ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Έλεγχος των συσκευών ενός smarthome με φωνητικές εντολές

Βασάλος Μάρκος

Εισηγητής: Γεώργιος Διλιντάς, Καθηγητής

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Έλεγχος των συσκευών ενός smart home με φωνητικές εντολές

Μάρκος Βασάλος
Α.Μ. 40733

Εισηγητής:

Γεώργιος Διλιντάς, Καθηγητής

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Βασάλος Μάρκος με αριθμό μητρώου 40733, φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω από καρδιάς όλους τους φίλους και την οικογένειά μου που με στήριξαν και συνεχίζουν να με στηρίζουν σε κάθε μου εγχείρημα.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία έχει ως σκοπό την υλοποίηση ενός τρόπου ελέγχου των έξυπνων συσκευών ενός έξυπνου σπιτιού, στον οποίο οι εντολές θα δίνονται φωνητικά.

Η εργασία χωρίζεται σε 5 κεφάλαια. Στο 1ο γίνεται μια ιστορική αναδρομή του έξυπνου σπιτιού και του τρόπου που αναπτύσσεται μέχρι και σήμερα. Στο 2ο κεφάλαιο γίνεται παρουσίαση και επεξήγηση όλων των θεωρητικών γνώσεων, αρχών και γλωσσών προγραμματισμού που χρησιμοποιήθηκαν για να δημιουργηθεί το πρακτικό μέρος της πτυχιακής εργασίας. Στο 3ο κεφάλαιο αναλύονται τα χαρακτηριστικά της πλακέτας και του ρελέ ελέγχου των συσκευών , στο 4ο κεφάλαιο, εξηγείται αναλυτικά η πρακτική υλοποίηση της παρούσας πτυχιακής εργασίας ενώ στο 5ο κεφάλαιο παρατίθενται τα συμπεράσματα και κάποιες μελλοντικές επεκτάσεις.

ABSTRACT

The present thesis concerns the development of a way to control smarthome devices using the voice of the user.

Thesis is divided into 5 chapters. In the 1st one a chronology of smarthome is described. The 2nd chapter is dedicated to the theoretical background which presents and describes all the techniques and programming languages that were used in the application. In the 3rd chapter are analyzed the board and the relay module that were used. The 4th chapter, the implementation of the application of current thesis is analyzed, and lastly in the 5th chapter, the conclusions and some future extensions are presented.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Έξυπνο σπίτι, Smart Home, Φωνητικός Έλεγχος, Voice control, Αυτοματισμός Σπιτιού, Home Automation, Απομακρυσμένος Έλεγχος, Remote control

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

Η.Υ.: Ηλεκτρονικός Υπολογιστής

IP: Internet Protocol (address)

IoT: Internet of Things

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

Τη σημερινή εποχή, οι γρήγοροι ρυθμοί ζωής δημιουργούν την ανάγκη αναζήτησης όλο και περισσότερων τρόπων απλούστευσης τη ζωής, από περιττές και βαρετές διαδικασίες. Σημαντικότατο ρόλο σε αυτή τη προσπάθεια προσφέρει η ραγδαία εξέλιξη της τεχνολογίας, η οποία προσφέρει τα μέσα, ώστε να αυτοματοποιηθούν όλες εκείνες οι διαδικασίες που φαίνονται βαρετές ή "χωρίς νόημα".

Συγκεκριμένα τα τελευταία χρόνια όλες οι συσκευές, από τα κινητά τηλέφωνα μέχρι και οι οικιακές συσκευές γίνονται πιο "έξυπνες" και δεν απαιτούν πάντα την παρουσία ή την ενέργεια του χρήστη για να λειτουργήσουν. Αυτός ο κλάδος, ερευνητικά, αναφέρεται ως "Internet of Things" και τα αντικείμενα που επηρεάζονται από αυτόν, έχουν αρχίσει να εξαπλώνονται και να ξεπερνούν τις ηλεκτρονικές συσκευές, επιδρώντας και στον τρόπο ενδυμασίας ή και γενικότερα, στον ευρύτερο τρόπο ζωής.

Πώς λοιπόν θα μπορούσε να μείνει ανεπηρέαστο κάτι τόσο βασικό, όσο ο χώρος στον οποίο μένουμε; Το έξυπνο σπίτι ή βιβλιογραφικά αναφερόμενο ως "smarthome" είναι το σπίτι του μέλλοντος το οποίο αναμένεται να προσφέρει πλήρη έλεγχο όλων των οικιακών, και μη, συσκευών, για λογαριασμό του χρήστη, προσφέροντας έτσι εξοικονόμηση χρόνου, μεγαλύτερη άνεση, καλύτερο βιοτικό επίπεδο, ασφάλεια κλπ.[3]

1.1 Τί είναι το Έξυπνο Σπίτι (Smarthome);

Η αλήθεια είναι ότι υπάρχουν πολλοί και διαφορετικοί (σε κάποια σημεία τους) ορισμοί οι οποίοι αφορούν το έξυπνο σπίτι. Ένα από τα προβλήματα στο να δοθεί ένας καθολικός ορισμός, είναι το γεγονός ότι το έξυπνο σπίτι ακόμη εξελίσσεται και αποτελεί περισσότερο μία έννοια, παρά ένα χειροπιαστό αντικείμενο με συγκεκριμένα στάνταρ και υλοποίηση. Τα λεξικά Oxford Dictionaries δίνουν τον εξής ορισμό για το έξυπνο σπίτι: *«Ένα σπίτι εξοπλισμένο με φωτισμό, θέρμανση και ηλεκτρονικές συσκευές που μπορούν να ελεγχθούν ασύρματα από κινητό ή υπολογιστή»*. Ο συγκεκριμένος ορισμός απαιτεί οι έξυπνες συσκευές να είναι ουσιαστικά ασύρματα ελεγχόμενες, κάτι που δεν τις κάνει απαραίτητα "έξυπνες", για να μην αναφέρουμε και άλλα συστατικά που λείπουν, όπως η διαχείριση των συσκευών από το ίδιο το σπίτι. Στην επιστημονική βιβλιογραφία, συναντάμε τον ορισμό του Valtchev και Frankov (2002) οι οποίοι ορίζουν το έξυπνο σπίτι ως : *«Ένα σπίτι ή χώρος διαβίωσης ο οποίος περιέχει την τεχνολογία για να επιτρέψει σε συσκευές και συστήματα να ελέγχονται αυτόματα».* Εδώ πάλι δεν υπάρχει πουθενά η αναφορά του ασύρματου ελέγχου ή της επικοινωνίας. [4] Προσπαθώντας λοιπόν να δώσουμε έναν ορισμό αρκετά ολοκληρωμένο και επεξηγηματικό, θα μπορούσαμε να πούμε ότι : *«Το έξυπνο σπίτι είναι μία οικιακή εγκατάσταση όπου οικιακές και άλλων ειδών συσκευές μπορούν να ελεγχθούν αυτόματα, από απόσταση, από οποιοδήποτε μέρος του κόσμου με τη χρήση ενός κινητού τηλεφώνου ή άλλης δικτυωμένης συσκευής. Ένα έξυπνο σπίτι έχει τις συσκευές του διασυνδεδεμένες μέσω του διαδικτύου, και μπορεί να ελέγχει λειτουργίες όπως πρόσβαση ασφαλείας στο σπίτι, θερμοκρασία, φωτισμό κ.α»* .[1]

Τα σύγχρονα έξυπνα σπίτια, προσφέρουν στους κατοίκους τους πολλές διευκολύνσεις, συμπεριλαμβανομένων της εξοικονόμησης χρόνου και χρημάτων. Το πιο σημαντικό όμως στοιχείο, είναι ότι οι ίδιες περιφερειακές μονάδες μπορούν να χρησιμοποιούνται για πολλές χρήσεις και έτσι ο εκάστοτε χρήστης μπορεί να δημιουργήσει τα δικά του σενάρια χρήσης, πολλαπλασιάζοντας και εμπλουτίζοντας τα ήδη υπάρχοντα και προσαρμόζοντας το σύστημα με τέτοιο τρόπο, ώστε να εξυπηρετεί και να καλύπτει τις δικές του ξεχωριστές ανάγκες. Για παράδειγμα, οι αισθητήρες κίνησης μπορούν να αξιοποιηθούν για τον έλεγχο του φωτισμού και για τον έλεγχο της θερμοκρασίας, αλλά και για το σύστημα συναγερμού. Ένα ακόμα

παράδειγμα αποτελούν οι αισθητήρες διοξειδίου του άνθρακα, οι οποίοι μπορούν να χρησιμοποιούνται για την ρύθμιση του εξαερισμού εσωτερικών χώρων αλλά και για τον έλεγχο του συστήματος πυρόσβεσης, είτε αυτόνομα είτε σε συνάρτηση με άλλους αισθητήρες (πχ θερμοκρασίας). Τα σενάρια χρήσης συνήθως εξαρτώνται και από το διαθέσιμο τεχνολογικό εξοπλισμό και τις δυνατότητες της εκάστοτε εγκατάστασης, αλλά ουσιαστικά μπορούν να είναι άπειρα. Μερικά ενδεικτικά σενάρια χρήσης είναι τα παρακάτω:

- **Διαχείριση Φωτισμού:** Αυτοματοποιημένη αυξομείωση έντασης ή μήκους κύματος φωτισμού (αλλαγή χρώματος και έντασης αναλόγως τη μέρα ή τα καιρικά φαινόμενα), προ - ρυθμισμένα επίπεδα φωτισμού για τις ώρες τις μέρας και αυτοματοποιημένη ενεργοποίηση – απενεργοποίηση.
- **Διαχείριση Ασφαλείας:** Προστασία από εισβολείς και χρήση εξωτερικής κάμερας ή ακόμη και θυροτηλεφώνου, για παρακολούθηση του εσωτερικού ή εξωτερικού χώρου. Προστασία από τυχόν πυρκαγιά, πλημμύρα ή βραχυκύκλωμα με αισθητήρες θερμοκρασίας, υγρασίας και διοξειδίου. Ακόμη και σε περίπτωση πτώσης τάσεως, το έξυπνο σπίτι θα είναι σε θέση να απομονώσει τις ευαίσθητες συσκευές, ώστε να μην πάθουν ζημιά από τις αυξομειώσεις του δικτύου. Θα μπορεί να εκτελεί όλα τα απαραίτητα σενάρια ώστε να προστατευτεί μόνο του σε κάθε περίπτωση, χωρίς την ανθρώπινη παρουσία ή ενέργεια, με έγκαιρη ενημέρωση του ιδιοκτήτη.
- **Διαχείριση κλιματισμού, θέρμανσης, αερισμού κλπ.:** Είτε το έξυπνο σπίτι, είτε ο ιδιοκτήτης από απόσταση, θα είναι σε θέση να διαχειρίζονται πλήρως τις συνθήκες διαβίωσης, αυξομειώνοντας τη θερμοκρασία ή απενεργοποιώντας εντελώς τη θέρμανση ή ψύξη, όταν αυτές δεν χρειάζονται (π.χ. όταν θα ανοίγει κάποιο παράθυρο στο χώρο). Θα μπορούν ακόμα να ρυθμίζονται τυχόν περσίδες ή σκίαστρα, ώστε να υλοποιηθεί ο οικονομικότερος ή ο γρηγορότερος τρόπος υλοποίησης του επιθυμητού αποτελέσματος (π.χ. με τη ρύθμιση των σκίαστρων και άνοιγμα των παραθύρων για ρεύμα αέρα, η θερμοκρασία θα πέσει πιο αργά αλλά πιο οικονομικά στο χώρο, απ’ ό, τι με τη χρήση του κλιματισμού).
- **Διαχείριση ενεργειακού αποτυπώματος:** Το έξυπνο σπίτι του μέλλοντος θα είναι σε θέση να διαχειρίζεται πλήρως το ενεργειακό αποτύπωμα του σπιτιού και θα μπορεί να λειτουργεί όσο το δυνατόν πιο “οικονομικά”, τόσο

για τον ιδιοκτήτη, όσο και για το ίδιο το περιβάλλον. Ο χρήστης θα έχει τη δυνατότητα να παρακολουθεί πλήρως πόση ενέργεια ξόδεψε και σε ποιες συσκευές/λειτουργίες, έχοντας πολλαπλά οφέλη, τόσο οικονομικά, όσο και σε θέματα περιβαλλοντικής ευαισθητοποίησης. Τα μελλοντικά αυτά σπίτια θα έχουν και τρόπους να φτιάχνουν τη δική τους ενέργεια, είτε από εναλλακτικές μορφές (π.χ. ηλιακά πάνελ), είτε από νέες μεθόδους παραγωγής ενέργειας (π.χ. υδρογόνο ή ηλεκτρόλυση), οπότε θα μπορούν ιδανικά να λειτουργούν εντελώς αυτόνομα ή όσο το δυνατόν πιο οικονομικά.

- **Διαχείριση πολυμέσων και τηλεπικοινωνιών :** Η διαχείριση πολυμέσων αποτελεί άλλο ένα σημαντικό στοιχείο του έξυπνου σπιτιού. Ο χρήστης θα μπορεί να μεταφέρει τα πολυμέσα από τη μία συσκευή στην άλλη με πολύ μεγάλη ευκολία, καθώς όλες οι συσκευές θα είναι συνδεδεμένες μεταξύ τους. Συγκεκριμένα, θα μπορεί να μεταφέρει βίντεο ή φωτογραφίες που βλέπει στο κινητό του, στην οθόνη του ψυγείου ή της τηλεόρασης, ενώ το βίντεο θα παίζει, χωρίς να υπάρχει διακοπή. Σχεδόν κάθε επιφάνεια και κάθε οθόνη στο σπίτι (ακόμη και αυτές των οικιακών συσκευών), θα μπορούν να προβάλουν το περιεχόμενο κάποιας άλλης συσκευής. Το ίδιο θα μπορεί να γίνεται και με την επικοινωνία του χρήστη, καθώς κατά την είσοδό του στο σπίτι, θα μπορεί να αφήνει το κινητό του να φορτίσει σε κάποιο δωμάτιο, αλλά τα μηνύματά και τα e-mail του να προβάλλονται στην οθόνη του δωματίου που θα βρίσκεται εκείνη τη στιγμή. Επιπλέον, την ίδια στιγμή θα μπορούν να πραγματοποιούνται και οι κλήσεις του, και να επικοινωνεί από τυχόν ενσωματωμένα μικρόφωνα ή ηχεία, που θα υπάρχουν στο σπίτι ή πάνω στις συσκευές. Το σπίτι θα διαχειρίζεται πλήρως την επικοινωνία έχοντας ακόμη και τη δυνατότητα να στέλνει τον ήχο σε άλλη συσκευή και να καταγράφει τη φωνή του χρήστη από αλλού.

Τα παραπάνω σενάρια μας δίνουν μια μικρή ένδειξη των δυνατοτήτων που θα μπορεί να προσφέρει το έξυπνο σπίτι του μέλλοντος. Αυτό που προκύπτει όμως από τα παραπάνω είναι ότι όλες οι συσκευές θα είναι διασυνδεδεμένες μεταξύ τους και σχεδόν όλες οι λειτουργίες θα μπορούν να γίνονται αυτοματοποιημένα και από απόσταση.

1.2 Ιστορική αναδρομή

Επισήμως η λέξη «έξυπνο» (Smart) χρησιμοποιήθηκε για πρώτη φορά σε αναφορά για τεχνολογικά επιτεύγματα, κατά τη δεκαετία του '70. Αναφερόταν σε στρατιωτικά προϊόντα, όπως βόμβες ή πυραύλους που καθοδηγούσαν τον εαυτό τους προς το στόχο («έξυπνες» βόμβες). Κατά την τεχνολογική άνθιση της δεκαετίας του '80 η λέξη «έξυπνο» απέκτησε άλλες προεκτάσεις: αναφερόταν σε συσκευές που εμπεριείχαν μικροτσίπ, όπως οι υπολογιστές και οι προηγμένες οικιακές συσκευές. Βέβαια αυτό άλλαξε με την πάροδο του χρόνου και πλέον σήμερα δεν αποκαλούμε έναν σύγχρονο υπολογιστή «έξυπνο», παρόλο που οι σημερινοί υπολογιστές είναι εκθετικά ισχυρότεροι από εκείνους της δεκαετίας του '80. [5] Αν και τα πρώτα «καλωδιωμένα σπίτια» κατασκευάζονταν από χομπίστες, από τις αρχές του 1960, ο όρος «Έξυπνο Σπίτι» (ή αλλιώς “Smart House”) χρησιμοποιήθηκε επίσημα πρώτη φορά το 1984 από την Αμερικάνικη Ένωση Κατασκευαστών Σπιτιών. Αυτή η Ένωση ίδρυσε μια ομάδα ενδιαφέροντος με το προαναφερόμενο όνομα, προκειμένου να ωθήσει την ενσωμάτωση της απαραίτητης τεχνολογίας στο σχεδιασμό των νέων σπιτιών. Όλα τα παραπάνω αποτέλεσαν σημαντικό παράγοντα στην εξέλιξη του έξυπνου σπιτιού, διότι ένα σπίτι δεν χαρακτηρίζεται έξυπνο με κριτήρια το αν είναι φιλικό προς το περιβάλλον, αν κατασκευάζεται από καινοτόμα υλικά ή εάν χρησιμοποιεί ηλιακή ενέργεια και ανακυκλώνει το νερό που καταναλώνεται. Αν και ένα έξυπνο σπίτι περιλαμβάνει πολλά ή και όλα τα παραπάνω, αυτό που το κάνει “έξυπνο” είναι οι δια-δραστικές τεχνολογίες που περιέχει. Στις αρχές του 20ού αιώνα, παρατηρήθηκε μια μεγάλη αύξηση του ρυθμού αλλαγής της οικιακής τεχνολογίας και μέχρι το τέλος του 20ού αιώνα πολλά σπίτια συνδέονταν και σε άλλες υπηρεσίες διασκέδασης και πληροφόρησης μέσω του Η.Υ. Αυτό αποτέλεσε την βάση πάνω στην οποία αναπτύχθηκε το έξυπνο σπίτι. [6]

Σήμερα παρόλο που το ενδιαφέρον για το έξυπνο σπίτι μεγαλώνει όλο και περισσότερο, δεν υπάρχει η αναμενόμενη ανάπτυξή του. Αυτό συμβαίνει διότι:

- η αρχική επένδυση που απαιτείται από τον καταναλωτή είναι υψηλή, περιορίζοντας το αγοραστικό κοινό στα μεσαία και υψηλά στρώματα. Οι πιθανοί

αγοραστές θα πρέπει πρώτα να πειστούν για τα οφέλη τα οποία θα έχουν και στη συνέχεια να προχωρήσουν στη δημιουργία του έξυπνου σπιτιού.

- οι κατασκευαστές θα πρέπει να βρουν τρόπους να εφαρμόσουν την τεχνολογία αυτή σε ήδη υπάρχουσες κατοικίες και κτίρια. Το παραπάνω εγχείρημα είναι πιο ακριβό σε σχέση με τη δικτύωση ενός κτιρίου τη στιγμή που χτίζεται.
- οι προμηθευτές έχουν υιοθετήσει μια πιο "κοντόφθαλμη" προσέγγιση και δεν δείχνουν αρκετό ενδιαφέρον για τις πραγματικές ανάγκες του χρήστη. Ο σημερινός καταναλωτής θέλει συστήματα τα οποία θα τον βοηθήσουν να διαχειριστεί τις καθημερινές του υποχρεώσεις προσφέροντας μείωση του κόπου, απλούστευση της διαδικασίας, ευκολία στη χρήση, απομακρυσμένο έλεγχο και μείωση του κόστους. Υπάρχει ένα μεγάλο κενό μεταξύ των απαιτήσεων των πελατών και των προϊόντων που υπάρχουν ήδη στην αγορά.
- οι προμηθευτές δεν έχουν ασχοληθεί ιδιαίτερα στο να αξιολογήσουν την χρηστικότητα των προϊόντων τους. Όπως επισημαίνει και ο Barlow (1997), η αξιολόγηση δεν είναι κάτι απλό, λόγω της ποικιλομορφίας του πληθυσμού των χρηστών, της ποικιλίας του περιεχομένου χρήσης και της προ απαιτούμενης εκπαίδευσης που χρειάζεται για τη χρήση του. [6][7][8][9][10]
- λόγω της έλλειψης ενός κοινού πρωτοκόλλου (standard), οι εταιρείες που ασχολούνται με το έξυπνο σπίτι επικεντρώνονται πιο πολύ σε απλά συστήματα on-off (π.χ. απομακρυσμένη ενεργοποίηση - απενεργοποίηση), τα οποία δεν απαιτούν επιπλέον δικτυακή εγκατάσταση.

1.3 Πρωτόκολλα Επικοινωνίας

Παρόλο που δεν υπάρχει κάποιο επίσημο standard όσον αφορά τα πρωτόκολλα επικοινωνίας μεταξύ των συσκευών, και ο καθένας μπορεί να δοκιμάσει την δική του υλοποίηση, υπάρχουν μερικά ευρέως διαδεδομένα πρωτόκολλα επικοινωνίας τα οποία αναλύονται εκτενώς παρακάτω:

- **X10**:

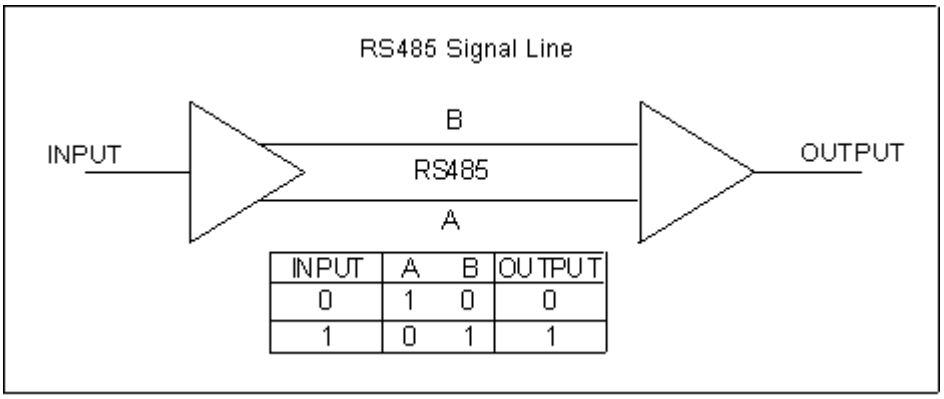
  Το πρωτόκολλο αυτό χρησιμοποιεί την ήδη υπάρχουσα καλωδίωση του ρεύματος για τη μεταφορά των δεδομένων μεταξύ των συσκευών. Ένα από τα μεγάλα πλεονεκτήματά του αποτελεί το γεγονός ότι δεν απαιτεί επιπλέον καλωδίωση για την επικοινωνία των συσκευών (ανάλογα πάντα με την πολυπλοκότητα της υλοποίησης).

- **Ethernet**:

  Το Ethernet, ευρέως διαδεδομένο για τις δικτυακές συνδεσμολογίες, μπορεί να χρησιμοποιηθεί και για τη διασύνδεση και την επικοινωνία μεταξύ των συσκευών του έξυπνου σπιτιού.

- **RS-485**:

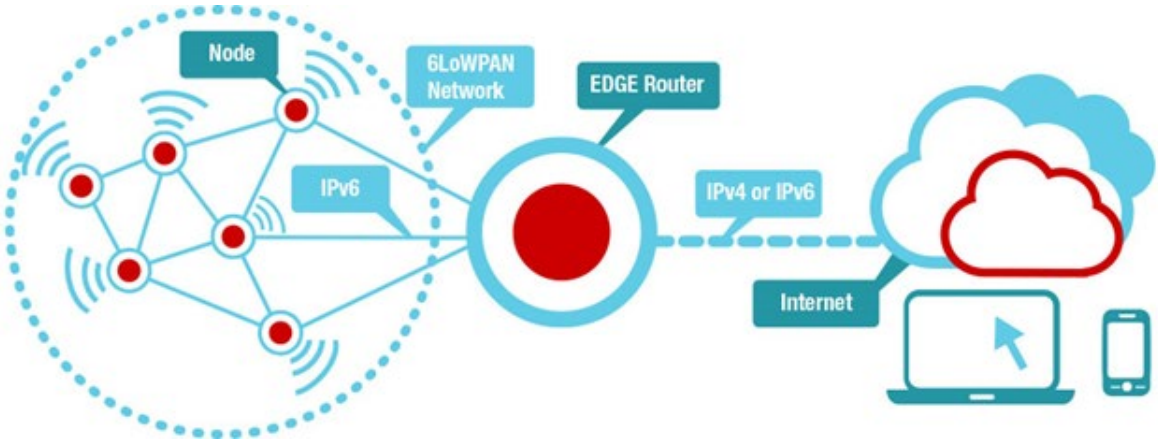
  Το πρωτόκολλο αυτό απαιτεί έναν πομπό και έναν δέκτη για την μεταφορά των μηνυμάτων. Για τη μεταφορά ενός σήματος απαιτούνται δύο γραμμές (για δύο σήματα 4, για τέσσερα σήματα οκτώ κοκ.). Δύο από τα σημαντικότερα πλεονεκτήματά του αποτελούν η ικανότητά του να μεταφέρει σήματα σε μεγάλες αποστάσεις και μέσα από περιβάλλοντα με έντονο ηλεκτρικό θόρυβο, καθιστώντας το ιδανικό για βιομηχανικές εφαρμογές.



INPUT	A	B	OUTPUT
0	1	0	0
1	0	1	1

- **6LoWPAN**:

  Το πρωτόκολλο αυτό αποτελεί ακρωνύμιο του: *IPv6 over Low power Wireless Personal Area Networks*, και λειτουργεί με την χρήση IP. Ο λόγος δημιουργίας αυτού του πρωτοκόλλου ήταν η χρήση του σε συσκευές χαμηλής κατανάλωσης ενέργειας και με μειωμένες επεξεργαστικές δυνατότητες, οι οποίες όμως δεν θα έπρεπε να μείνουν εκτός του IoT. Στο πρωτόκολλο αυτό χρησιμοποιείται ενθυλάκωση (encapsulation) και συμπίεση κεφαλίδας (header compression) τα οποία επιτρέπουν στα πακέτα να μεταδίδονται μέσω IP και δημιουργείται ένα πλήρες δίκτυο με κόμβους, που η διευθυνσιοδότησή τους γίνεται μόνο με IP από άκρη σε άκρη (end-to-end). Επίσης οι συσκευές αυτές έχουν τη δυνατότητα να επικοινωνούν ασύρματα μεταξύ τους, δημιουργώντας έτσι μία τοπολογία δικτύου πλέγματος (mesh network), η οποία μπορεί να οδηγηθεί από ένα ή διάφορα σημεία προς το ρούτερ, και από εκεί στον έξω κόσμο και στην συσκευή του τελικού χρήστη.

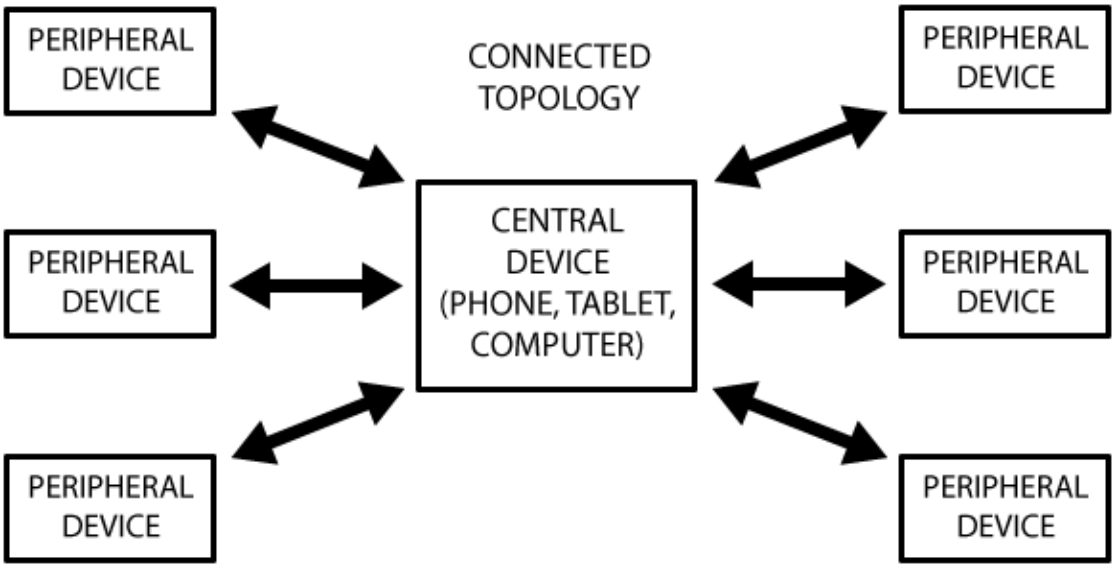


Σχήμα 1.1: Τοπολογία δικτύου με χρήση πρωτοκόλλου 6LoWPAN

- **Bluetooth LE (BLE)**

  Το Bluetooth LE (ή αλλιώς Bluetooth low energy) αποτελεί την τεχνολογία Bluetooth που προσφέρει τετραπλάσιο εύρος, διπλάσια ταχύτητα επικοινωνίας μεταξύ των συσκευών και ακόμη μεγαλύτερο data broadcasting capacity. Το Bluetooth LE θα κυκλοφορήσει στην αγορά ως Bluetooth 5 και όλα τα παραπάνω χαρακτηριστικά θα είναι απαραίτητα προκειμένου να χρησιμοποιηθεί πιο ευρέως σε smart home τεχνολογίες. Το Bluetooth υποστηρίζει μια αστεροειδή τοπολογία δικτύου μεταξύ των συσκευών, στην οποία η κάθε περιφερειακή συσκευή μπορεί

να συνδεθεί μόνο σε μία κεντρική, αλλά η κεντρική συσκευή μπορεί να έχει πάνω της πολλές περιφερειακές συσκευές και η επικοινωνία να γίνεται από και προς τις δύο κατευθύνσεις (βλέπε σχήμα 1.2).



Σχήμα 1.2: Τοπολογία δικτύου με χρήση πρωτοκόλλου Bluetooth LE.

1.4 Σκοπός της Εργασίας

Ο σκοπός της παρούσας έρευνας είναι να μελετηθεί η δυνατότητα υλοποίησης ενός συστήματος για έξυπνο έλεγχο των συσκευών ενός σπιτιού με φωνητικές εντολές. Η συγκεκριμένη υλοποίηση θα μπορεί να λειτουργήσει εντός ενός ήδη υπάρχοντος σπιτιού, παρέχοντας ευκολία στον έλεγχο του συστήματος, χωρίς να υπάρχει η απαίτηση δημιουργίας επιπρόσθετης εφαρμογής ελέγχου. Επίσης, η υπάρχουσα υλοποίηση θα προσφέρει στο χρήστη τη δυνατότητα ελέγχου από οποιαδήποτε συσκευή που χρησιμοποιεί περιηγητή στο διαδίκτυο, καθιστώντας την εύκολα προσβάσιμη από laptop, desktop, tablet, κινητό τηλέφωνο. Ο χρήστης θα πρέπει απλά να πει την επιθυμητή εντολή.

Η συγκεκριμένη υλοποίηση δημιουργήθηκε με σκοπό να ωθήσει τις ήδη υπάρχουσες υλοποιήσεις για έξυπνα σπίτια (αλλά και νέων συσκευών γενικότερα) να χρησιμοποιήσουν τις φωνητικές εντολές, ως γενικότερο τρόπο ελέγχου. Η ομιλία είναι μία εγγενής ιδιότητα την οποία ο άνθρωπος έχει μάθει να χρησιμοποιεί και να ελέγχει από πολύ μικρή ηλικία, οπότε του είναι και πιο φυσικό και εύκολο να κάνει χρήση της, για να ελέγξει το περιβάλλον γύρω του. Οι συσκευές της σημερινής εποχής, γίνονται όλο και πιο πολύπλοκες και απαιτούν όλο και περισσότερες ειδικές γνώσεις από τον άνθρωπο, προκειμένου να τις χειριστεί. Οι φωνητικές εντολές, λοιπόν, αποτελούν τον πιο απλό και εύκολο τρόπο ελέγχου, και αν αξιοποιηθούν πλήρως, θα μπορέσουν να εκμεταλλευτούν όχι μόνο το τι λέει ο εκάστοτε χρήστης, αλλά και τον τρόπο, χρόνο που το λέει. Επίσης, με την όλο και μεγαλύτερη προσπάθεια εξέλιξης και ανάπτυξης της τεχνητής νοημοσύνης, ο τρόπος ελέγχου μέσω της ομιλίας είναι τουλάχιστον αναμενόμενος.

Η συγκεκριμένη εργασία χρησιμοποιεί ήδη υπάρχοντα πρωτόκολλα και τρόπους υλοποίησης θέλοντας να δείξει πώς αυτά μπορούν να ενσωματωθούν με το κομμάτι του φωνητικού ελέγχου. Ακόμη, αποτελεί μέρος της γενικότερης προσπάθειας ενσωμάτωσης των εντολών, μέσω της ομιλίας, και σε άλλες υλοποιήσεις που ακολουθούν τις ίδιες αρχές λειτουργίας. Το παραπάνω, όχι μόνο θα κάνει τις σύγχρονες αυτές εφαρμογές πιο προσιτές στον σημερινό άνθρωπο (λόγω της απλούστευσης των διαδικασιών και του τρόπου χρήσης), αλλά και θα βοηθήσει στην εξέλιξη της χρήσης των φωνητικών εντολών, ως γενικότερου τρόπου ελέγχου

ηλεκτρονικών συσκευών, βοηθώντας σημαντικά στην εξέλιξη της τεχνητής νοημοσύνης. Το τελευταίο θα επιτευχθεί μέσα από τη δημιουργία ενός συστήματος, στο οποίο ο άνθρωπος θα μπορεί να μιλάει πιο φυσικά (χωρίς να λέει συγκεκριμένες λέξεις κάθε φορά). Έτσι, θα διαμορφωθεί ένα σύστημα που θα αντιλαμβάνεται το ουσιαστικό νόημα της κάθε πρότασης, χωρίς αυτή να είναι διατυπωμένη με συγκεκριμένο τρόπο. Ιδανικότερα, το σύστημα θα μπορεί να απαντάει ή και να ρωτάει περισσότερες πληροφορίες για την εντολή που δίδεται από το χρήστη, εξελισσόμενο από τη μεταξύ τους "συναναστροφή".

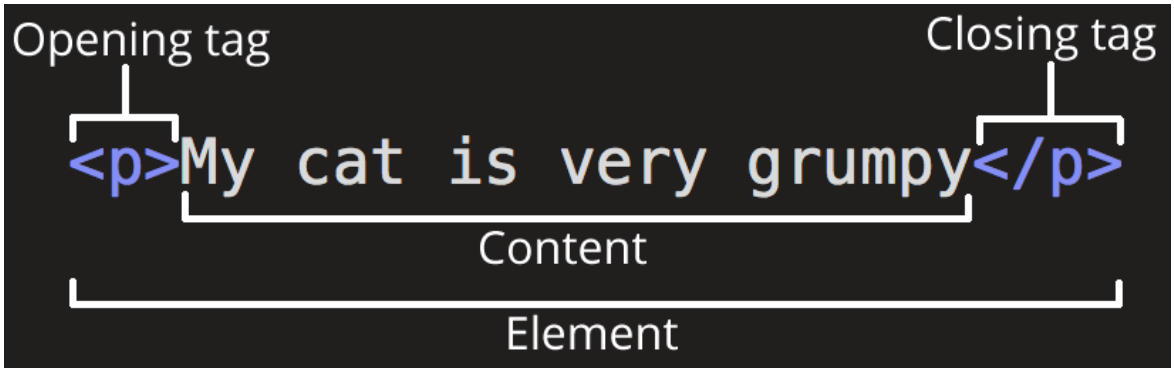
ΚΕΦΑΛΑΙΟ 2: ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Σε αυτό το κεφάλαιο αναλύονται τα εργαλεία και οι τεχνικές που χρησιμοποιήθηκαν καθώς και όλες οι απαιτούμενες γνώσεις για την υλοποίηση της παρούσας πτυχιακής εργασίας.

2.1 HTML

Η HTML (HyperText Markup Language – γλώσσα σήμανσης υπερκειμένου) δεν είναι γλώσσα προγραμματισμού όπως πολλοί συχνά την μπερδεύουν, αλλά *γλώσσα σήμανσης* η οποία υπαγορεύει στον περιηγητή (browser) πώς να δομήσει την εκάστοτε ιστοσελίδα που επισκεπτόμαστε στο διαδίκτυο. Η HTML αποτελείται από μια σειρά από στοιχεία (elements), τα οποία χρησιμοποιούνται για περικλείσουν, τυλίξουν ή *σημάνουν* διαφορετικά μέρη του περιεχομένου ώστε να εμφανίζονται με συγκεκριμένο τρόπο ή να λειτουργούν με συγκεκριμένο τρόπο. Οι ετικέτες (tags) μπορούν να μετατρέψουν το περιεχόμενο που περικλείουν σε σύνδεσμο (link) σε μια άλλη ιστοσελίδα, να δώσουν στυλ **bold** σε λέξεις, να ενσωματώσουν εικόνες στην σελίδα μας και πολλά ακόμα. Αξίζει να σημειωθεί επίσης ότι τις περισσότερες φορές οι ετικέτες εμφανίζονται σε ζευγάρια, την ετικέτα ανοίγματος και την ετικέτα κλεισίματος (κάποιες ετικέτες δεν έχουν ετικέτα κλεισίματος).

Για παράδειγμα εάν θέλαμε να γράψουμε και να εμφανίσουμε την πρόταση My cat is very grumpy θα την περικλείαμε σε ετικέτες παραγράφου ως εξής:



Εικόνα 2.1: Τα κύρια μέρη ενός στοιχείου HTML

Όπως φαίνεται και παραπάνω, τα κύρια μέρη ενός HTML στοιχείου είναι:

1. **Η ετικέτα ανοίγματος** (opening tag): Η ετικέτα αυτή αποτελείται από το όνομα του στοιχείου (στην περίπτωσή μας το p που υποδηλώνει παράγραφο), “τυλιγμένο” μέσα σε **γωνιακές παρενθέσεις**. Αυτό υποδηλώνει από πού ξεκινά ή από πού τίθεται σε ισχύ το στοιχείο - στην περίπτωσή μας από την αρχή της παραγράφου.
2. **Η ετικέτα κλεισίματος** (closing tag): Είναι το ίδιο με την ετικέτα ανοίγματος, εκτός από το ότι χρησιμοποιεί το διαχωριστικό / πριν από το όνομα του στοιχείου. Αυτό δηλώνει που τελειώνει το στοιχείο – σε αυτή την περίπτωση στο τέλος της παραγράφου.
3. **Το περιεχόμενο** (content): Αυτό είναι το περιεχόμενο του στοιχείου όπου στην περίπτωσή μας είναι μόνο κείμενο.
4. **Το στοιχείο** (element): Η ετικέτα ανοίγματος, συν την ετικέτα κλεισίματος, συν το περιεχόμενο μας κάνουν το στοιχείο.

2.1.1 Ανατομία ενός HTML εγγράφου

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <title>My test page</title>
6    </head>
7    <body>
8      <p>This is my page</p>
9    </body>
10 </html>
```

Εικόνα 2.2 : Σκελετός ενός HTML εγγράφου

Ο παραπάνω κώδικας αναλύεται ως εξής:

1. **<!DOCTYPE html>** : Το doctype λειτουργεί ως υπερσύνδεσμος (link) σε ένα σετ κανόνων που πρέπει όλα τα html έγγραφα να περιέχουν ώστε να είναι έγκυρα. Η πιο απλή μορφή του είναι η παραπάνω και χρησιμοποιείται πάντα σε κάθε έγγραφο HTML.
2. **<html></html>** : Το στοιχείο <html> περικλείει όλο το περιεχόμενο όλης της σελίδας και πολλές φορές είναι γνωστό ως στοιχείο ρίζα (root element).

3. **<head></head>** : Το στοιχείο <head> λειτουργεί σαν ένα δοχείο για όλα τα πράγματα που χρειάζεται να συμπεριληφθούν στην HTML σελίδα μας τα οποία δεν αποτελούν το περιεχόμενο της σελίδας. Τέτοια πράγματα είναι για παράδειγμα λέξεις κλειδιά, αρχεία CSS που περιέχουν κανόνες μορφοποίησης του περιεχομένου μας (θα αναλύσουμε τα αρχεία CSS παρακάτω),αρχεία JavaScript (τα οποία θα αναλυθούν παρακάτω), την κωδικοποίηση που θα χρησιμοποιηθεί η οποία αφορά την γλώσσα στην οποία θα γραφτεί το περιεχόμενο κλπ. Γενικότερα στις ετικέτες αυτές συμπεριλαμβάνονται ότι χρειάζεται προκειμένου να εμφανιστεί σωστά το περιεχόμενο της σελίδας μας.
4. **<meta charset="utf-8">** : Αυτό το στοιχείο ορίζει το σετ των χαρακτήρων που θα χρησιμοποιηθούν στο περιεχόμενο της σελίδας στην κωδικοποίηση utf-8 η οποία περιλαμβάνει τους χαρακτήρες από τις περισσότερες ανθρώπινες γλώσσες. Ουσιαστικά μπορεί να χειριστεί το περιεχόμενο οποιουδήποτε κειμένου μπορεί να εισαχθεί.
5. **<title></title>** : Αυτό το στοιχείο ορίζει τον τίτλο της σελίδας, ο οποίος είναι ο τίτλος που εμφανίζεται στην καρτέλα του περιηγητή όταν ανοίγουμε την σελίδα. Να σημειωθεί ότι και αυτό δηλώνεται μέσα στο στοιχείο <head> το οποίο αναλύσαμε προηγουμένως.
6. **<body></body>** : Το στοιχείο <body> περιέχει *όλο* το περιεχόμενο που θέλουμε να δείξουμε στους διαδικτυακούς χρήστες που επισκέπτονται την σελίδα μας, είτε αυτό είναι κείμενο, εικόνα, βίντεο, παιχνίδι, ήχος ή οτιδήποτε άλλο.

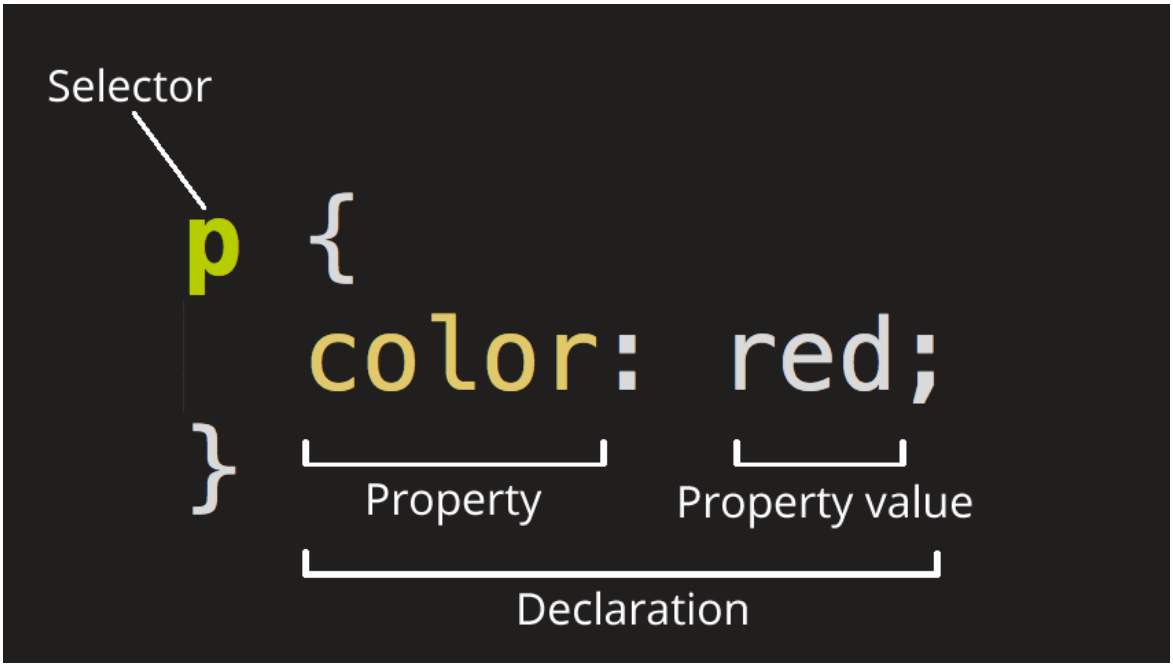
Οι παραπάνω είναι οι πιο βασικές ετικέτες που χρειάζονται για την σωστή εμφάνιση ενός HTML εγγράφου. Εάν θέλετε να δείτε μια λίστα με όλες τις ετικέτες HTML ακολουθήστε τον υπερσύνδεσμο https://www.w3schools.com/tags/.

2.2 CSS

Η CSS (Cascading Style Sheets – επικαλυπτόμενα φύλλα στυλ) είναι μία γλώσσα στυλ η οποία χρησιμοποιείται για να περιγράψει το πώς θα παρουσιαστεί (στιλιστικά) ένα έγγραφο HTML, δηλαδή να διαμορφώσει τα χρώματα, τη στοίχιση, τη θέση ενός στοιχείου κλπ., με ποια σειρά θα φορτωθούν τα στοιχεία στην οθόνη κλπ.

2.2.1 Ανατομία ενός κανόνα CSS

Στη CSS ορίζουμε σετ κανόνων (ruleset) ή αλλιώς κανόνες (rules) για κάποιο στοιχείο στο HTML έγγραφό μας. Για παράδειγμα:



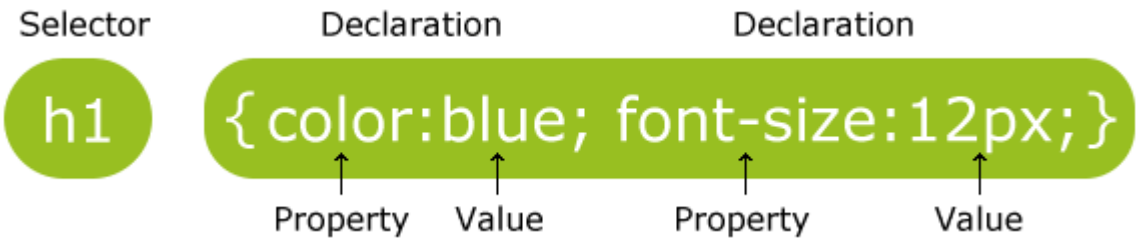
Εικόνα 2.3 : Παράδειγμα CSS κανόνα

1. **Επιλογέας (selector)** : Ο επιλογέας είναι το όνομα του HTML στοιχείου στην αρχή του σετ κανόνων και επιλέγει το στοιχείο(α) που πρόκειται να μορφοποιηθούν (για παραπάνω από ένα στοιχεία χρησιμοποιούμε το κόμμα). Στο συγκεκριμένο παράδειγμα ο επιλογέας αφορά την παράγραφο (διότι χρησιμοποιείται το στοιχείο της παραγράφου p). Για τη μορφοποίηση ενός διαφορετικού HTML στοιχείου, απλά αλλάξτε τον επιλογέα.

2. **Δήλωση (declaration)** : Ένας κανόνας όπως για παράδειγμα color: red; (χρώμα : κόκκινο) ο οποίος ορίζει ποια **ιδιότητα** του στοιχείου θέλουμε να αλλάξουμε.
3. **Ιδιότητες (properties)** : Τρόποι με τους οποίους μπορείς να μορφοποιήσεις ένα δοθέν στοιχείο HTML. (Σε αυτή την περίπτωση το color είναι μία ιδιότητα των στοιχείων <p>.) Με το CSS επιλέγουμε τους κανόνες που θέλουμε να επηρεάσουμε εντός του κανόνα.
4. **Τιμή ιδιότητας (property value)** : Στα δεξιά της ιδιότητας μετά την άνω κάτω τελεία, έχουμε την **τιμή της ιδιότητας**, με την οποία επιλέγεται μία από τις πολλές διαθέσιμες τιμές για μία δοθείσα ιδιότητα (υπάρχουν πολλές τιμές για την ιδιότητα color εκτός από το red(κόκκινο)).

Επίσης πρέπει να σημειωθεί ότι:

- Κάθε σετ κανόνων (εκτός από τον επιλογέα) πρέπει να βρίσκεται εντός των άγκιστρων ({ }).
- Εντός κάθε δήλωσης, θα πρέπει να χρησιμοποιείται η άνω κάτω τελεία (:) για να διαχωρίσει την ιδιότητα από την τιμή που της ανατίθεται.
- Εντός ενός σετ κανόνων, θα πρέπει να χρησιμοποιείται το ελληνικό ερωτηματικό (;) ώστε να διαχωρίσετε μια δήλωση από την επόμενη



Εικόνα 2.4: Παράδειγμα CSS κανόνα με δύο δηλώσεις

2.3 JavaScript

Η JavaScript είναι μια αντικειμενοστραφής και σκριπτοειδής (scripting language) γλώσσα προγραμματισμού η οποία δίνει τη δυνατότητα να ελεγχθεί δυναμικά το περιεχόμενο μίας ιστοσελίδας. Χάρη σε αυτήν μπορούμε να δημιουργήσουμε δυναμικές ιστοσελίδες, δηλαδή να αλλάζει η μορφή και το περιεχόμενο της ιστοσελίδας μας ανάλογα με τις ενέργειες του χρήστη ή ανάλογα με το χρόνο κλπ. Μπορούμε για παράδειγμα να εκτελέσουμε μια συγκεκριμένη ενέργεια όταν ο χρήστης κάνει κλικ σε κάποια συγκεκριμένη περιοχή της ιστοσελίδας. Μία άλλη συνηθισμένη πρακτική είναι να τροποποιούμε ή να προσθέτουμε κώδικα **CSS** ή/και **HTML** που αναλύσαμε προηγουμένως. Μπορούμε δηλαδή να αλλάξουμε το χρώμα μιας παραγράφου όταν το ποντίκι του χρήστη περνά από πάνω της να εμφανίσουμε κάποια εικόνα ή ένα αναδυόμενο πλαίσιο κειμένου.

2.3.1 Παράδειγμα κώδικα JavaScript

Έστω ότι έχουμε το ακόλουθο κομμάτι HTML κώδικα:

```
1 | <p>Player 1: Chris</p>
```

Εικόνα 2.5: HTML κώδικας στον οποίο θα προσθέσουμε λειτουργικότητα με JavaScript

Το οποίο θα εμφανιστεί στην οθόνη μας ως εξής:

Player 1: Chris

Τώρα προσθέτοντας λίγο κώδικα CSS μορφοποιούμε την παράγραφο αυτή ώστε να μοιάζει με κουμπί

```css
p {
  font-family: 'helvetica neue', helvetica, sans-serif;
  letter-spacing: 1px;
  text-transform: uppercase;
  text-align: center;
  border: 2px solid rgba(0,0,200,0.6);
  background: rgba(0,0,200,0.3);
  color: rgba(0,0,200,0.6);
  box-shadow: 1px 1px 2px rgba(0,0,200,0.4);
  border-radius: 10px;
  padding: 3px 10px;
  display: inline-block;
  cursor:pointer;
}
```

Εικόνα 2.6: Κώδικας CSS με τον οποίο μορφοποιούμε την προηγούμενη παράγραφο

Με τους προηγούμενους κώδικες, η παράγραφός μας θα εμφανιστεί ως εξής:

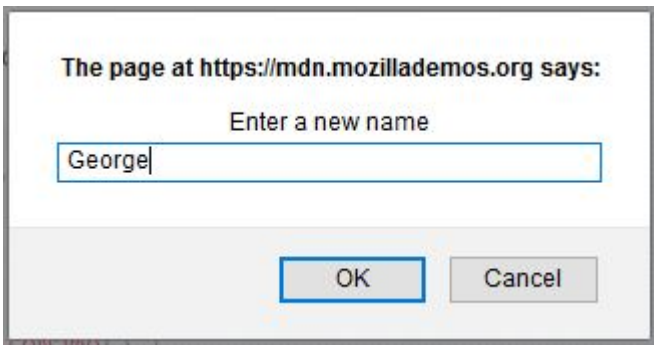
Εικόνα 2.7 : Εικόνα της παραγράφου αφότου μορφοποιήθηκε με CSS.

Τέλος θα προσθέσουμε τον κώδικα JavaScript με τον οποίο μπορούμε να δώσουμε λειτουργικότητα στην παράγραφο ώστε να συμπεριφέρεται ως κουμπί.

```javascript
var para = document.querySelector('p');

para.addEventListener('click', updateName);

function updateName() {
  var name = prompt('Enter a new name');
  para.textContent = 'Player 1: ' + name;
}
```

Εικόνα 2.8: Κώδικας JavaScript ο οποίος προσθέτει λειτουργικότητα στην παράγραφο.

Μετά από την ενσωμάτωση και του κώδικα JavaScript, η παράγραφος μας δίνει τη δυνατότητα να κάνουμε κλικ σε αυτή και εμφανίζει ένα παράθυρο διαλόγου στο οποίο μπορούμε να εισάγουμε ένα διαφορετικό όνομα το οποίο εμφανίζεται στη θέση του προηγούμενου.



Εικόνα 2.9: Παράθυρο που εμφανίζεται αφότου πατήσουμε την παράγραφο, όπου μπορούμε να εισάγουμε το νέο όνομα.

Εικόνα 2.10: Η παράγραφος μετά την αλλαγή του ονόματος.

Το παραπάνω παράδειγμα JavaScript κώδικα (βλέπε εικόνα 2.8) αναλύεται ως εξής:

1. `var para = document.querySelector('p');`
    - `document.querySelector('p')`: Αυτή η μέθοδος, παίρνει το αναγνωριστικό που δηλώνουμε στις παραμέτρους της μεθόδου (παράμετροι λέγονται τα ορίσματα που εισάγονται όταν καλείται μια μέθοδος, στην προκειμένη περίπτωση το ‘p’), και επιλέγει όλα τα στοιχεία του εγγράφου (εδώ αναφέρεται στο HTML έγγραφο) που ταιριάζουν με αυτό. Στην προκειμένη περίπτωση έχουμε μόνο ένα στοιχείο παραγράφου (‘p’) άρα θα επιλεγεί αυτό. Να σημειωθεί ότι αυτή η μέθοδος υπάρχει ήδη στην JavaScript, όπως και άλλες που θα δούμε παρακάτω.
    - `var para =` : Το λεκτικό var χρησιμοποιείται για να δηλώσει μία μεταβλητή οποιουδήποτε τύπου. Δηλαδή στο συγκεκριμένο παράδειγμα

η μεταβλητή para μπορεί να έχει μέσα νούμερα, γράμματα, λογικές τιμές (true,false) κλπ. Το ίσο χρησιμοποιείται για ανάθεση τιμής, δηλαδή στην μεταβλητή para αποθηκεύεται το περιεχόμενο της παραγράφου p που επιλέξαμε πριν, δηλαδή το «Player 1: Chris».

2. `para.addEventListener('click', updateName);` : Η μέθοδος `addEventListener` δημιουργεί ένα συμβάν σε μία ενέργεια που ορίζει ο προγραμματιστής. Σαν πρώτο όρισμα παίρνει την ενέργεια κατά την οποία θα πρέπει να εκτελέσει, ότι του δίνεται στο δεύτερο όρισμα. Στην συγκεκριμένη περίπτωση, σαν δεύτερο όρισμα δίνουμε το όνομα μίας μεθόδου. Όλα τα παραπάνω ανατίθενται στη μεταβλητή `para`. Συνοψίζοντας λοιπόν η παραπάνω γραμμή κώδικα, υπαγορεύει ότι μόλις γίνει κλικ πάνω στην μεταβλητή para (η οποία πλέον περιέχει όλη την παράγραφο από το βήμα 1, άρα είναι σαν να κάνουμε κλικ την παράγραφο) να εκτελεσθεί η μέθοδος `updateName`.
3. `function updateName()`: Με το λεκτικό function γίνεται η δήλωση μιας μεθόδου, ενώ το όνομα που ακολουθεί είναι το όνομα της μεθόδου. Οι δύο παρενθέσεις δείχνουν τα ορίσματα που παίρνει η μέθοδος (στην προκειμένη δεν έχει). ***Σημείωση:*** Τα άγκιστρα που ακολουθούν {} δείχνουν που αρχίζει και που τελειώνει η μέθοδος και περιλαμβάνουν μέσα τους όλο τον κώδικα της μεθόδου.
4. `var name = prompt('Enter a new name');`
   - `prompt('Enter a new name');` : Η μέθοδος prompt είναι άλλη μια ήδη υπάρχουσα μέθοδος στην JavaScript η οποία εμφανίζει στον χρήστη αυτό το παράθυρο που είδαμε στην **Εικόνα 2.9** όπου μας προτρέπει να εισάγουμε κείμενο μέσα στο πεδίο κειμένου.
   - `var name` : Όπως αναφέραμε και πριν στο βήμα 1, με το λεκτικό var δημιουργούμε μία μεταβλητή στην οποία εκχωρούμε το κείμενο που εισήγαγε ο χρήστης όπως φαίνεται και στην **Εικόνα 2.9**.
5. `para.textContent = 'Player 1: ' + name;`
   - `'Player 1: ' + name;` : Με αυτή την εντολή ενώνουμε τις συμβολοσειρές ‘Player 1: ‘ και ότι έχει προσθέσει ο χρήστης στο πεδίο κειμένου (το οποίο το αποθηκεύσαμε στην προαναφερθείσα μεταβλητή name) σε μία, η οποία στο παράδειγμά μας θα είναι ‘Player 1: George’.

Ο τελεστής + είναι αυτός που είναι υπεύθυνος για τη συνένωση των συμβολοσειρών και στη γλώσσα JavaScript χρησιμοποιείται και για την συνένωση συμβολοσειρών (και για πρόσθεση αλλά σε αριθμητικά δεδομένα).

- `para.textContent =` : Η εντολή αυτή (και συγκεκριμένα ο τελεστής =) εκχωρεί την τιμή της προηγούμενης συμβολοσειράς στην μεταβλητή para. Η μέθοδος `textContent` θέτει το κείμενο της μεταβλητής, οπότε δίνοντάς της την καινούρια συμβολοσειρά, αντικαθιστά την προηγούμενη που είχαμε βάλει από τον κώδικα (δηλαδή το 'Player 1: Chris').

Παρακάτω παρατίθεται ολόκληρος ο κώδικας που υλοποιεί το συγκεκριμένο παράδειγμα:

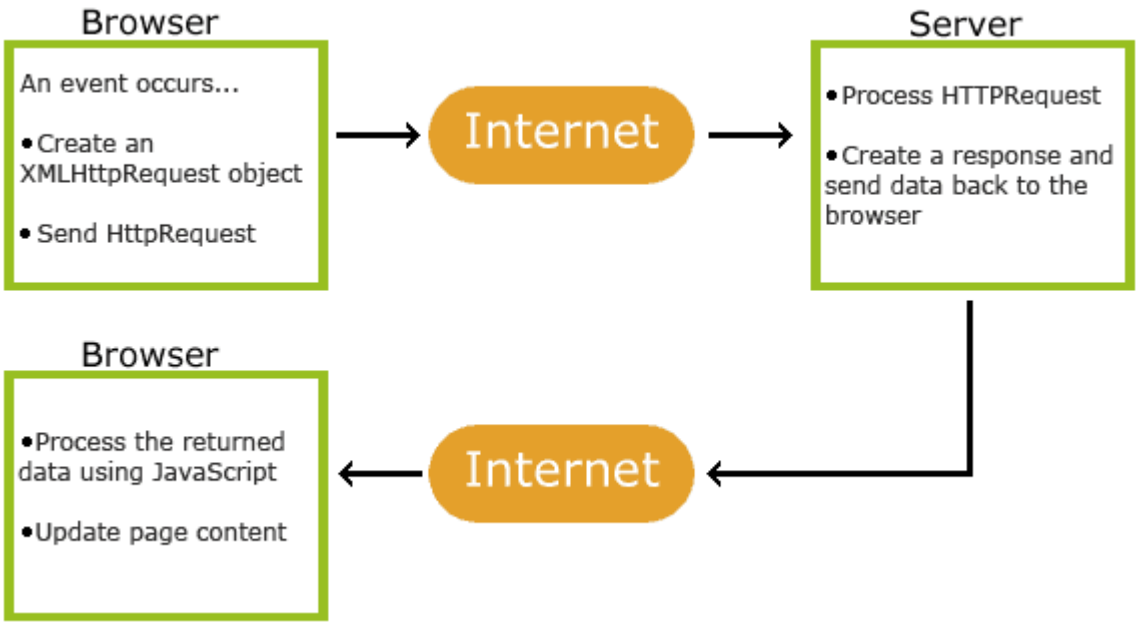
```
<!DOCTYPE HTML>
<html lang="en">
  <head>
    <title>JavaScript label example</title>
    <style>
      p {
        font-family: 'helvetica neue', helvetica, sans-serif;
        letter-spacing: 1px;
        text-transform: uppercase;
        text-align: center;
        border: 2px solid rgba(0,0,200,0.6);
        background: rgba(0,0,200,0.3);
        color: rgba(0,0,200,0.6);
        box-shadow: 1px 1px 2px rgba(0,0,200,0.4);
        border-radius: 10px;
        padding: 3px 10px;
        display: inline-block;
        cursor: pointer;
      }
    </style>
  </head>
  <body>
    <p>Player 1: Chris</p>

    <script>
```

```
      var para = document.querySelector('p');
      para.addEventListener('click', updateName);
      function updateName() {
        var name = prompt('Enter a new name');
        para.textContent = 'Player 1: ' + name;
      }
    </script>
  </body>
</html>
```

2.4 AJAX

Το AJAX (Asynchronous JavaScript And XML) δεν αποτελεί μια νέα γλώσσα προγραμματισμού, αλλά μία τεχνική που χρησιμοποιείται πάρα πολύ στην JavaScript και την οποία χρησιμοποιούμε και στην συγκεκριμένη υλοποίηση και αξίζει να αναφερθεί. Το πλεονέκτημα που προσφέρει η χρήση της συγκεκριμένης τεχνικής, είναι ουσιαστικά ότι δίνει τη δυνατότητα επικοινωνίας με τον server και ανταλλαγής περιεχομένου με αυτόν καθώς και δυνατότητα ανανέωσης περιεχομένου ή τμήματος μιας ιστοσελίδας χωρίς να ανανεώσουμε όλη την ιστοσελίδα. Αυτή η δυνατότητα είναι πολύ σημαντική διότι το να ανανεώσουμε όλη τη σελίδα μπορεί να είναι κοστοβόρο τόσο σε πόρους όσο και σε χρόνο (για παράδειγμα εάν η σελίδα μας περιέχει μεγάλο όγκο δεδομένων από φωτογραφίες, βίντεο κλπ., είναι ασύμφορο να τα φορτώσουμε ξανά μόνο και μόνο για να ανανεώσουμε ένα μέρος μιας ιστοσελίδας).

Παρακάτω δίνεται ένα παράδειγμα χρήσης ως εξής:



Εικόνα 2.11: Σχεδιάγραμμα λειτουργίας τεχνικής AJAX

1. Ένα συμβάν προκύπτει σε μια ιστοσελίδα (πχ πάτημα κουμπιού αφού έχει φορτωθεί η σελίδα)

2. Ένα αντικείμενο XMLHttpRequest δημιουργείται με JavaScript
   - Το αντικείμενο XMLHttpRequest υποστηρίζεται από όλους τους σύγχρονους περιηγητές (browser) και χρησιμοποιείται για ανταλλαγή δεδομένων με τον server στο παρασκήνιο. Αυτό σημαίνει ότι μπορεί να επικοινωνήσει με τον server και κατά τη διάρκεια περιήγησης του χρήστη σε μια ήδη φορτωμένη σελίδα και όχι μόνο κάθε φορά που φορτώνει μια σελίδα.

3. Το αντικείμενο XMLHttpRequest στέλνει ένα αίτημα στον server
   - Στέλνουμε ένα αίτημα (GET ή POST) στο οποίο μπορούμε να συμπεριλάβουμε τυχόν μεταβλητές ή άλλα δεδομένα (όπως url) που έχουν προκύψει από κάποια ενέργεια του χρήστη και να μεταφερθούν και αυτά στο server
4. Ο server επεξεργάζεται το αίτημα
5. Ο server στέλνει μια απάντηση στην ιστοσελίδα
   - Η απάντηση που στέλνει ο server μπορεί να παρθεί με δύο μεθόδους. Την **responseText** και την **responseXML**. Η Πρώτη επιστρέφει οτιδήποτε έχει στείλει ο server πίσω ως απάντηση υπό μορφή string της JavaScript ενώ η δεύτερη χρησιμοποιεί έναν ενσωματωμένο αναλυτή xml (xml parser) ο οποίος επιστρέφει την απάντηση του server ως ένα αντικείμενο XML.
6. Η απάντηση διαβάζεται με JavaScript
7. Εκτελείται η κατάλληλη ενέργεια (για παράδειγμα ανανέωση τμήματος της σελίδας) μέσω JavaScript.

2.5 PHP

Η PHP είναι μια ευρέως χρησιμοποιούμενη σκριπτοειδής (scripting language) γλώσσα ανοικτού κώδικα (open source) η οποία χρησιμοποιείται ευρέως για ανάπτυξη ιστοσελίδων και μπορεί να ενσωματωθεί μέσα στην HTML με την χρήση των ειδικών συμβόλων που δηλώνουν την αρχή και το τέλος ενός κώδικα php. Για παράδειγμα:

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
            echo "Hi, I'm a PHP script!";
        ?>

    </body>
</html>
```

Εικόνα 2.12: Παράδειγμα κώδικα HTML με χρήση php.

Ο προηγούμενος κώδικας θα εμφανίσει μία html σελίδα με το μήνυμα : Hi, I'm a PHP script!. Όπως παρατηρούμε στον παραπάνω κώδικα, τα σύμβολα `<?php` και `?>` δείχνουν την αρχή και το τέλος του κώδικα php. Αξίζει επίσης να σημειωθεί ότι ο παραπάνω κώδικας θα δημιουργήσει ένα στοιχείο παραγράφου (<p>) μέσα στο οποίο θα είναι το προαναφερθέν μήνυμα.

Αυτό που κάνει την php να διαφέρει από την JavaScript είναι ότι ο php κώδικας εκτελείτε στον server (server – side) όπου παράγει html κώδικα ο οποίος ενσωματώνεται στην html σελίδα η οποία στέλνεται στην συνέχεια στον χρήστη. Η JavaScript από την άλλη εκτελεί και παράγει τον html κώδικα στο μηχάνημα του χρήστη (client – side) χρησιμοποιώντας τους πόρους του μηχανήματος του χρήστη.

2.6 API

Το λεκτικό API αποτελεί συντομογραφία του **Application Programming Interface** (Διεπαφή Προγραμματισμού Εφαρμογών). Τα API χρησιμοποιούνται προκειμένου να επιτρέπουν την επικοινωνία μεταξύ διαφόρων προγραμμάτων. Τα API παρέχουν ουσιαστικά ένα σύνολο από εργαλεία, όπως μεθόδους, υπορουτίνες, πρωτόκολλά κλπ. τα οποία χρησιμοποιούνται από τους προγραμματιστές στην ανάπτυξη ενός λογισμικού ή μιας εφαρμογής. Για παράδειγμα, κάνοντας copy ένα κείμενο από έναν κειμενογράφο και επικολλώντας το σε κάποιο άλλο πρόγραμμα όπως το Excel, από πίσω χρησιμοποιείται ένα API για την επικοινωνία αυτών των δύο προγραμμάτων μεταξύ τους, ή την επικοινωνία τους με το ίδιο το λειτουργικό. Επίσης ένα άλλο παράδειγμα είναι η χρήση των Google Maps όπου από όποια συσκευή και να τους χρησιμοποιήσουμε, μπορούν να βρίσκουν την τοποθεσία μας ή να μας δίνουν οδηγίες προς ένα σημείο. Εκείνη την ώρα η συσκευή μας χρησιμοποιεί το **Geolocation API** για να λάβει την τοποθεσία μας και να μας φέρει τις οδηγίες που ζητήσαμε. Τα API λοιπόν αυτό που κάνουν είναι να διαθέτουν στον προγραμματιστή συγκεκριμένα εργαλεία τα οποία μπορεί να χρησιμοποιήσει όταν φτιάχνει την εφαρμογή του και να ενσωματώσει πχ το συγκεκριμένο API της Google ώστε να παρέχει εντοπισμό θέσης μέσα στην εφαρμογή του. Κάτι που πρέπει να τονιστεί εδώ είναι ότι ο προγραμματιστής συνήθως δεν μπορεί να τροποποιήσει την λειτουργικότητα του API (εκτός αν μιλάμε για κάποιο API ανοικτού κώδικα) πέρα από τις επιλογές που το ίδιο διαθέτει στον προγραμματιστή, ούτε να δει τον κώδικα που τρέχει όταν το χρησιμοποιεί. Ο προγραμματιστής απλά χρησιμοποιεί τις μεθόδους και τα εργαλεία που το API του παρέχει και περιμένει την καθορισμένη έξοδο που επιστρέφεται.

Ένα API που χρησιμοποιείται στην συγκεκριμένη πτυχιακή είναι το Google Speech API, το οποίο δέχεται ως είσοδο ένα τύπο αρχείου (wave ή flack) μαζί με κάποιες άλλες παραμέτρους όπως το ρυθμό δειγματοληψίας, τον αριθμό των καναλιών κλπ., επεξεργάζεται το αρχείο ήχου και το μετατρέπει σε κείμενο το οποίο επιστρέφει.

2.7 JSON

Τα αρχικά JSON προκύπτουν από το JavaScript Object Notation και το JSON αναπαριστά ένα **τρόπο αποθήκευσης και ανταλλαγής δεδομένων σε κείμενο** το οποίο χρησιμοποιεί τον τρόπο γραφής της γλώσσας JavaScript.

Όταν θέλουμε να ανταλλάξουμε δεδομένα μεταξύ server περιηγητή (browser) το μόνο που συνήθως μπορούμε να στείλουμε εύκολα και γρήγορα είναι το απλό κείμενο. Χρησιμοποιώντας λοιπόν το JSON μπορούμε να μετατρέψουμε οποιοδήποτε αντικείμενο και δομή δεδομένων σε JSON το οποίο μετά το στέλνουμε στον server, ο οποίος το αποκωδικοποιεί και μπορεί να το επεξεργαστεί.

Ένα παράδειγμα δημιουργίας ενός JSON αντικειμένου είναι το παρακάτω:

```
var myObj = { "name":"John", "age":31, "city":"New York" };
```

Εικόνα 2.12: Απλή μορφή JSON αντικειμένου

Όπως βλέπουμε υπάρχει μια συσχέτιση κλειδιού/τιμής που διαχωρίζονται μεταξύ τους με κόμμα. Αυτή είναι μια απλή μορφή ενός JSON αντικειμένου. Πέρα από αυτό ένα αντικείμενο JSON μπορεί να περιέχει και πίνακα τιμών ή εάν ολόκληρο αντικείμενο.

```
{
"employees":[ "John", "Anna", "Peter" ]
}
```

Εικόνα 2.13: JSON αντικείμενο το οποίο περιέχει πίνακα τιμών

```
{"employees":[
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"Anna", "lastName":"Smith" },
    { "firstName":"Peter", "lastName":"Jones" }
]}
```

Εικόνα 2.14: JSON αντικείμενο το οποίο περιέχει ένα άλλο αντικείμενο.

Αξίζει εδώ να σημειωθεί ότι ένα JSON αντικείμενο μπορεί να αρχίσει να γίνεται πολύπλοκο όταν αρχίζει να περιέχει πολύπλοκες δομές, δηλαδή πίνακα ο οποίος

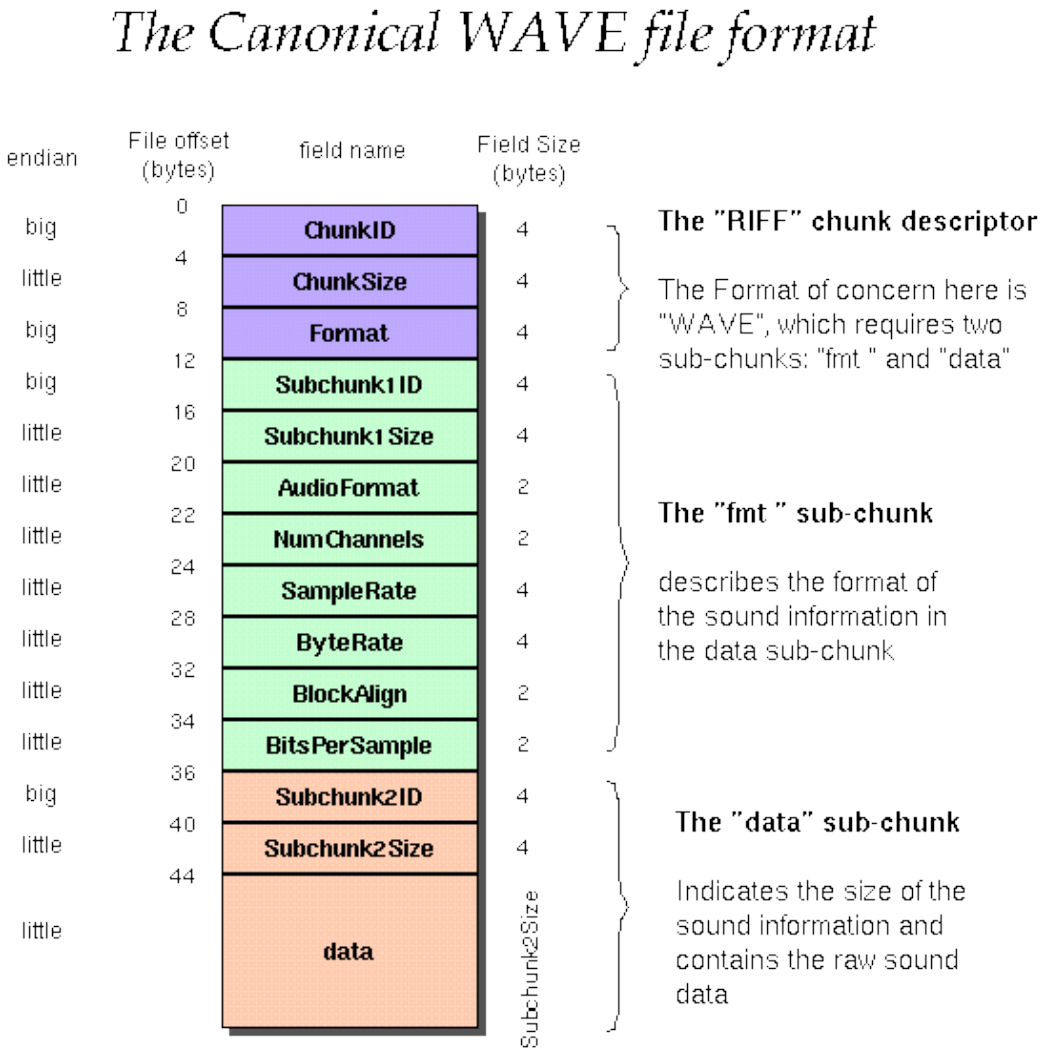
μέσα του έχει υποπίνακα ή αντικείμενα τα οποία εσωκλείουν και άλλα υποαντικείμενα ή/και υποαντικείμενα υποαντικειμένων.

Οι συναρτήσεις parse() και stringify()

Δύο πολύ σημαντικές συναρτήσεις όταν χρησιμοποιούμε JSON αντικείμενα είναι οι parse και stringify. Με την stringify μετατρέπουμε ένα JSON αντικείμενο σε κείμενο (string) το οποίο το κάνουμε συνήθως πριν στείλουμε τα δεδομένα, διότι δεν γίνεται να αποσταλούν σαν καθαρό JSON αντικείμενο. Αντίστοιχα, η μέθοδος parse παίρνει ως όρισμα ένα string αντικείμενο και επιστρέφει ένα αντικείμενο JSON (αρκεί το string που δώσαμε ως όρισμα να είναι ένα έγκυρο JSON αντικείμενο, όπως δείξαμε παραπάνω).

2.8: Πρότυπο .WAV αρχείου

Είναι σημαντικό να αναφερθούμε στη μορφή ενός .wav αρχείου το οποίο δημιουργούμε στο πρακτικό μέρος της εφαρμογής. Ένα αρχείο ήχου μορφής wave έχει την παρακάτω μορφή:



Εικόνα 2.15: Το πρότυπο του .wav αρχείου

Όπως βλέπουμε παραπάνω, ένα αρχείο ήχου μορφής wave αποτελείται από τρία μέρη (chunks). Το RIFF, το fmt και το data. Στην παραπάνω εικόνα φαίνεται πόσα byte πιάνει κάθε υποτμήμα καθενός από τα τρία παραπάνω τμήματα. Παρακάτω εξηγούμε αναλυτικά τι δεδομένα χρειάζεται το καθένα.

RIFF chunk

- **ChunkID:** Περιέχει το λεκτικό “RIFF” σε μορφή ASCII. Κάθε χαρακτήρας είναι 1 byte.
- **ChunkSize:** Αυτό το πεδίο περιέχει το μέγεθος του συνολικού αρχείου το οποίο είναι όλα τα τμήματα και τα υποτμήματα μαζί (το οποίο είναι 44 byte) συν το μέγεθος των δεδομένων που θα ηχογραφήσουμε από το μικρόφωνο του χρήστη.
- **Format:** Αυτό το πεδίο περιέχει το λεκτικό “WAVE” σε χαρακτήρες ASCII. Κάθε χαρακτήρας έχει μέγεθος 1 byte.

Format chunk

- **Subchunk1ID:** Περιέχει τους χαρακτήρες “fmt” συμπεριλαμβανομένου του κενού σε χαρακτήρες ASCII. Κάθε χαρακτήρας έχει μέγεθος 1 byte.
- **Subchunk1Size:** Περιέχει το μέγεθος, σε bytes του υπόλοιπου format chunk, τα υποτμήματα δηλαδή που ακολουθούν αμέσως μετά από αυτό το πεδίο.
- **Audio format:** Καθορίζει τον τύπο του αρχείου. Αν έχουμε αρχείο .wav όπως εδώ η τιμή αυτού του πεδίο θα είναι 1. Τιμές διαφορετικές του 1 υποδεικνύουν κάποια συμπίεση.
- **NumChannels:** Υποδεικνύει τον αριθμό των διαφορετικών καναλιών που χρησιμοποιούνται. Για ένα κανάλι (mono) η τιμή του πεδίου είναι ένα. Για δύο κανάλια (stereo) η τιμή είναι 2 κλπ.
- **SampleRate:** Περιέχει τον αριθμό των δειγμάτων ανά μονάδα χρόνου. Συνήθως σε δευτερόλεπτα.
- **ByteRate:** Περιέχει τον αριθμό των bit ανά δείγμα διαιρεμένος με το 8 (διότι 8 bit είναι ένα byte) και πολλαπλασιασμένος με τον αριθμό δειγματοληψίας που προαναφέραμε.
- **BlockAlign:** Περιέχει τον αριθμό των bit ανά δείγμα, διαιρεμένο με το 8 και πολλαπλασιασμένο με τον αριθμό των καναλιών.
- **BitsPerSample:** Περιέχει τον αριθμό των bit που χρησιμοποιούνται σε κάθε δείγμα. Για παράδειγμα, αν χρησιμοποιείτε 8 bit ανά δείγμα, η τιμή του πεδίου είναι 8.

Data chunk

- **Subchunk2ID:** Περιέχει το λεκτικό “data” σε μορφή ASCII. Κάθε χαρακτήρας καταλαμβάνει 1 byte.
- **SubchunkSize:** Περιλαμβάνει το μέγεθος των δεδομένων ήχου που καταχωρούνται στο αμέσως επόμενο πεδίο. Ο αριθμός αυτός θα μπορούσε να θεωρηθεί ίδιος με τον αριθμό των bits ανά δείγμα, διαιρεμένος με τον αριθμό τον δειγμάτων και πολλαπλασιασμένος με τον αριθμό των καναλιών.
- **Data:** Περιέχει τα ηχητικά δεδομένα.

Όλα τα πεδία μαζί έχουν σταθερό μέγεθος 44 byte όταν μιλάμε για απλό wave αρχείο. Επίσης θα πρέπει να δοθεί προσοχή στο endianness του κάθε πεδίου. Κάποια έχουν αρχιτεκτονική little endian ενώ άλλα big endian. Αυτό έχει να κάνει με το αν το σημαντικό ψηφίο κάθε αριθμού γράφεται στην αρχή ή στο τέλος του.

ΚΕΦΑΛΑΙΟ 3: RASPBERRY PI & RELAY MODULE

Το raspberry pi είναι ένας υπολογιστής σε μέγεθος πιστωτικής κάρτας. Αποτελείται ουσιαστικά από μία πλακέτα η οποία περιέχει όλα τα βασικά στοιχεία που χρειάζεται ένας υπολογιστής για να λειτουργήσει (επεξεργαστή, επεξεργαστή γραφικών, μνήμη Ram, μνήμη συστήματος, θύρες usb, υποδοχή Ethernet για ίντερνετ κλπ.) και αναπτύχθηκε με στόχο να προωθηθεί η γνώση των υπολογιστών σε ακόμη περισσότερα σπίτια, σχολεία αλλά και στις αναπτυσσόμενες χώρες κάνοντας την απόκτηση ενός υπολογιστή πολύ πιο εύκολη και οικονομική. Πρακτικά, το raspberry pi είναι ένας μικρός υπολογιστής ο οποίος μπορεί να χρησιμοποιηθεί σχεδόν για τα πάντα.
Για παράδειγμα :

- Κατασκευή μετεωρολογικού σταθμού
- Ρομποτική (πχ έλεγχος ρομποτικού βραχίονα)
- Υπολογιστή γενικής χρήσης
- Media Center
- Security System (ελέγχοντας μια απλή webcam)

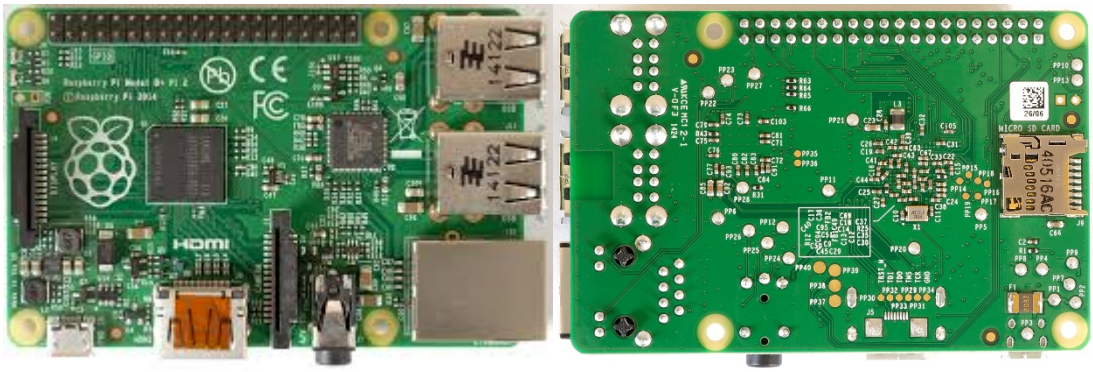
Όλα τα παραπάνω είναι επιτεύξιμα μέσω των GPIO Pins του raspberry pi τα οποία επιτρέπουν να συνδέσουμε πληθώρα αισθητήρων και άλλων συσκευών τις οποίες μπορούμε μετά να ελέγξουμε μέσω του raspberry pi. Θα αναφερθούμε σε αυτά πιο αναλυτικά παρακάτω.

3.1 Raspberry Pi 2 Model B+

Στην ενότητα αυτή παρουσιάζεται το μοντέλο της πλακέτας που χρησιμοποιήθηκε, καθώς και τα χαρακτηριστικά του.

3.1.1 Χαρακτηριστικά

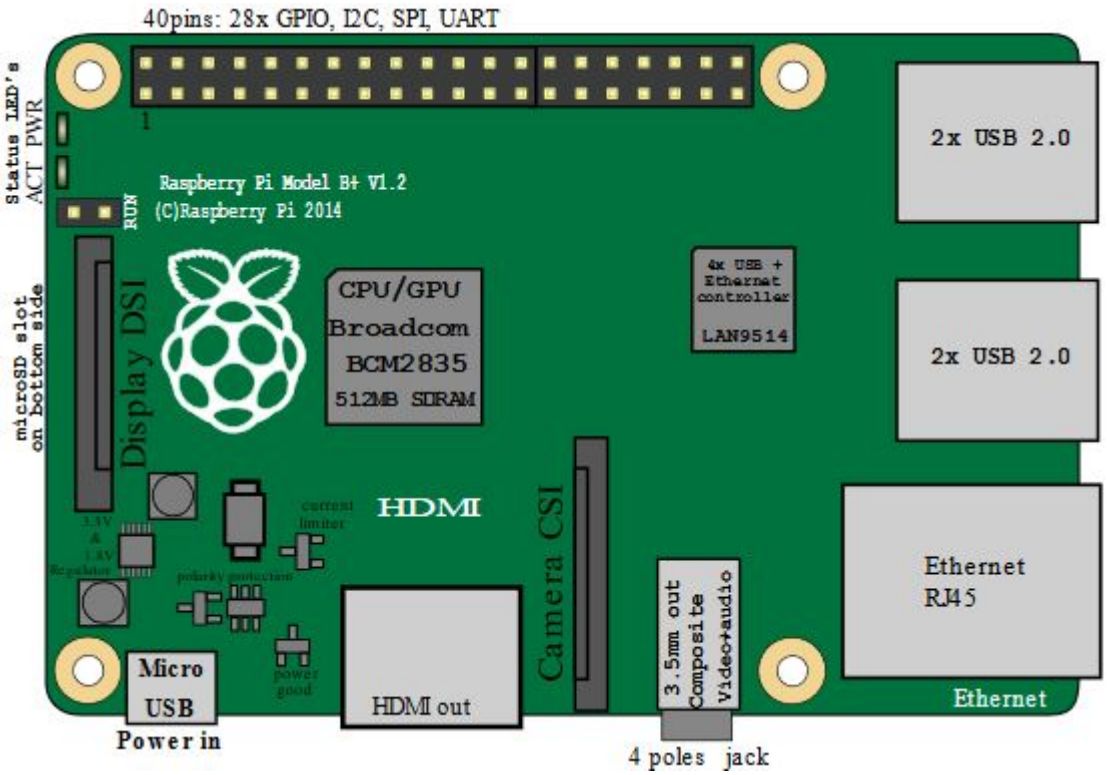
Το Raspberry Pi που χρησιμοποιήθηκε σε αυτή την εφαρμογή είναι το Raspberry Pi 2 Model B+ με τα κάτωθι χαρακτηριστικά.



Εικόνα 3.1: Εμπρόσθια και οπίσθια όψη της πλακέτας

Architecture	ARMv7-A (32-bit)
SoC	Broadcom BCM2836
CPU	900 MHz 32-bit quad-core ARM Cortex-A7
GPU	Broadcom VideoCore IV @ 250 MHz (BCM2837: 3D part of GPU @ 300 MHz, video part of GPU @ 400 MHz) OpenGL ES 2.0 (BCM2835, BCM2836: 24 GFLOPS / BCM2837: 28.8 GFLOPS) MPEG-2 and VC-1 (with license), 1080p30 H.264/MPEG-4 AVC high-profile decoder and encoder (BCM2837: 1080p60)
Memory (SDRAM)	1 GB (μοιράζεται με την GPU)
USB 2.0	4
Video input	15-pin MIPI camera interface (CSI)
Video outputs	HDMI (rev 1.3), composite video (3.5 mm TRRS jack), MIPI display interface (DSI) για LCD πάνελ
Audio inputs	I²S
Audio outputs	Αναλογική μέσω θύρας 3.5 mm, ψηφιακή μέσω HDMI και μέσω I²S
On-board storage	Θύρα MicroSDHC
On-board network	10/100 Mbit/s Ethernet (8P8C)
Low-level peripherals	40pins: 28x GPIO, I2C, SPI, UART
Power ratings	220 mA (1.1 W) κατά μέσο όρο σε κατάσταση αδράνειας, 820 mA (4.1 W) μέγιστη κατανάλωση υπό πίεση (πληκτρολόγιο και οθόνη συνδεδεμένα)

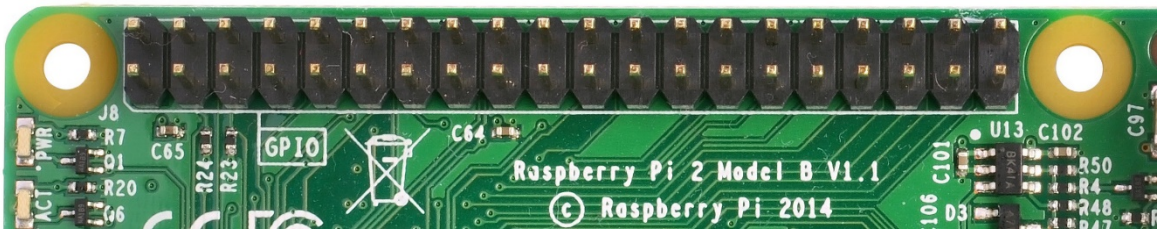
Power source	5 V μέσω MicroUSB ή κεφαλής GPIO
Size	85.60 mm × 56.5 mm (3.370 in × 2.224 in), δεν συμπεριλαμβάνονται τα βύσματα σύνδεσης τα οποία προεξέχουν.
Weight	45 g (1.6 oz)



Εικόνα 3.2: Σκαρίφημα μπροστινής όψης του raspberry pi 2 model B με επισήμανση των θυρών εισόδου και εξόδου

3.2 GPIO Pins

Ένα από τα πιο “δυνατά” και χρήσιμα χαρακτηριστικά του raspberry pi είναι οι ακροδέκτες (pin) GPIO (General Purpose Input Output) οι οποίοι βρίσκονται στο πάνω μέρος της πλακέτας.



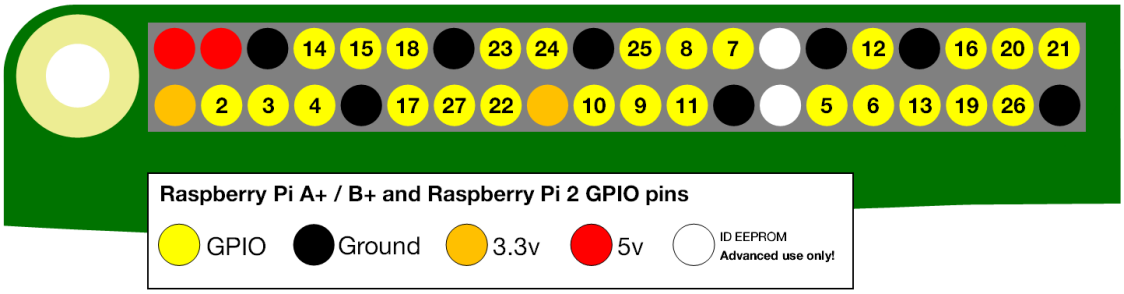
Εικόνα 3.3: GPIO pins

Οι ακροδέκτες αυτοί αποτελούν τη φυσική διεπαφή μεταξύ του raspberry και του έξω κόσμου. Στην πιο απλή τους εκδοχή μπορούν να οριστούν ως διακόπτες τους οποίους μπορούμε να ενεργοποιούμε η να απενεργοποιούμε προγραμματιστικά. Από τους συνολικά 40 ακροδέκτες, οι 26 είναι ακροδέκτες γενικού σκοπού, και τα υπόλοιπα είναι ακροδέκτες τροφοδοσίας και γείωσης (συν δύο ακροδέκτες ID EEPROM οι οποίοι προορίζονται μόνο για προχωρημένη χρήση).

3.2.2 Αρίθμηση και θέσεις των GPIO.

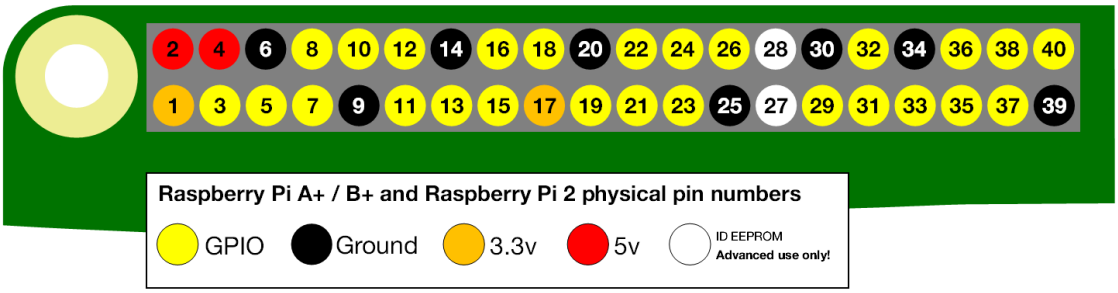
Όταν προγραμματίζουμε τα GPIO pins, υπάρχουν δύο διαφορετικοί τρόποι για να αναφερθούμε σε αυτά.

Ένας τρόπος είναι με την **αρίθμηση GPIO**. Αυτή η αρίθμηση αναφέρεται στα GPIO pins με τον τρόπο που τα βλέπει ο υπολογιστής. Η αρίθμηση με αυτόν τον τρόπο δεν βγάζει ιδιαίτερο νόημα στους ανθρώπους, διότι δεν είναι συνεχής και δεν έχει συγκεκριμένη σειρά, και έτσι είναι δύσκολο να τη θυμάται κάποιος. Η αρίθμηση αυτή φαίνεται στην παρακάτω εικόνα.



Εικόνα 3.6: Σκαρίφημα των GPIO pin με την αρίθμηση GPIO.

Ο δεύτερος τρόπος είναι η **φυσική αρίθμηση**. Αυτός ο τρόπος χρησιμοποιεί την συνηθισμένη σειριακή αρίθμηση που όλοι οι άνθρωποι χρησιμοποιούν και ξεκινάει από το pin 1 (αυτό που είναι πιο κοντά στην κάρτα SD) μετρώντας κατά μήκος και εναλλάξ πάνω κάτω καθώς κατεβαίνουμε.



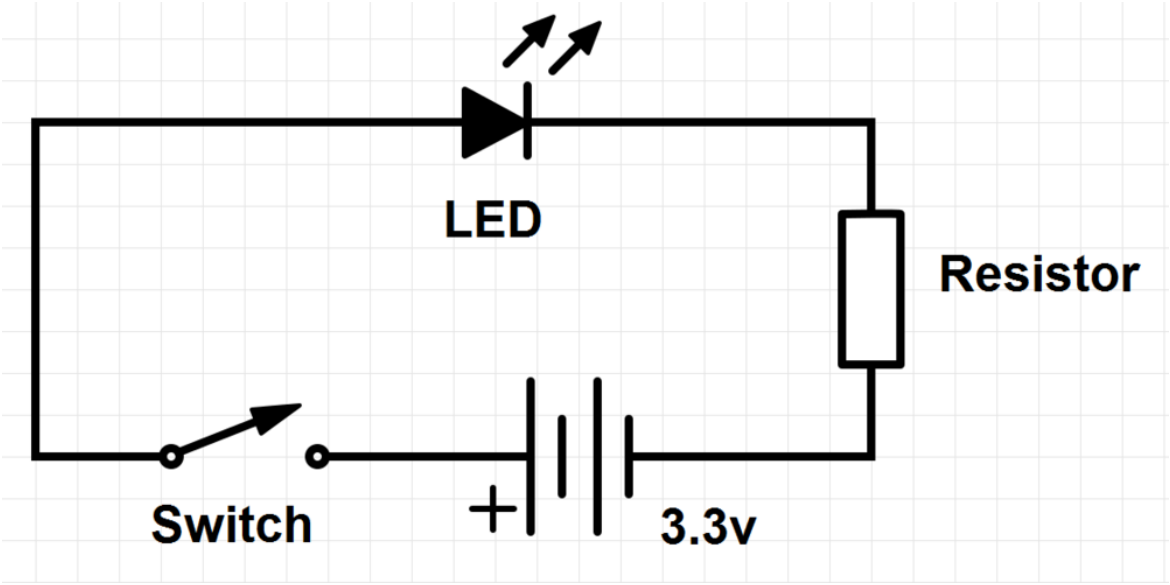
Εικόνα 3.7: Σκαρίφημα των GPIO pin με την φυσική αρίθμηση

Από τα δύο συστήματα, το σύστημα της **φυσικής αρίθμησης** αναφέρεται πιο πολύ στους αρχάριους, ενώ συστήνεται να χρησιμοποιείται το σύστημα αρίθμησης των GPIO. Όπως επίσης θα παρατηρήσατε, η αρίθμηση GPIO αριθμεί μόνο pins τα οποία είναι χρησιμοποιούμενα προγραμματιστικά και όχι όλα όπως κάνει η φυσική αρίθμηση.

3.2.1 Τρόποι λειτουργίας και παραδείγματα χρήσης των GPIO

Τα GPIO pins, μπορούν να χρησιμοποιηθούν τόσο σαν είσοδοι δεδομένων όσο και σαν έξοδοι. Στην παρούσα εφαρμογή χρησιμοποιούμε τα pins σαν εξόδους, οπότε θα αναλύσουμε αυτόν τον τρόπο χρήσης.

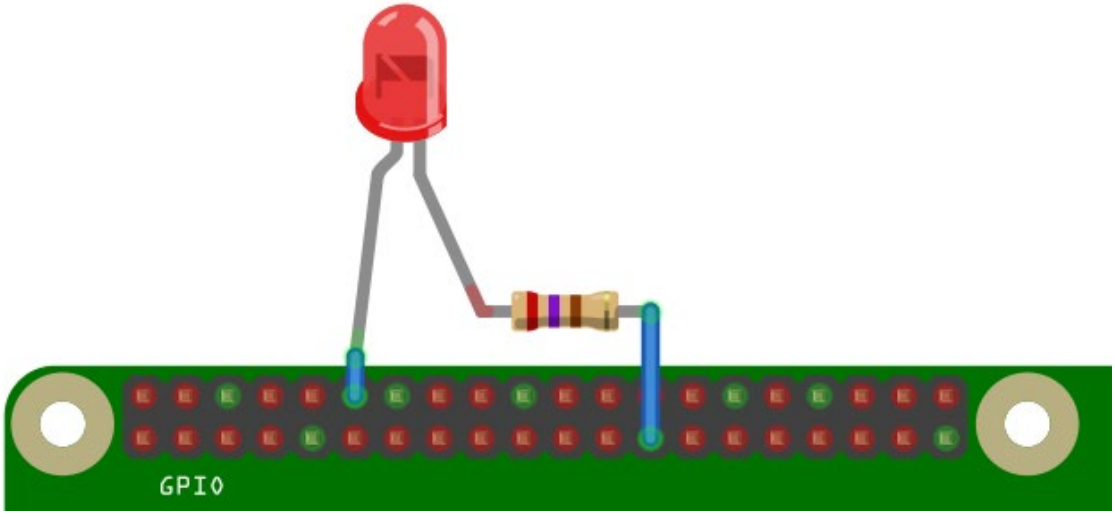
Έστω ότι έχουμε το παρακάτω απλό ηλεκτρικό κύκλωμα το οποίο αποτελείται από μία μπαταρία συνδεδεμένη σε ένα λαμπάκι (LED) μεταξύ των οποίων παρεμβάλλεται μία αντίσταση (προκειμένου να προστατέψει το λαμπάκι) και τέλος ένας διακόπτης για να ανοιγοκλείνουμε το κύκλωμα.



Εικόνα 3.4: Απλό ηλεκτρικό κύκλωμα με ένα led, τροφοδοσία και διακόπτη

Αν θέλουμε να φτιάξουμε το ίδιο κύκλωμα με την βοήθεια του raspberry pi, τότε αυτό αντικαθιστά **και τον διακόπτη και την μπαταρία** στο παραπάνω κύκλωμα. Κάθε ακίδα (pin) μπορεί να είναι ενεργοποιημένη ή απενεργοποιημένη, δηλαδή να έχει τάση (HIGH) ή όχι (LOW). Όταν το pin έχει τάση, η τάση που δίνει είναι 3.3 volt, ενώ όταν δεν έχει, δίνει τάση 0 volt.

Παρακάτω φαίνεται το αντίστοιχο κύκλωμα υλοποιημένο με τη χρήση του raspberry pi. Το LED είναι συνδεδεμένο σε μία GPIO ακίδα (η οποία δίνει σταθερή τάση 3,3 volt πχ η ακίδα νούμερο 1 με τη φυσική αρίθμηση) και στη γείωση (η οποία έχει τάση 0 volt και λειτουργεί σαν αρνητικός πόλος της μπαταρίας).

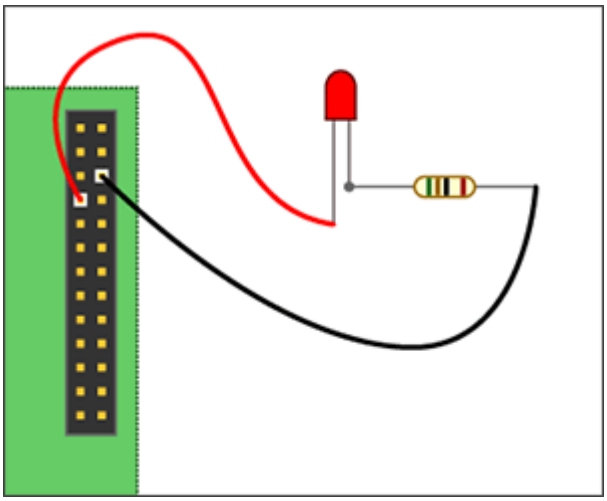


Εικόνα 3.5: Αντίστοιχο κύκλωμα με διακόπτη, τροφοδοσία και led με τη χρήση του raspberry pi.

Μόλις λοιπόν ενεργοποιήσουμε το raspberry pi μας το led αμέσως θα ανάψει και θα μείνει σταθερά αναμμένο.

Αν θέλαμε να ελέγξουμε προγραμματιστικά το raspberry pi χωρίς να πρέπει να ενεργοποιούμε και να απενεργοποιούμε το raspberry pi για να ανάψει και να σβήσει το led, θα πρέπει να γράψουμε τον κώδικα με τον οποίο θα το ελέγξουμε. Ο πιο εύκολος τρόπος να το κάνουμε, είναι με τη χρήση ενός κώδικα (script) γραμμένο σε python. Επίσης θα πρέπει να αλλάξουμε θέση στην άνοδο του led (το θετικό και μακρύτερο ποδαράκι) και να το βάλουμε σε ένα GPIO pin το οποίο προγραμματίζεται, όπως το Pin με τον αριθμό 7 (με τη χρήση της φυσικής αρίθμησης) το οποίο θα χρησιμοποιήσουμε στο παράδειγμά μας.

Έχοντας λοιπόν ένα κύκλωμα όπως το παρακάτω :



Εικόνα 3.6: Κύκλωμα για έλεγχο του led μέσω κώδικα

Γράφουμε τον εξής κώδικα:

```
import RPi.GPIO as GPIO ## Εισάγουμε την βιβλιοθήκη GPIO
GPIO.setmode(GPIO.BOARD) ## Δήλωση χρήσης φυσικής αρίθμησης των GPIO
GPIO.setup(7,GPIO.out) ## Δήλωση του pin 7 ως pin εξόδου
GPIO.output(7,True) ## Ενεργοποιούμε το pin 7 το οποίο δίνει τάση
```

Αφού λοιπόν γράψουμε το παρακάτω πρόγραμμα σε έναν κειμενογράφο και το αποθηκεύσουμε με την κατάληξη .py (ώστε ο υπολογιστής να καταλάβει ότι πρόκειται για κώδικα python) το τρέχουμε και θα δούμε το led να ανάβει. Αν αντίστοιχα θέλαμε να το απενεργοποιήσουμε χωρίς να κλείσουμε το raspberry pi, το μόνο που θα έπρεπε να κάνουμε είναι να αλλάξουμε την τελευταία εντολή και να πούμε στο raspberry pi να σταματήσει να έχει το Pin 7 σε λογικό 1 και να δίνει στην έξοδο αυτή λογικό 0, κάτι το οποίο θα ήταν αντίστοιχο του να ανοίξουμε τον διακόπτη στο προηγούμενο κύκλωμά μας χωρίς το raspberry pi (εικόνα 3.4).

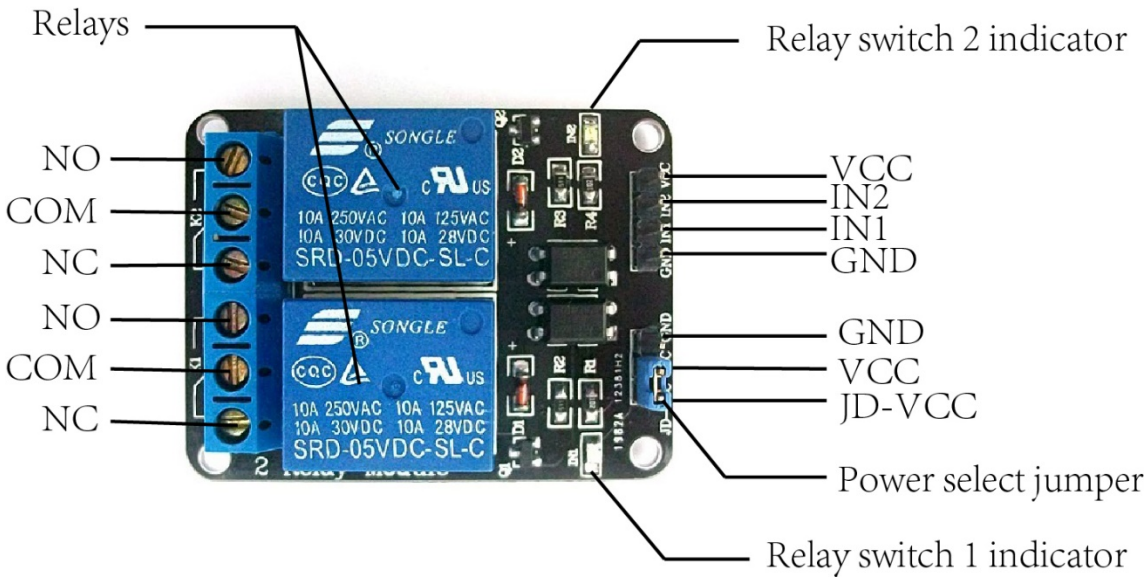
Το πρόγραμμα για την απενεργοποίηση του led θα είχε ως εξής:

```
import RPi.GPIO as GPIO ## Εισάγουμε την βιβλιοθήκη GPIO
GPIO.setmode(GPIO.BOARD) ## Δήλωση χρήσης φυσικής αρίθμησης των GPIO
GPIO.setup(7,GPIO.out) ## Δήλωση του pin 7 ως pin εξόδου
GPIO.output(7,False) ## Απενεργοποιούμε το pin 7 το οποίο δίνει τάση
```

Αν παρατηρήσουμε και συγκρίνουμε τον κώδικά που ανάβει το led με αυτόν που το σβήνει, θα δούμε ότι ουσιαστικά το μόνο που αλλάζει είναι η τελευταία γραμμή στην οποία ο ένας κώδικας υπαγορεύει την ενεργοποίηση του Pin 7 σε λογικό 1 ενώ ο άλλος σε λογικό 0 το οποίο σταματά το Pin 7 από το να δίνει τάση στο κύκλωμα και είναι σαν να “ανοίγει” τον διακόπτη.

3.3 Raspberry Pi Relay Module

Προκειμένου να μπορέσουμε να ελέγξουμε τις συσκευές μέσω του raspberry pi, χρησιμοποιούμε ένα ρελέ (relay) το οποίο μπορεί να τροφοδοτήσει συσκευές με μεγάλες τάσεις και το οποίο ελέγχεται προγραμματιστικά μέσω των GPIO pins του raspberry pi τα οποία αναφέραμε παραπάνω. Το module το οποίο θα χρησιμοποιήσουμε είναι το παρακάτω:



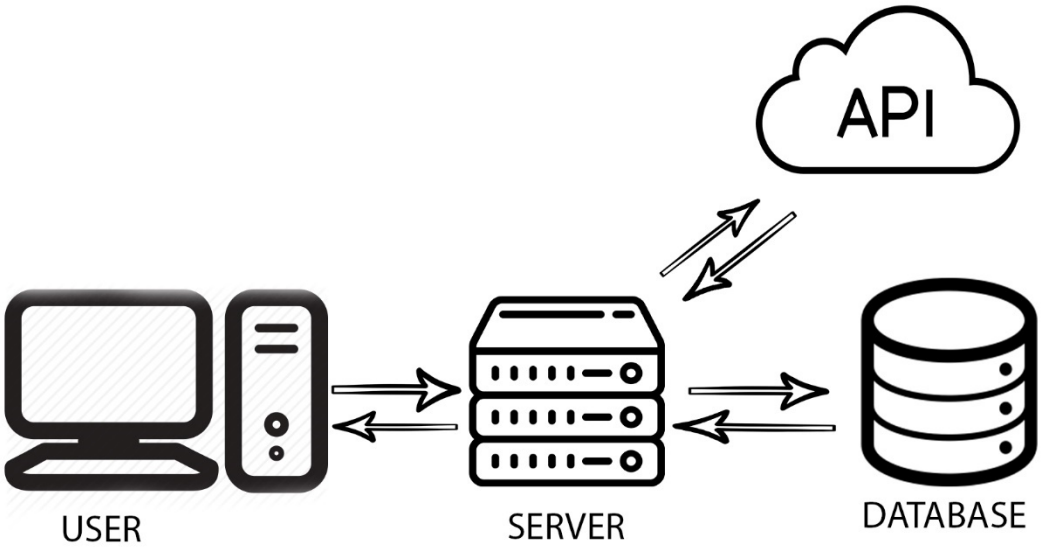
Εικόνα 3.7: Το relay module το οποίο θα χρησιμοποιηθεί για τον έλεγχο των συσκευών.

Το παραπάνω relay module αποτελείται από δύο ρελέ τα οποία ελέγχονται από το raspberry pi. Πιο αναλυτικά:

- **COM:** Αυτός η επαφή είναι ο “κοινός”. Αποτελεί πάντα το ένα από τα δύο άκρα του κυκλώματός μας. Όταν το ρελέ δεν έχει ενεργοποιηθεί, ο κοινός δημιουργεί κύκλωμα με το NC. Ενώ όταν το ρελέ ενεργοποιηθεί, τότε ο κοινός κλείνει κύκλωμα με το NO.
- **NO:** Αυτό είναι το άκρο του ρελέ το οποίο είναι συνήθως ανοικτό (Normally Open). Όταν ενεργοποιήσουμε το ρελέ το άκρο αυτό θα κλείσει κύκλωμα με τον κοινό (CO).
- **NC:** Το άκρο αυτό είναι συνήθως κλειστό, δηλαδή κλείνει κύκλωμα με τον κοινό (NC). Όταν το ρελέ ενεργοποιηθεί, τότε ο κοινός θα κλείσει κύκλωμα με το άλλο άκρο (NO) και το κύκλωμα αυτό θα είναι ανοικτό.
- **VCC:** Εδώ συνδέουμε το ρελέ στα 5V του raspberry pi.

- **GND:** Σε αυτό το Pin συνδέουμε τη γείωση του raspberry pi
- **IN1:** Αυτό το Pin ελέγχει το ένα από τα δύο ρελέ. Όταν δώσουμε τάση από το raspberry pi μας σε αυτό το pin τότε το ρελέ θα ενεργοποιηθεί ενώ όταν δώσουμε λογικό 0 θα γυρίσει στην προηγούμενη κατάσταση.
- **IN2:** Πραγματοποιεί την ίδια λειτουργία με το IN1 για το δεύτερο ρελέ.

ΚΕΦΑΛΑΙΟ 4: ΠΡΑΚΤΙΚΗ ΥΛΟΠΟΙΗΣΗ



Εικόνα 4 : Αρχιτεκτονική δομή της πρακτικής υλοποίησης

Στην παρούσα πτυχιακή, ο χρήστης συνδέεται σε μια ιστοσελίδα που φορτώνεται τοπικά από το raspberry pi (το οποίο είναι ο τοπικός μας server). Με το που συνδεθεί, θα ερωτηθεί εάν θέλει να αφήσει την εφαρμογή να χρησιμοποιήσει το μικρόφωνο που είναι συνδεδεμένο στη συσκευή του. (ο χρήστης μπορεί να μπει από οποιαδήποτε συσκευή έχει περιηγητή). Από εκεί έχει τη δυνατότητα να επιλέξει τη γλώσσα στην οποία θα δώσει τις φωνητικές εντολές, (οι διαθέσιμες γλώσσες είναι αγγλικά και ελληνικά), και πατώντας το αντίστοιχο κουμπί ξεκινάει να ηχογραφείται. Όταν ο χρήστης δώσει την φωνητική εντολή, πρέπει να πατήσει το κουμπί παύσης της ηχογράφησης, προκειμένου η εντολή του να μετατραπεί μέσω του API Google Speech της Google, το οποίο χρησιμοποιούμε για να μετατρέψουμε τις φωνητικές μας εντολές σε κείμενο. Όταν η εντολή μετατραπεί σε κείμενο, αποστέλλεται στον server ο οποίος επικοινωνεί με τη βάση δεδομένων του και συγκρίνει την εντολή με τις διαθέσιμες εντολές που έχει στη βάση και προσπαθεί να αναγνωρίσει με ποια ταιριάζει περισσότερο. Αν βρεθεί εντολή με ποσοστό ομοιότητας πάνω από 85%, με αυτή που είπε ο χρήστης, τότε ο server, τερματίζει

την αναζήτηση και εκτελεί την αντίστοιχη εντολή, η οποία μέσω των GPIO pins του raspberry pi τα οποία επικοινωνούν με ένα ρελέ (έτσι ώστε να μπορέσουμε να ελέγξουμε συσκευές με μεγάλη κατανάλωση όπως πχ ένας βραστήρας) και ελέγχοντας το ρελέ, ελέγχουν και την συσκευή η οποία είναι συνδεδεμένη πάνω του.

4.1 Επεξήγηση του κώδικα

Ο κώδικας χωρίζεται σε τρία μέρη τα οποία είναι:

- Το front-end της εφαρμογής μας και εμπεριέχει τον κώδικα που δημιουργεί την αρχική σελίδα που βλέπει ο χρήστης και που ηχογραφεί και εμφανίζει στο χρήστη τα αποτελέσματα της ηχογράφησης.
- Το back-end της εφαρμογής, το οποίο αντιστοιχεί στον κώδικα που εκτελείται στον διακομιστή (server) και είναι υπεύθυνος για τους ελέγχους των φωνητικών εντολών με τη βάση και την εγκυρότητα αυτών, καθώς και την εκτέλεση των τελικών script όταν η εντολή είναι έγκυρη και υπάρχει στη βάση.
- Τα script τα οποία τρέχουν για να εκτελέσουν την κάθε εντολή που έχουμε αποθηκευμένη στη βάση μας.

Αρχικά όταν θέλουμε να τρέξουμε την εφαρμογή θα ζητήσουμε από τον περιηγητή (browser) να φέρει την αντίστοιχη σελίδα στην οποία γίνεται η καταγραφή των εντολών και η παρουσίαση των αποτελεσμάτων. Ο χρήστης πατώντας το κουμπί record ξεκινά να ηχογραφεί την εντολή που θα δώσει, και μόλις τελειώσει, πατάει το κουμπί stop record και στη συνέχεια εμφανίζονται στα τρία πλαίσια των αποτελεσμάτων η εντολή που έδωσε ο χρήστης σε κείμενο, η εντολή που είναι αποθηκευμένη στη βάση δεδομένων και το αντίστοιχο ποσοστό ομοιότητας μεταξύ των εντολών. Η σελίδα αυτή είναι η VoiceCommands.html και ο κώδικας της σελίδας αυτής, αποτελείται από τρία μέρη τα οποία θα αναλυθούν ξεχωριστά. Το html κώδικα, τον CSS κώδικα και τον JavaScript κώδικα.

4.1.1 Front – End, Html Κώδικας

Το πρώτο κομμάτι που θα αναλύσουμε είναι ο html κώδικας ο οποίος ουσιαστικά εμπεριέχεται εντός των ετικετών <body></body>.

```
29      <body>
30
31          <div align = "center">
32              <div align = "center" class = "body">
33                  <p>Πατήστε record για να ηχογραφήσετε την εντολή και stop recording για να σταματήσετε την ηχογράφηση.</p>
34                  <input type = "button" value = "record" id = "rec">
35                  <input type = "button" value = "stop recording" id = "stopRec">
36              </div>
37              <p align="center"> Εντολή Χρήστη</p>
38              <div id="result" class="displaybox"> </div>
39              <p align="center"> Αποθηκευμένη Εντολή</p>
40              <div id="command" class="displaybox"> </div>
41              <p align="center"> Ποσοστό Ομοιότητας</p>
42              <div id="percent" class="displaybox"> </div>
43              <form>
44                  <input type="radio" id = "en" name="language" value="English" checked> English<br>
45                  <input type="radio" id = "el" name="language" value="Greek">Ελληνικά<br>
46              </form>
47          </div>
48      </body>
```

Εικόνα 4.1: Html κώδικας της σελίδας VoiceCommands

Με αυτόν τον κώδικα δημιουργείται η βασική εμφάνιση της σελίδας. Ξεκινώντας να αναλύουμε τον κώδικα από πάνω προς τα κάτω, αρχικά ορίζουμε ένα στοιχείο ** ** το οποίο λειτουργεί σαν ένα πλαίσιο μέσα στο οποίο θα δημιουργηθεί όλη η σελίδα. Το χαρακτηριστικό align ορίζει την στοίχιση του ίδιου του στοιχείου και του περιεχομένου του καθώς και η τιμή **“center”** που τις δίνουμε ορίζει ότι το πλαίσιο θα είναι στοιχισμένο στη μέση της σελίδας. Στη συνέχεια ορίζουμε άλλο ένα πλαίσιο στο οποίο δίνουμε την κλάση body. Η κλάση αυτή δίνει συγκεκριμένη μορφοποίηση την οποία έχουμε ορίσει με css και αναλύουμε παρακάτω. Με το στοιχείο **<p>** δημιουργούμε μια παράγραφο στη σελίδα. Το στοιχείο **<input>** χρησιμοποιείται για να δηλώσει ένα πεδίο εισαγωγής δεδομένων. Στο συγκεκριμένο σημείο ορίζουμε ότι το πεδίο εισαγωγής θα είναι στην πραγματικότητα το κουμπί έναρξης της ηχογράφησης (**type=”button”**) του οποίου το κείμενο θα λέει “record” το οποίο ορίζουμε με το χαρακτηριστικό **value=”record”** και δίνουμε ένα μοναδικό αναγνωριστικό (id) στο κουμπί μας με τιμή rec. Αντίστοιχα κάνουμε και για το κουμπί παύσης της ηχογράφησης. Στη συνέχεια δημιουργούμε τρείς παραγράφους και κάτω από την κάθε μια, ένα ορθογώνιο πλαίσιο στο οποίο δίνουμε την κλάση **class=”displaybox”** και ένα διαφορετικό αναγνωριστικό (id) στο καθένα. Εντός αυτών των πλαισίων, θα εμφανίζεται στον χρήστη το αντίστοιχο μήνυμα, δηλαδή ένα πλαίσιο για την εντολή που έδωσε ο χρήστης, ένα πλαίσιο για την αντίστοιχη εντολή που βρήκε το πρόγραμμα στη βάση, καθώς και το ποσοστό ομοιότητάς (σε περίπτωση που υπερβαίνει το 80%. Τέλος δημιουργούμε δύο ακόμη κουμπιά τα

οποία είναι **type=”radio”** και είναι κουμπιά επιλογής γλώσσας με τα αντίστοιχα αναγνωριστικά και το κείμενο του κάθε κουμπιού. Το χαρακτηριστικό **checked** μας δείχνει ποιο από τα δύο κουμπιά θα είναι επιλεγμένα από πριν μόλις φορτωθεί η σελίδα.

4.1.2 Front – End, CSS Κώδικας

Το δεύτερο κομμάτι είναι ο css κώδικας ο οποίος μας επιτρέπει να διαμορφώσουμε την εμφάνιση της σελίδας μας.

```
7       <style>
8           .body
9           {
10              background: #fff;
11              font-family:Verdana,sans-serif;
12              font-size: 13pt;
13              font-weight: normal;
14          }
15          .displaybox
16          {
17              width: 300px;
18              height: 50px;
19              background-color: #ffffff;
20              border: 2px solid #000000;
21              line-height: 2.5em;
22              margin-top: 25px;
23              font-size: 12pt;
24              font-weight: bold;
25
26          }
27      </style>
```

Εικόνα 4.2: CSS κώδικας της σελίδας VoiceCommands

Εδώ όπως βλέπουμε μέσα στις ετικέτες <style></style> βρίσκονται οι δύο κλάσεις (body και displaybox) που αναφέραμε πριν μέσα στον html κώδικα. Αρχικά να σημειωθεί ότι κάθε κλάση που δημιουργούμε στο CSS ξεκινάει με την τελεία (.).

1. .body
   - background: ορίζουμε το background χρώμα της κλάσης σε άσπρο (το #fff αντιστοιχεί στην δεκαεξαδική τιμή του άσπρου χρώματος).
   - font-family: Ορίζουμε την γραμματοσειρά που θα χρησιμοποιεί αυτή η κλάση. Στο συγκεκριμένο παράδειγμα ορίσαμε δύο, που σημαίνει ότι εάν

για οποιονδήποτε λόγο δεν υποστηρίζεται η πρώτη από τον περιηγητή, θα εφαρμοστεί η δεύτερη.

- font-size: Ορίζει το μέγεθος της γραμματοσειράς. Η μονάδα μέτρησης είναι σε pt (point) όπου 1pt αντιστοιχεί στο 1/72 της ίντσας.
- font-weight: Το “βάρος” της γραμματοσειράς, όπου ουσιαστικά ορίζει εάν η γραμματοσειρά μας θα είναι bold ή semi-bold κλπ.

2. .displaybox
   - width: Ορίζει το πλάτος, το οποίο αντιστοιχεί σε 300px (pixel).
   - height: Ορίζει το ύψος το οποίο αντιστοιχεί σε 50px (pixel).
   - background-color: ορίζει το χρώμα που θα υπάρχει στο background όπου εδώ χρησιμοποιούμε την δεκαεξαδική τιμή του άσπρου.
   - border: Ορίζει τα περιθώρια του κάθε πλαισίου τα οποία θα έχουν πάχος 2px (pixel) σε κάθε μεριά, η γραμμή θα είναι μονοκόμματη (solid) και το χρώμα των ορίων είναι μαύρο (στην δεκαεξαδική του τιμή).
   - line-height: Ορίζει το ύψος γραμμής του κειμένου σε μονάδα μέτρησης em. Η συγκεκριμένη μονάδα μέτρησης δεν έχει σταθερό μέγεθος από τη φύση της και προσαρμόζεται κάθε φορά στο τρέχον μέγεθος της τρέχουσας γραμματοσειράς. Για παράδειγμα 1em μπορεί να είναι ίσο με 12 pt ή άλλες φορές να είναι ίσο με 24pt.
   - margin-top: Το περιθώριο που αφήνει η κλάση στο πάνω μέρος της, όπου εδώ αντιστοιχεί σε 25px.
   - font-size: Ομοίως με παραπάνω.
   - font-weight: Ομοίως με παραπάνω.

4.1.3 Front – End, JavaScript κώδικας

Το τρίτο κομμάτι του front – end αποτελεί το τμήμα κώδικα σε JavaScript το οποίο φαίνεται παρακάτω και βρίσκεται εντός των ετικετών <script></script>. Λόγω του μεγέθους του κώδικα, θα παρατίθεται σε μικρά κομμάτια και θα εξηγείται στο καθένα. Ολόκληρος ο κώδικας παρατίθεται στο παράρτημα Α.

```
48          var audioCntxt;
49          var sampleRate;
50          var recorder = null;
51          var recording = false;
52          var monochannel = [];
53          var recordingLength = 0;
54          var voiceCommand;
55          if (!navigator.getUserMedia)
56              navigator.getUserMedia = navigator.getUserMedia || navigator.webkitGetUserMedia ||
57                                       navigator.mozGetUserMedia || navigator.msGetUserMedia;
58
59          if (navigator.getUserMedia){
60              navigator.getUserMedia({audio:true}, success, function(e) {
61              alert('Error capturing audio.' + e);
62              console.log(e);
63              });
64          } else alert('getUserMedia not supported in this browser.');
65
```

Εικόνα 4.3: Κομμάτι κώδικα JavaScript 0/9. Η μέθοδος navigator.getUserMedia()

Ξεκινώντας λοιπόν τον JavaScript κώδικα, αρχικά ορίζουμε τις μεταβλητές μας. Υπενθυμίζεται ότι στην JavaScript δεν χρειάζεται να ορίσουμε τον τύπο της μεταβλητής. Η γλώσσα το καταλαβαίνει αυτόματα από τον τύπο των δεδομένων που ανατίθεται.

Η μέθοδος navigator.getUserMedia()

Η μέθοδος αυτή χρησιμοποιείται για να συλλάβει τον ήχο ή/και την εικόνα από το μικρόφωνο ή/και την κάμερα του χρήστη (εφόσον έχει). Η μέθοδος αυτή σιγά σιγά έχει αρχίσει να αντικαθίσταται από την navigator.mediaDevices.getUserMedia() η οποία όμως δεν υποστηρίζεται πλήρως στο διάστημα συγγραφής αυτής της πτυχιακής εργασίας. Αρχικά λοιπόν ελέγχουμε εάν η μέθοδος αυτή υποστηρίζεται από τον browser. Αν όχι, δοκιμάζουμε κάποια άλλα προθέματα (prefixes) μέσω των οποίων πολλοί browser υποστηρίζουν αυτή τη μέθοδο. Γενικά κάθε browser δεν υποστηρίζει τις ίδιες μεθόδους με τον ίδιο τρόπο για την κατασκευή ιστοσελίδων. πχ ο Mozilla χρησιμοποιεί την mozGetUserMedia() αντί για την προαναφερόμενη μέθοδο. Αφού λοιπόν βρούμε την ίδια ή κάποια παραλλαγή της που να υποστηρίζεται, την καλούμε περνώντας της τρία ορίσματα. Τον τύπο πολυμέσων που θέλουμε (εδώ ζητάμε μόνο τον ήχο άρα audio:true), μια μέθοδο που θα εκτελεστεί σε περίπτωση που η κλήση της γίνει με επιτυχία (την οποία ονομάσαμε success), και τέλος μία μέθοδο που θα εκτελεστεί σε περίπτωση που παρουσιαστεί κάποιο πρόβλημα η οποία μας τυπώνει ένα μήνυμα και μας εμφανίζει το σφάλμα. Αν η μέθοδος τελικά δεν υποστηρίζεται, εμφανίζουμε αντίστοιχο μήνυμα στο χρήστη (στο κομμάτι κώδικα else).

Η μέθοδος success

Παρακάτω αναλύουμε την μέθοδο success η οποία θα τρέξει μετά από μια επιτυχημένη κλήση της μεθόδου getUserMedia.

```
67          function success(stream)
68          {
69              console.log ("success is enabled");
70              //we create an audioContext (sth like an Audio canvas),
71              audioCntxt = new (window.AudioContext || window.webkitAudioContext)();
72              //sampleRate is by default at 44100Hz
73              sampleRate = audioCntxt.sampleRate;
74              console.log ('sample rate is'+sampleRate);
75              //window.source is used in firefox only due to a bug
76              window.source = audioCntxt.createMediaStreamSource(stream);
77              var volume = audioCntxt.createGain();
78              volume.value = 1;
79              window.source.connect(volume);
80
81
82
83              var bufferSize = 2048;
84              //creates a script processor node. params are: buffer size, input channels, output channels
85              recorder = audioCntxt.createScriptProcessor(bufferSize, 1, 1);
86              //every time buffersize fills out, this function is called
87              recorder.onaudioprocess = function (e)
88              {
89                  if (!recording) return;
90                  var mono =  e.inputBuffer.getChannelData (0);
91                  monochannel.push (new Float32Array(mono));
92                  recordingLength += bufferSize;
93              }
94              volume.connect(recorder);                                   //here we connect the recorder
95
96          }
```

Εικόνα: 4.4: Κομμάτι κώδικα JavaScript 1/9. Η μέθοδος success.

Αρχικά δημιουργούμε ένα audioContext (πάλι δοκιμάζοντας διαφορετικά prefixes) το οποίο είναι κάτι σαν ένας καμβάς, αλλά για ήχο. Μέσα σε ένα audioContext μπορούμε να κάνουμε πολλά πράγματα με το stream του ήχου που παίρνουμε από το μικρόφωνο του χρήστη, όπως οπτικοποίηση δεδομένων (πχ vu meter) κλπ. Στη συνέχεια παίρνουμε το ρυθμό δειγματοληψίας από τον audioContext το οποίο χρησιμοποιεί by default 44100Hz. Κατόπιν παίρνουμε τη ροή (stream) του ήχου που ηχογραφούμε από το χρήστη και μπορούμε πλέον να τη χρησιμοποιήσουμε όπως θέλουμε. Αξίζει εδώ να αναφερθεί ότι λόγω ενός bug στον browser Mozilla, πρέπει να συσχετίσουμε αυτή το stream ήχου με το παράθυρο της εφαρμογής προκειμένου η διάρκεια ζωής του αντικειμένου να είναι όση και του παραθύρου της εφαρμογής (δηλαδή όταν ο χρήστης κλείσει το παράθυρο, τότε να διακόπτεται το stream του ήχου) διότι ο Mozilla δυστυχώς μετά από ένα συγκεκριμένο χρονικό διάστημα σταματούσε τη ροή του ήχου. Τέλος μπορούμε χρησιμοποιώντας το audioContext που αναφέραμε πριν, να φτιάξουμε διάφορους κόμβους (nodes) επεξεργασίας του stream του ήχου. Για παράδειγμα εδώ δημιουργούμε ένα node έντασης (volume) στο οποίο δίνουμε τιμή και μετά το

ενώνουμε με το window.source με το οποίο έχουμε συνδέσει την εισερχόμενη ροή (stream) ήχου. Θα μπορούσαμε να δημιουργήσουμε και άλλα nodes εάν θέλαμε να επεξεργαστούμε τον ήχο αυτό, όπως ένα βαθυπερατό φίλτρο το οποίο θα έκοβε τις υψηλές συχνότητες.

Στη συνέχεια, δημιουργούμε έναν buffer που του ορίζουμε μέγεθος ίσο με 2048 bytes. Έπειτα δημιουργούμε έναν εγγραφέα ήχου (recorder) που θα καταγράφει και θα αποθηκεύει το stream ήχου που έρχεται από το μικρόφωνο του χρήστη. Για να το καταφέρουμε αυτό χρησιμοποιούμε την μέθοδο createScriptProcessor του audioContext η οποία παίρνει σαν ορίσματα τον buffer Που δημιουργήσαμε, αριθμό καναλιών εισόδου, και αριθμό καναλιών εξόδου. Αφού φτιάξουμε τον recorder, καθορίζουμε την μέθοδο που θα εκτελείτε κάθε φορά που θα γεμίζει ο buffer. Αυτή η μέθοδος λέγεται onaudioproccess. Όταν καλείται, ελέγχει αρχικά την μεταβλητή recording ώστε να ξέρει ότι όντως γίνεται ηχογράφηση, αλλιώς η μέθοδος σταματά. Αν η μεταβλητή είναι αληθής (true), θα παίρνουμε τα δεδομένα που έχει ο buffer και τα αποθηκεύουμε στην μεταβλητή monochannel με μορφή πινάκων του οποίου τα δεδομένα αναπαρίστανται με αριθμούς κινητής υποδιαστολής 32-bit, το οποίο επιτυγχάνεται με την εντολή new **Float32Array**. Τέλος στην recordingLength προσθέτουμε τον εαυτό της συν το μέγεθος του buffer προκειμένου να ξέρουμε συνολικά πόσα byte ήταν όλη η ηχογράφηση. Τέλος συνδέουμε το stream στο node του recorder που μόλις δημιουργήσαμε.

Ηχογράφηση της εντολής – το κουμπί record

Καθώς ο χρήστης θα πατήσει το κουμπί για να ξεκινήσει η ηχογράφηση της εντολής, θα πρέπει κληθεί η παρακάτω μέθοδος η οποία “πυροδοτείτε” από το πάτημα του κουμπιού record.

```
197     document.getElementById('rec').onclick = function()
198     {
199         console.log ('recording now!');
200         recording = true;
201         monochannel.length =0;
202
203     }
```

Εικόνα 4.5: Κομμάτι κώδικα JavaScript 2/9. Μέθοδος εκκίνησης ηχογράφησης

Όταν λοιπόν το κουμπί record πατηθεί, εκτυπώνουμε ένα μήνυμα στην κονσόλα ότι ξεκίνησε η ηχογράφηση και θέτουμε την μεταβλητή recording την οποία προαναφέραμε πριν σε αληθής (true) και το μήκος το πίνακα monochannel σε 0 διότι ξεκινά μια καινούργια ηχογράφηση.

Ολοκλήρωση της ηχογράφησης – το κουμπί stop record

Μόλις ο χρήστης ολοκληρώσει την φωνητική του εντολή και πατήσει το κουμπί stop record τρέχει η παρακάτω μέθοδος. (Λόγω του μεγέθους της, θα παρατίθεται και θα εξηγείται σε κομμάτια).

```
110         document.getElementById('stopRec').onclick = function()
111         {
112             console.log("recorder stopped");
113             recording = false;
114             var monoBuffer = mergeArrays(monochannel,recordingLength);
115             // we create our wav file
116             var buffer = new ArrayBuffer(44 + monoBuffer.length*2);
117             var view = new DataView(buffer);
```

Εικόνα 4.6: Κομμάτι κώδικα JavaScript 3/9. Μέθοδος λήξης ηχογράφησης

Μόλις ο χρήστης πατήσει το κουμπί stop record εκτελείται η παραπάνω μέθοδος. Αρχικά εμφανίζουμε ένα μήνυμα στην κονσόλα ότι η ηχογράφηση σταμάτησε και θέτουμε τη μεταβλητή recording την οποία προαναφέραμε σε ψευδή (false). Έπειτα ορίζουμε την μεταβλητή monoBuffer στην οποία αποθηκεύουμε το αποτέλεσμα που επιστρέφει η μέθοδος mergeArrays, την οποία εξηγούμε παρακάτω στις βοηθητικές μεθόδους, και η οποία επιστρέφει έναν πίνακα οποίος περιέχει τα δεδομένα όλης της ηχογράφησης σε ένα πίνακα Float32Array. Το επόμενο βήμα είναι να δημιουργήσουμε το αρχείο ήχου μορφής .wav το οποίο θα στείλουμε στο Google Speech API το οποίο προαναφέραμε και θα μετατρέψει τον ήχο σε κείμενο. Για αυτό θα πρέπει να κατασκευάσουμε το αρχείο ήχου αυτής της μορφής το οποίο αποτελείται από διαφορετικά κομμάτια (chunks) και υποκομμάτια (subchunks), τοποθετημένα με συγκεκριμένη σειρά και στο τέλος αυτών είναι τα raw δεδομένα που ηχογραφήσαμε από το μικρόφωνο του χρήστη. Έτσι δημιουργούμε την μεταβλητή buffer η οποία έχει ένα πίνακα ο οποίος στο τέλος της επεξεργασίας θα είναι το αρχείο .wav μας και στον οποίο ορίζουμε απλά το μέγεθός του να είναι ίσο με 44 byte (τόσα byte καταλαμβάνουν συνολικά οι κεφαλίδες και οι υποκεφαλίδες του .wav αρχείου) συν το μέγεθος του monoBuffer που περιέχει τα δεδομένα της

ηχογράφησης. Λόγω της ιδιαιτερότητας αυτού του πίνακα ο οποίος δεν μας επιτρέπει να τροποποιήσουμε τα δεδομένα του κατευθείαν, χρησιμοποιούμε ένα αντικείμενο DataView του οποίου του δίνουμε ως όρισμα τον πίνακα που φτιάξαμε, και το οποίο αναλαμβάνει να τροποποιεί τα δεδομένα του πίνακα αυτού παρέχοντας ταυτόχρονα ευκολία στην καταχώρηση των δεδομένων από μεριάς του προγραμματιστή (πχ ο προγραμματιστής δεν θα πρέπει να λάβει υπόψη του τον τρόπο αναπαράστασης των δεδομένων που γράφει στον πίνακα δηλαδή το endianness).

Δημιουργία του .wav αρχείου

Ξεκινώντας να δημιουργούμε το .wav αρχείο μας, ακολουθούμε τη δομή του η οποία όπως αυτή περιεγράφηκε στο 2ο κεφάλαιο και ξεκινάμε να δημιουργούμε τα chunks του αρχείου.

```
119                 // RIFF chunk descriptor
120                 writeLetters(view, 0, 'RIFF');
121                 view.setUint32(4, 44 + monoBuffer.length*2, true);
122                 writeLetters(view, 8, 'WAVE');
123                 // FMT sub-chunk
124                 writeLetters(view, 12, 'fmt ');
125                 view.setUint32(16, 16, true);
126                 view.setUint16(20, 1, true);
127                 // mono (1 channels)
128                 view.setUint16(22, 1, true);
129                 view.setUint32(24, sampleRate, true);
130                 //we use setUint32 or 16 depending on the number of bytes we want to write on the buffer
131                 //32 for 4 bytes 16 for 2 bytes. check canonical wave file format.
132                 view.setUint32(28, sampleRate * 2, true);
133                 view.setUint16(32, 2, true);
134                 view.setUint16(34, 16, true);
```

Εικόνα 4.7: Κομμάτι κώδικα JavaScript 4/9. RIFF και FMT chunks

Ακολουθώντας το πρότυπο του .wav αρχείου χρησιμοποιούμε τρεις μεθόδους για να γράψουμε σωστά τη δομή του: την βοηθητική μέθοδο writeLetters την οποία αναλύουμε παρακάτω στις βοηθητικές μεθόδους την μέθοδο setUint32 και setUint16. Οι δυο τελευταίες μέθοδοι του αντικειμένου DataView, κάνουν ακριβώς το ίδιο πράγμα απλά η μία χρησιμοποιείται για την εγγραφή 4 byte στον πίνακα ενώ η άλλη για την εγγραφή 2 byte (υπενθυμίζεται ότι 1byte = 8bit, άρα 2byte = 16bit και 4byte = 32bit). Τα ορίσματα που απαιτούν αυτές οι δύο μέθοδοι για να λειτουργήσουν σωστά είναι:

1. offset, που ορίζει σε πιο σημείο (μετρώντας με byte) θα πρέπει να πάει να καταχωρήσει την τιμή της η κάθε μέθοδος.

2. value, που περιέχει την τιμή που θα πρέπει να καταχωρήσει η μέθοδος στον πίνακα
3. little endian, αυτή η τιμή είναι προαιρετική και παίρνει τιμές αληθής/ψευδής και υπαγορεύει τον τρόπο γραφής της τιμής (αν βάλουμε true η τιμή γράφεται ως little endian ενώ αν βάλουμε false η τιμή γράφεται ως big endian).

Το ποιες τιμές πρέπει να γράψουμε και που, μας το υπαγορεύει το πρότυπο του .wav αρχείου. Στο offset κάθε φορά προσθέτουμε όσα byte έχουμε γράψει μέχρι εκείνο το σημείο συν το μέγεθος της ακριβώς προηγούμενης τιμής που γράψαμε στον πίνακα και αυτό το νούμερο θα είναι η θέση της επόμενης εγγραφής, δηλαδή αμέσως μετά από την προηγούμενη.

Τέλος για να ολοκληρωθεί το .wav αρχείο, πρέπει να δημιουργήσουμε και το chunk των δεδομένων (data chunk). Αυτό θα έχει ως εξής:

```
135                 // data sub-chunk
136                 writeLetters(view, 36, 'data');
137                 view.setUint32(40, monoBuffer.length*2, true);
138
139                 // write the PCM samples
140                 var lng = monoBuffer.length;
141                 var index = 44;
142                 var volume = 1;
143                 for (var i = 0; i<lng; i++)
144                 {
145                     view.setInt16(index, monoBuffer[i] * (0x7FFF *volume), true);
146                 //index is +=2 because, every sample i write has 2bytes
147                 //length (remember i use setInt16, so 16bit) so the offset is 2
148                     index += 2;
149
150                 }
```

Εικόνα 4.8: Κομμάτι κώδικα JavaScript 4/9. Data chunk.

Εδώ δημιουργούμε το τελευταίο chunk του .wav αρχείου το οποίο περιέχει όλα τα δεδομένα που ηχογραφήσαμε από το μικρόφωνο του χρήστη τα οποία ξεκινάμε να γράφουμε στο τέλος του πίνακά μας (μέσω του DataView το οποίο προαναφέραμε).

Δημιουργία του BLOB

Το blob είναι ένα αρχείο το οποίο όμως δημιουργείται στην μνήμη του υπολογιστή και όχι στον σκληρό μας δίσκο, και μπορούμε να κατεβάσουμε και να αποθηκεύσουμε σε περίπτωση που το χρειαζόμαστε.

```
153                 // our final binary blob
154                 var blob = new Blob ( [ view ], { type : 'audio/wav' } );
155                 var xhr =new XMLHttpRequest();
156                 // here we create a formData object for easier creation of key/value pairs
157                 var fd =new FormData();
158                 if (document.getElementById('el').checked)
159                 {
160                     fd.append("language","greek");
161                     console.log ("language is greek");
162                 }
163                 else
164                 {
165                     fd.append("language","english");
166                     console.log("language is english");
167                 }
168                 fd.append("rec",blob);//we pass the blob with a name "rec"
169                 fd.append("rate",sampleRate);//we pass the samplerate with a name "rate"
```

Εικόνα 4.9: Κομμάτι κώδικα JavaScript 5/9. Δημιουργία του blob αρχείου.

Όπως βλέπουμε και στον παραπάνω κώδικα, δημιουργούμε ένα αντικείμενο τύπου Blob και του δίνουμε σαν παράμετρο το DataView (το οποίο περιέχει τον πίνακα με το .wav αρχείο) και δηλώνουμε στην πρώτη παράμετρο τον τύπο των δεδομένων που περνάμε. Στην συνέχεια δημιουργούμε ένα αντικείμενο **XMLHttpRequest** το οποίο αναλύουμε παρακάτω, καθώς και ένα αντικείμενο **FormData.** Το αντικείμενο αυτό μας βοηθά να δημιουργήσουμε συσχετισμούς κλειδιού – τιμής όπως θα γινόταν κατά την υποβολή μιας φόρμας, για διάφορες τιμές στον κώδικα τις οποίες θέλουμε να περάσουμε στο back-end. Δηλαδή όπως βλέπουμε και στον κώδικα, αν έχει επιλεγεί το radio button με την ελληνική γλώσσα, δημιουργούμε το κλειδί “language” εντός της μεταβλητής fd και αναθέτουμε την τιμή “greek” σε αυτό το κλειδί, αλλιώς αν έχουν επιλεγεί τα αγγλικά, αναθέτουμε την τιμή “english”. Όταν αυτή η μεταβλητή σταλεί στο back-end το οποίο θα επεξεργαστεί τα δεδομένα, θα χρησιμοποιήσουμε αυτό το κλειδί για να πάρουμε την τιμή της μεταβλητής και έτσι θα ξέρουμε ποια ήταν η επιλεγμένη γλώσσα. Αυτό το κάνουμε επειδή δεν κάνουμε submit κάποια φόρμα, αλλά χρησιμοποιούμε την τεχνική ajax, και θέλουμε κάποια από τα δεδομένα μας να έχουν ένα συγκεκριμένο κλειδί ώστε να μπορέσουμε να πάρουμε την τιμή του. Τέλος δημιουργούμε δύο ακόμη κλειδιά, ένα με το ίδιο το blob αρχείο και το άλλο με το ρυθμό δειγματοληψίας που χρησιμοποιήσαμε.

Αποστολή των δεδομένων (XMLHttpRequest)

Στη συνέχεια θα στείλουμε τα δεδομένα στο back – end το οποίο θα αναλάβει να επικοινωνήσει με το Google Speech API της Google και να μας επιστρέψει την εντολή σε κείμενο.

```
xhr.open("POST","phpFiles/upload_raw_wav.php",false);
xhr.onreadystatechange = function()
{
    if (xhr.readyState == 4)
    {
        console.log ("readyState = 4");
        if (xhr.status == 200)
        {
            console.log("Server returned: ",xhr.responseText);
            var subStrng = xhr.responseText.substring(14);//with this i delete the 14 first characters of the resposne
            //i do this to delete a non-whitespace character from the response that couldn't be parsed
            //either way, the first letters contain an empty response, so it's not important or needed.
            voiceCommand = JSON.parse(subStrng);
            document.getElementById("result").innerHTML= voiceCommand.result[0].alternative[0].transcript;
        }
    }
};
xhr.send(fd);
fd.delete('rec');
fd.delete('rate');
```

Εικόνα 4.10: Κομμάτι κώδικα JavaScript 6/9. Αποστολή δεδομένων με XMLHttpRequest.

Ξεκινάμε αρχικοποιώντας το XMLHttpRequest, στο οποίο ορίζουμε το αρχείο το οποίο θα επεξεργαστεί τα δεδομένα και αν θέλουμε το αίτημα να είναι ασύγχρονο (αυτό πρακτικά σημαίνει εάν θα συνεχίσει να εκτελείται το υπόλοιπο πρόγραμμα ή εάν θα περιμένει μέχρι να ολοκληρωθεί η αποστολή των δεδομένων και η λήψη της απάντησης). Στην συνέχεια διαμορφώνουμε την onreadystatechange η οποία θα καλείται κάθε φορά που αλλάζει η κατάσταση readyState του XMLHttpRequest. Οι καταστάσεις που υπάρχουν συνολικά είναι 5 και είναι οι εξής

0. Το αίτημα δεν έχει αρχικοποιηθεί ακόμη.
1. Δημιουργία σύνδεσης με τον sever
2. Το αίτημα έχει ληφθεί
3. Επεξεργασία του αιτήματος
4. Το αίτημα έχει τελειώσει και η απάντηση είναι έτοιμη.

Αντίστοιχα 3 από τα κυριότερες καταστάσεις απαντήσεων του server είναι οι εξής:

- 200: “OK”
- 403: “Forbidden” (απαγορευμένο)
- 404: “Page not found” (η σελίδα δεν βρέθηκε)

Εδώ πρέπει να αναφέρουμε ότι το readyState δεν σχετίζεται άμεσα με το status. δηλαδή αν το readystate γίνει 4 δε σημαίνει ότι και η σελίδα βρήκε τον πόρο που ζητήσαμε ή όλα πήγαν καλά κατά την επεξεργασία των δεδομένων και μπορεί να μας επιστρέψει 404 status αντί για 200. Οπότε για να είμαστε σίγουροι ότι όλα

πήγαν καλά, ελέγχουμε την κατάσταση της μεταβλητής readyState και αν είναι 4 και το status 200 τότε παίρνουμε τα δεδομένα από τη μέθοδο responseText του XMLHttpRequest. Να σημειωθεί εδώ ότι οι πρώτοι 14 χαρακτήρες της απάντησης είναι κενοί οπότε τους αποκόπτουμε με την εντολή substring(14). Στην συνέχεια διαβάζουμε τα δεδομένα που πήραμε ως ένα JSON αντικείμενο διότι η μέθοδος που καλέσαμε μας επιστρέφει τα δεδομένα σε μορφή JSON, και στην συνέχεια ενημερώνουμε το περιεχόμενο του πεδίου με τίτλο «Εντολή Χρήστη» στο οποίο δώσαμε αναγνωριστικό (id) το λεκτικό “result”. Τέλος, αφού έχουμε φτιάξει όλα τα παραπάνω, στέλνουμε τα δεδομένα με τη μέθοδο send στην οποία δίνουμε ως όρισμα το αντικείμενο FormData που περιέχει τις τιμές και τα κλειδιά τους.

Η μορφή του JSON που μας επιστρέφει η google το οποίο περιέχει σε κείμενο, την φωνητική μας εντολή έχει την εξής μορφή:


```
{
  "result":[
    {
      "alternative":[
        {
          "transcript":"this is a test",
          "confidence":0.97321892
        },
        {
          "transcript":"this is a test for"
        }
      ],
      "final":true
    }
  ],
  "result_index":0
}
```


Εικόνα 4.10.1: Παράδειγμα μορφής JSON που μας επιστρέφει το Google Speech API

Όταν το Google Speech API δεν είναι 100% σίγουρο για την μετατροπή που μας κάνει, επιστρέφει μια απάντηση όπως η παραπάνω. Μέσα στο αντικείμενο alternative, προσθέτει την παράμετρο confidence το οποίο είναι ένα ποσοστό σιγουριάς για την εγκυρότητα της μετατροπής σε σχέση με αυτό που ηχογραφήσαμε. Επειδή πάνω πάνω μπαίνει πάντα το λεκτικό με το μεγαλύτερο

ποσοστό σιγουριάς, εμείς πάντα παίρνουμε το πρώτο από την απάντηση που μας έρχεται με την εντολή: voiceCommand.result[0].alternative[0].transcript. Τέλος σβήνουμε τα κλειδιά rec και rate που δώσαμε με το FormData διότι θα ξαναχρησιμοποιήσουμε το αντικείμενο για νέο XMLHttpRequest και το να κουβαλάμε περιττά δεδομένα στο αίτημά μας κάνει τον όγκο δεδομένων μεγαλύτερο και πιο αργή την αποστολή των δεδομένων χωρίς λόγο.

Σύγκριση του αποτελέσματος με τη βάση

Αφού λάβουμε το αποτέλεσμα από το Google Speech API, στέλνουμε με ένα δεύτερο XMLHttpRequest την φωνητική εντολή ώστε να γίνει σύγκριση με τις εντολές της βάσης και να βρεθεί το αντίστοιχο ποσοστό ομοιότητας με τις εντολές της βάσης. Να σημειωθεί ότι αν δεν υπάρξει ποσοστό ομοιότητας πάνω από 85% με κάποια από τις εντολές στη βάση, τότε δεν θα επιστραφεί τίποτα, σαν να μην υπήρξε καμία εντολή που να ταιριάζει. Ο κώδικας είναι ο εξής:

```
188                 fd.append('voiceCommand',voiceCommand.result[0].alternative[0].transcript);
189                 xhr.open("POST","phpFiles/database_fetch.php",true);
190                 xhr.onreadystatechange = function()
191                 {
192                     if (xhr.readyState == 4)
193                     {
194                         console.log ("readyState = 4");
195                         if (xhr.status == 200)
196                         {
197                             console.log("Server returned: ",xhr.responseText);
198                             var dbResponse = JSON.parse(xhr.responseText);
199                             document.getElementById("command").innerHTML= dbResponse.command;
200                             document.getElementById("percent").innerHTML= dbResponse.percentage;
201                             console.log(dbResponse.shellExecMessages);
202                         }
203                     }
204                 };
205                 xhr.send(fd);
206 
207             };
```

Εικόνα 4.11: Κομμάτι κώδικα JavaScript 7/9. Αποστολή εντολής για σύγκριση.

Ακολουθώντας την ίδια λογική που περιγράψαμε προηγουμένως, περνάμε την εντολή που δώσαμε και το back – end αναλαμβάνει να κάνει τον έλεγχο στη βάση. Αν βρεθεί εντολή με ποσοστό ομοιότητας μεγαλύτερο από 85% τότε επιστρέφεται η εντολή της βάσης καθώς και το ποσοστό ομοιότητας, αλλιώς επιστρέφεται μηδέν, σαν να μην υπήρχε καν η εντολή ή κάποια που να τις μοιάζει. Στη συνέχεια πάμε και βάζουμε τις αντίστοιχες τιμές που επιστράφηκαν σε κάθε περίπτωση στα αντίστοιχα πεδία της σελίδας.

Βοηθητικές Μέθοδοι

Η πρώτη από τις δύο βοηθητικές μεθόδους είναι η **mergeArrays** η οποία είναι η παρακάτω:

```
216         function mergeArrays(channelBuffer, recordingLength)
217         {
218             var result = new Float32Array(recordingLength);
219             var offset = 0;
220             var lng = channelBuffer.length;
221             for (var i = 0; i < lng; i++)
222             {
223                 var buffer = channelBuffer[i];
224                 result.set(buffer, offset);
225                 offset += buffer.length;
226             }
227             return result;
228         }
```

Εικόνα 4.12: Κομμάτι κώδικα JavaScript 8/9. Η βοηθητική μέθοδος mergeArrays.

Όταν κάνουμε την ηχογράφηση της φωνής του χρήστη, έχουμε ορίσει ένα συγκεκριμένο μέγεθος buffer όπως φαίνεται και στην εικόνα 4.4 ο οποίος μόλις γεμίσει, καλείται η συνάρτηση onaudioprocess η οποία τον αδειάζει και αποθηκεύει τα δεδομένα στον πίνακα monoChannel. Κάθε φορά που γίνεται αυτό, μέσα στον monoChannel αποθηκεύουμε ένα πίνακα μορφής Float32Array όπως αναλύουμε και πιο πάνω. Και επειδή δε ξέρουμε τη συνολική διάρκεια της φωνητικής εντολής του χρήστη, κάθε φορά που γεμίζει ο buffer αποθηκεύουμε τον πίνακα που περιέχει στον υπερπίνακα monoChannel. Στο τέλος λοιπόν της ηχογράφησης ο πίνακας monoChannel περιέχει πολλούς μικρούς πίνακες που ο καθένας έχει το μέγεθος του buffer. Δουλειά της βοηθητικής μεθόδου mergeArrays λοιπόν είναι να πάρει όλους αυτούς τους πίνακες και να τους κάνει έναν, ώστε να έχουμε την ηχογράφησή μας σε ένα πίνακα όπως θα έπρεπε να είναι κανονικά αν ξέραμε τη διάρκεια ή τα δεδομένα που θα έδινε ο χρήστης. Ορίζουμε την μεταβλητή result η οποία θα έχει μέγεθος όσο όλη η ηχογράφηση της οποίας το μέγεθος περνάμε ως παράμετρο. Στη συνέχεια ορίζουμε το offset ίσο με μηδέν για να ξεκινήσουμε να γράφουμε από την αρχή του πίνακα, και βάζουμε και το μήκος του monoChannel (ο οποίος έχει περάσει ως όρισμα στην μεταβλητή channelBuffer). Τέλος σε μία δομή επανάληψης περνάμε ένα-ένα όλα τα στοιχεία του channelBuffer και τα δεδομένα του τα βάζουμε στην μεταβλητή result. Έτσι όταν τελειώσει η επανάληψη, θα έχουμε πάρει τον πίνακα result ο οποίος θα έχει όλες τις τιμές μέσα σε ένα πίνακα.

Η δεύτερη βοηθητική μέθοδος είναι η writeLetters, της οποίας η λειτουργία είναι απλή. Μας βοηθάει να γράφουμε τα αντίστοιχα λεκτικά στην αρχή κάθε chunk. Ο κώδικας είναι ο εξής:

```
229         function writeLetters(view, offset, string)
230         {
231             var lng = string.length;
232             for (var i = 0; i < lng; i++)
233             {
234                 view.setUint8(offset + i, string.charCodeAt(i));
235             }
236         }
```

Εικόνα 4.13: Κομμάτι κώδικα JavaScript 9/9. Η βοηθητική μέθοδος writeLetters.

Η μέθοδος παίρνει ως ορίσματα το **DataView**, το offset (δηλαδή σε ποια θέση μετρώντας από την αρχή του πίνακα view θα πρέπει να γραφτεί το λεκτικό), και το λεκτικό (string) που θέλουμε να γραφτεί, πχ το “RIFF’. Αρχικά παίρνουμε το μήκος του string το οποίο στην ουσία αποτελεί μια ακολουθία χαρακτήρων και χρησιμοποιώντας μια δομή επανάληψης παίρνουμε έναν-έναν όλους τους χαρακτήρες του string. Στη συνέχεια χρησιμοποιώντας την συνάρτηση charCodeAt μετατρέπουμε τον κάθε χαρακτήρα στο αντίστοιχο ASCII ισοδύναμό του και το γράφουμε κατευθείαν στο view στην αντίστοιχη θέση.

4.2 Back – End, PHP κώδικας

Σε αυτό το κομμάτι θα αναλύσουμε τον κώδικα που τρέχει μέσα στον server και εκτελεί τους ελέγχους καθώς και την επικοινωνία με το API της Google. Θα ξεκινήσουμε αναλύοντας την κλάση SpeechToText, η οποία υλοποιεί το βασικό μοντέλο της μετατροπής του κειμένου σε φωνή.

```php
<?php
    class SpeechToText
    {

        function convert($file,$rate,$lang)
        {

            if (!(empty ($file)))//empty returns bool
            {
                if (!(empty ($lang)))
                {
                    if (empty ($rate))
                    {
                        echo ("error, sampleRate not specified.
                        pass the filepath as first argument in convert,
                        and sample rate as second");


                    }
```

Εικόνα 4.14: Κλάση SpeechToText 1/0. Δήλωση κλάσης

Ξεκινώντας την κλάση δημιουργούμε την μια και μοναδική μέθοδο της κλάσης, τη μέθοδο convert. Αυτή η μέθοδος δέχεται τρία ορίσματα. Το μονοπάτι (path) του αρχείου (το οποίο έχουμε στην μνήμη και όχι αποθηκευμένο σε κάποιο φυσικό μέσο αποθήκευσης), τον ρυθμό δειγματοληψίας και τη γλώσσα που επέλεξε ο χρήστης. Ξεκινώντας ελέγχουμε ότι έχουν δοθεί και τα τρία προκειμένου να τα χρησιμοποιήσουμε.

```php
                else
                {
                    if ($lang == "english")
                    {
                        $conversionUrl = "https://www.google.com/speech-api/v2/recognize
                        ?output=json&lang=en-us&key=AIzaSyDCy0qY7CP2oDglZ-rcoF2AoEzCnpu80as";
                    }
                    else
                    {
                        $conversionUrl = "https://www.google.com/speech-api/v2/recognize
                        ?output=json&lang=el-gr&key=AIzaSyDCy0qY7CP2oDglZ-rcoF2AoEzCnpu80as";
                    }
```

Εικόνα 4.14: Κλάση SpeechToText 2/0: Δημιουργία του url

Εάν τα στοιχεία είναι όλα σωστά, προχωράμε στον έλεγχο της γλώσσας η οποία είναι ένα από τα τρία στοιχεία που χρειαζόμαστε για τη δημιουργία της διεύθυνσης(url) που θα χρησιμοποιήσουμε για να επικοινωνήσουμε με το Google Speech API. Το link αυτό αποτελείται από τρείς παραμέτρους:

- **output:** Δηλώνει τη μορφή στην οποία θέλουμε να πάρουμε την απάντηση από το API.
- **lang:** Δηλώνει τη γλώσσα στην οποία είναι η ηχογραφημένη μας εντολή και δηλώνεται με το συνδυασμό του locale της γλώσσας ακολουθούμενο από μία παύλα και το country region λεκτικό, δηλαδή για τα ελληνικά το el-gr
- **key:** Δηλώνει το κλειδί το οποίο έχει καταγράψει το χρήστη μας και μας δίνει τη δυνατότητα να χρησιμοποιήσουμε αυτό το API. Να σημειωθεί ότι το διάστημα στο οποίο γράφτηκε αυτή η πτυχιακή το API αυτό δεν ήταν διαθέσιμο για ευρεία χρήση και η δυνατότητα χρήσης του επιτρεπόταν μόνο για σκοπούς έρευνας και ανάπτυξης και όχι για εμπορική χρήση, έτσι λοιπόν αν κάποιος θέλει να το χρησιμοποιεί, θα πρέπει να είναι registered ως Google Developer και να έχει εκδώσει ένα αντίστοιχο key το οποίο επιτρέπει συγκεκριμένο αριθμό αιτημάτων τη μέρα.

```
$filePath = $file;
$sampleRate = trim($rate);
$args['file'] = new CurlFile ($filePath);
$ch = curl_init($conversionUrl);
curl_setopt($ch, CURLOPT_POST, true);
//contains the voice_file and declares a post
curl_setopt($ch, CURLOPT_POSTFIELDS,$args);
// --data-binary IS necessary if u want to pass
//the result as a string and not to just display it*
curl_setopt($ch, CURLOPT_BINARYTRANSFER, TRUE);
//in order to pass the result to curl_exec()
//rather than output (echo) the result.
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
// --header
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: audio/l16; rate='.$sampleRate.';'));
$result = curl_exec($ch);
if ($result) // execute curl post
{
    curl_close($ch); // close connection
    //echo "success";
    return $result;
}
else
{
    return "error with curl_exec";
}
```

Εικόνα 4.15: Κλάση SpeechToText 3/0. Ρύθμιση παραμέτρων.

Προχωρώντας, αρχίζουμε να θέτουμε τις παραμέτρους που χρειάζονται να σταλούν μαζί με το αίτημα στο API. Ο τρόπος που θα χρησιμοποιήσουμε για να επικοινωνήσουμε με το API είναι μέσω curl. Το curl είναι μια εντολή η οποία μας επιτρέπει να στέλνουμε δεδομένα σε ένα server μέσω ίντερνετ και μπορούμε να τη χρησιμοποιήσουμε χωρίς να χρειαζόμαστε τη χρήση browser. Ξεκινάμε λοιπόν να δημιουργούμε το αίτημά μας χρησιμοποιώντας την CurlFile δίνοντάς της ως παράμετρο το αρχείο για να το ενσωματώσουμε στο αίτημά μας. Αφού γίνει αυτό, χρησιμοποιούμε την curl_init για να αρχικοποιήσουμε το curl αίτημά μας. Για να θέσουμε διαφορετικές παραμέτρους στο αίτημά μας, χρησιμοποιούμε την curl_setopt η οποία παίρνει τρία ορίσματα. Την μεταβλητή στην οποία έχουμε αρχικοποιήσει το curl_init για να ενσωματώσει τις παραμέτρους στο curl αίτημα, την παράμετρο που θέλουμε να θέσουμε , και την τιμή της παραμέτρου που θέλουμε να αναθέσουμε. Οι παράμετροι που θέτουμε είναι οι εξής:

- **CURLOPT_POST** : Ορίζει εάν το αίτημα θα είναι POST
- **CURLOPT_POSTFIELDS**: Περιλαμβάνει τα δεδομένα που θέλουμε να στείλουμε σε ένα curl αίτημα
- **CURLOPT_BINARYTRANSFER**: Αν οριστεί σε true μας επιστρέφει τα binary data της απάντησης του αιτήματός μας τα οποία χρειαζόμαστε για να τα επεξεργαστούμε μετέπειτα. Επίσης απαιτείται η χρήση της όταν χρησιμοποιείται η CURLOPT_RETURNTRANSFER.
- **CURLOPT_RETURNTRANSFER**: Αν οριστεί σε true μας επιστρέφει το αποτέλεσμα σε string αντί να το τυπώνει κατευθείαν.
- **CURLOPT_HTTPHEADER**: Ορίζουμε την κεφαλίδα (header) που παίρνει ως όρισμα έναν πίνακα ο οποίος περιέχει πληροφορίες για το αίτημά μας, δηλαδή τον τύπο του περιεχομένου του αιτήματος, τον αριθμό δειγματοληψίας κ.λπ.

Τέλος εκτελούμε το αίτημα με την curl_exec και την μεταβλητή στην οποία κρατήσαμε την αρχικοποίηση ως παράμετρο. Αν η εκτέλεση του αιτήματος επιτύχει, η curl_exec θα επιστρέψει δεδομένα, τα οποία τα γυρνάμε επιστρέφουμε στην συνάρτηση η οποία κάλεσε την μέθοδο convert, αλλιώς επιστρέφουμε ένα μήνυμα σφάλματος. Περισσότερες πληροφορίες για τις παραμέτρους του αιτήματος curl μπορείτε να βρείτε εδώ : http://php.net/manual/en/function.curl-setopt.php

Ανέβασμα των δεδομένων και λήψη της εντολής σε κείμενο

Συνεχίζοντας τη ροή του προγράμματος από το κεφάλαιο **4.1.3**, αφού ο χρήστης ηχογραφήσει την φωνητική του εντολή, το post αίτημα που στέλνεται στον server, στέλνεται στο αρχείο raw_wav_data.php το οποίο θα περιγράψουμε παρακάτω. Το αρχείο αυτό αρχικοποιεί και χρησιμοποιεί την κλάση SpeechToText την οποία περιγράψαμε παραπάνω.

```php
<?php
    if ($_FILES['rec']['error'] > 0 )
    {
        echo 'Problem: ';
        switch ($_FILES['rec']['error'])
        {
            case 1: echo 'File exceeded upload_max_filesize';
                break;
            case 2: echo 'File exceeded max_file_size';
                break;
            case 3: echo 'File only partially uploaded';
                break;
            case 4: echo 'No file uploaded';
                break;
            case 6: echo 'Cannot upload file: No temp directory specified';
                break;
            case 7: echo 'Upload failed: Cannot write to disk';
                break;
        }
        exit;
```

Εικόνα 4.16: Κώδικας αρχείου raw_wav_data.php. Έλεγχος για σφάλματα

Αρχικά ελέγχουμε για τυχόν σφάλμα κατά την αποστολή του αρχείου το οποίο θα βρίσκεται στο $_FILES['rec']['error'] και τυπώνουμε ένα αντίστοιχο μήνυμα, ανάλογα με τον αριθμό του κάθε σφάλματος. Αν ο κώδικας δεν περιέχει σφάλματα, προχωράμε στην αξιοποίηση των δεδομένων.

```php
if (is_uploaded_file ($_FILES['rec']['tmp_name']))
{
    if (isset ($_POST['rate']))
    {
        include 'SpeechToText.php';
        $sttx = new SpeechToText();
        //this gives us the full path to the file we uploaded.
        $filePath = $_FILES['rec']['tmp_name'];
        $sampleRate = $_POST['rate'];
        $language = $_POST['language'];
        $result = $sttx->convert($filePath,$sampleRate,$language);
        echo $result;

    }
}
else
{
    echo "problem with upload";
}
?>
```

Εικόνα 4.17: Κώδικας αρχείου raw_wav_data.php. Μετατροπή φωνής σε κείμενο.

Ελέγχουμε για να δούμε ότι έχει όντως σταλεί το αρχείο σε αυτό το path. Να σημειωθεί ότι η εντολή $_FILES['rec']['tmp_name'] μας δίνει το πλήρες μονοπάτι (path) όπου βρίσκεται το αρχείο στη μνήμη του server. Αν λοιπόν έχουμε και τον ρυθμό δειγματοληψίας τότε κάνουμε include το αρχείο SpeechToText.php. Αυτό μας επιτρέπει να χρησιμοποιήσουμε την κλάση αυτή και πιο συγκεκριμένα την μέθοδο convert η οποία μας ενδιαφέρει. Αρχικοποιούμε λοιπόν την μεταβλητή $sttx και μετά καλούμε τη μέθοδο convert την οποία αναλύσαμε παραπάνω δίνοντάς της τα τρία ορίσματα που χρειάζεται, το path του αρχείου, τον ρυθμό δειγματοληψίας και τη γλώσσα και αυτή κάνει κλήση στο Google Speech API και μας επιστρέφει το αποτέλεσμα το οποίο επιστρέφουμε και μείς με τη σειρά μας πίσω στον JavaScript κώδικα, στο πρώτο XMLHttpRequest, όπως φαίνεται και στην εικόνα **4.10**.

Σύγκριση της εντολής με τη βάση

Αφού μετατρέψουμε την εντολή του χρήστη και την εμφανίσουμε, σειρά έχει η σύγκριση της εντολής με τη βάση, ώστε να βρεθεί τυχόν ομοιότητα με τις αποθηκευμένες εντολές που έχουμε στην βάση.

```php
<?php
$language = $_POST['language'];
$voiceCommand = $_POST['voiceCommand'];
$percent=0;
$matchedCommand = 'No match is found';
$shellExecMessages = 'empty';
```

Εικόνα 4.18: Κώδικας αρχείου database_fetch. Αρχικοποίηση μεταβλητών

```php
try
{
    $db = new PDO('mysql:host=127.0.0.1;dbname=Voice_Commands;charset=utf8', 'praktiki', 'test1234');
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

}
catch (PDOException $e)
{
    echo $e->getMessage();
}
```

Εικόνα 4.19: Κώδικας αρχείου database_fetch. Δημιουργία σύνδεσης με τη βάση.

Αφού δημιουργήσουμε τις μεταβλητές, αρχικοποιούμε τη σύνδεσή μας με τη βάση χρησιμοποιώντας το interface PDO της PHP το οποίο μας επιτρέπει να συνδεθούμε στην βάση δημιουργώντας prepared statements. Εάν η σύνδεσή μας με τη βάση αποτύχει τότε ο κώδικας θα εκτελέσει το μπλοκ κώδικα catch, αλλιώς θα προχωρήσει παρακάτω.

```php
if ($language == "greek")
{
    $query = "select * from valid_el_commands";
}
else
{
    $query = "select * from valid_en_commands";
}
try
{
    $result = $db->prepare($query);
    $result->execute();
}
catch (PDOException $e)
{
    echo $e->getMessage();
}
```

Εικόνα 4.20: Κώδικας αρχείου database_fetch. Δημιουργία του query.

Ελέγχουμε τη γλώσσα και δημιουργούμε το αντίστοιχο query αναλόγως τη γλώσσα, διότι κρατάμε τις έγκυρες εντολές της κάθε γλώσσας σε ξεχωριστό πίνακα.

Αφού δημιουργηθεί το σωστό query το εκτελούμε και σε περίπτωση σφάλματος θα εκτελεστεί ο κώδικας στο catch μπλοκ. Το παραπάνω query που εκτελέσαμε θα μας φέρει όλες τις εγγραφές οι οποίες υπάρχουν μέσα στον πίνακα που ζητήσαμε.

```
38  while ($row = $result->fetch(PDO::FETCH_OBJ)) {
39      global $percent, $matchedCommand;
40      similar_text($row->command, $voiceCommand, $percent);
41      if ($percent >= 85) {
42          $matchedCommand = $row->command;
43          break;
44      }
45      $percent = 0;
46  }
```

Εικόνα 4.21: Κώδικας αρχείου database_fetch. Σύγκριση ομοιότητας εντολών

Δημιουργώντας μια δομή επανάληψης προσπελαύνουμε όλες τις εντολές της βάσης και χρησιμοποιώντας την μέθοδο similar_text συγκρίνουμε την εντολή της βάσης με την εντολή που μας επέστρεψε το Google Speech API. Το ποσοστό ομοιότητας μπαίνει στην μεταβλητή $percent. Αν το ποσοστό αυτό είναι μεγαλύτερο ή ίσο με 85% τότε η εντολή αποθηκεύεται στην μεταβλητή $matchedCommand και η σύγκριση σταματά.

```
47  switch ($row->id){
48      case (1):
49          $shellExecMessages = shell_exec('sudo /python_files/close_the_light.py 2>&1');
50
51          break;
52      case (2):
53          $shellExecMessages = shell_exec('sudo /python_files/open_the_light.py 2>&1');
54
55          break;
56  }
```

Εικόνα 4.22: Κώδικας αρχείο database_fetch. Εκτέλεση script της κάθε εντολής

Κάθε εντολή που επιλέγεται έχει ένα μοναδικό id. Αναλόγως με αυτό το id, υπάρχει και ένα script το οποίο λέει στο raspberry pi τι πρέπει να εκτελέσει. Οι εντολές που έχουμε αυτή τη στιγμή στη βάση μας είναι οι εντολές “άναψε το φως” και “σβήσε το φως” με id 2 και 1 αντίστοιχα. οπότε μόλις επιλεχθεί η εντολή εκτελούμε με βάση το id της το αντίστοιχο script.

```python
#!/usr/bin/env python
import RPi.GPIO as GPIO
# to use Raspberry Pi board pin numbers
GPIO.setmode(GPIO.BOARD)
# set up GPIO output channel
GPIO.setup(11, GPIO.OUT)
GPIO.output(11,GPIO.LOW)

```

Εικόνα 4.22: Κώδικας για άνοιγμα των φώτων

```python
#!/usr/bin/env python
import RPi.GPIO as GPIO
# to use Raspberry Pi board pin numbers
GPIO.setmode(GPIO.BOARD)
# set up GPIO output channel
GPIO.setup(11, GPIO.OUT)
GPIO.output(11,GPIO.HIGH)

```

Εικόνα 4.23: Κώδικας για σβήσιμο των φώτων

τα script τρέχουν χρησιμοποιώντας την μέθοδο shell_exec και δίνοντας σαν όρισμα το path του αρχείου.

```php
$jsonArray = array('command' => $matchedCommand, 'percentage' => $percent, 'shellExecMessages' =>$shellExecMessages);
echo json_encode($jsonArray);

?>
```

Εικόνα 4.23 Κώδικας αρχείο database_fetch. Δημιουργία της απάντησης για τον client

Στη συνέχεια, δημιουργούμε ένα πίνακα με την εντολή της βάσης, το ποσοστό ομοιότητας και το μήνυμα που εμφανίστηκε κατά την εκτέλεση του αντίστοιχου script (αυτό πιο πολύ το περνάμε μπροστά και το τυπώνουμε στην κονσόλα για λόγους αποσφαλμάτωσης) και μετά μετατρέπουμε τον πίνακα σε ένα json String με τη συνάρτηση json_encode. Το String αυτό κάνοντάς το echo θα επιστραφεί στον client ο οποίος το ζήτησε και θα το μετατρέψει ξανά σε πίνακα για να πάρει τις τιμές και να τις εμφανίσει στον χρήστη.

ΚΕΦΑΛΑΙΟ 5: ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΤΑΣΕΙΣ

Στο συγκεκριμένο κεφάλαιο συμπεριλαμβάνονται κάποιες γενικές ιδέες, συμπεράσματα από την κατασκευή, προτάσεις και προοπτικές, δυνατότητες εξέλιξης της κατασκευής καθώς και πιθανοί κίνδυνοι/προβλήματα που πιθανώς ελλοχεύουν.

5.1 Συμπεράσματα

Στην παρούσα πτυχιακή, υλοποιήθηκε ένας τρόπος διαχείρισης συσκευών έξυπνου σπιτιού με χαμηλό κόστος η οποία στη βασική δομή της. Επιπλέον αξιοποιεί την υπάρχουσα υλικοτεχνική υποδομή που υπάρχει στα περισσότερα σπίτια και με μικρές αναβαθμίσεις μπορεί να αποτελέσει την βάση πάνω στην οποία θα εξελιχθούν και θα αναπτυχθούν οι δυνατότητες του έξυπνου σπιτιού. Ένα από τα μεγάλα προβλήματα τα οποία προκύπτουν άμεσα και από την βιβλιογραφική έρευνα, είναι η έλλειψη ενός παγκόσμιου προτύπου, πάνω στο οποίο θα μπορέσει να γίνει περεταίρω ανάπτυξη του έξυπνου σπιτιού. Αυτό έχει ως συνέπεια την αδυναμία ενσωμάτωσης ενός συγκεκριμένου πρωτοκόλλου επικοινωνίας από τους κατασκευαστές, προκειμένου να προσθέσουν δυνατότητες στις συσκευές τους οι οποίες θα μπορούσαν να αξιοποιηθούν από ένα smarthome και όχι από μεμονωμένες συσκευές με συγκεκριμένες δυνατότητες (πχ. η επικοινωνία μέσω NFC σε κάποιες συσκευές η οποία προς το παρόν είναι εφικτή στο ευρύ κοινό μόνο με τη χρήση κινητών που υποστηρίζουν αυτή τη δυνατότητα).

5.2 Κίνδυνοι και προβληματισμοί

Η όλο και αυξανόμενη τάση για προσαρμογή και εξέλιξη των χώρων κατοικίας μας σε έξυπνα σπίτια δυστυχώς δεν έχει μόνο οφέλη για τον τελικό χρήστη αλλά και κινδύνους, όπως κάθε τεχνολογικό επίτευγμα. Η δυνατότητα ελέγχου του σπιτιού μας μέσω τεχνολογιών δικτύων και ίντερνετ και ειδικά η δυνατότητα απομακρυσμένου ελέγχου, δεν διαχωρίζει και πολύ το σπίτι μας από ένα διακομιστή (server) ή ένα μικρό δίκτυο. Αναφέρομαι κυρίως στο θέμα της ασφάλειας. Θα πρέπει μέσα στον σχεδιασμό αυτών των τεχνολογιών να ενσωματωθούν firewalls και νέα συστήματα ασφάλειας προκειμένου να καταστεί αδύνατο για κάποιον να καταφέρει να συνδεθεί με το οικιακό σας δίκτυο. Σκεφτείτε να μπορούσε κάποιος να συνδεθεί μέσα σε ένα έξυπνο σπίτι. Θα μπορούσε

πρακτικά να ελέγξει όλο το σπίτι και οι πόρτες να ανοίγουν οικειοθελώς, χωρίς να χρειαστεί να κάνει κάποια διάρρηξη, ή να μπορέσει να συλλέξει και να εκμεταλλευτεί όλα αυτά τα δεδομένα τα οποία χειρίζονται όλοι οι αισθητήρες ενός έξυπνου σπιτιού.

5.3 Δυνατότητες εξέλιξης

Η παρούσα πτυχιακή παρουσιάζει κάποιες πολύ βασικές δυνατότητες που μπορεί να έχει ένα έξυπνο σπίτι αλλά στην πλήρη μορφή της οι δυνατότητες είναι ατελείωτες. Ένα βασικό ζήτημα το οποίο προαναφέρθηκε είναι η δυνατότητα αλληλεπίδρασης του ανθρώπου και του σπιτιού μέσω του φυσικού λόγου. Αυτό θα δώσει τη δυνατότητα για πολύ πιο φυσική ενσωμάτωση των νέων αυτών τεχνολογιών στον άνθρωπο ενώ παράλληλα θα ενισχύσει την αποτελεσματικότητα του συστήματος. Αυτό θα μπορούσε να επιτευχθεί μέσω της χρήσης γραμματικών δένδρων μέσω των οποίων το σύστημα μπορεί να αναγνωρίσει τα διάφορα μέρη του λόγου και έτσι να προβεί σε πιο ακριβή αποτελέσματα. Για παράδειγμα οι φράσεις “ο Γιάννης έγγραψε ένα γράμμα” και “ένα γράμμα γράφτηκε από τον Γιάννη” είναι μεν διαφορετικές αλλά οδηγούν στην ίδια εννοιολογική σημασία όπως αυτή προκύπτει από την θέση των μερών του λόγου (ουσιαστικό, υποκείμενο αντικείμενο κλπ.). Μέσω αυτής της δυνατότητας, ο άνθρωπος θα μπορεί να μιλάει πιο φυσικά (και όχι με προκαθορισμένες εντολές που θα πρέπει να ειπωθούν αυτούσιες) και το σύστημα να καταλαβαίνει τη σημασία του τι είπε και ότι να στέκεται στη σειρά των λέξεων και των φράσεων που χρησιμοποιήθηκαν. Μια ακόμη εξέλιξη που βασίζεται στα προηγούμενα είναι η χρήση της τεχνητής νοημοσύνης (Artificial Intelligence) μέσω της οποίας το σύστημα θα μπορεί όχι μόνο να αναλύει καλύτερα την κάθε πρόταση και να φτάνει στο πραγματικό νόημα και ζητούμενο, αλλά και θα μπορεί να μαθαίνει και να εξελίσσεται μέσα από την αλληλεπίδραση ανθρώπου – μηχανής, μαθαίνοντας πχ τις συνήθειες και τον τρόπο ομιλίας του κάθε χρήστη παρέχοντας του τη δυνατότητα να κάνει προτάσεις στο χρήστη (όπως να του προτείνει να ανάψει το θερμοσίφωνα επειδή σύμφωνα με τα στατιστικά ξέρει ότι ο συγκεκριμένος χρήστης απολαμβάνει πάντα ένα ζεστό μπάνιο πριν τον ύπνο) αντί να εκτελεί μόνο εντολές.

ΠΑΡΑΡΤΗΜΑ Α’

ΓΛΩΣΣΑΡΙ

Η.Υ : Ηλεκτρονικός Υπολογιστής

6LoWPAN: IPv6 over Low-Power Wireless Personal Area Networks

BLE: Bluetooth Low Energy

HTML: Hyper Text Markup Language

UTF-8: Unicode Transformation Format – 8-bit

CSS: Cascading Style Sheets

JS: JavaScript

AJAX: Asynchronous JavaScript And XML

PHP: Αρχικά σήμαινε Personal Home Page αλλά πλέον χρησιμοποιείται το ακρωνύμιο PHP: Hypertext Pre-processor

API: Application Programming Interface

JSON: JavaScript Object Notation

GPIO: General Purpose Input Output

URL: Uniform Resource Locator

cURL: Command line URL

PDO: PHP Data Objects

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] http://www.investopedia.com/terms/s/smart-home.asp

[2] https://en.wikipedia.org/wiki/Home_automation

[3] Κωνσταντίνος Ν. Αρκουλής, "Smart Home (Έξυπνο Σπίτι)", *Πτυχιακή εργασία*, Τμήμα Ηλεκτρονικών Υπολογιστών Συστημάτων, ΤΕΙ Πειραιά, 2015.

[4] Järvinen, Hannu, "Web Technology based Smart Home Interoperability", Διδακτορική διατριβή, Department of Computer Science, Aalto University, 2015

[5] Μαρίνα Αρντίτ, Μπικίκου Ευγενία, "Έξυπνο Σπίτι – Smarthome", Πτυχιακή Εργασία, Τμήμα Ηλεκτρονικών Υπολογιστικών Συστημάτων, ΤΕΙ Πειραιά, 2015

[6] Richard Harper, "Inside the Smart Home", Springer-Verlag London, 2003

[7] Gann, D., Barlow, J., & Venables, T. (1999). Digital Futures: Making Homes Smarter, Coventry/York: Chartered Institute of Housing and Joseph Rowntree Foundation

[8] Barlow, J. (1997). "Smart Homes Project. "User Needs Analysis", Report to the Joseph Rowntree Foundation. Mimeo

[9] Barlow, J., & Gann, D. (1998). "A Changing Sense of Place: Are Integrated IT Systems Reshaping the Home?", paper presented to the Technological Futures, Urban Futures Conference, Durham, 23–24 April.

[10] Meyer, S., & Schulze, E. (1996). "The Smart Home in the 1990s. Acceptance and Future Usage in Private Households in Europe", in The Smart Home: Research Perspectives, The European Media Technology and Everyday Life Network (EMTEL), Working Paper No. 1, University of Sussex, Brighton.