbar-default">
    <div class="container-fluid">
        <!-- Collect the nav links, forms, and other content
for toggling -->
        <div class="collapse navbar-collapse" id="bs-
example-navbar-collapse-1">
            <ul class="nav navbar-nav">
                <li><a href="/">Home</a></li>
                <li><a href="/dashboard">Dashboard</a></li>
```

```
            </ul>

            <ul class="nav navbar-nav navbar-right">
                <li><a href="/login">Log In</a></li>
                <li><a href="/logout">Log Out</a></li>
            </ul>
        </div><!-- /.navbar-collapse -->
    </div><!-- /.container-fluid -->
</nav>

<div class="container row">
    <div class="jumbotron col-sm-4 pull-center">
        <form action="/login" method="post">
            <div>
                <label>Username:</label>
                <input type="text" name="username"/>
            </div>
            <div>
                <label>Password:</label>
                <input type="password" name="password"/>
            </div>
            <div>
                <input class="btn btn-primary" type="submit"
value="Log In"/>
            </div>
        </form>
    </div>
</div>
</body>
</html>
```

Dashboard.ejs

```
<html>
    <head>
        <title>Patrol System - Dashboard</title>
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bo
otstrap.min.css" integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdE
jh4u" crossorigin="anonymous">

      <script src="https://code.jquery.com/jquery-
1.11.2.min.js"></script>
      <script src="https://code.jquery.com/jquery-migrate-
1.2.1.min.js"></script>
      <script
```

```
src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.0.3/
socket.io.js"></script>
    </head>
    <body class="container">
        <div class="page-header">
            <h1>Patrol System - Dashboard</h1>
        </div>

        <nav class="navbar navbar-default">
            <div class="container-fluid">
            <div class="navbar-header">
            <a class="navbar-brand" href="#">User: <%=
username %>     |   </a>
            </div>

                <!-- Collect the nav links, forms, and other
content for toggling -->
                <div class="collapse navbar-collapse"
id="bs-example-navbar-collapse-1">
                    <ul class="nav navbar-nav">
                        <li><a href="/">Home</a></li>
                        <li><a
href="/dashboard">Dashboard</a></li>
                    </ul>
                    <ul class="nav navbar-nav navbar-right">

                        <li><a href="/login">Log In</a></li>
                        <li><a href="/logout">Log
Out</a></li>
                    </ul>
                </div><!-- /.navbar-collapse -->
            </div><!-- /.container-fluid -->
        </nav>

        <div class="container row">

            <span id="info"> </span>

            <div class="sss">
            <span>Θερμοκρασία: </span> <span id="temp">
</span>
            </div>
            <div class="sss">
            <span>Υγρασία: </span> <span id="hum"> </span>
            </div>
            </br>
            <button id="led">Στέλνω σήμα στο
Raspberry</button>
```

```
        </div>


    <script>
     $(function () {

      //var socket =
io('http://papaxristos.eparko.com:8080');
     var socket = io('http://localhost:8080');

     //Για να στείλουμε στον server μέσω του μοναδικού
καναλιού socket.id
     $( "#led" ).click(function() {
        socket.emit('led', socket.id);
     });

     setTimeout(func, 1000);
     function func() {
     //Οταν ο χρήστης συνδέεται στην ιστοσελίδα (dashboard)
καλείται η
     //και στέλνει στο κανάλι socket.id το username του
χρήστη
     //έτσι θα υπάρχει μοναδική αντιστοίχιση user και
socket.id
     //μεταξύ site-server
         socket.emit(socket.id, '<%= username %>');

     //Για να λάβουμε από τον Server τιμές
     socket.on('k'+socket.id, function(type,msg){
          if(type == 'temp') $('#temp').text(msg);
        if(type == 'hum') $('#hum').text(msg);
        if(type == 'info') $('#info').text(msg);
       });

     }

     });
    </script>

    </body>
</html>
```