ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

"Smart Home με Raspberry Pi Python και Android Application Java"

ΑΘΑΝΑΣΙΟΣ ΝΑΘΑΝΑΗΛ

ΕΠΙΒΛΕΠΩΝ: Χριστοδούλου Π. Σωτήρης, Λέκτορας

ΠΑΤΡΑ 2020

1. Περιεχόμενα

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Πάτρα

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Χριστοδούλου Π. Σωτήρης,

2. Τζήμας Γιάννης,

3. Παρασκευάς Μιχάλης,

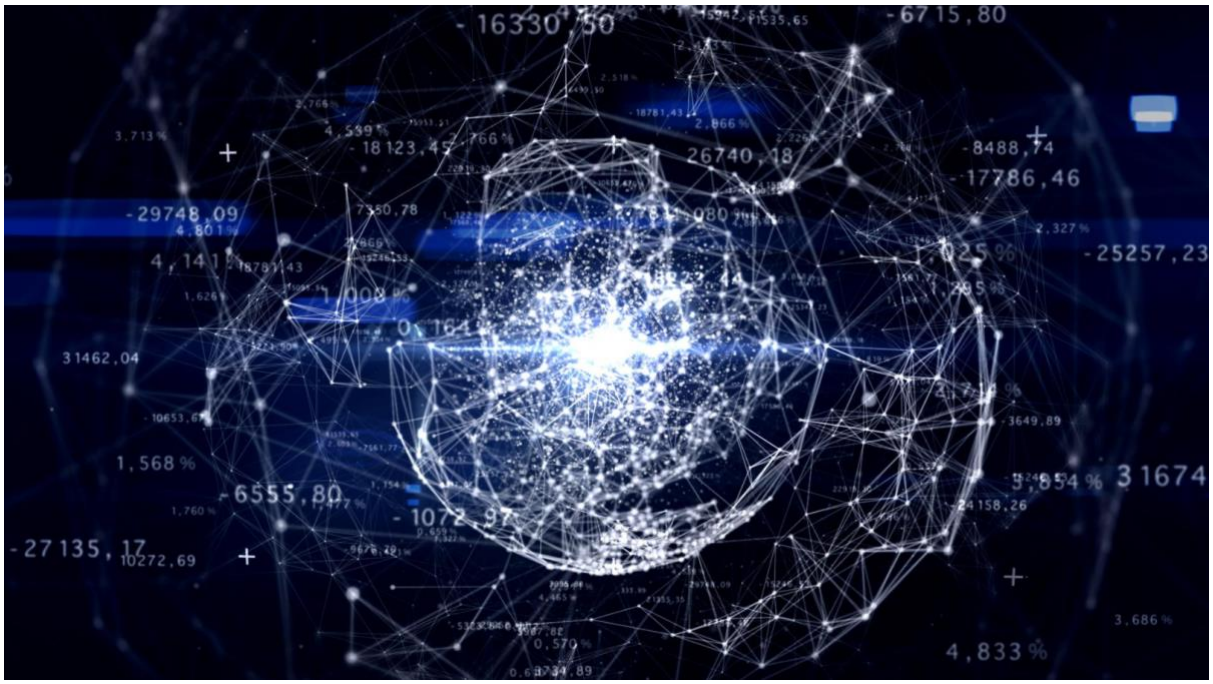
Υπεύθυνη Δήλωση Φοιτητή

Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη εργασία.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πελοποννήσου δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

1. Εισαγωγή

1.1 *Η έλευση του Internet of Things στις ζωές μας*



https://www.forbes.com/sites/alextapscott/2016/08/08/the-internet-of-things-needs-a-ledger-of-things/

Στη σύγχρονη εποχή, η εξέλιξη της τεχνολογίας έχει φτάσει στο σημείο όπου πολλές συσκευές της καθημερινότητάς μας μπορούν να επικοινωνούν μεταξύ τους και να ανταλλάσσουν πολύτιμες πληροφορίες. Τέτοιου είδους συσκευές μπορούν να είναι τα smartphones, οι υπολογιστές, τα φορητά gadget ή ακόμα και το ψυγείο, τα φώτα, τα παράθυρα και κάθε άλλη συσκευή που μπορείτε να σκεφτείτε. Αυτό επιτυγχάνεται με τη χρήση ειδικών ενσωματωμένων αισθητήρων που συλλέγουν δεδομένα και εκτελούν συγκεκριμένες ενέργειες εντός ενός καθορισμένου δικτύου. Φανταστείτε τον εαυτό σας να είναι σε θέση να ρυθμίσει αυτόματα τη θέρμανση ή τον φωτισμό του σπιτιού σας χωρίς να πατήσετε οποιοδήποτε κουμπί. Αυτό μπορεί να επεκταθεί ακόμα και σε τεράστια εργοστάσια όπου ένας ειδικός εξοπλισμός θα ειδοποιεί αμέσως το προσωπικό συντήρησης σε περίπτωση επικείμενης μηχανικής βλάβης. Με άλλα λόγια, το «Διαδίκτυο των πραγμάτων» είναι το τεχνολογικό μέλλον που όχι μόνο θα διευκολύνει τη ζωή μας, αλλά θα κάνει τον κόσμο μας πιο ασφαλή και αποτελεσματικό.

1.2 *Τι είναι το Internet of Things (IoT);*

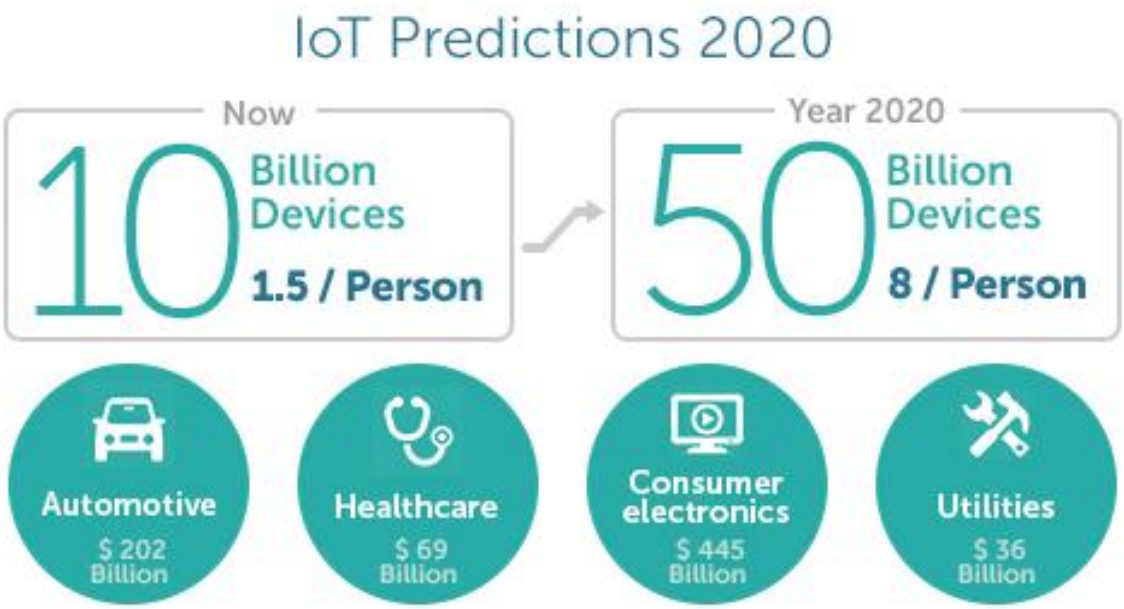
Με τον όρο internet of thins περιγράφουμε την δυνατότητα **κάθε τύπου συσκευή να συνδέεται στο ίντερνετ**: από smartphones και laptops, μέχρι αυτοκίνητα, χύτρες και λάμπες. Εννοείται πως στην «εξίσωση» προστίθενται και όλες εκείνες οι συσκευές που μπορεί να μην έχουν καμία άμεση εφαρμογή στην καθημερινότητά μας, ωστόσο και αυτά με κάποιον τρόπο βρίσκουν τον δρόμο τους για τον παγκόσμιο ιστό (βλ. φανάρια δρόμων, διασταυρώσεις γραμμών τρένων, κινητήρες αεροπλάνων κ.α.). αν κάτι έχει διακόπτη, η λογική λέει πως μπορεί να αποτελέσει μέρος του internet of things –αν δεν αποτελεί ήδη…

Η φιλοσοφία του internet of things θέλει κάθε αντικείμενο που περικλείει τις ζωές μας να είναι online σε ένα κολοσσιαίο δίκτυο, μέσω του οποίου δεν θα αλληλεπιδρούν μεταξύ τους μοναχά οι άνθρωποι αλλά και οι μηχανές. Ενδεικτικά ομιλώντας, η εταιρεία αναλύσεων Gartner, υπολογίζει με αρκετή δόση μετριοπάθειας πως το 2020 περί τις **26 δισ. συσκευές θα είναι συνδεδεμένες** στο διαδίκτυο – και λέμε έτσι διότι άλλες προβλέψεις τοποθετούν τον πήχη στα 100 δισ. συσκευών για την ίδια περίοδο.

Δεν χρειάζεται πολλή φαντασία για αντιληφθεί κανείς την **εφαρμογή του internet of things στις ζωές μας**. Το γεγονός πως μπορούμε να χειριστούμε την τηλεόραση μέσω του smartphone μας, το firmware update του αυτοκινήτου μας μέσω USB stick, το κόνσεπτ του e-banking, οι σκούπες που συνοδεύονται από iOS & Android app, οι LED λάμπες που προγραμματίζονται άνετα μέσω iPhone, αυτά και άλλα πολλά είναι απλοϊκές, καθημερινές εκφάνσεις του internet of things και του πώς αυτό έχει μπει για τα καλά στις ζωές μας. Αν θέλετε να πάμε σε λίγο πιο σύνθετα παραδείγματα, μπορούμε να πάρουμε τη λειτουργία των φαναριών στον δρόμο, τα συστήματα ελέγχου στις μηχανές της Αστυνομίας, την τηλεμετρία σε μετρό και λεωφορεία και τις φωτεινές πινακίδες με τα μηνύματα κειμένου στις μεγάλες λεωφόρους και εθνικές οδούς.

Ο κανόνας είναι πως *«αν κάτι μπορεί να συνδεθεί, θα συνδεθεί»*. Δεν χρειάζεται να ανησυχείτε μιας που ο τελικός χρήστης, **ο καταναλωτής, μόνο κερδισμένος θα βγει από την περαιτέρω ανάπτυξη του internet of things**. Σκεφτείτε π.χ. το αυτοκίνητό σας να είναι συνδεδεμένο με την ατζέντα σας ώστε να ξέρει τα ραντεβού σας και άρα τι ώρα πρέπει να είστε πού. Θα μπορεί να κατεβάσει δεδομένα για την κίνηση στους δρόμους και να επιλέξει αυτόματα τη διαδρομή που θα ακολουθήσετε προκειμένου να φτάσετε στην ώρα σας στον προορισμό σας –κι αν δεν προλαβαίνετε, να σας προτρέπει μέσω του –επίσης συνδεδεμένου- smartphone σας να καλέσετε για να ενημερώσετε. Δεν θα ήταν άσχημα αν το wearables που φοράτε όλη μέρα, το wristband ή το smartwatch ας πούμε, ενημέρωνε σε πραγματικό χρόνο τον ιατρικό σας φάκελο στον οποίο είχε πρόσβαση ο γιατρός σας με τα βασικά σας στοιχεία (πίεση, παλμοί κλπ) και μάλιστα τον ειδοποιούσε σε περίπτωση που κάτι πήγαινε στραβά;

Το internet of things δεν εξαντλείται απλά στα φανάρια των δρόμων, την οδηγική εμπειρία των αυτοκινήτων ή το περίφημο έξυπνο/συνδεδεμένο σπίτι. **Βρίσκει εφαρμογή σχεδόν παντού** κι αν δεν το κάνει ήδη σε κάποιο βαθμό, ετοιμαστείτε γιατί θα το πραγματοποιήσει στο μέλλον, εγγύς ή απώτερο. Το μεγάλο στοίχημα του internet of things δεν είναι να προσελκύσει κι άλλες συσκευές στις τάξεις του: νομοτελειακά αυτό θα συμβεί αργά ή γρήγορα αφού είναι σχεδόν αδύνατη η κυκλοφορία νέων μοντέλων (δεν παίζει ρόλο η προϊοντική κατηγορία) χωρίς τουλάχιστον επαρκείς λύσεις στον τομέα της συνδεσιμότητας. Το ζητούμενο είναι η ασφάλεια του απίθανου όγκου δεδομένων που διακινείται σε καθημερινή βάση διαδικτυακά. Αν κάποιος είχε πρόσβαση στα ψηφιακά δεδομένα μας, θα μπορούσε να ξεγυμνώσει τις ζωές μας εν ριπή οφθαλμού και αυτό είναι κάτι που κανείς δεν θέλει. Το internet of things είναι ασφαλές αλλά μπορεί και πρέπει να γίνει ασφαλέστερο.



https://redwiredc.co.uk/the-internet-of-things-iot-and-data-centres-future-today/

Το IoT θα προσφέρει μια νέα αξία στις ζωές των καταναλωτών, μετατρέποντας καθημερινά αντικείμενα σε έξυπνες συνδεδεμένες συσκευές που κατανοούν αποτελεσματικά τους καταναλωτές τους και προσφέρουν μεγάλα οφέλη και υπηρεσίες που ικανοποιούν τις ανάγκες και τις επιθυμίες των ανθρώπων. Για παράδειγμα, το σύστημα ασφαλείας του σπιτιού σας θα ανιχνεύει το χρόνο που ξυπνάτε το πρωί και θα στέλνει ένα μήνυμα στην καφετιέρα σας για να προετοιμάσει τον καφέ σας!

2. Σχετικές λειτουργίες, προβλήματα και λύσεις

2.1 *Έρευνα σχετικά με τις λειτουργίες, τα συστήματα και τις τεχνολογίες*

Όταν καταφέραμε να συνδέσουμε τελικά περισσότερα αντικείμενα από ό, τι ανθρώπους στο Διαδίκτυο, ανοίξαμε ένα τεράστιο παράθυρο που μας έδωσε την ευκαιρία να δημιουργήσουμε εφαρμογές σε τομείς όπως η αυτοματοποίηση και η επικοινωνία από μηχανή σε μηχανή. Στην πραγματικότητα, οι δυνατότητες είναι σχεδόν ατελείωτες.

Οι τομείς εφαρμογής περιλαμβάνουν: τον πολεοδομικό σχεδιασμό, την διαχείριση αποβλήτων, το περιβάλλον, την κοινωνική αλληλεπίδραση, την αντιμετώπιση καταστάσεων έκτακτης ανάγκης, τις έξυπνες αγορές, τις οικιακές συσκευές, τον αυτοματισμό και πολλά άλλα. Τα παρακάτω παραδείγματα υπογραμμίζουν μερικούς από τους τρόπους με τους οποίους το IoT καθιστά τη ζωή των ανθρώπων ευκολότερη, πιο βολική και πιο παραγωγική.

2.1.1 Έξυπνα Σπίτια

https://grayhats.in/what-is-home-automation-and-how-does-it-work/

Τα μελλοντικά έξυπνα σπίτια θα έχουν "συνείδηση" για το τι συμβαίνει μέσα σε ένα κτίριο, επηρεασμένο κυρίως από τρεις πτυχές: τη χρήση των πόρων, την ασφάλεια και την άνεση. Ο στόχος είναι να επιτευχθούν καλύτερα επίπεδα διαβίωσης καθώς και μείωση του συνολικού κόστους. Επιπλέον, τα έξυπνα σπίτια θα είναι σε θέση να αντιμετωπίσουν επαρκώς ζητήματα ασφάλειας όπως η ανίχνευση πυρκαγιάς ή η εισβολή κλοπής. Διαφορετικές οντότητες θα βρίσκονται σε θέση να συνεργάζονται στο κάθε σπίτι, όπως πάροχοι Διαδικτύου, κατασκευαστές συσκευών, τηλεπικοινωνιακοί φορείς, πάροχοι υπηρεσιών οπτικοακουστικών μέσων, εταιρείες παροχής υπηρεσιών ασφαλείας ή ακόμη ιδρύματα κοινής ωφελείας.

- Χρήση ενέργειας και νερού

Η κατανάλωση ενέργειας και νερού παρακολουθείται για να παράγει τα κατάλληλα συμπεράσματα για τη μείωση των πόρων και του κόστους.

- Συσκευές τηλεχειρισμού

Ενεργοποιήστε και απενεργοποιήστε τις συσκευές από απόσταση για να αποφύγετε ατυχήματα και να εξοικονομήσετε ενέργεια.

- Συστήματα ανίχνευσης εισβολής

Παρακολουθείστε την κατάσταση των παραθύρων και των θυρών (κλειδωμένη / ξεκλείδωτη) για να αποφύγετε παραβιάσεις και να αποτρέψετε τους εισβολείς.

- Έλεγχος πρόσβασης περιμέτρου

Ελέγξτε την πρόσβαση σε απαγορευμένες ζώνες και εντοπίστε άτομα που βρίσκονται σε μη εγκεκριμένες περιοχές.

2.1.2 Έξυπνες Πόλεις

https://ecoworld.co/ecoworld-city-corporation.html

Η έξυπνη πόλη είναι μια αστική περιοχή που είναι φιλική προς το περιβάλλον και προωθεί την καλή ποιότητα ζωής. Ορισμένα από τα χαρακτηριστικά μιας έξυπνης πόλης περιλαμβάνουν την οικονομία, τους πολίτες, το κράτος, τα μεταφορικά μέσα και το περιβάλλον. Πολλοί τομείς και βιομηχανίες θα επωφεληθούν από μια πιο ψηφιοποιημένη και έξυπνη πόλη.

- Ασφάλεια και πρόληψη
Παρακολούθηση των δονήσεων και την κατάσταση των υλικών σε κτίρια, γέφυρες και ιστορικά μνημεία.

- Χάρτες αστικού θορύβου
Παρακολούθηση θορύβου σε συνωστισμένους χώρους όπως μπαρ και κεντρικές περιοχές Σε πραγματικό χρόνο.

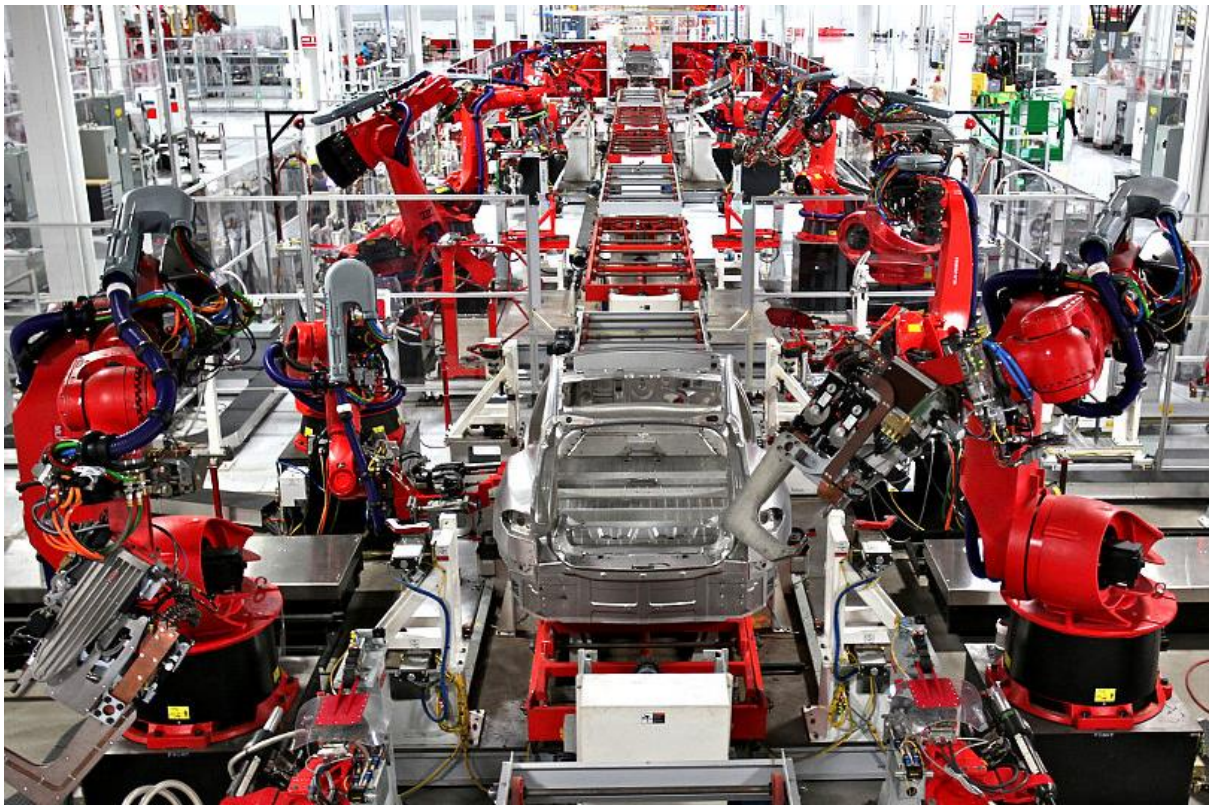
- Ανίχνευση Smartphone
Εντοπισμός συσκευών iPhone, Android ή Windows Phone και γενικά κάθε συσκευής που υποστηρίζει λειτουργείες όπως Wi-Fi ή Bluetooth.

- Έξυπνο φωτισμό
Ευφυής και προσαρμοστικός φωτισμός στους δρόμους με βάση την κίνηση και τις καιρικές συνθήκες.

-Διαχείριση των αποβλήτων
Παρακολούθηση του επιπέδου των απορριμμάτων για να βελτιστοποιηθεί η διαδρομή για την περισυλλογή των αποβλήτων.

- Έξυπνοι δρόμοι
Ευφυείς αυτοκινητοδρόμους με προειδοποιητικά μηνύματα ανάλογα με τις κλιματικές συνθήκες και τα απροσδόκητα γεγονότα, όπως τα ατυχήματα ή η κυκλοφοριακή συμφόρηση.

2.1.3 Έξυπνη Βιομηχανία



https://www.forbes.com/sites/amitchowdhry/2014/07/22/tesla-motors-idles-california-plant-to-prepare-for-electric-suv-line/

Στην παγκόσμια αλυσίδα εφοδιασμού, οι εταιρείες θα μπορούν να παρακολουθούν όλα τα προϊόντα τους μέσω ετικετών αναγνώρισης ραδιοσυχνοτήτων (RFID). Ως εκ τούτου, το κόστος λειτουργικότητας θα μειωθεί και η παραγωγικότητα θα αυξηθεί ραγδαία. Επίσης, η συντήρηση των μηχανών θα διευκολυνθεί από τους ανάλογους αισθητήρες, επιτρέποντας την παρακολούθηση σε πραγματικό χρόνο. Γενικά, η πληροφορική θα παρέχει αυτοματοποιημένες διαδικασίες που συνεπάγονται δραστική μείωση του αριθμού των εργαζομένων που απαιτούνται. Οι εργαζόμενοι θα αντικατασταθούν από ένα σύνολο σαρωτών barcode, αισθητήρες, ενεργοποιητές και σύνθετη ρομποτική. Χωρίς αμφιβολία, αυτές οι νέες τεχνολογίες θα προσφέρουν αρκετές ευκαιρίες για νέες θέσεις εργασίας. Ο προγραμματισμός και η επισκευή αυτού του εξοπλισμού θα είναι μια πολύ κοινή δουλειά στο μέλλον. Το IoT θα αυτοματοποιήσει ορισμένες σημαντικές διαδικασίες του βιομηχανικού περιβάλλοντος, όπως:

- **Η ποιότητα του εσωτερικού αέρα**
  Παρακολούθηση των τοξικών αερίων και το επίπεδο οξυγόνου στο χώρο εργασίας για να εξασφαλιστεί η ασφάλεια των εργαζομένων και των αγαθών.

- **Παρακολούθηση θερμοκρασίας**
  Θα πραγματοποιούνται έλεγχοι για τη θερμοκρασία μέσα στα ψυγεία που περιέχουν ευαίσθητα προϊόντα.

- **Εσωτερική παρακολούθηση**
  Εσωτερική αξιολόγηση δωματίου με χρήση ενεργών ZigBee και παθητικών ετικετών (RFID / NFC)

2.1.4 Έξυπνο Περιβάλλον

Το έξυπνο περιβάλλον διαπνέεται από την ιδέα ενός φυσικού κόσμου που είναι πλούσιος και εμφανώς συνυφασμένος με αισθητήρες, ενεργοποιητές, οθόνες και υπολογιστικά δεδομένα, τα οποία ενσωματώνονται αρμονικά στα καθημερινά αντικείμενα της ζωής μας και είναι συνδεδεμένα συνεχώς μέσω δικτύου.

- **Δασική πυρανίχνευση**
  Παρακολούθηση των αερίων καύσης και των συνθηκών που προκαλούν πυρκαγιά για τον εντοπισμό επικίνδυνων περιοχών.

- **Μόλυνση του αέρα**
  Έλεγχος των εκπομπών CO2 που παράγουν τα εργοστάσια και οι εξατμίσεις των αυτοκινήτων.

- **Παρατήρηση των επιπέδων χιονιού**
  Μέτρηση των δονήσεων της γης και των επιπέδων χιονιού με ενημερώσεις σε πραγματικό χρόνο για την αποφυγή του κινδύνου μιας χιονοστιβάδας.

- **Πρόωρη ανίχνευση σεισμών**
  Έλεγχος σε συγκεκριμένες περιοχές κραδασμών.

2.1.5 Μεταφορά και Εφοδιασμός

https://www.openpr.com/news/1880721/third-party-logistics-market-is-booming-worldwide-by-leading

Κατά τη μεταφορά προμηθειών, το IoT βελτιώνει το παγκόσμιο σύστημα ανίχνευσης και την αυτόματη αναγνώριση των εμπορευμάτων. Αυξάνει επίσης την ενεργειακή απόδοση και κατά συνέπεια μειώνει την κατανάλωση ενέργειας. Έτσι, το IoT αναμένεται να φέρει μεγάλες αλλαγές στην παγκόσμια αλυσίδα εφοδιασμού μέσω της «έξυπνης» κίνησης του φορτίου. Αυτό θα επιτευχθεί με τον συνεχή συγχρονισμό των πληροφοριών εφοδιασμού και την αδιάλειπτη παρακολούθηση σε πραγματικό χρόνο, η οποία θα καταστήσει την αλυσίδα εφοδιασμού διαφανή, ορατή και ελεγχόμενη, επιτρέποντας την έξυπνη και αποτελεσματική επικοινωνία μεταξύ ανθρώπων και αγαθών.

- **Μποτιλιάρισμα**
  Παρακολούθηση οχημάτων και πεζών βελτιστοποιώντας με αυτό τον τρόπο τη διαδικασία οδήγησης.

- **Έξυπνος χώρος στάθμευσης**
  Παρακολούθηση των ελεύθερων χώρων στάθμευσης.

- **Ποιότητα των συνθηκών φόρτωσης**
  Παρακολούθηση δονήσεων προκειμένου να διασφαλιστεί ότι τα δέματα παραμένουν κλειστά για όλη τη διάρκεια του ταξιδιού.

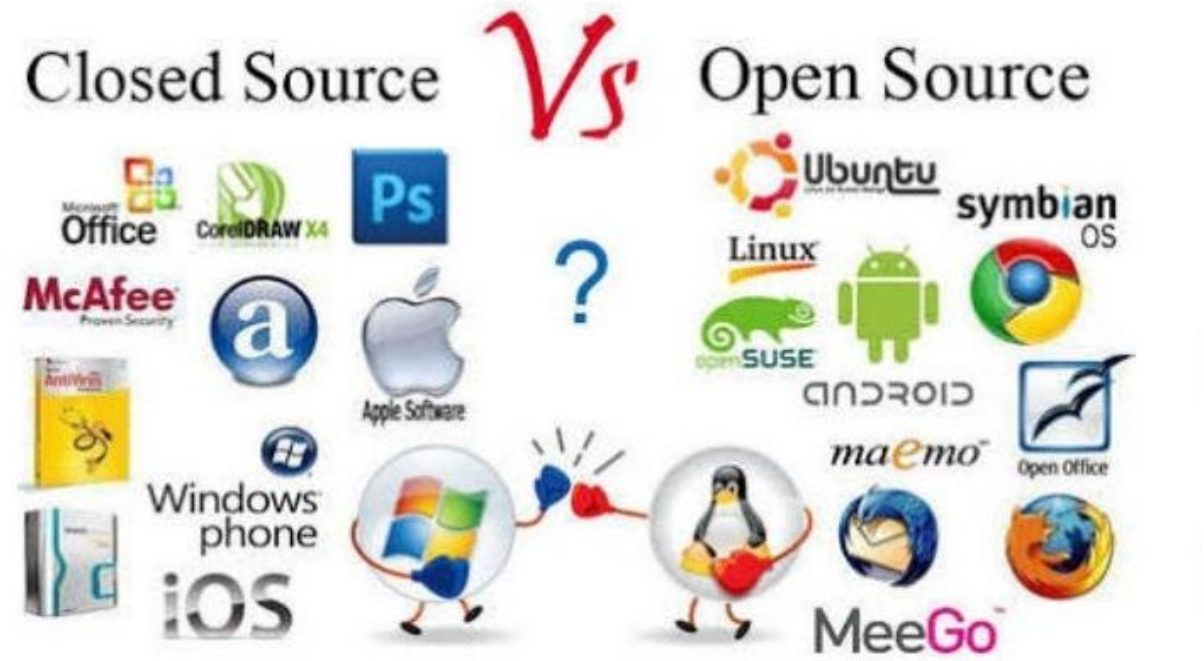
- **Τοποθέτηση αντικειμένων**
  Αναζήτηση μεμονωμένων αντικειμένων σε αποθήκες ή λιμάνια.

- **Ανίχνευση**
  Προειδοποίηση για τη μεταφορά εύφλεκτων αγαθών και απομόνωση εκείνων που εμπεριέχουν εκρηκτικό υλικό.

- **Παρακολούθηση προϊόντων**
  Έλεγχος των διαδρομών των ευαίσθητων προϊόντων όπως τα φάρμακα, τα κοσμήματα ή τα επικίνδυνα προϊόντα.

2.2 *Προβλήματα και Λύσεις*

2.2.1 Υψηλό Κόστος Αγοράς και "Κλειστού" Τύπου Κώδικας



https://medium.com/@chaomengting/is-open-source-software-more-reliable-or-secure-than-closed-source-software-5470732d7b90

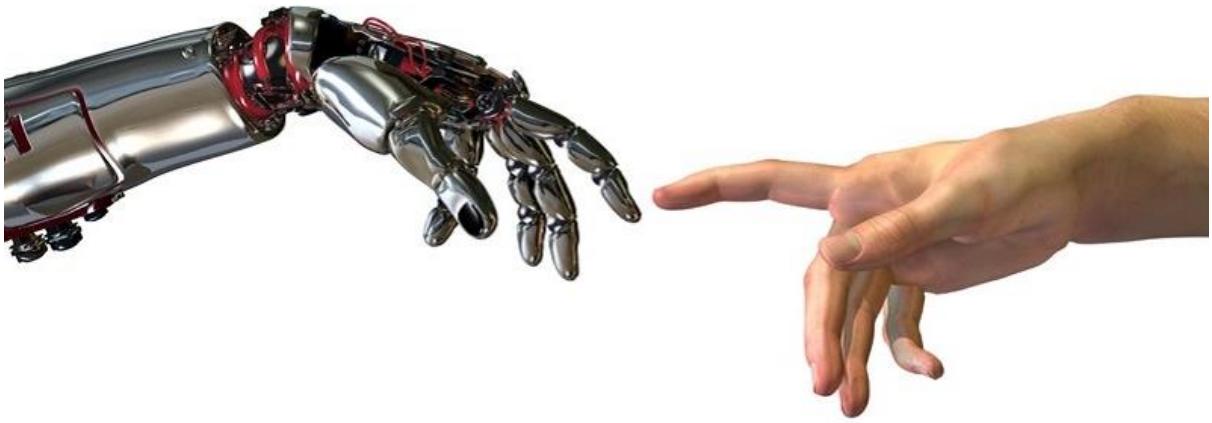
Σύμφωνα με τα παραδείγματα που παρουσιάζονται παραπάνω, συνειδητοποιούμε ότι οι χρήσεις του Διαδικτύου στην καθημερινότητά μας είναι πολύ σημαντικές και απεριόριστες. Ωστόσο, το κόστος αγοράς ενός τέτοιου εξοπλισμού για οικιακή χρήση από έναν μέσο χρήστη είναι πολύ υψηλό. Εκτός από την αγορά του υλικού των έξυπνων συσκευών, είναι σημαντικό να αγοράσετε το σωστό λογισμικό που θα συνδέει όλες αυτές τις συσκευές με την χρήση του Διαδικτύου. Αυτό το λογισμικό μπορεί να είναι είτε μια εφαρμογή στη συσκευή smartphone μας είτε μια διαδικτυακή εφαρμογή είτε μια πλατφόρμα αναγνώρισης φωνής. Η τιμή για όλα τα παραπάνω υλικά είναι πολύ υψηλή και αποτρεπτική για τον μέσο χρήστη.

Επιπλέον, ο κώδικας του λογισμικού που προσφέρουν οι μεγάλες εταιρείες είναι κλειστός που σημαίνει ότι δεν μπορούμε να παρέμβουμε και να αλλάξουμε ορισμένα κομμάτια κώδικα σύμφωνα με τις δικές μας ανάγκες και προτιμήσεις. Αυτό συμβαίνει επειδή οι μεγάλοι προμηθευτές δεν θέλουν όλοι να έχουν πρόσβαση και να γνωρίζουν πώς δημιουργήθηκε το προϊόν τους. Αυτή είναι η βασική έννοια του κλειστού κώδικα που δεν μπορεί να τροποποιηθεί.

2.2.2 Μια διαφορετική προσέγγιση

Βασικός σκοπός αυτής της διπλωματικής εργασίας είναι να προτείνει λύσεις στα παραπάνω προβλήματα και να επιτρέψει στους καθημερινούς μέσους χρήστες υπολογιστών να δημιουργήσουν το δικό τους έξυπνο σπίτι με το χαμηλότερο δυνατό κόστος. Αυτό μπορεί να γίνει με την αγορά ενός μόνο Raspberry Pi που μπορούμε να βρούμε σήμερα σε πολύ χαμηλή τιμή των 40 ευρώ. Χρησιμοποιώντας το υπάρχον δίκτυο του σπιτιού μας, μπορούμε να συνδέσουμε όλες τις έξυπνες συσκευές στο Διαδίκτυο και να τις διαχειριστούμε εξ αποστάσεως. Επίσης, το λογισμικό που θα δημιουργήσουμε θα βασίζεται αποκλειστικά σε τεχνολογίες ανοιχτού κώδικα, έτσι ώστε όλοι να μπορούν να χειριστούν και να προσαρμόσουν τον κώδικα όπως επιθυμεί σύμφωνα με τις δικές του ανάγκες. Επειδή η γραφική διεπαφή χρήστη (GUI) διαδραματίζει πολύ σημαντικό ρόλο στον τρόπο με τον οποίο οι χρήστες αλληλεπιδρούν με τις συσκευές τους, θα δημιουργήσουμε 3 διαφορετικές εφαρμογές που καλύπτουν το πλήρες φάσμα των αναγκών του κάθε χρήστη.

2.2.3 Αλληλεπίδραση Ανθρώπου – Υπολογιστή



https://onlineinfodash.blogspot.com/2017/01/is-technology-making-human-more-busy.html

Ο κύριος στόχος της αλληλεπίδρασης Ανθρώπου-Υπολογιστή είναι η βελτίωση της επικοινωνίας μεταξύ χρηστών και υπολογιστών με τον πιο κατάλληλο και φιλικό σχεδιασμό για το χρήστη, προσαρμοσμένο πάντα στις ανάγκες του ανθρώπου. Η αλληλεπίδραση Ανθρώπου-Υπολογιστή βασίζεται στην κατανόηση των αναγκών των χρηστών, των σωστών οδηγιών σχεδιασμού διεπαφών και πολύ χρήσιμων μεθόδων αξιολόγησης. Η αλληλεπίδραση μεταξύ χρηστών και υπολογιστών εμφανίζεται στο επίπεδο της διεπαφής του χρήστη, μέσω κατάλληλου λογισμικού και υλικού. Ένα τυπικό παράδειγμα είναι οι χαρακτήρες και τα αντικείμενα που εμφανίζονται από το λογισμικό σε μια οθόνη υπολογιστή, καθώς και οτιδήποτε εισάγουν ως πληροφορία οι χρήστες μέσω περιφερειακών συσκευών όπως το πληκτρολόγιο, το ποντίκι, η οθόνη αφής ή ακόμα και η ανθρώπινη φωνή. Παρακάτω παρουσιάζουμε τους 3 διαφορετικούς τύπους της γραφικής διεπαφής χρήστη που θα αναπτύξουμε.

2.2.4 “Native” Εφαρμογές (Android, Windows Phone, iOS)

Οι “Native” εφαρμογές δημιουργούνται και σχεδιάζονται αποκλειστικά για να τρέχουν σε ένα συγκεκριμένο λειτουργικό σύστημα, όπως το Android, τα Windows κ.λπ. Μπορούμε να τα κατεβάσουμε ή να τα αγοράσουμε από μια ηλεκτρονική πηγή όπως το Google Play ή το GitHub. Μπορούμε επίσης να δημιουργήσουμε τη δική μας εφαρμογή από το μηδέν και να την εγκαταστήσουμε απευθείας στο smartphone, το tablet μας ή σε οποιαδήποτε άλλη συσκευή που διαθέτουμε. Στις επόμενες ενότητες θα δημιουργήσουμε τη δική μας εφαρμογή Android, μέσω της οποίας θα μπορούμε να διαχειριστούμε και να αλληλοεπιδράμε με διάφορες έξυπνες οικιακές συσκευές.

2.2.5 “Web based” Εφαρμογές

https://might-web.com/web-technology-is-important-for-achievement-and-development-in-todays-business-market/

Οι εφαρμογές που βασίζονται στο Web χρησιμοποιούν τη γλώσσα HTML5 για να δημιουργήσουν ένα ψηφιακό περιβάλλον υψηλής ευκρίνειας. Η γλώσσα CSS χρησιμοποιείται για να βελτιώσει αισθητικά την εμφάνιση και την αίσθηση της εφαρμογής μας και η JavaScript χρησιμοποιείται για να δώσει περισσότερες λειτουργίες και δυνατότητες στο λογισμικό μας. Έτσι, οι εφαρμογές που βασίζονται στο Web μπορούν να λειτουργούν σε οποιαδήποτε πλατφόρμα που υποστηρίζει ένα σύγχρονο πρόγραμμα περιήγησης, όπως τα Windows 10, το Ubuntu, το Mac OS, το Android, το iOS κλπ. Στα επόμενα κεφάλαια, θα δημιουργήσουμε μια Web Based Application χρησιμοποιώντας HTML5, CSS, JavaScript και PHP που τρέχει σε οποιοδήποτε σύγχρονο πρόγραμμα περιήγησης. Με αυτόν τον τρόπο, θα είμαστε σε θέση να διαχειριστούμε από απόσταση και να δίνουμε συγκεκριμένες εντολές στο Raspberry Pi.

2.2.6 Εφαρμογές Φωνητικών Εντολών

https://www.brigittesstudio.com/vocal-toning

Η αναγνώριση φωνής είναι ο υποτομέας της υπολογιστικής γλωσσολογίας που αναπτύσσει μεθοδολογίες και τεχνολογίες που επιτρέπουν την αναγνώριση και τη μετάφραση της ομιλούμενης γλώσσας σε κείμενο για ηλεκτρονικούς υπολογιστές. Είναι επίσης γνωστή ως "αυτόματη αναγνώριση ομιλίας" (ASR), "αναγνώριση ομιλίας υπολογιστή" ή απλά "ομιλία σε κείμενο" (STT). Ενσωματώνει γνώσεις και έρευνες στους τομείς της γλωσσολογίας , της πληροφορικής και της ηλεκτρολογίας .

Ορισμένα συστήματα αναγνώρισης ομιλίας απαιτούν "εκπαίδευση" (που ονομάζεται επίσης "εγγραφή") όπου ένας μεμονωμένος ομιλητής διαβάζει κείμενο ή απομονωμένο λεξιλόγιο στο σύστημα. Το σύστημα αναλύει τη συγκεκριμένη φωνή του ατόμου και το χρησιμοποιεί για να τελειοποιήσει την αναγνώριση της ομιλίας αυτού του ατόμου, με αποτέλεσμα μεγαλύτερη ακρίβεια. Τα συστήματα που δεν χρησιμοποιούν την εκπαίδευση ονομάζονται συστήματα "ανεξάρτητα από ομιλητή". Τα συστήματα που χρειάζονται εκπαίδευση για να λειτουργήσουν ονομάζονται συστήματα "εξαρτημένα από ομιλητής".

Οι εφαρμογές που ελέγχονται με φωνή, μας επιτρέπουν να εκτελούμε διάφορες εργασίες χρησιμοποιώντας φωνητικές εντολές μειώνοντας την ανάγκη πληκτρολόγησης. Στην πραγματικότητα, δεν χρειάζεται καν να χρησιμοποιήσουμε το πληκτρολόγιό μας ή το ποντίκι μας, διότι εκφράζοντας συγκεκριμένες λέξεις-κλειδιά, η εφαρμογή γνωρίζει ακριβώς ποιες πληροφορίες θα πρέπει να ανακτήσει. Αυτή η τεχνολογία μπορεί επίσης να προσφέρει αμέτρητες ευκαιρίες σε άτομα με ειδικές ανάγκες που θα τους βοηθήσει να ζουν άνετα και ανεξάρτητα στα σπίτια τους. Στο τελευταίο μέρος του έργου μου, θα δημιουργήσουμε ένα σύστημα αναγνώρισης φωνής, το οποίο θα κατανοεί συγκεκριμένες φωνητικές εντολές και θα ενεργοποιεί ανάλογα τα αντίστοιχα συμβάντα στο Raspberry Pi.

3. Linux και λογισμικό "Ανοιχτού" τύπου κώδικα

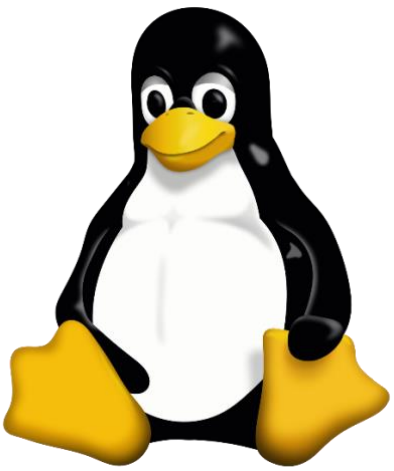
3.1 *Η σημαντικότητα των λογισμικών "Ανοιχτού" τύπου κώδικα*

https://www.kynetics.com/docs/2018/open-source-software-and-licensing/

Το όλο μας έργο βασίζεται σε τεχνολογίες ανοιχτού κώδικα. Το Open Source είναι ένας όρος που υποδεικνύει ότι ο κώδικας ενός προγράμματος, ενός εργαλείου, ενός πλαισίου ή ενός λειτουργικού συστήματος είναι ορατός και προσβάσιμος από όλους. Με αυτόν τον τρόπο, κάποιος μπορεί ελεύθερα να χρησιμοποιεί, να αντιγράφει, να διανέμει και να τροποποιεί τον κώδικα ανάλογα με τις ανάγκες του / της.

3.2 *Τι είναι το Linux;*

Το Linux είναι ένα λειτουργικό σύστημα Unix που δημιουργήθηκε αρχικά από τον Linus Torvalds και στη συνέχεια αναπτύχθηκε με τη βοήθεια προγραμματιστών από όλο τον κόσμο. Τα μεγαλύτερα πλεονεκτήματα της είναι η ταχύτητα και η σταθερότητα. Έχει απεριόριστες δυνατότητες, απίστευτη ευελιξία και μεγάλη ποικιλία λογισμικού. Αυτό που κάνει το Linux διαφορετικό από τα περισσότερα άλλα λειτουργικά συστήματα, είναι η επαναστατική τεχνολογία που κρύβεται πίσω από αυτό και η μεγάλη αποδοχή που έχει βρει μέσα στην αγορά.



3.3 *Διανομές Linux*

Ένα από τα καλύτερα χαρακτηριστικά του Linux είναι οι ατελείωτες διαθέσιμες επιλογές που προσφέρουν στους χρήστες του. Ο πυρήνας, οι βοηθητικές εφαρμογές, τα διάφορα GUI (Graphical User Interfaces) και η μεγάλη ποικιλία προγραμμάτων συνθέτουν αυτό που ονομάζουμε "διανομή".

https://linuxnewbieguide.org/overview-of-chapters/chapter-3-choosing-a-linux-distribution/

4. Raspberry Pi

4.1 *Τα βασικά για το Raspberry Pi;*

Το Raspberry Pi είναι ένα ολοκληρωμένο υπολογιστικό σύστημα με μέγεθος πιστωτικής κάρτας. Η έκδοση που χρησιμοποιήσαμε για να διεξαγάγουμε όλα τα πειράματά μας ήταν η τελευταία: Raspberry Pi 3 Model B. Παρά το μικρό όγκο του, το Raspberry Pi διαθέτει επεξεργαστή ARMv8 τετραπλού πυρήνα (64 bit) που χρονομετρείται στα 1200MHz, έναν πυρήνα γραφικών 3D VideoCore IV με θύρα Ethernet, διασύνδεση δικτύου 802.11n ασύρματου δικτύου LAN, συνδυασμένη υποδοχή ήχου 3.5 mm και σύνθετο βίντεο, διασύνδεση κάμερας (CSI), θύρα USB, , Bluetooth 4.1, υποδοχή κάρτας Micro SD (push-pull αντί push-push στην προηγούμενη έκδοση) και 40 ακροδέκτες γενικής χρήσης για σύνδεση με άλλα ηλεκτρονικά και περιφερειακά (pin GPIO). Το λειτουργικό σύστημα αποθηκεύεται στην κάρτα micro SD που πρέπει να έχει μέγεθος τουλάχιστον 4 GB και τουλάχιστον επίπεδο 4. Επιπλέον, θα χρειαστούμε επίσης έναν προσαρμογέα για να συνδέσετε την κάρτα micro SD στον υπολογιστή μας (για να εγκαταστήσετε το λειτουργικό σύστημα), ένα καλώδιο HDMI, ένα καλώδιο Ethernet, ένα ποντίκι, ένα πληκτρολόγιο και μια οθόνη. Μετά τη συγκέντρωση όλων αυτών των απαραίτητων στοιχείων, είμαστε έτοιμοι να προχωρήσουμε στην εγκατάσταση του λογισμικού.

4.2 *Οι Χρησιμότητες του Raspberry Pi στην καθημερινότητά μας*

Όλες οι βασικές λειτουργίες ενός σύγχρονου υπολογιστή μπορούν επίσης να βρεθούν στο Raspberry Pi, επιτρέποντας σε έναν άπειρο χρήστη να περιηγηθεί στο διαδίκτυο, να μεταδίδει το περιεχόμενό του και να κάνει τα πρώτα του βήματα στον προγραμματισμό. Παρακάτω σας παρουσιάζω ορισμένες βασικές χρήσεις αυτής της μικρής αλλά θαυματουργής συσκευής στον σημερινό κόσμο.

4.2.1 Media Center

Συνδέστε το Raspberry Pi σε μια τηλεόραση και δημιουργήστε εύκολα και αβίαστα το δικό σας κέντρο πολυμέσων! Με αυτόν τον τρόπο, θα μπορείτε να προβάλλετε τις αγαπημένες σας ταινίες και να διαχειρίζεστε το περιεχόμενο πολυμέσων σας με μερικά μόνο κλικ. Η διανομή OSMC Open Source

Linux είναι προσαρμοσμένη για αυτό το σκοπό και θα εντυπωσιαστείτε από την ποιότητα του περιεχομένου.

4.2.2 Προσωπικός υπολογιστής (PC)

https://www.theinquirer.net/inquirer/news/2449118/microsoft-quick-off-the-blocks-with-windows-10-for-the-raspberry-pi-3

Το Raspberry Pi μπορεί να μετατραπεί σε ένα σταθερό και φορητό περιβάλλον PC με περιορισμένες δυνατότητες αλλά επαρκή δύναμη για να καλύψει τις ανάγκες του μέσου, καθημερινού χρήστη. Απλά συνδέστε την οθόνη, το ποντίκι και το πληκτρολόγιο και είστε έτοιμοι να αρχίσετε να χρησιμοποιείτε το νέο προσωπικό σας υπολογιστή ανεμπόδιστα. Θα μπορείτε ακόμη και να κάνετε λήψη και εγκατάσταση των Windows 10, νόμιμα και εντελώς δωρεάν.

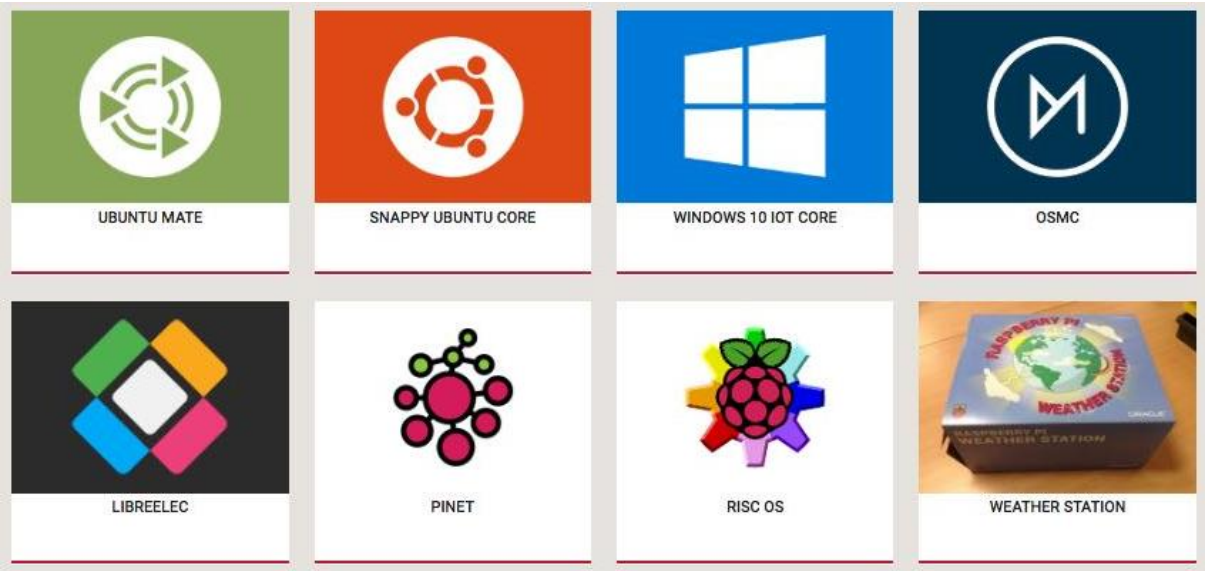
4.2.3 Μηχανή για Torrents

Εκείνοι που συμμετέχουν ενεργά στην torrent κοινότητα μπορούν να χρησιμοποιήσουν το Raspberry Pi ως αποκλειστικό μηχάνημα για τον διαμοιρασμό των torrent. Καταναλώνει πολύ λίγη ενέργεια, ώστε να μπορείτε να λειτουργείτε 24 ώρες το 24ωρο και χωρίς να περιορίζει τους πόρους του υπολογιστή σας.

4.2.4 Προγραμματισμός

Το Raspberry Pi είναι ο ιδανικός τρόπος για να κάνετε τα πρώτα σας βήματα στον προγραμματισμό. Υπάρχουν πολλές γλώσσες για να διαλέξετε. Αρχικά μπορείτε να χειριστείτε κάτι απλοϊκό και στο μέλλον μπορείτε να προχωρήσετε με εξελιγμένες δημιουργίες όπως η ρομποτική

4.2.5 Λειτουργικά συστήματα



https://raspberry-valley.azurewebsites.net/OS-for-Raspberry-Pi/

Με το Raspberry Pi, θα μπορείτε εύκολα να ανακαλύψετε, να εγκαταστήσετε και να χρησιμοποιήσετε τα διάφορα λειτουργικά συστήματα στην αγορά σήμερα. Linux, Windows 10, Firefox OS και πολλά άλλα είναι διαθέσιμα για να επιλέξετε!

4.2.6 Παιχνίδια

Όταν πρόκειται για απλά παιχνίδια που δεν απαιτούν προηγμένο υλικό, το Raspberry Pi είναι η καλύτερη επιλογή. Μπορείτε να παίξετε όλα τα ρετρό παιχνίδια όπως το Super Mario χωρίς καθυστέρηση ή διακοπές. Μπορείτε ακόμη να δημιουργήσετε το δικό σας Minecraft Server και να αλληλεπιδράσετε με άλλους παίκτες στον δικό σας εικονικό κόσμο.

5. Τι θα χρειαστούμε

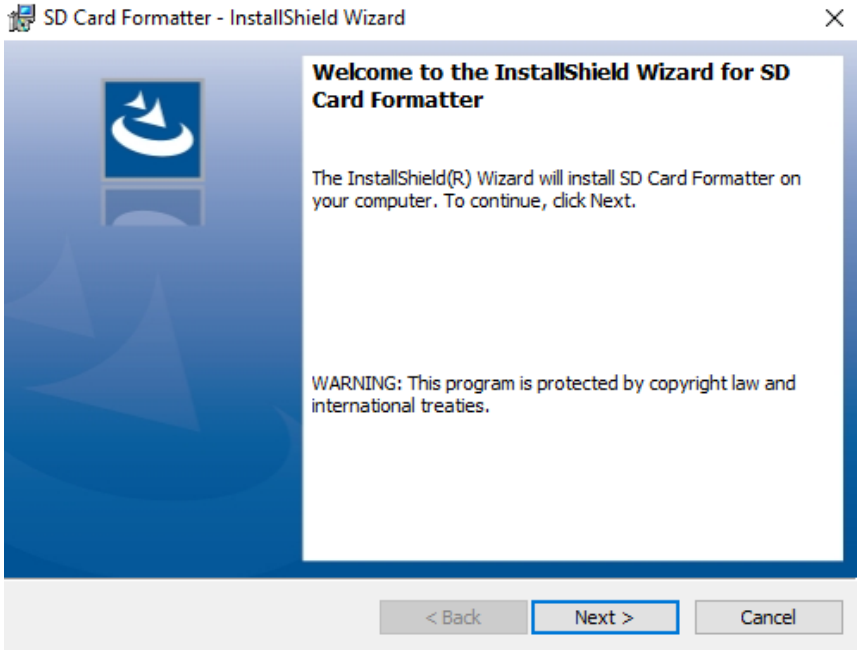
5.1 *Raspbian - Noobs (Το λειτουργικό σύστημα)*

Η διανομή Linux που επιλέξαμε να εγκαταστήσουμε στο Raspberry Pi ονομάζεται "Noobs - Raspbian" και είναι ένα λειτουργικό σύστημα βασισμένο στο Debian, το οποίο περιέχει άφθονο λογισμικό προεγκατεστημένο για προγραμματισμό, εκπαίδευση και γενική χρήση. Μπορούμε να βρούμε και να κατεβάσουμε αυτό το λογισμικό πατώντας εδώ.

Το πρώτο πράγμα που πρέπει να κάνουμε είναι να συνδέσουμε την κάρτα Micro SD με τον προσαρμογέα στον υπολογιστή μας για να το διαμορφώσουμε σωστά. Το "Noobs" συνιστά να χρησιμοποιήσετε την επίσημη αίτηση που θα βρείτε εδώ:

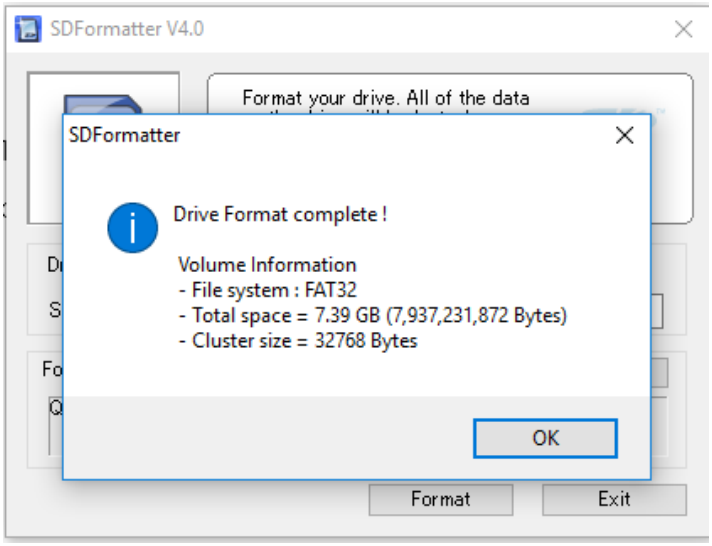
Εγκαθιστούμε την εφαρμογή κάνοντας κλικ στο κουμπί "**Accept**" στο κάτω μέρος της σελίδας, όπως φαίνεται στην παραπάνω εικόνα.

Στη συνέχεια, εκτελούμε το αρχείο "**SD_CardFormatter.exe**", εγκαθιστούμε και τελικά τρέχουμε την εφαρμογή - η εγκατάσταση είναι απλή και δεν θα επιχειρήσει να εγκαταστήσει προγράμματα τρίτων.

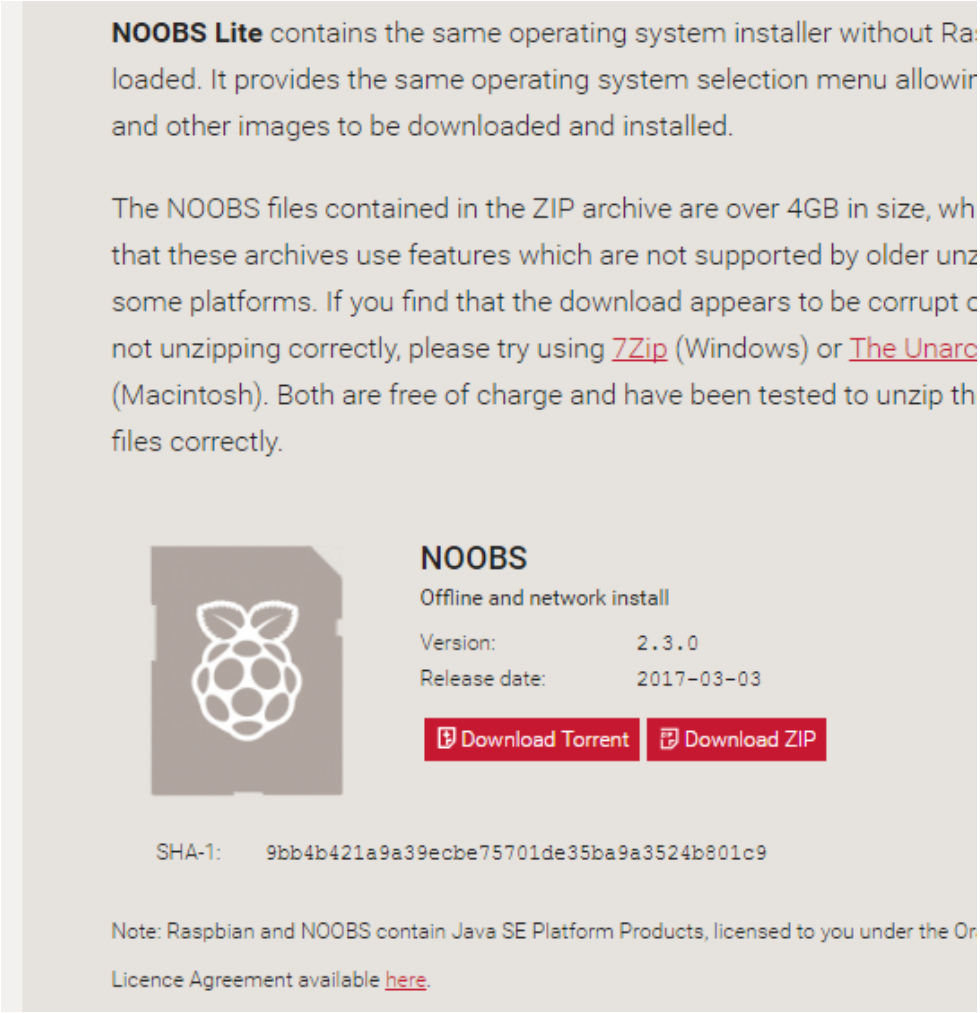


Αφού ξεκινήσουμε το πρόγραμμα, κάνουμε κλικ στο κουμπί "**Option**" Στο παράθυρο της εφαρμογής και αλλάζουμε τη μορφή προσαρμογής μεγέθους σε "**ON**".

Στη συνέχεια, κάνουμε κλικ στο κουμπί **Format** και σε λίγα δευτερόλεπτα η διαδικασία διαμόρφωσης θα ολοκληρωθεί.



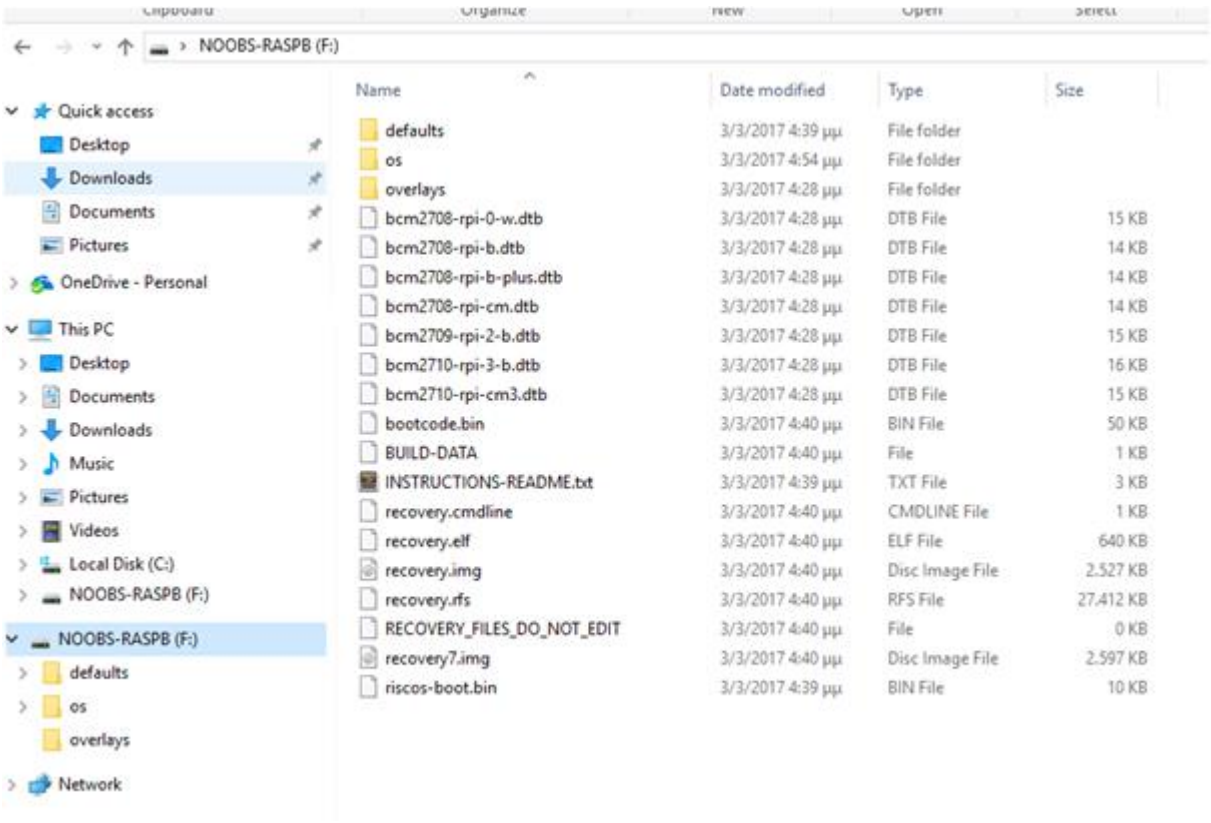
Μετά την ολοκλήρωση της διαμόρφωσης της κάρτας SD card μας, επισκεπτόμαστε την ακόλουθη διεύθυνση και κατεβάζουμε την τελευταία έκδοση του Noobs, είτε απευθείας από τον ιστότοπο με τη μορφή συμπιεσμένου αρχείου .zip είτε μέσω Torrent.



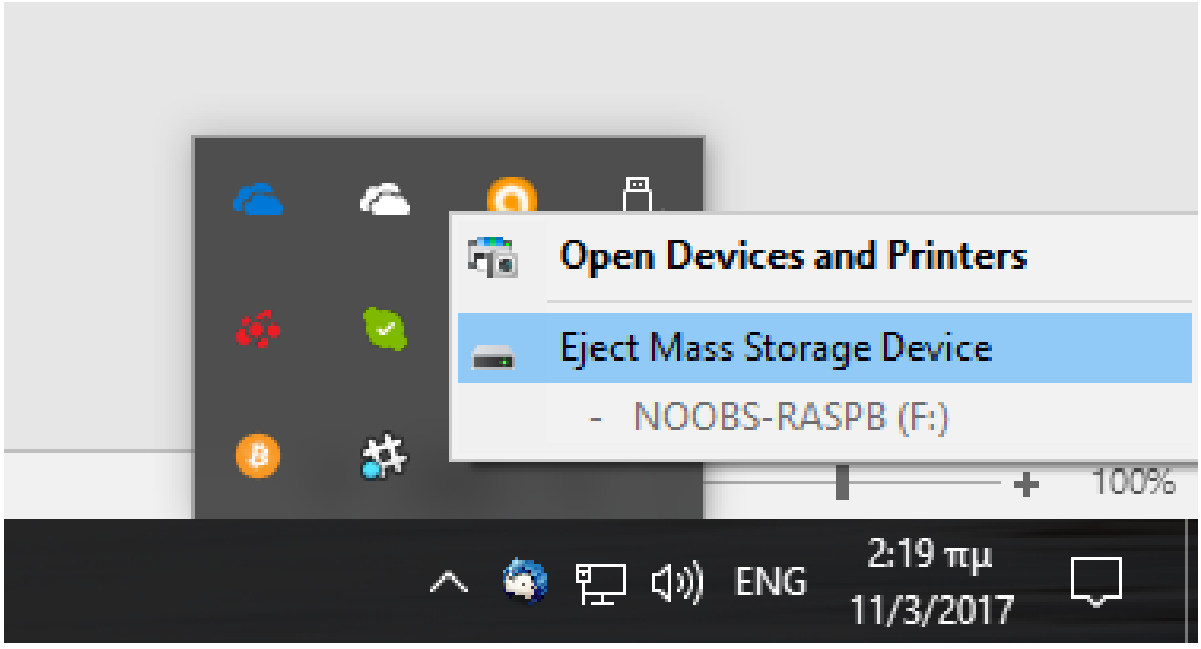
Ο Noobs είναι το πιο δημοφιλές λειτουργικό σύστημα για εγκατάσταση στο Raspberry Pi και προσφέρει πολλές δυνατότητες.

5.2 *Εγκατάσταση τουNoobs-Raspbian*

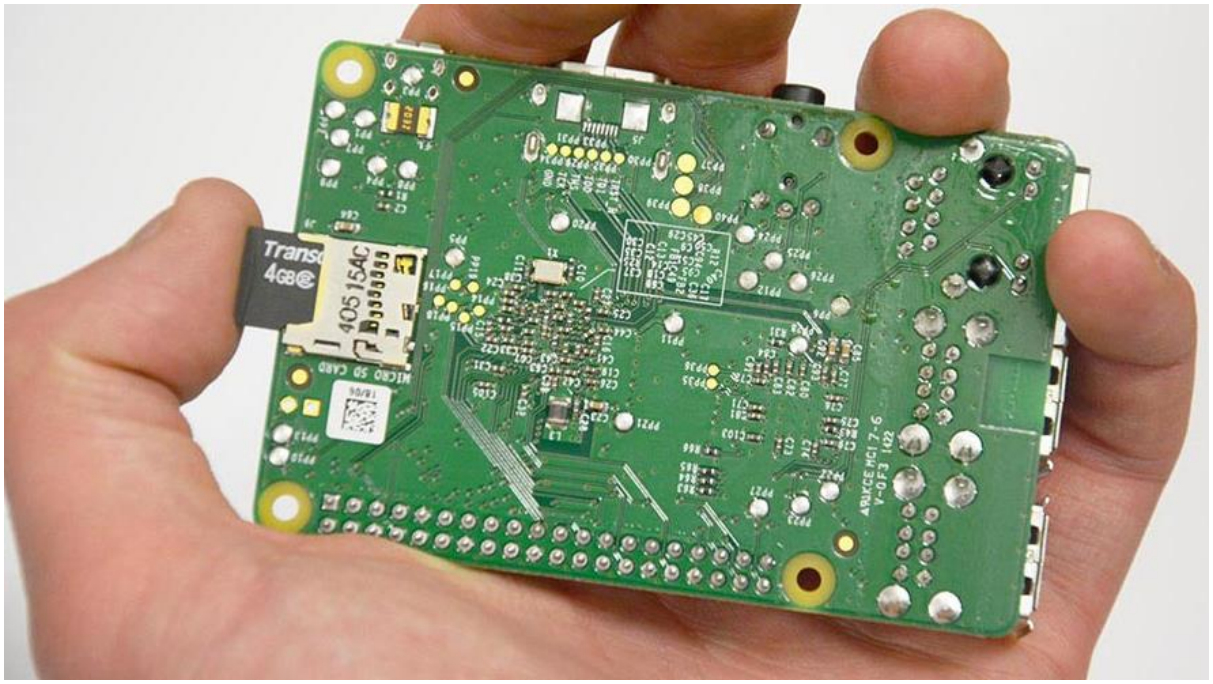
Μετά τη λήψη του συμπιεσμένου αρχείου .zip του Noobs, εξάγουμε όλο το περιεχόμενό του σε ένα νέο φάκελο. Στη συνέχεια περιηγούμαστε σε αυτόν τον φάκελο και αντιγράφουμε όλα του τα αρχεία στην κάρτα SD μας.



Αφού ολοκληρωθεί η διαδικασία αντιγραφής, αφαιρούμε με ασφάλεια την κάρτα SD, για να ελαχιστοποιήσουμε τη δυνατότητα καταστροφής δεδομένων.



Τώρα πρέπει να τοποθετήσουμε την κάρτα Micro SD μέσα στο Raspberry Pi. Γυρίστε το ανάποδα για να μπορέσετε να τοποθετήσετε το Micro SD στην κατάλληλη υποδοχή. Υπάρχει μόνο ένας τρόπος τοποθέτησης και αφού τοποθετηθεί σωστά, θα "κλειδώσει". Για να αφαιρέσετε την κάρτα SD, την πατάμε προς τα μέσα για να ξεκλειδώσετε.

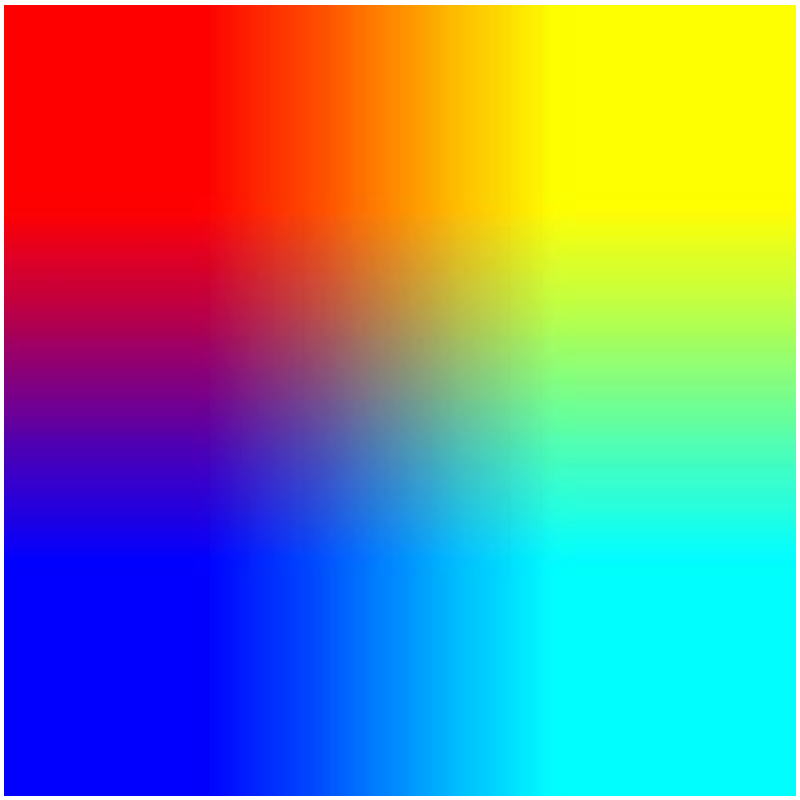


https://www.alphr.com/features/391627/how-to-set-up-a-raspberry-pi-b

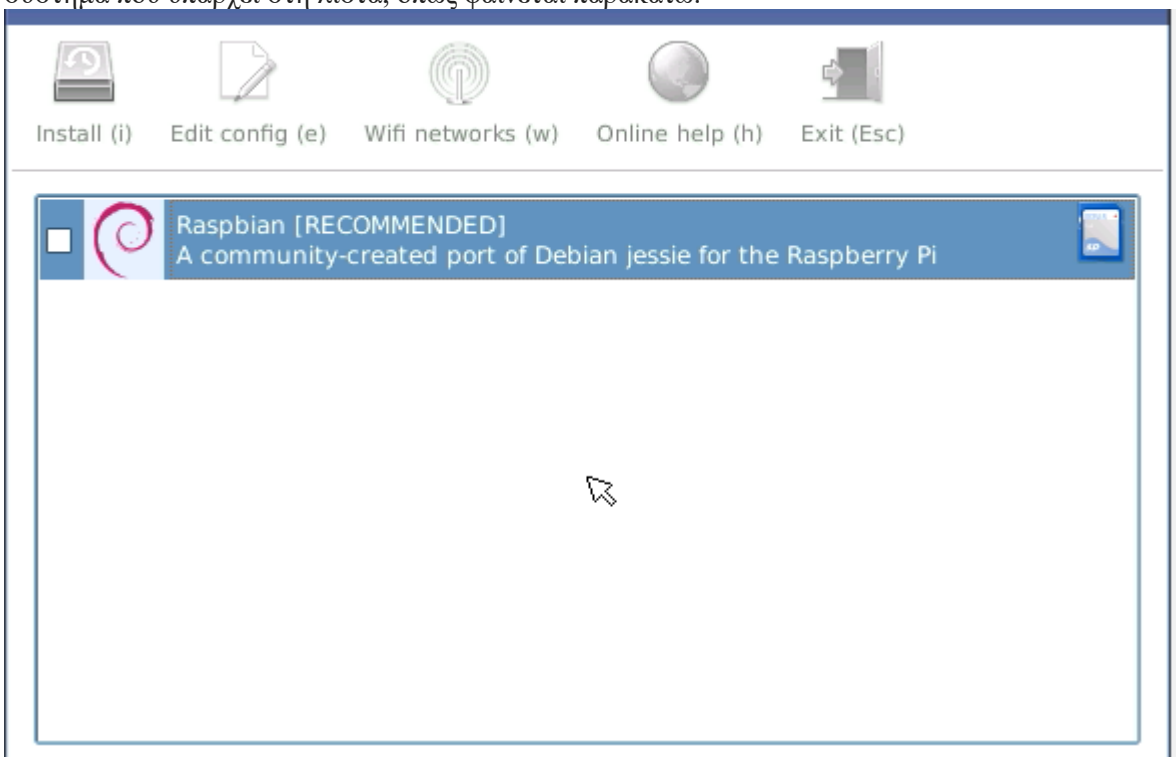
Μετά τη σύνδεση ενός ποντικιού και ενός πληκτρολογίου στο Raspberry Pi χρησιμοποιούμε ένα καλώδιο HDMI για τη σύνδεση με την οθόνη. Τώρα είμαστε έτοιμοι να βάλουμε το τροφοδοτικό. Όπως θα παρατηρήσετε, δεν υπάρχει διακόπτης On / Off στο Raspberry Pi. Πρέπει να το βάλουμε στην πρίζα και θα ξεκινήσει άμεσα.

5.3 *Η πρώτη εκκίνηση του Raspberry Pi*

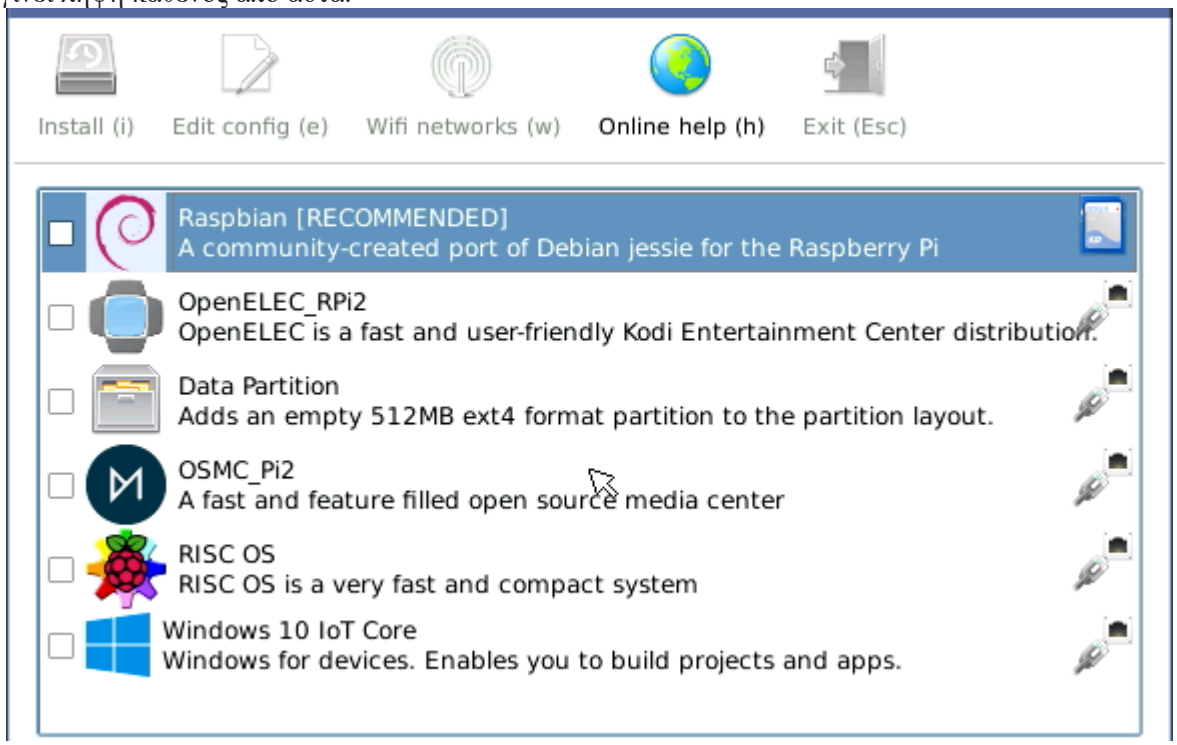
Όταν ξεκινά το Raspberry Pi, το πρώτο πράγμα που μπορούμε να δούμε είναι μια οθόνη με πολλά διαφορετικά χρώματα, γνωστή και ως οθόνη ουράνιο τόξο.



Εάν το Raspberry Pi δεν είναι συνδεδεμένο στο Internet, το Raspbian θα είναι το μόνο λειτουργικό σύστημα που υπάρχει στη λίστα, όπως φαίνεται παρακάτω.



Από την άλλη πλευρά, εάν το Raspberry Pi είναι συνδεδεμένο στο Διαδίκτυο, θα μπορέσουμε επίσης να δούμε εναλλακτικά λειτουργικά συστήματα για να διαλέξετε. Πρώτον, πρέπει να περιμένουμε να γίνει λήψη καθενός από αυτά.

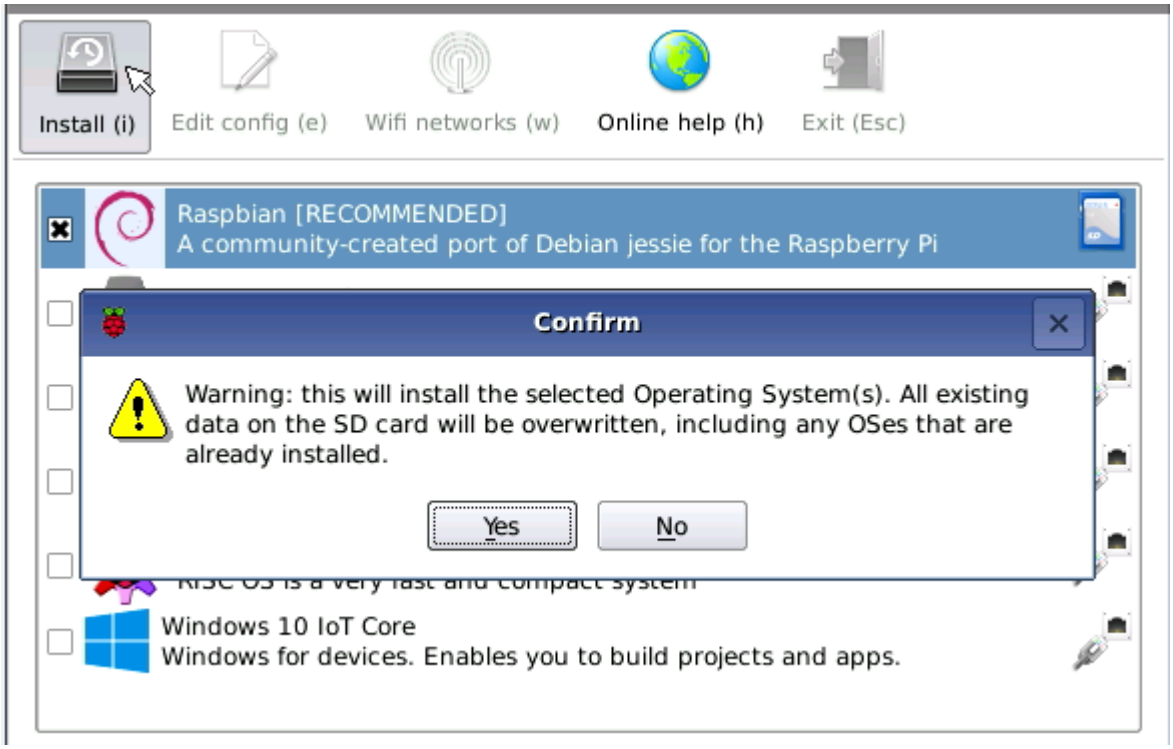


Αυτά τα λειτουργικά συστήματα περιλαμβάνουν:

1) Το Raspbian, ως παραλλαγή του Debian για το Raspberry Pi
2) Το OpenELEC, το οποίο είναι μια έκδοση του Kodi για το Raspberry Pi
3) Το OSMC που έχει την ίδια λειτουργικότητα με το Kodi αλλά με διαφορετική εμφάνιση
4) Το RISC OS, το οποίο είναι ένα πλήρες λειτουργικό σύστημα που δεν βασίζεται στο Linux
5) Ο πυρήνας 10 IoT των Windows

Μετά από αυτή τη σύντομη ανάλυση όλων των διαθέσιμων επιλογών στη λίστα, το πρώτο βήμα είναι να ορίσετε μια γλώσσα (π.χ. Αγγλικά ΗΠΑ) και μια διάταξη πληκτρολογίου στο κάτω μέρος της οθόνης.

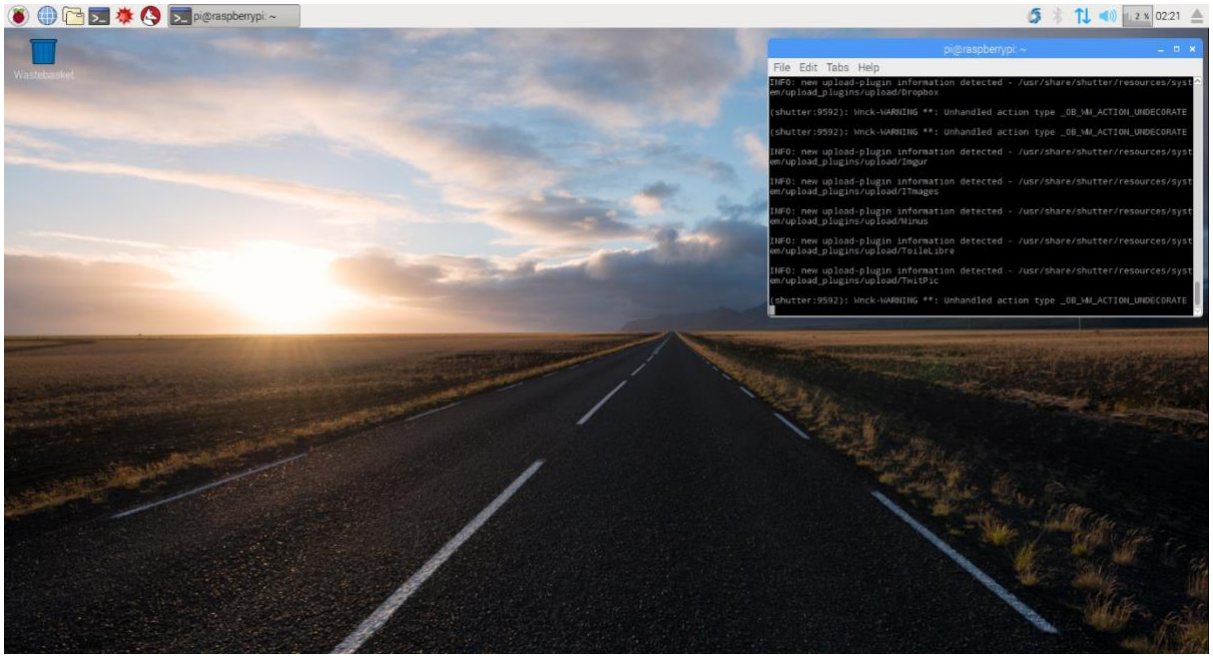
Ελέγχοντας την επιλογή Raspbian και κάνοντας κλικ στο Install, το σύστημα μας προειδοποιεί ότι θα διαγραφούν όλα τα περιεχόμενα της κάρτας SD.



Μόλις επιλέξετε την επιλογή "Ναι", η εγκατάσταση θα ξεκινήσει. Θα χρειαστεί λίγος χρόνος για να ολοκληρωθεί ανάλογα με την ταχύτητα της κάρτας SD. Μια αργή κάρτα Class 4 έχει ρυθμό μεταφοράς δεδομένων περίπου 1-1,5MB / s, ενώ μια γρήγορη κάρτα μπορεί να φτάσει έως και 5,8MB / s.

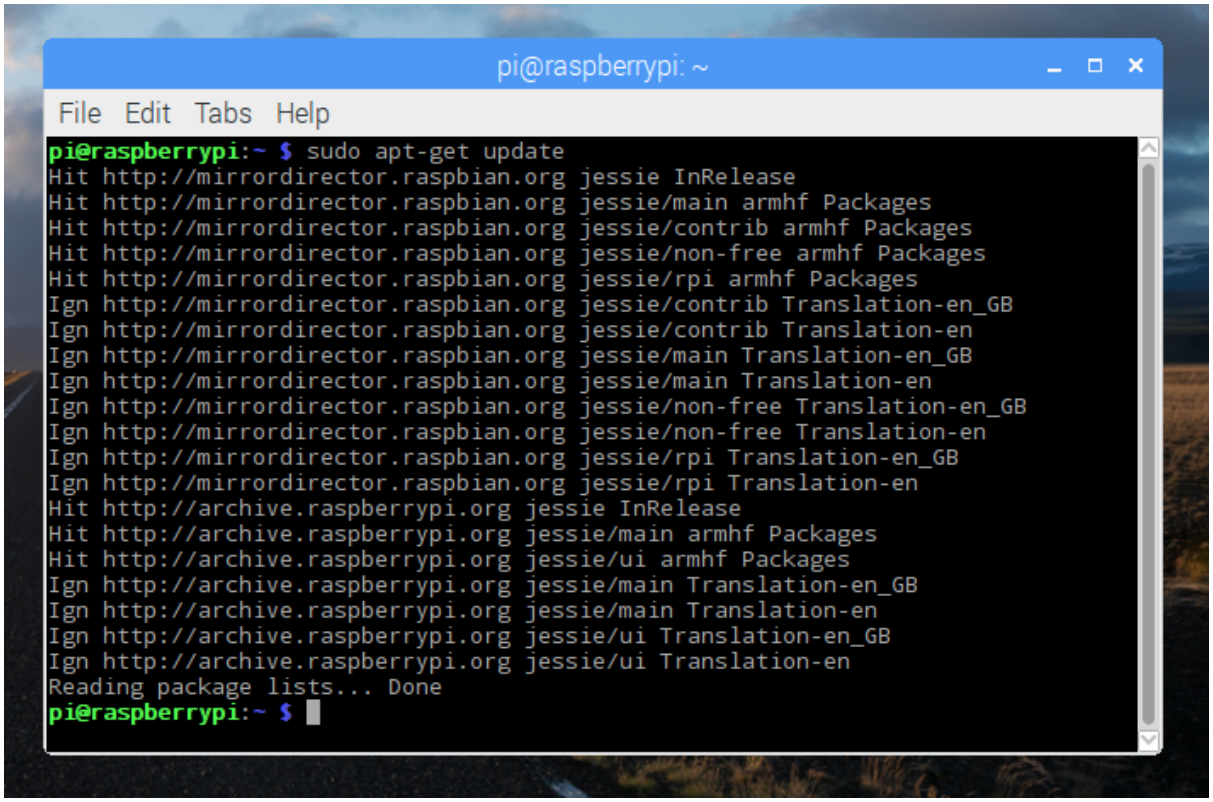
Στο τέλος, αν όλα πήγαν καλά, το σύστημα θα μας δείξει το μήνυμα ότι το λειτουργικό σύστημα (ή τα λειτουργικά συστήματα, εάν επιλέξατε πολλαπλά) εγκαθίστανται με επιτυχία. Στην περίπτωσή μας, εγκαταστήσαμε μόνο Raspbian στο σύστημά μας.

Αφού κάνετε κλικ στο κουμπί OK, η Raspberry Pi θα ξεκινήσει ξανά. Κατά την επόμενη εκκίνηση, θα συνδεθούμε στο Raspbian περιβάλλον.

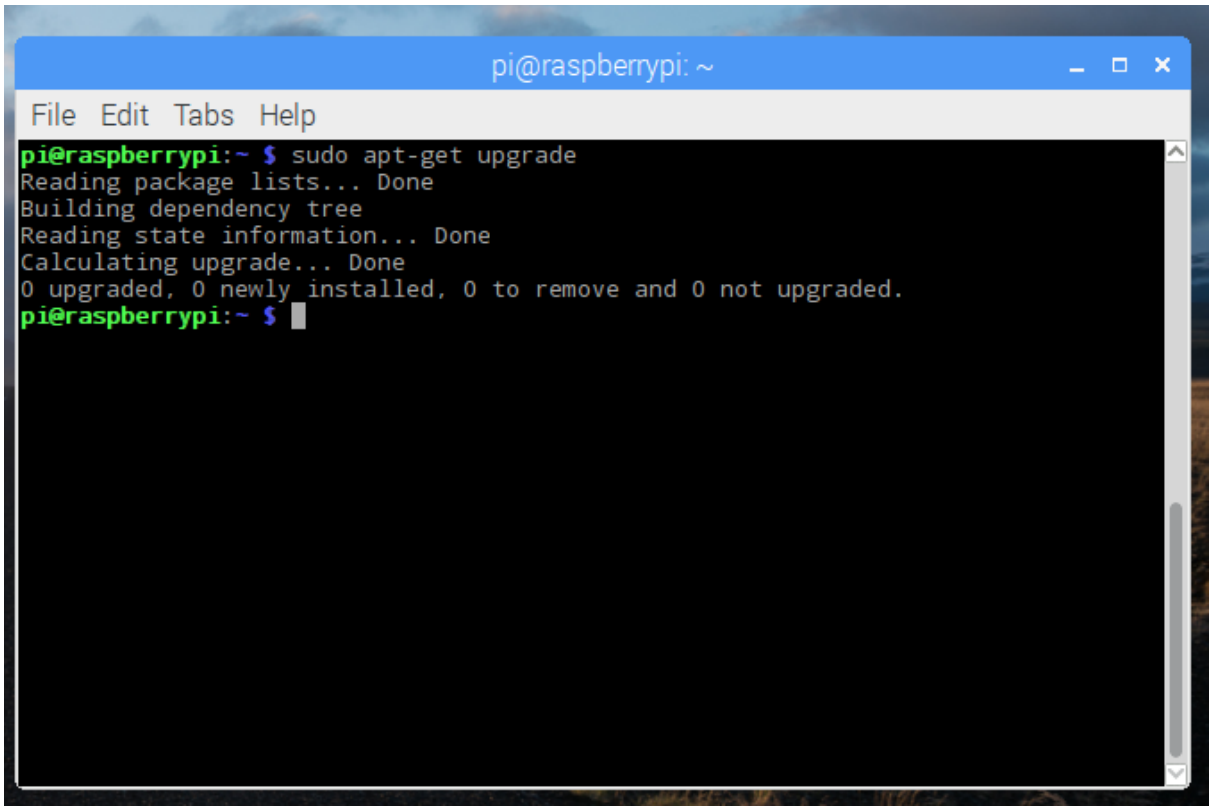


Έτσι, πρώτα θα εκτελέσουμε δύο βασικές εντολές Linux για να ενημερώσουμε πλήρως το περιβάλλον του Linux και να κατεβάσουμε την τελευταία έκδοση των πακέτων.

1) sudo apt-get update

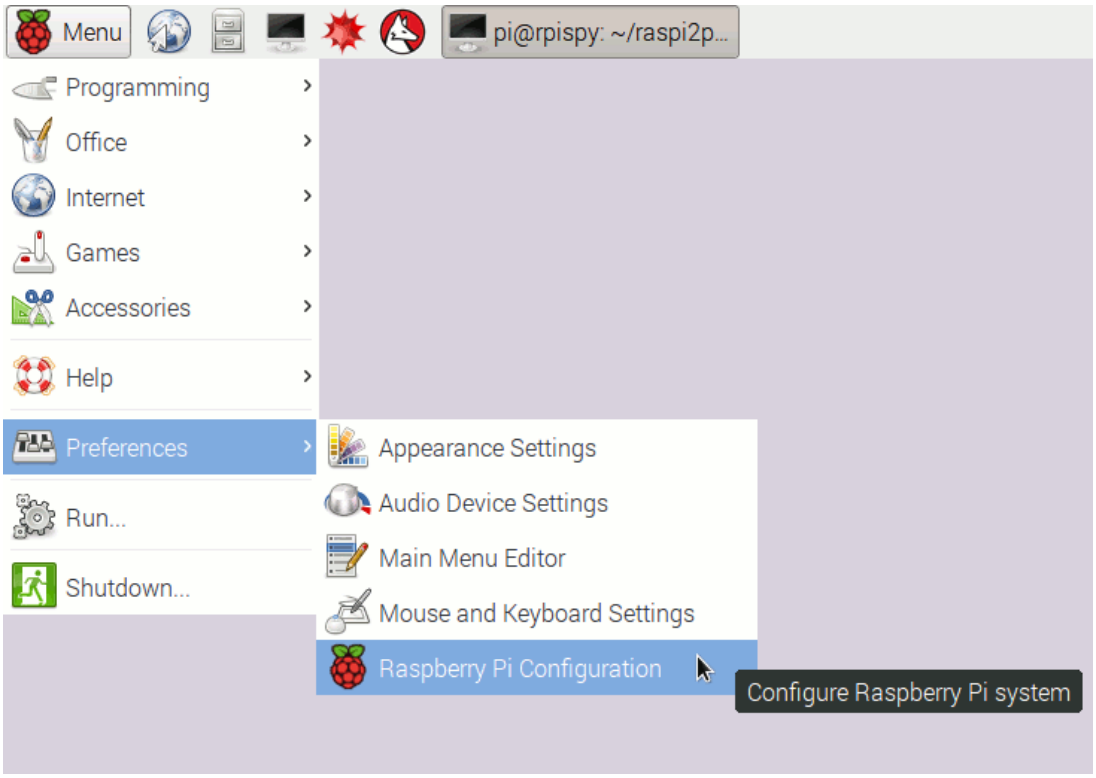


2) sudo apt-get upgrade

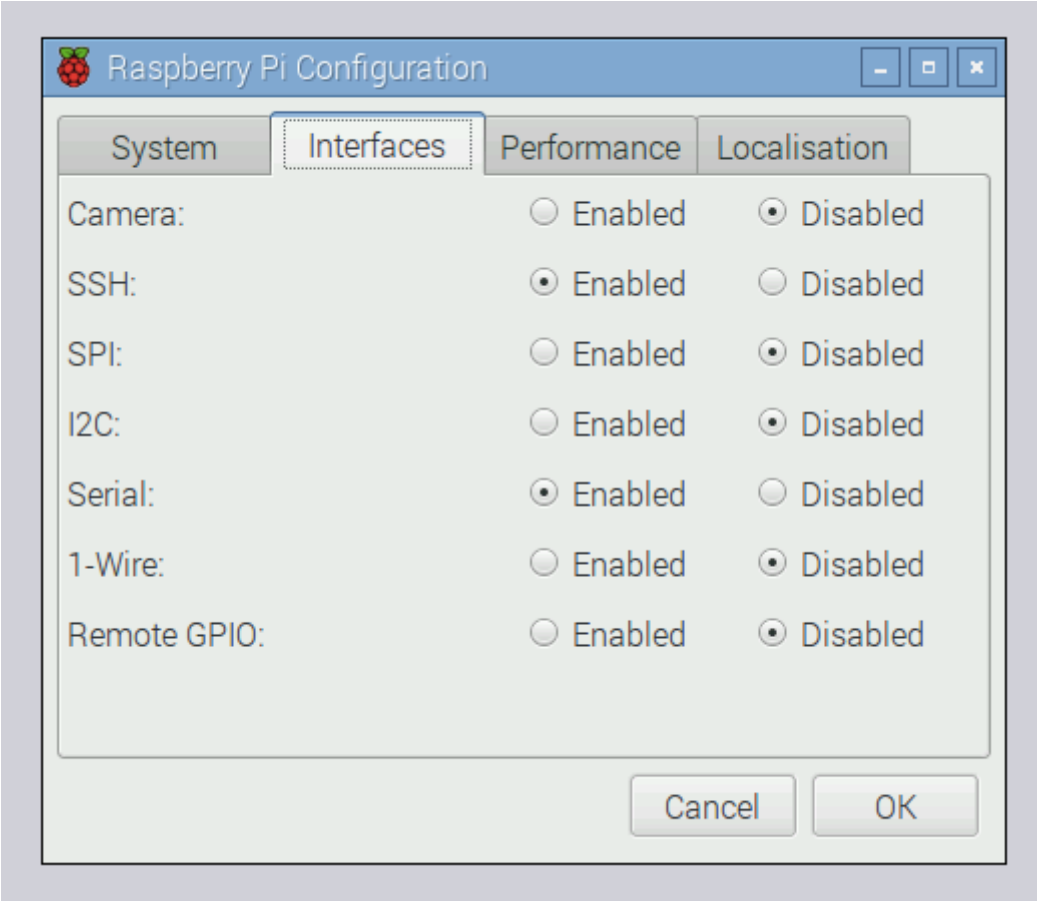


5.4 *Ενεργοποιούμε την πρόσβαση SSH στο Raspberry Pi*

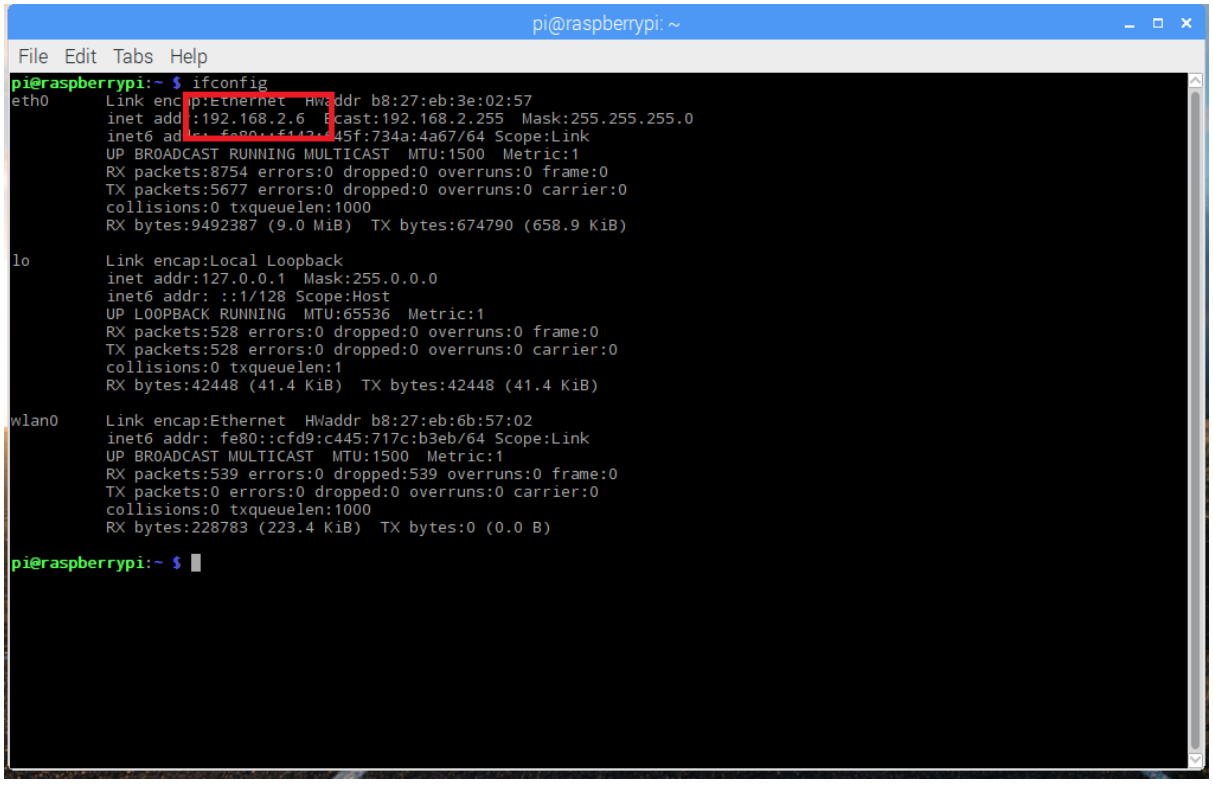
Μετά τη νέα εγκατάσταση του Raspbian παρατηρούμε ότι ο SSH Server είναι απενεργοποιημένος από προεπιλογή. Για να το ενεργοποιήσετε, θα επισκεφτούμε το **Main Menu → Preferences → Raspberry Pi Configuration**



Στη συνέχεια, κάντε κλικ στην καρτέλα "**Διεπαφές**" και επιλέγουμε "**Enabled**" στο πεδίο "**SSH**"



Τώρα είμαστε έτοιμοι να συνδεθούμε με το Raspberry Pi από άλλο μηχάνημα μέσω SSH. Πρώτον, σημειώνουμε την τρέχουσα διεύθυνση IP του Raspberry Pi πληκτρολογώντας **ifconfig**



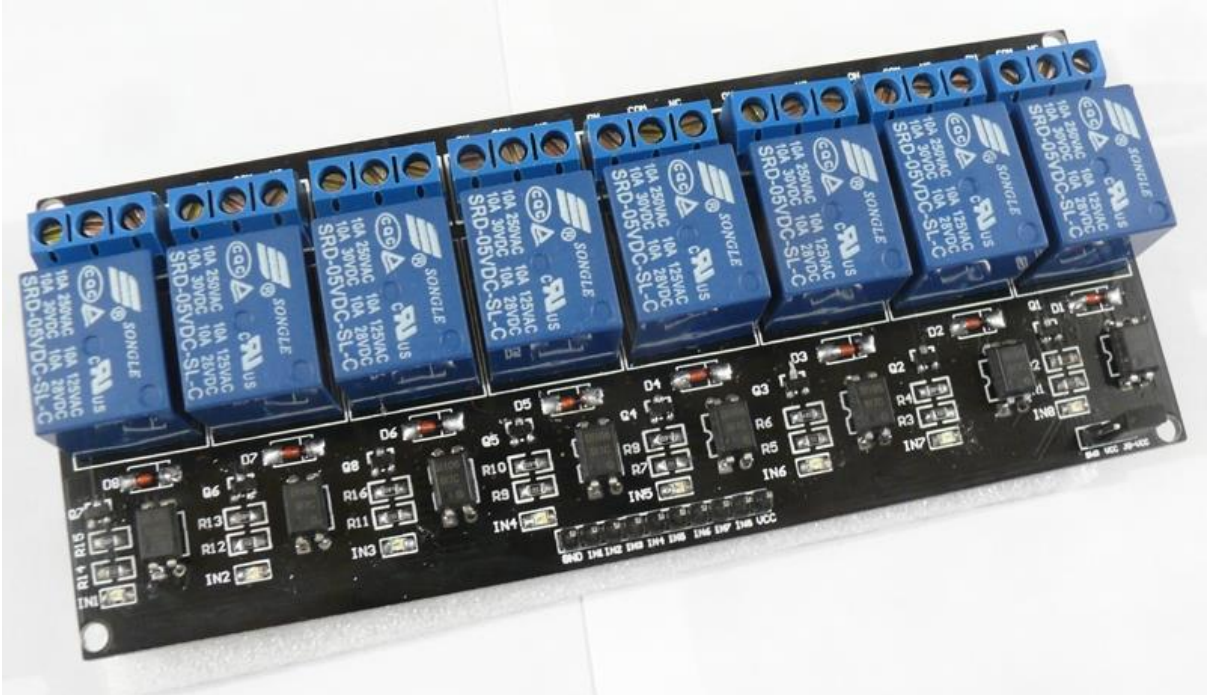
Και στη συνέχεια θα συνδεθούμε ως χρήστης **pi** χρησιμοποιώντας τον προεπιλεγμένο κωδικό πρόσβασης που είναι **raspberry**

```
pi@raspberrypi: ~
bill@DESKTOP-1FIK3R0:~$ ssh pi@192.168.2.6
The authenticity of host '192.168.2.6 (192.168.2.6)' can't be established.
ECDSA key fingerprint is 35:d1:1a:c1:a8:1a:c9:5c:c3:87:d6:64:a6:68:f3:d7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.6' (ECDSA) to the list of known hosts.
pi@192.168.2.6's password: raspberry

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Mar 23 20:15:34 2017

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

pi@raspberrypi:~ $
```

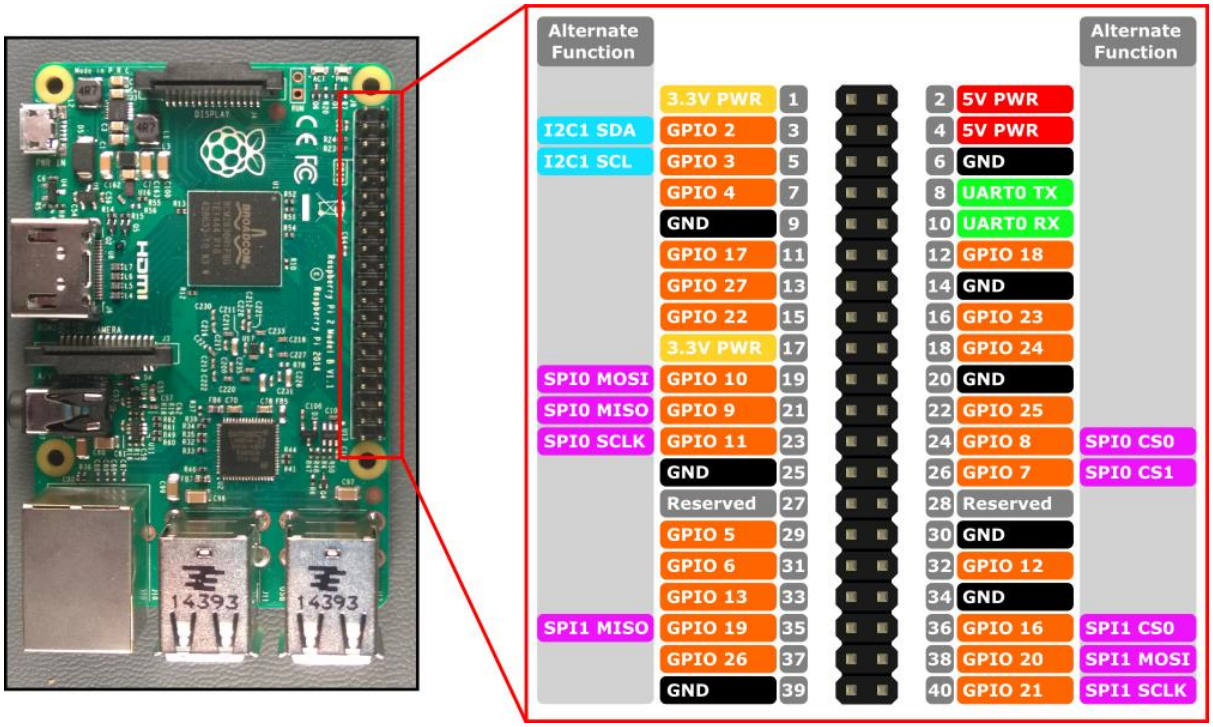
5.5 *Συνδέουμε ένα ρελέ με 8 κανάλια*

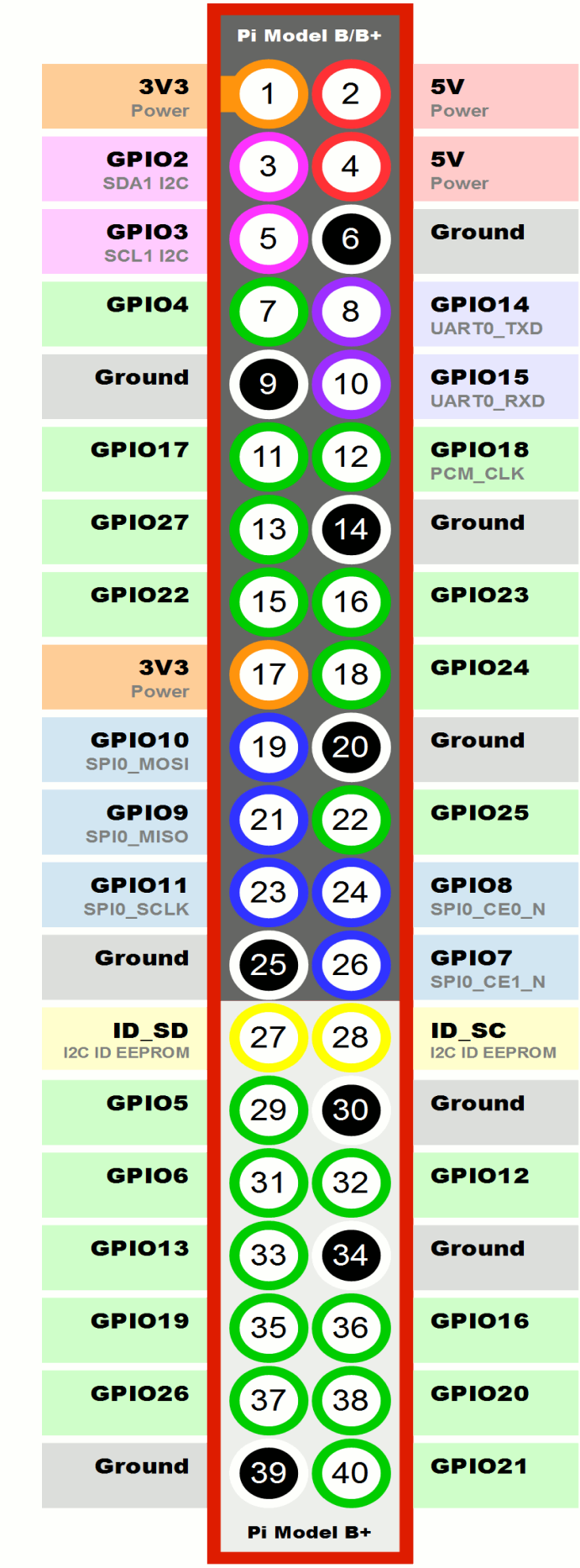
Ο λόγος που συνδέουμε ένα ρελέ 8 καναλιών στο Raspberry Pi μας είναι να μπορούμε να διαχειριστούμε οποιαδήποτε ηλεκτρική συσκευή ελέγχοντας την τρέχουσα ροή που διέρχεται από κάθε κανάλι του ρελέ. Για παράδειγμα, συνδέοντας μια λυχνία τύπου LED σε μια από τις υποδοχές του ρελέ, θα μπορέσουμε να καθορίσουμε το πότε θα είναι ενεργοποιημένη αυτή η λάμπα (δηλ. Αν θα επιτρέπουμε ηλεκτρική ενέργεια να περνά μέσα από αυτήν) ή πότε απενεργοποιημένη (δηλ. ο ηλεκτρισμός δεν περνά μέσα από αυτήν). Ο τρόπος σύνδεσης του ρελέ με το Raspberry Pi είναι με τη χρήση των ακροδεκτών (pins) GPIO (Γενικής χρήσης εισόδου / εξόδου).

Ένα ρελέ 8 καναλιών

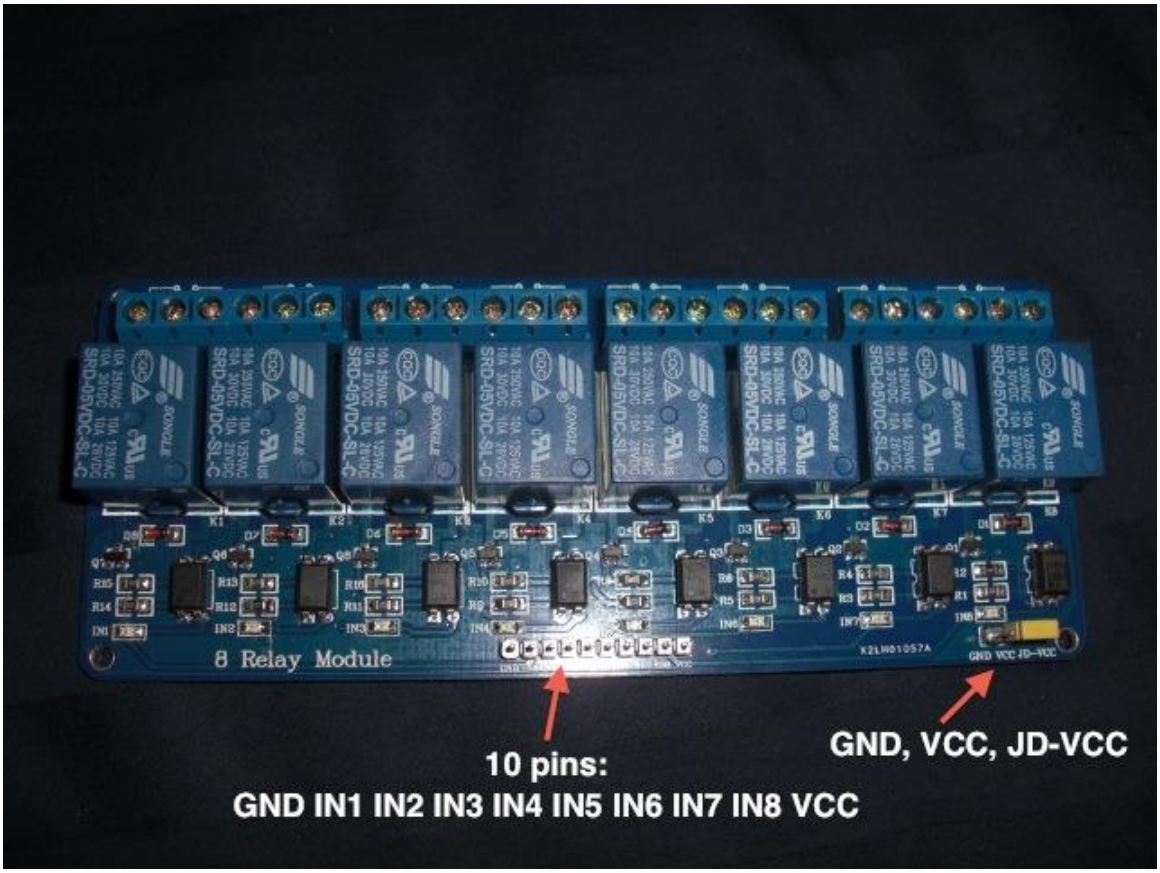


GPIO Pins στο Raspberry Pi:


Alternate Function
3.3V PWR 1
2 5V PWR
I2C1 SDA GPIO 2 3
4 5V PWR
I2C1 SCL GPIO 3 5
6 GND
GPIO 4 7
8 UART0 TX
GND 9
10 UART0 RX
GPIO 17 11
12 GPIO 18
GPIO 27 13
14 GND
GPIO 22 15
16 GPIO 23
3.3V PWR 17
18 GPIO 24
SPI0 MOSI GPIO 10 19
20 GND
SPI0 MISO GPIO 9 21
22 GPIO 25
SPI0 SCLK GPIO 11 23
24 GPIO 8 SPI0 CS0
GND 25
26 GPIO 7 SPI0 CS1
Reserved 27
28 Reserved
GPIO 5 29
30 GND
GPIO 6 31
32 GPIO 12
GPIO 13 33
34 GND
SPI1 MISO GPIO 19 35
36 GPIO 16 SPI1 CS0
GPIO 26 37
38 GPIO 20 SPI1 MOSI
GND 39
40 GPIO 21 SPI1 SCLK
Alternate Function



Οι ακροδέκτες των οκτώ καναλιών του ρελέ:

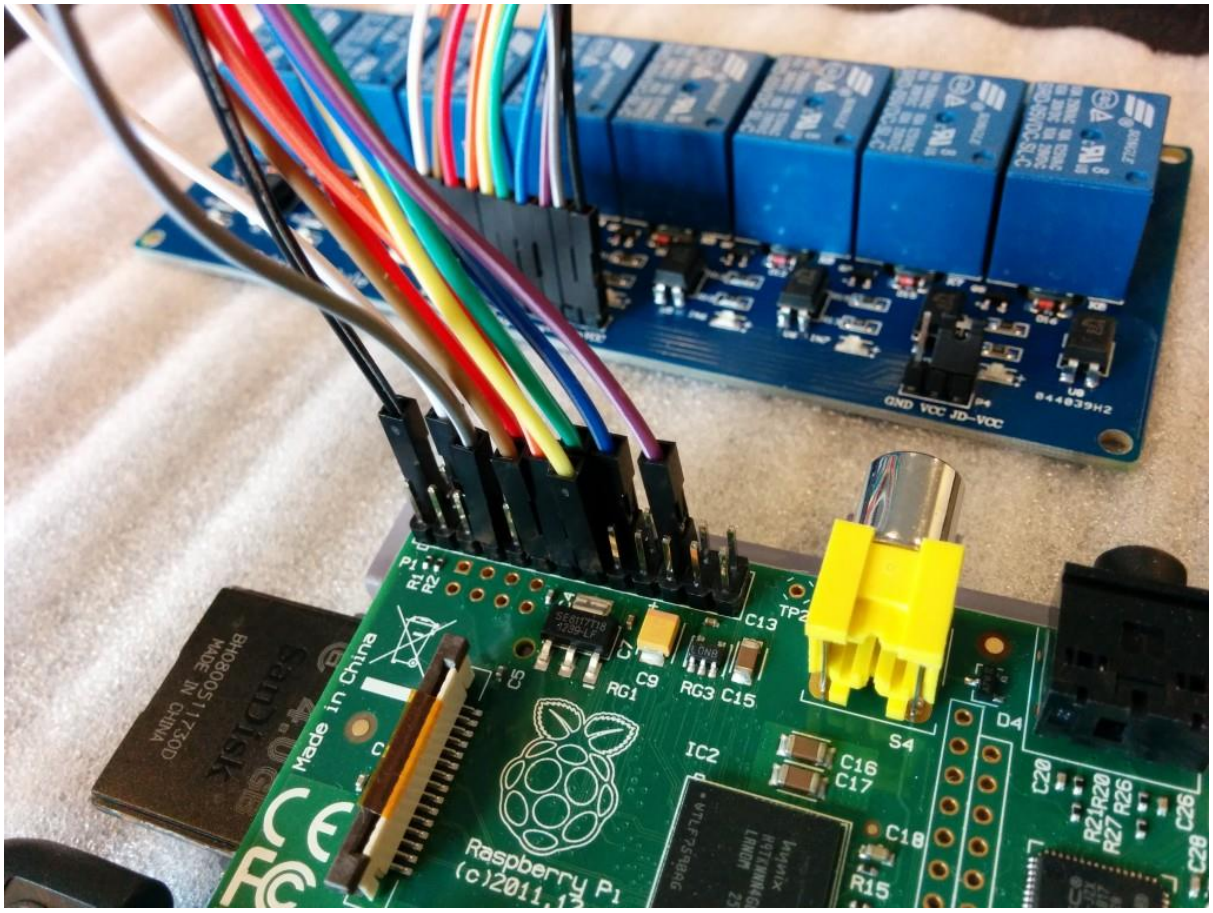


Επεξήγηση των ακροδεκτών (pins) του Ρελέ (από αριστερά προς τα δεξιά):

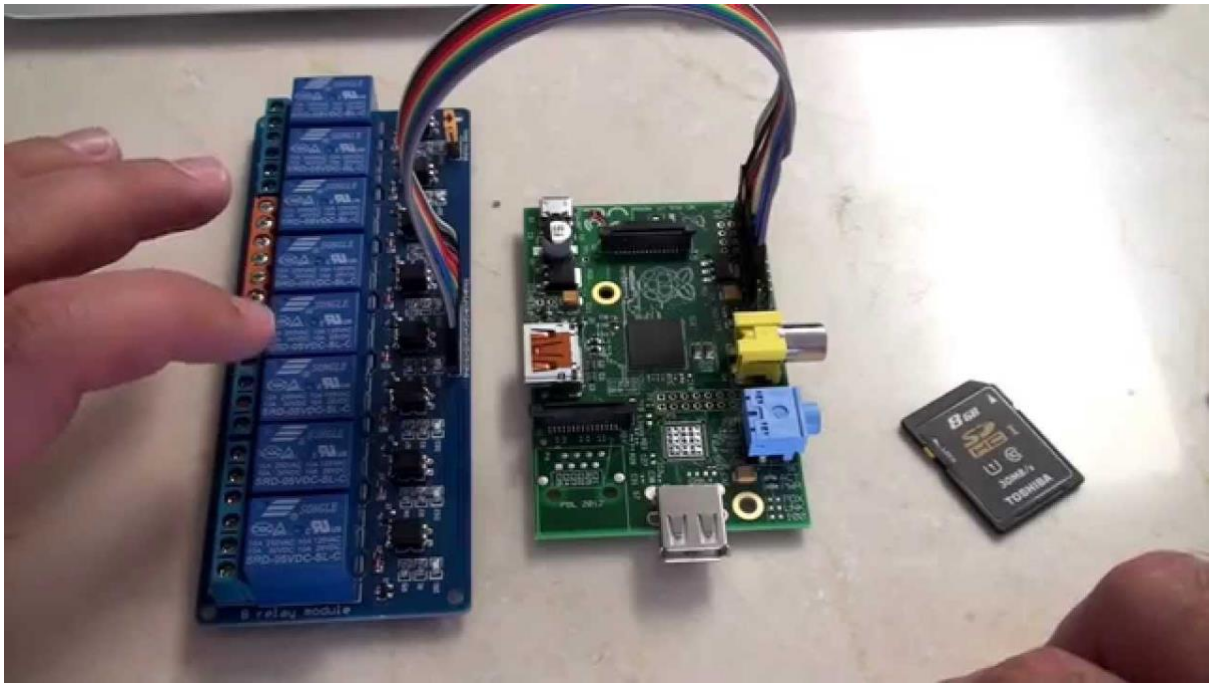
1ο ακροδέκτης (pin) → Γείωση (GND)
2ος – 9ος ακροδέκτης (pin) → IN1 - IN8 → Όλα τα διαθέσιμα κανάλια / εισόδους για τη σύνδεση οποιασδήποτε συσκευής
10ος ακροδέκτης → Τροφοδοσία ρεύματος (VCC)

Για να συνδεθείτε με επιτυχία το Relay με το Raspberry Pi, πρέπει να χρησιμοποιήσουμε μερικά καλώδια GPIO και να ενώσουμε:

1) Το VCC Pin στο ρελέ με το δεύτερο PIN στο Raspberry Pi (Ισχύς 5 Volt)
2) Το GND του ρελέ με τον ακροδέκτη γείωσης του Raspberry Pi
3) Όλες οι 8 είσοδοι του ρελέ (IN1-IN8) με 8 GPIO Raspberry Pi (μπορείτε να επιλέξετε όποιο από τα GPIO (pins) επιθυμείτε)

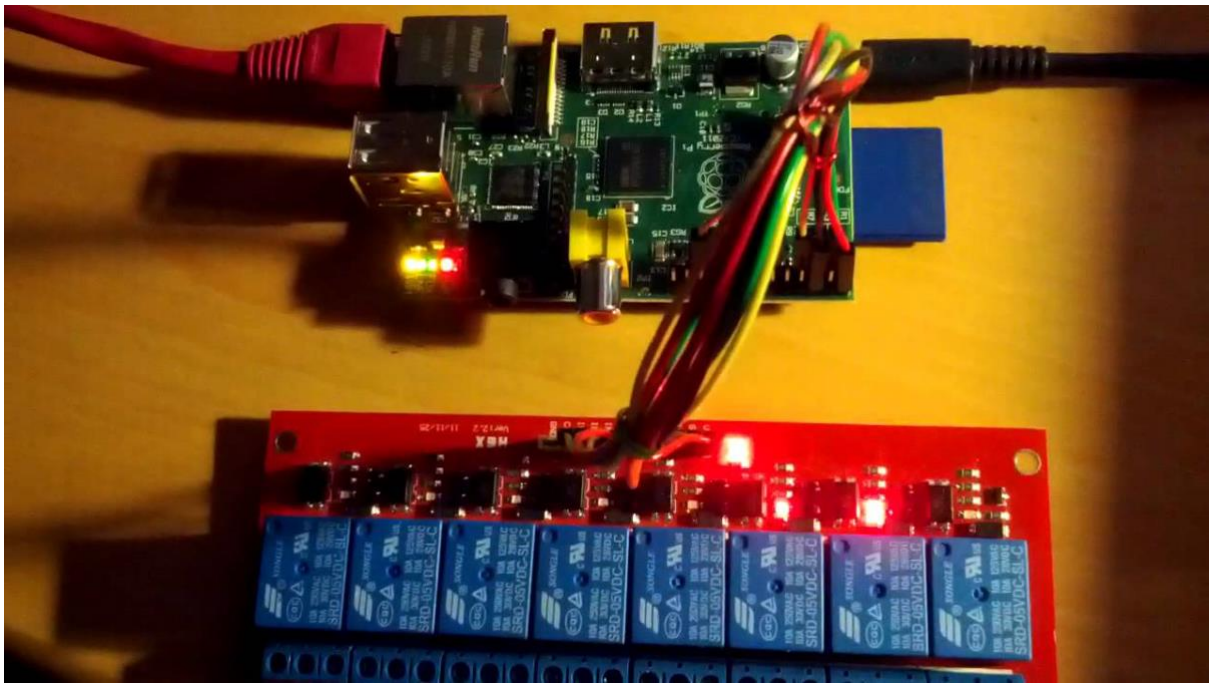


Εδώ φαίνεται το αποτέλεσμα



5.6 *Χρησιμοποιούμε την γλώσσα Python για να ελέγχουμε την παροχή ισχύος στο κάθε κανάλι του Relay*

Δημιουργώντας ένα script με την python, μπορούμε να καθορίσουμε την κατάσταση της ηλεκτρικής ενέργειας κάθε καναλιού στο ρελέ. Όταν το ρεύμα ρέει μέσω ενός καναλιού, ανάβει το LED φως αυτού του συγκεκριμένου καναλιού.



Στον παρακάτω python κώδικα που ακολουθεί δημιουργήσαμε μια λίστα που ονομάζεται "**pinList**", συμπεριλαμβανομένων όλων των αριθμών Pin GPIO που χρησιμοποιήθηκαν στο προηγούμενο βήμα. Στη συνέχεια, επιτρέπουμε στο ρεύμα να περνάει μέσα από κάθε PIN, ένα την φορά, με καθυστέρηση δύο δευτερολέπτων. Πάνω από κάθε εντολή, έχουμε συμπεριλάβει κάποιες παρατηρήσεις που εξηγούν τη λειτουργικότητά της

Σημείωση:

GPIO.LOW → Το κύκλωμα επιτρέπει τη διέλευση ηλεκτρικής ενέργειας

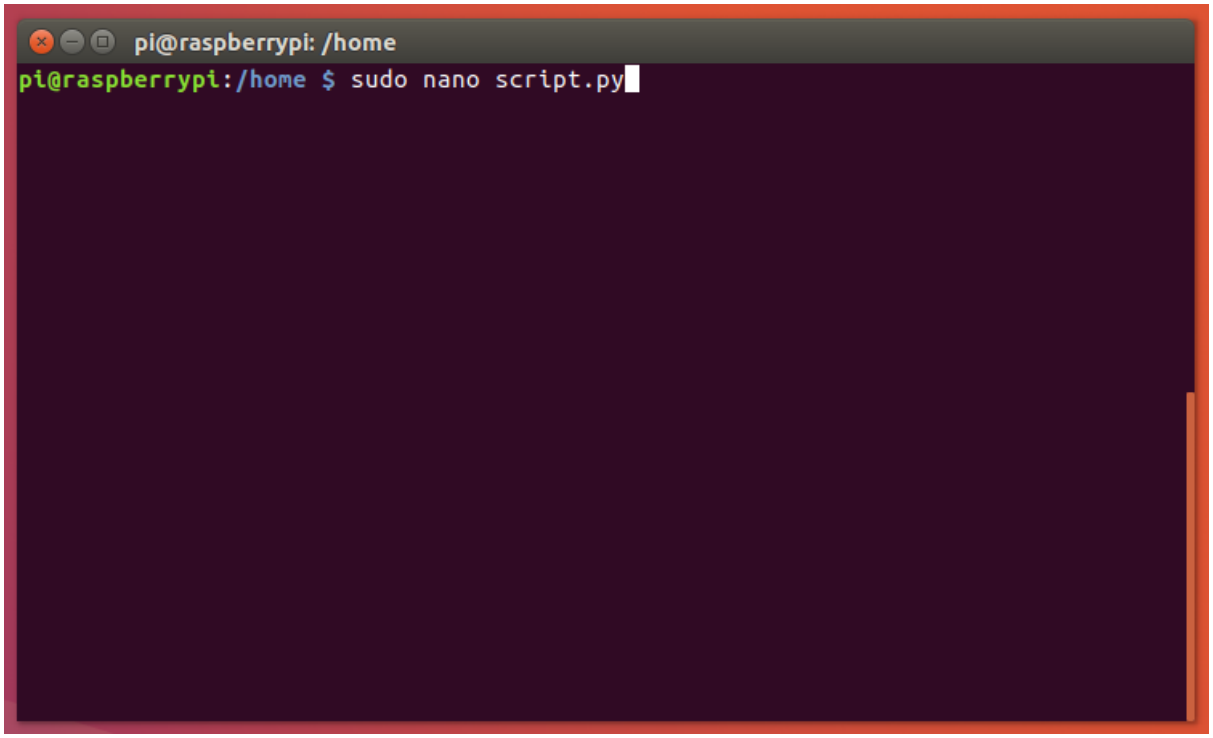
GPIO.HIGH → Το κύκλωμα δεν επιτρέπει τη διέλευση ηλεκτρικής ενέργειας

Κώδικας Python:

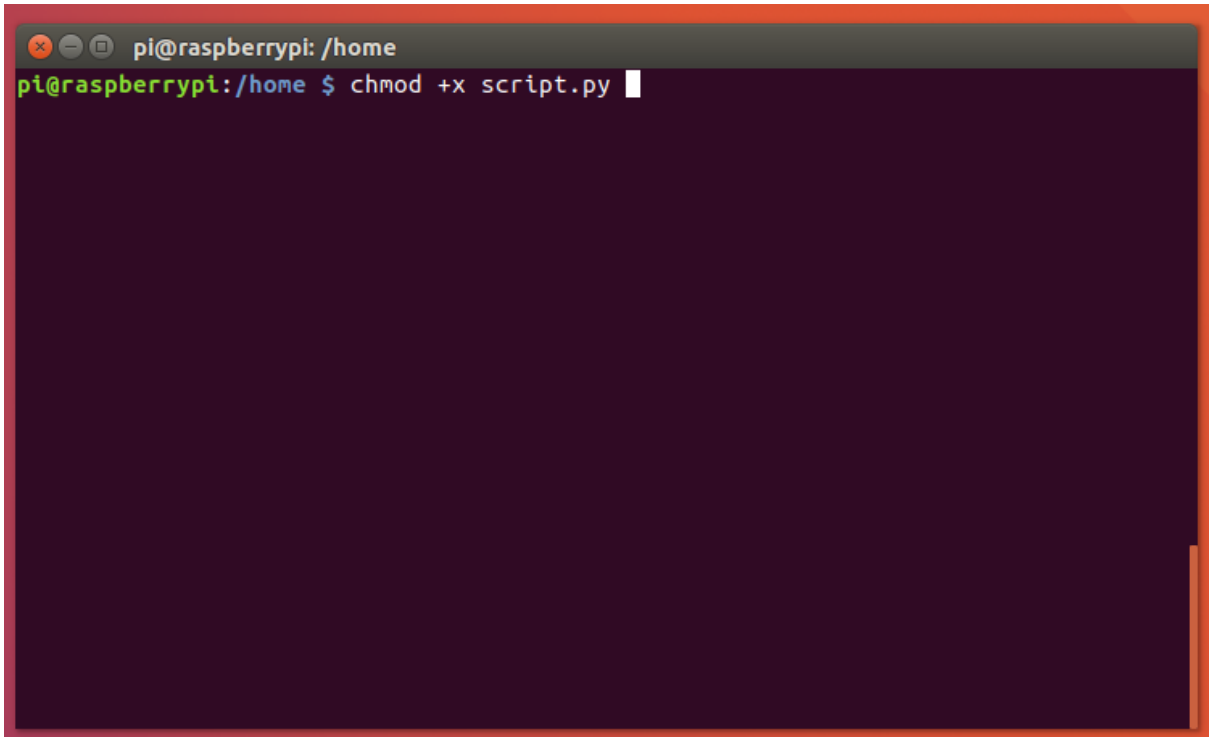
```
1 #!/usr/bin/python
 2 import RPi.GPIO as GPIO
 3 import time
 4
 5 GPIO.setmode(GPIO.BCM)
 6
 7 # init list with pin numbers
 8 pinList = [21, 26, 20, 19, 16, 13, 6, 12]
 9
```

```
10 # loop through pins and set mode and state to 'low'
11 for i in pinList:
12     GPIO.setup(i, GPIO.OUT)
13     GPIO.output(i, GPIO.HIGH)
14
15 # time to sleep between operations in the main loop
16 SleepTimeL = 0.2
17
18 # main loop
19 try:
20   GPIO.output(21, GPIO.LOW)
21   print "INPUT 1 - GPIO: 21 - LIGHT IS ON"
22   time.sleep(SleepTimeL);
23   GPIO.output(26, GPIO.LOW)
24   print "INPUT 2 - GPIO: 26 - LIGHT IS ON"
25   time.sleep(SleepTimeL);
26   GPIO.output(20, GPIO.LOW)
27   print "INPUT 3 - GPIO: 20 - LIGHT IS ON"
28   time.sleep(SleepTimeL);
29   GPIO.output(19, GPIO.LOW)
30   print "INPUT 4 - GPIO: 19 - LIGHT IS ON"
31   time.sleep(SleepTimeL);
32   GPIO.output(16, GPIO.LOW)
33   print "INPUT 5 - GPIO: 16 - LIGHT IS ON"
34   time.sleep(SleepTimeL);
35   GPIO.output(13, GPIO.LOW)
36   print "INPUT 6 - GPIO: 13 - LIGHT IS ON"
37   time.sleep(SleepTimeL);
38   GPIO.output(6, GPIO.LOW)
39   print "INPUT 7 - GPIO: 6 - LIGHT IS ON"
40   time.sleep(SleepTimeL);
41   GPIO.output(12, GPIO.LOW)
42   print "INPUT 8 - GPIO: 12 - LIGHT IS ON"
43   time.sleep(SleepTimeL);
44   print "Good bye - All LIGHTS ARE OFF"
45
46
47 # End program cleanly with keyboard
48 except KeyboardInterrupt:
49   print "  Quit"
50
51   # Reset GPIO settings
52   GPIO.cleanup()
```

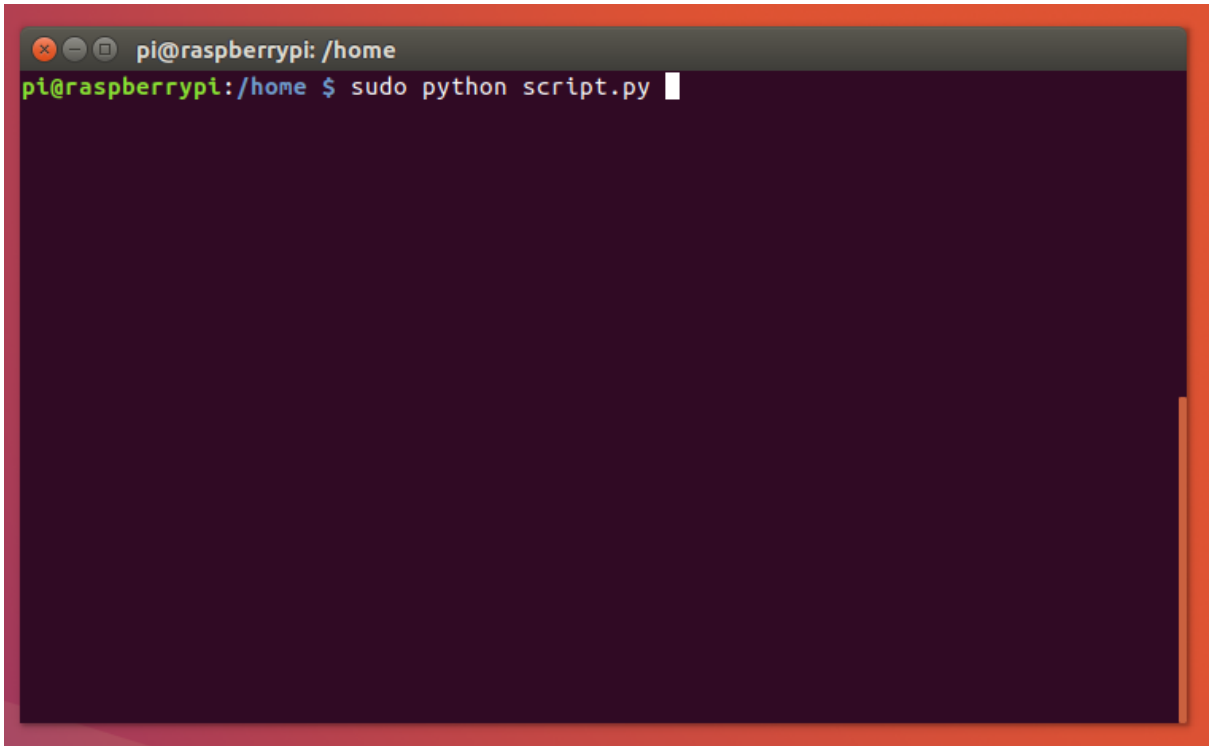
Για να εκτελέσουμε τον παραπάνω κώδικα, εισερχόμαστε ως χρήστης **pi** στο Raspberry Pi και δημιουργούμε ένα αρχείο στον αρχικό κατάλογο με την κατάληξη **.py** (για παράδειγμα **script.py**)



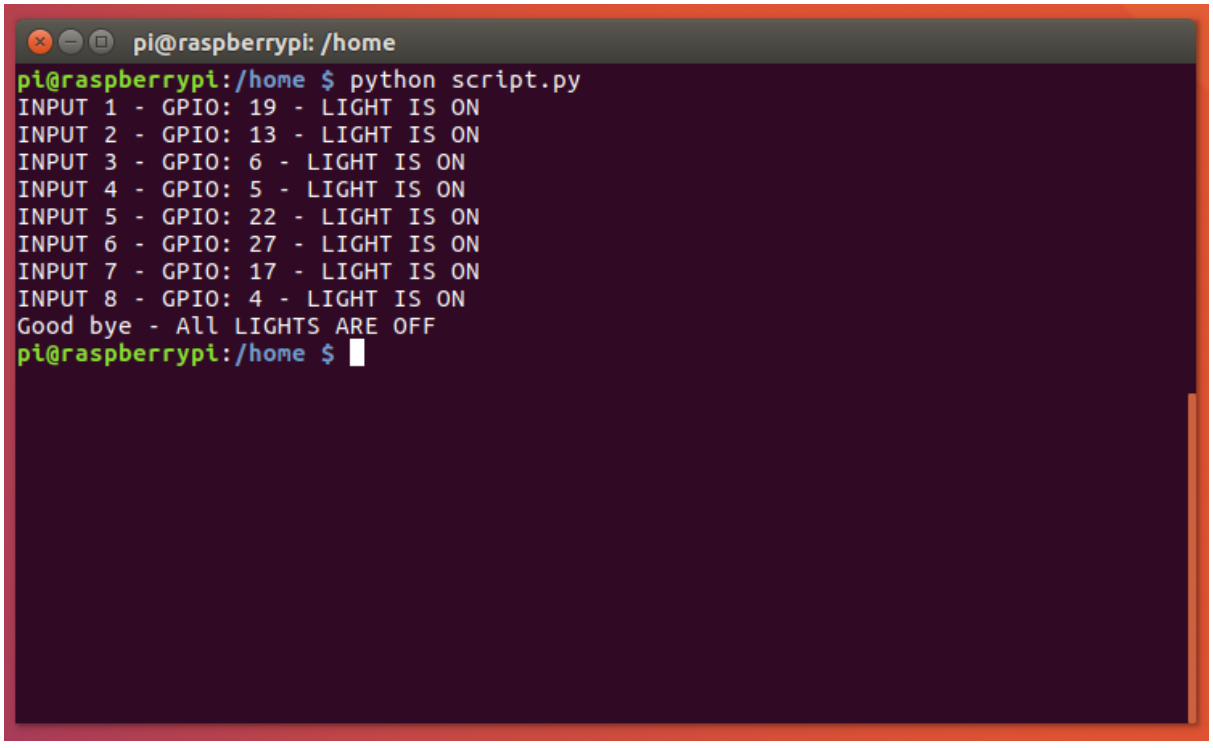
Στη συνέχεια, κάνουμε το αρχείο αυτό εκτελέσιμο πληκτρολογώντας:



Για να εκτελέσετε αυτό το αρχείο, δίνουμε την εντολή:



Σε αυτό το σημείο, θα πρέπει να δούμε την έξοδο που ακολουθεί στο τερματικό μας και θα πρέπει να παρατηρήσουμε ότι τα φώτα στο ρελέ ενεργοποιούνται το ένα μετά το άλλο με καθυστέρηση 0,2 δευτερολέπτων. Στο τέλος, όλα τα φώτα είναι απενεργοποιημένα. Στο επόμενο στιγμιότυπο οθόνης, μπορούμε να δούμε σε κάθε γραμμή τον αύξον αριθμό του εκάστοτε Pin εξόδου του ρελέ, τον αριθμό που έχει το εκάστοτε GPIO Pin και την κατάσταση της λυχνίας LED κάθε καναλιού.



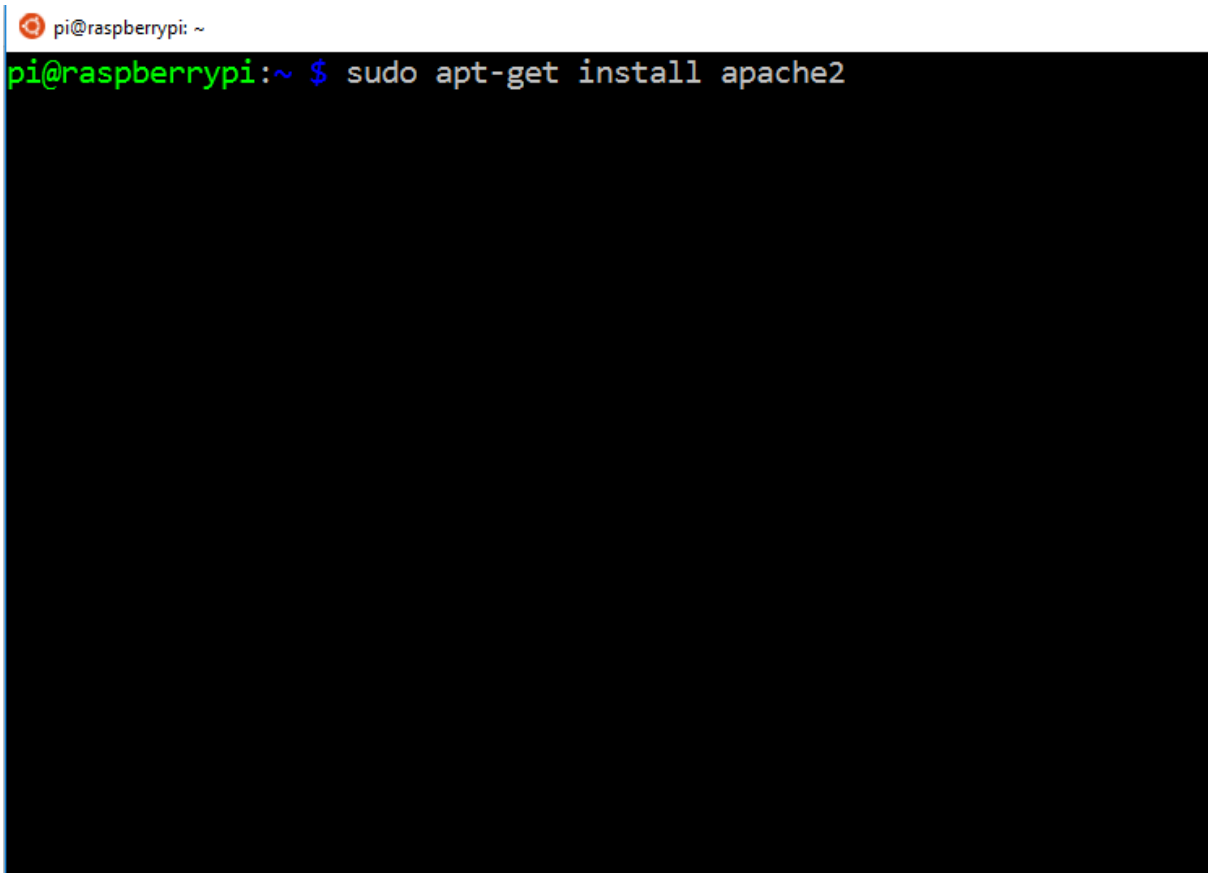
6. Εφαρμογή μέσω Web (χρησιμοποιώντας πακέτα TCP)

Ο κύριος στόχος μας είναι να δημιουργήσουμε μια διεπαφή ιστού γραμμένη σε HTML5, CSS, JavaScript και PHP μέσω της οποίας μπορούμε να επικοινωνούμε με το Raspberry Pi και να του δώσουμε διάφορες εντολές.

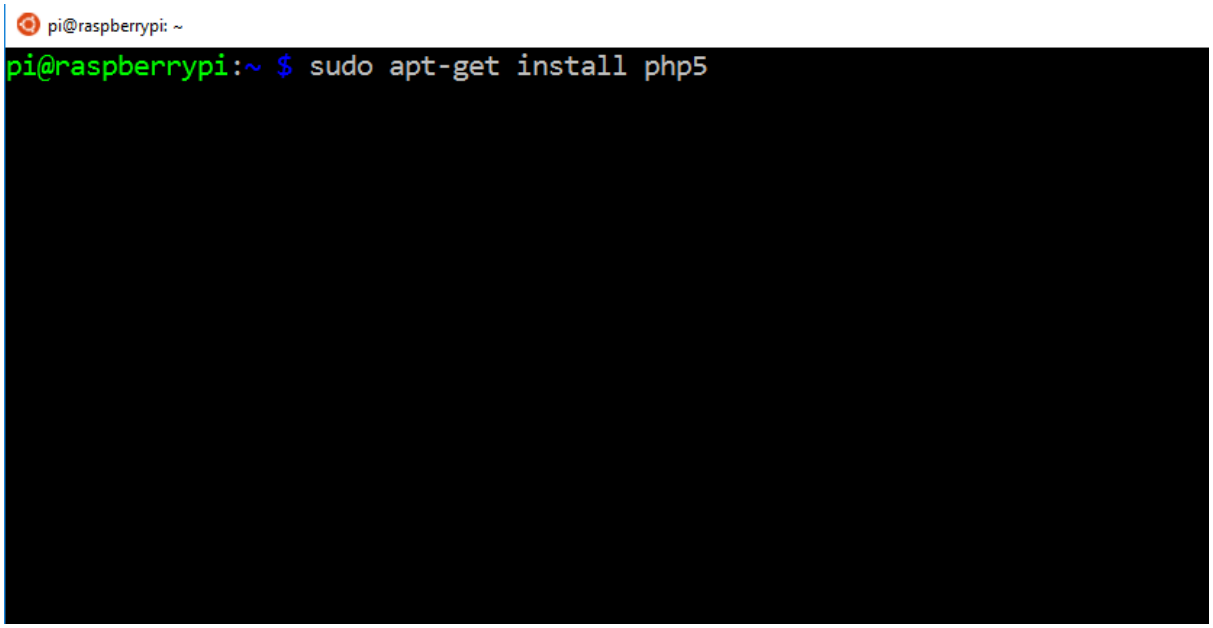
6.1 *Εγκατάσταση όλων των προαπαιτούμενων*

Για να μπορέσουμε να εκτελέσουμε την εφαρμογή μας, θα πρέπει να εγκαταστήσουμε τον Apache Web Server και την PHP. Για να γίνει αυτό, ανοίγουμε ένα τερματικό και δίνουμε τις ακόλουθες εντολές

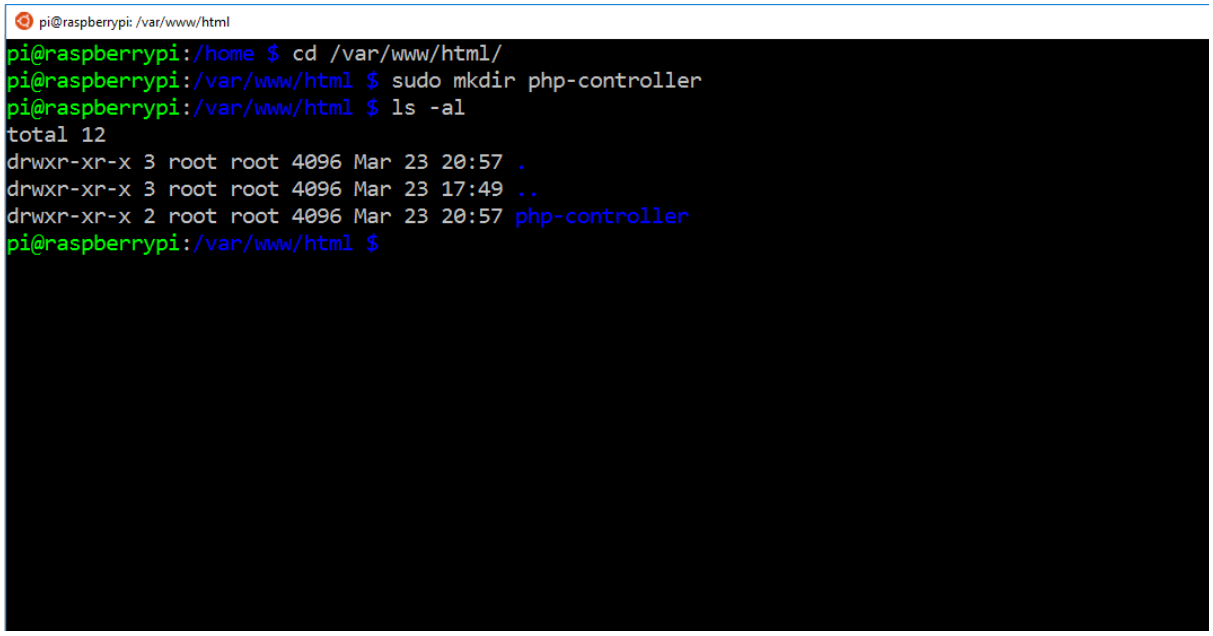
Αυτή η εντολή θα εγκαταστήσει το Apache2 Web Server



Αυτή η εντολή θα εγκαταστήσει την έκδοση 5 της PHP



Τώρα, μπορούμε να κατευθυνθούμε στον κατάλογο / var / www / html / όπου θα δημιουργήσουμε ένα νέο φάκελο που ονομάζεται **php-controller** ο οποίος θα περιέχει ολόκληρη την PHP εφαρμογή.

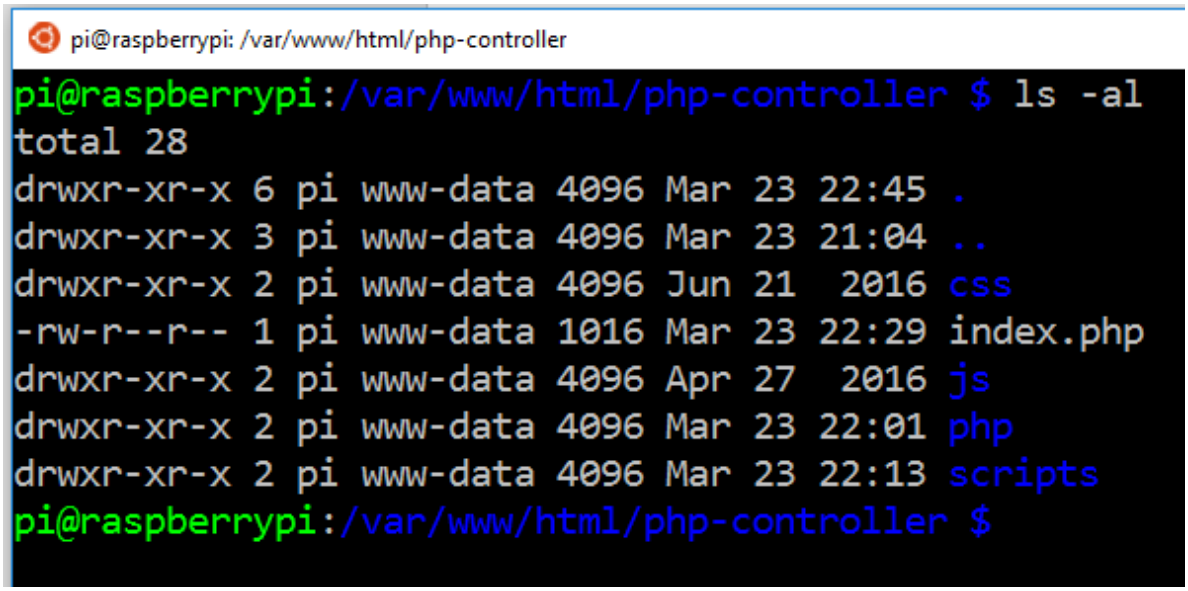


6.2 *Ανάπτυξη της PHP εφαρμογής*

Στην αρχή, δημιουργήσαμε μερικούς υποφάκελους για την καλύτερη οργάνωση των αρχείων μας. Όπως μπορούμε να παρατηρήσουμε από το παρακάτω screenshot έχουμε:

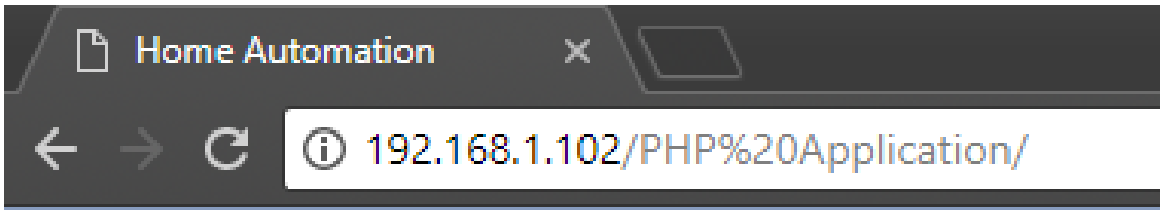
- Ένα φάκελο με όνομα **css** στο οποίο έχουμε τοποθετήσει όλα τα **css αρχεία** που θα χρησιμοποιηθούν για το **styling** της ιστοσελίδας μας.

- Ένα φάκελο με όνομα **js** που περιέχει όλα τα **αρχεία javascript**, συμπεριλαμβανομένης και της βιβλιοθήκης JQuery
- Ένα φάκελο που ονομάζεται **php** όπου έχουμε όλα τα **αρχεία PHP**
- Ένα φάκελο που ονομάζεται **scripts** όπου αποθηκεύουμε όλα τα Python scripts που εκτελούνται όταν πατάμε συγκεκριμένα κουμπιά της εφαρμογής
- Το αρχείο **index.php** που είναι η πρώτη σελίδα που κάποιος επισκέπτεται κατά την επίσκεψή μας στην εφαρμογή Ιστού

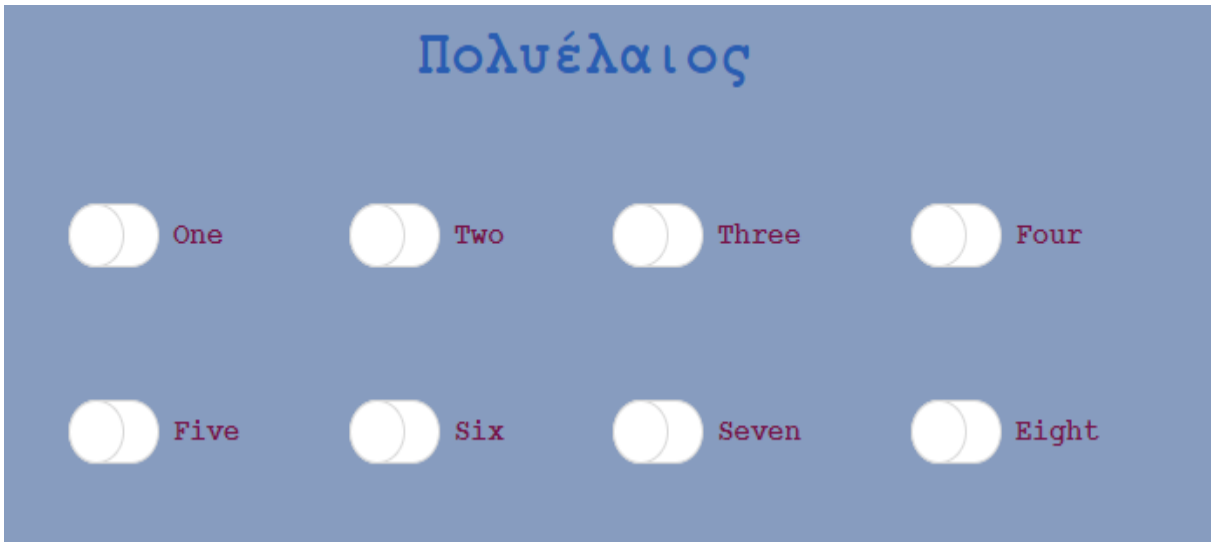


Για να επισκεφτούμε αυτή την ιστοσελίδα, ανοίγουμε ένα πρόγραμμα περιήγησης και πληκτρολογούμε τη **διεύθυνση IP** του Raspberry Pi (στη δική μας περίπτωση: 192.168.1.102), στη συνέχεια το σύμβολο της καθέτου και τέλος το όνομα του φακέλου στον οποίο τοποθετήσαμε όλα τα αρχεία μας ο φάκελος πρέπει να είναι μέσα στο / var / www / html και στην περίπτωσή μας, ονομάζεται: **PHP Application**)

Επομένως, η τελική μορφή του συνδέσμου πρέπει να είναι η εξής:



Η σελίδα προορισμού θα πρέπει να είναι παρόμοια με τα ακόλουθα:



Όπως φαίνεται στην παραπάνω εικόνα, υπάρχουν οχτώ ξεχωριστοί διακόπτες.

index.php:

```
1 <html>
 2  <head>
 3   <title> Home Automation</title>
 4
 5          <!-- we include all the appropriate javascript and css files -->
 6          <link rel="stylesheet" type="text/css" href= "css/Switch.css">
 7          <script type="text/javascript" src="js/jquery-1.12.3.min.js"> </script>
 8          <script type="text/javascript" src="js/functions.js"> </script>
 9  </head>
10
11  <body style="background-color:#879cbf">
12   <h1 align="center"><font face="Courier" size="6" color="295db2"> Πολυέλαιος </h1>
13
14   <table align="center" cellpadding="30">
15    <tr>
16     <td>
17      <input type='checkbox' class='ios8-switch ios8-switch-lg' id='checkbox-1'>
18      <label for='checkbox-1'><font color="761549"><b>One</b></font></label>
19     </td>
20     <td>
21      <input type='checkbox' class='ios8-switch ios8-switch-lg' id='checkbox-2'>
22      <label for='checkbox-2'><font color="761549"><b>Two</b></font></label>
23     </td>
24     <td>
25      <input type='checkbox' class='ios8-switch ios8-switch-lg' id='checkbox-3'>
26      <label for='checkbox-3'><font color="761549"><b>Three</font></label>
27     </td>
28     <td>
```

```
29          <input type='checkbox' class='ios8-switch ios8-switch-lg' id='checkbox-4'>
30          <label for='checkbox-4'><font color="761549"><b>Four</font></label>
31        </td>
32      </tr>
33      <tr>
34        <td>
35          <input type='checkbox' class='ios8-switch ios8-switch-lg' id='checkbox-5'>
36          <label for='checkbox-5'><font color="761549"><b>Five</font></label>
37        </td>
38        <td>
39          <input type='checkbox' class='ios8-switch ios8-switch-lg' id='checkbox-6'>
40          <label for='checkbox-6'><font color="761549"><b>Six</font></label>
41        </td>
42        <td>
43          <input type='checkbox' class='ios8-switch ios8-switch-lg' id='checkbox-7'>
44          <label for='checkbox-7'><font color="761549"><b>Seven</font></label>
45        </td>
46        <td>
47          <input type='checkbox' class='ios8-switch ios8-switch-lg' id='checkbox-8'>
48          <label for='checkbox-8'><font color="761549"><b>Eight</font></label>
49        </td>
50      </tr>
51    </table>
52  </body>
53 </html>
```

Η JQuery μας βοηθά να καλέσουμε τις αντίστοιχες σελίδες php μέσω του AJAX. Αυτή η λειτουργία υλοποιείται μέσα στο αρχείο **functions.js**.

Functions.js:

```
1 $(document).ready(function()
 2 {
 3   $( "#checkbox-1" ).click(function()
 4   {
 5     if($('#checkbox-1').is(':checked') )
 6       {
 7         $.ajax({url: "php/one-checked.php "});
 8       }
 9     else
10       {
11         $.ajax({url: "php/one-unchecked.php"});
12       }
13   });
14
15   $( "#checkbox-2" ).click(function()
16   {
```

```
17   if($('#checkbox-2').is(':checked') )
18     {
19       $.ajax({url: "php/two-checked.php"});
20     }
21   else
22     {
23       $.ajax({url: "php/two-unchecked.php"});
24     }
25 });
26
27 $( "#checkbox-3" ).click(function()
28 {
29   if($('#checkbox-3').is(':checked') )
30     {
31       $.ajax({url: "php/three-checked.php"});
32     }
33   else
34     {
35       $.ajax({url: "php/three-unchecked.php"});
36     }
37 });
38
39 $( "#checkbox-4" ).click(function()
40 {
41   if($('#checkbox-4').is(':checked') )
42     {
43       $.ajax({url: "php/four-checked.php"});
44     }
45   else
46     {
47       $.ajax({url: "php/four-unchecked.php"});
48     }
49 });
50
51 $( "#checkbox-5" ).click(function()
52 {
53   if($('#checkbox-5').is(':checked') )
54     {
55       $.ajax({url: "php/five-checked.php"});
56     }
57   else
58     {
59       $.ajax({url: "php/five-unchecked.php"});
60     }
61 });
62
63 $( "#checkbox-6" ).click(function()
```

```
64  {
65   if($('#checkbox-6').is(':checked') )
66     {
67       $.ajax({url: "php/six-checked.php"});
68     }
69   else
70     {
71       $.ajax({url: "php/six-unchecked.php"});
72     }
73  });
74
75  $( "#checkbox-7" ).click(function()
76  {
77   if($('#checkbox-7').is(':checked') )
78     {
79       $.ajax({url: "php/seven-checked.php"});
80     }
81   else
82     {
83       $.ajax({url: "php/seven-unchecked.php"});
84     }
85  });
86
87  $( "#checkbox-8" ).click(function()
88  {
89   if($('#checkbox-8').is(':checked') )
90     {
91       $.ajax({url: "php/eight-checked.php"});
92     }
93   else
94     {
95       $.ajax({url: "php/eight-unchecked.php"});
96     }
97  });
98 });
```

Προχωρώντας, το γραφικό μέρος της εφαρμογής καλύπτεται από τα αρχεία css που περιλαμβάνονται στο φάκελο "**css**". Το αρχείο **Switch.css** του CSS περιέχει τον κώδικα για τη διακόσμηση των διακοπτών που διαχειρίζονται τα κανάλια του ρελέ.

Switch.cs:

```
1 input[type="checkbox"].ios8-switch {
 2    position: absolute;
 3    margin: 8px 0 0 16px;
 4 }
 5 input[type="checkbox"].ios8-switch + label {
```

```
    position: relative;
    padding: 5px 0 0 50px;
    line-height: 2.0em;
}
input[type="checkbox"].ios8-switch + label:before {
    content: "";
    position: absolute;
    display: block;
    left: 0;
    top: 0;
    width: 40px; /* x*5 */
    height: 24px; /* x*3 */
    border-radius: 16px; /* x*2 */
    background: #fff;
    border: 1px solid #d9d9d9;
    -webkit-transition: all 0.3s;
    transition: all 0.3s;
}
input[type="checkbox"].ios8-switch + label:after {
    content: "";
    position: absolute;
    display: block;
    left: 0px;
    top: 0px;
    width: 24px; /* x*3 */
    height: 24px; /* x*3 */
    border-radius: 16px; /* x*2 */
    background: #fff;
    border: 1px solid #d9d9d9;
    -webkit-transition: all 0.3s;
    transition: all 0.3s;
}
input[type="checkbox"].ios8-switch + label:hover:after {
    box-shadow: 0 0 5px rgba(0,0,0,0.3);
}
input[type="checkbox"].ios8-switch:checked + label:after {
    margin-left: 16px;
}
input[type="checkbox"].ios8-switch:checked + label:before {
    background: #55D069;
}

/* LARGE */

input[type="checkbox"].ios8-switch-lg {
    margin: 10px 0 0 20px;
}
```

```
53 input[type="checkbox"].ios8-switch-lg + label {
54     position: relative;
55     padding: 7px 0 0 60px;
56     line-height: 2.3em;
57 }
58 input[type="checkbox"].ios8-switch-lg + label:before {
59     width: 50px; /* x*5 */
60     height: 30px; /* x*3 */
61     border-radius: 20px; /* x*2 */
62 }
63 input[type="checkbox"].ios8-switch-lg + label:after {
64     width: 30px; /* x*3 */
65     height: 30px; /* x*3 */
66     border-radius: 20px; /* x*2 */
67 }
68 input[type="checkbox"].ios8-switch-lg + label:hover:after {
69     box-shadow: 0 0 8px rgba(0,0,0,0.3);
70 }
71 input[type="checkbox"].ios8-switch-lg:checked + label:after {
72     margin-left: 20px; /* x*2 */
73 }
```

Επιπλέον, κάθε φορά που ο χρήστης κάνει κλικ σε ένα κουμπί, η αντίστοιχη σελίδα PHP καλείται μέσω AJAX. Όλες αυτές οι PHP σελίδες έχουν δύο βασικούς ρόλους. Ο πρώτος είναι να ενεργοποιήσουν το σωστό python script (χρησιμοποιώντας την εντολή php **shell_exec ()**) και το δεύτερο να εκτυπώσουν (**echo ()**) το κατάλληλο μήνυμα στην οθόνη. Έτσι, η δομή τους είναι σχεδόν παρόμοια, αλλά με μικρές διαφορές στο τυπωμένο κείμενο κάθε φορά.

one-checked.php → User turns ON the light switch 1

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-one-on.py");
3 echo "<pre>$output</pre>";
4 ?>
```

one-unchecked.php → User turns OFF the light switch 1

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-one-off.py");
3 echo "<pre>$output</pre>";
4 ?>
```

two-checked.php → User turns ON the light switch 2

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-two-on.py");
3 echo "<pre>$output</pre>";
4 ?>
```

two-unchecked.php → User turns OFF the light switch 2

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-two-off.py");
3 echo "<pre>$output</pre>";
4 ?>
```

tree-checked.php → User turns ON the light switch 3

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-three-on.py");
3 echo "<pre>$output</pre>";
4 ?>
```

tree-unchecked.php → User turns OFF the light switch 3

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-three-off.py");
3 echo "<pre>$output</pre>";
4 ?>
```

four-checked.php → User turns ON the light switch 4

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-four-on.py");
3 echo "<pre>$output</pre>";
4 ?>
```

four-unchecked.php → User turns OFF the light switch 4

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-four-off.py");
3 echo "<pre>$output</pre>";
4 ?>
```

five-checked.php → User turns ON the light switch 5

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-five-on.py");
3 echo "<pre>$output</pre>";
4 ?>
```

five-unchecked.php → User turns OFF the light switch 5

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-five-off.py");
3 echo "<pre>$output</pre>";
4 ?>
```

six-checked.php → User turns ON the light switch 6

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-six-on.py");
3 echo "<pre>$output</pre>";
4 ?>
```

six-unchecked.php → User turns OFF the light switch 6

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-six-off.py");
3 echo "<pre>$output</pre>";
4 ?>
```

seven-checked.php→ User turns ON the light switch 7

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-seven-on.py");
3 echo "<pre>$output</pre>";
4 ?>
```

seven-unchecked.php → User turns OFF the light switch 7

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-seven-off.py");
3 echo "<pre>$output</pre>";
4 ?>
```

eight-checked.php → User turns ON the light switch 8

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-eight-on.py");
3 echo "<pre>$output</pre>";
4 ?>
```

eight-unchecked.php → User turns OFF the light switch 8

```
1 <?php
2 $output = shell_exec("sudo python ../scripts/switch-eight-off.py");
3 echo "<pre>$output</pre>";
4 ?>
```

6.3 *Python Scripts*

switch-one-on.py → The First Relay Input Led Lamp - GPIO 12 turns ON
switch-one-off.py → The First Relay Input Led Lamp - GPIO 12 turns OFF

switch-two-on.py → The First Relay Input Led Lamp - GPIO 6 turns ON
switch-two-off.py → The First Relay Input Led Lamp - GPIO 6 turns OFF

switch-three-on.py → The First Relay Input Led Lamp - GPIO 13 turns ON
switch-three-off.py → The First Relay Input Led Lamp - GPIO 13 turns OFF

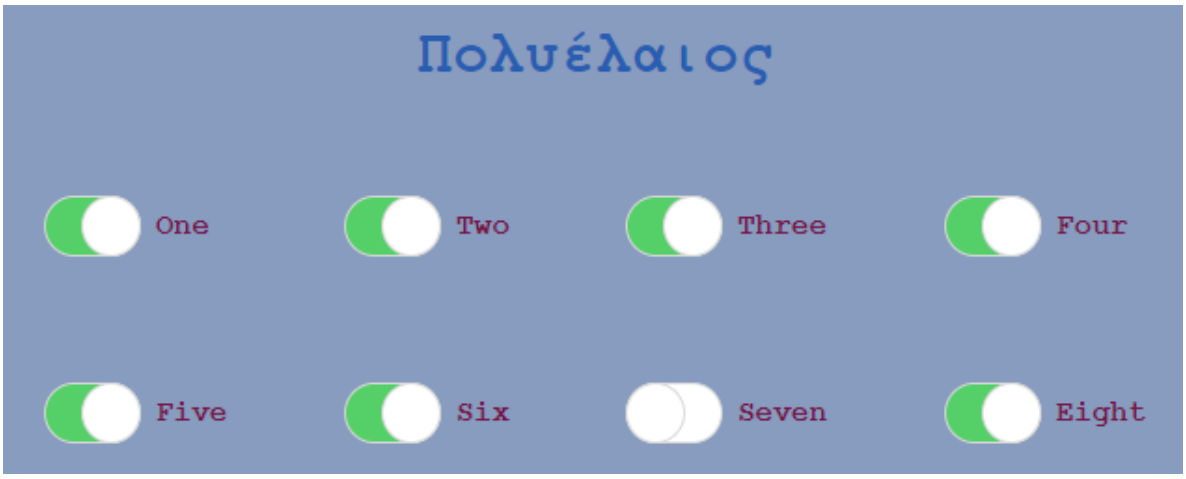
switch-four-on.py → The First Relay Input Led Lamp - GPIO 16 turns ON
switch-four-off.py → The First Relay Input Led Lamp - GPIO 16 turns OFF

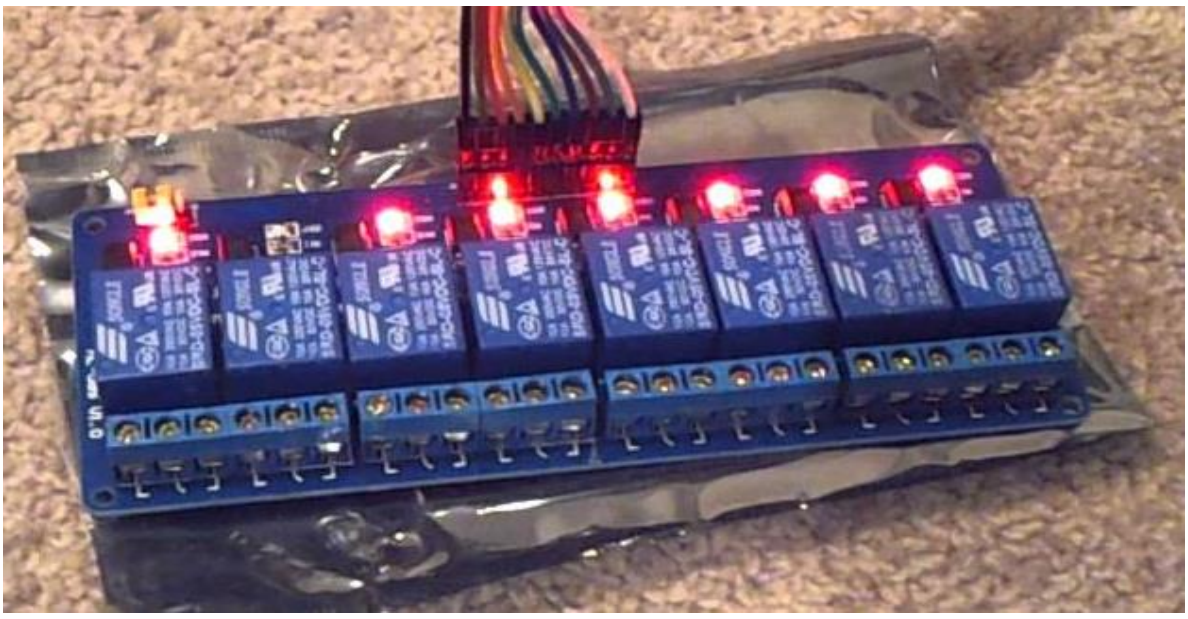
switch-five-on.py → The First Relay Input Led Lamp - GPIO 19 turns ON
switch-five-off.py → The First Relay Input Led Lamp - GPIO 19 turns OFF

switch-six-on.py → The First Relay Input Led Lamp - GPIO 20 turns ON
switch-six-off.py → The First Relay Input Led Lamp - GPIO 20 turns OFF

switch-seven-on.py → The First Relay Input Led Lamp - GPIO 26 turns ON
switch-seven-off.py → The First Relay Input Led Lamp - GPIO 26 turns OFF

switch-eight-on.py → The First Relay Input Led Lamp - GPIO 21 turns ON
switch-eight-off.py → The First Relay Input Led Lamp - GPIO 21 turns OFF





switch-one-on.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(12, GPIO.OUT)
GPIO.output(12, GPIO.HIGH)
try:
  GPIO.output(12, GPIO.LOW)
  # print "Relay Input 1 - Gpio: 12 - Led is on"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch-one-off.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(12, GPIO.OUT)
GPIO.output(12, GPIO.HIGH)

# main loop
try:
  GPIO.output(12, GPIO.HIGH)
  # print "Relay Input 1 - Gpio: 12 - Led is off"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch-two-on.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(6, GPIO.OUT)
GPIO.output(6, GPIO.HIGH)

try:
  GPIO.output(6, GPIO.LOW)
  # print "Relay Input 1 - Gpio: 6 - Led is on"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch- two -off.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(6, GPIO.OUT)
GPIO.output(6, GPIO.HIGH)

try:
  GPIO.output(6, GPIO.HIGH)
  # print "Relay Input 1 - Gpio: 6 - Led is off"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch-tree-on.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(13, GPIO.OUT)
GPIO.output(13, GPIO.HIGH)

try:
  GPIO.output(13, GPIO.LOW)
  # print "Relay Input 1 - Gpio: 13 - Led is on"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch- tree -off.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(13, GPIO.OUT)
GPIO.output(13, GPIO.HIGH)

try:
  GPIO.output(13, GPIO.HIGH)
  # print "Relay Input 1 - Gpio: 13 - Led is off"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch-four-on.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(16, GPIO.OUT)
GPIO.output(16, GPIO.HIGH)

try:
  GPIO.output(16, GPIO.LOW)
  # print "Relay Input 1 - Gpio: 16 - Led is on"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch- four -off.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(16, GPIO.OUT)
GPIO.output(16, GPIO.HIGH)

try:
  GPIO.output(16, GPIO.HIGH)
  # print "Relay Input 1 - Gpio: 16 - Led is off"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch-five-on.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(19, GPIO.OUT)
GPIO.output(19, GPIO.HIGH)

try:
  GPIO.output(19, GPIO.LOW)
  # print "Relay Input 1 - Gpio: 19 - Led is on"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch-five-off.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers
GPIO.setup(19, GPIO.OUT)
GPIO.output(19, GPIO.HIGH)

try:
  GPIO.output(19, GPIO.HIGH)
  # print "Relay Input 1 - Gpio: 19 - Led is off"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch-six-on.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(20, GPIO.OUT)
GPIO.output(20, GPIO.HIGH)

try:
  GPIO.output(20, GPIO.LOW)
  # print "Relay Input 1 - Gpio: 20 - Led is on"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch-six-off.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(20, GPIO.OUT)
GPIO.output(20, GPIO.HIGH)

try:
  GPIO.output(20, GPIO.HIGH)
  # print "Relay Input 1 - Gpio: 20 - Led is off"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch-seven-on.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(26, GPIO.OUT)
GPIO.output(26, GPIO.HIGH)

try:
  GPIO.output(26, GPIO.LOW)
  # print "Relay Input 1 - Gpio: 26 - Led is on"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch-seven-off.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(26, GPIO.OUT)
GPIO.output(26, GPIO.HIGH)

try:
  GPIO.output(26, GPIO.HIGH)
  # print "Relay Input 1 - Gpio: 26 - Led is off"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch-eight-on.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(21, GPIO.OUT)
GPIO.output(21, GPIO.HIGH)

try:
  GPIO.output(21, GPIO.LOW)
  # print "Relay Input 1 - Gpio: 21 - Led is on"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

switch-eight-off.py

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
# init list with pin numbers

GPIO.setup(21, GPIO.OUT)
GPIO.output(21, GPIO.HIGH)

try:
  GPIO.output(21, GPIO.HIGH)
  # print "Relay Input 1 - Gpio: 21 - Led is off"

# End program cleanly with keyboard
except KeyboardInterrupt:
  print "  Quit"

  # Reset GPIO settings
  GPIO.cleanup()
```

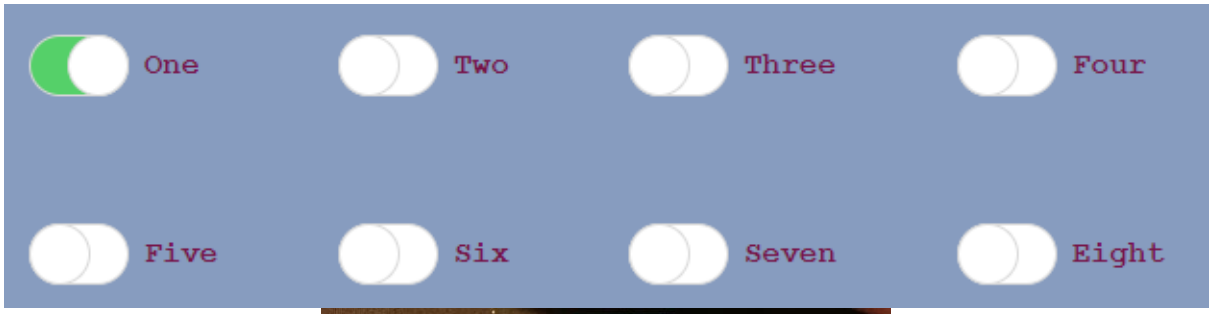
<u>Σημαντική σημείωση</u>:

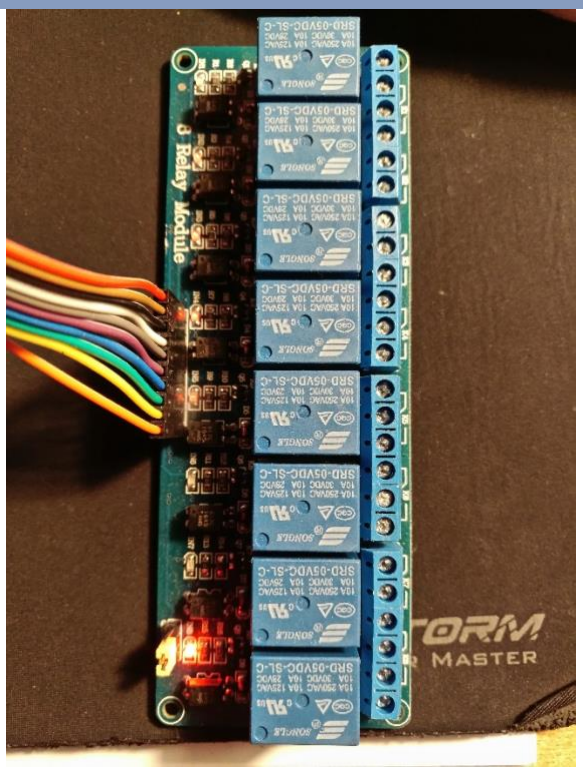
Τον Διακομιστής Apache τον διαχειρίζεται ο "**www-data**" χρήστη ο οποίος δεν συγκαταλέγεται μέσα στο αρχείο "**/etc/sudoers**", οπότε και δεν έχει τα δικαιώματα να "τρέξει" τα παραπάνω Python Scripts. Για να μπορέσει να εκτελέσει τα παραπάνω Python Scripts, θα πρέπει να του δώσουμε "**sudo**" δικαιώματα. Για να επιτευχθεί αυτό απλώς προσθέτουμε τον χρήστη "**www-data**" μέσα στο αρχείο με τους "**sudoers**" κάνοντας επεξεργασία στο αρχείο "**/etc/sudoers**" και προσθέτοντας την παρακάτω γραμμή από κώδικα στο τέλος του αρχείου:

```
www-data ALL=(ALL) NOPASSWD:ALL
```

6.4 *Δοκιμάζοντας την WEB εφαρμογή*

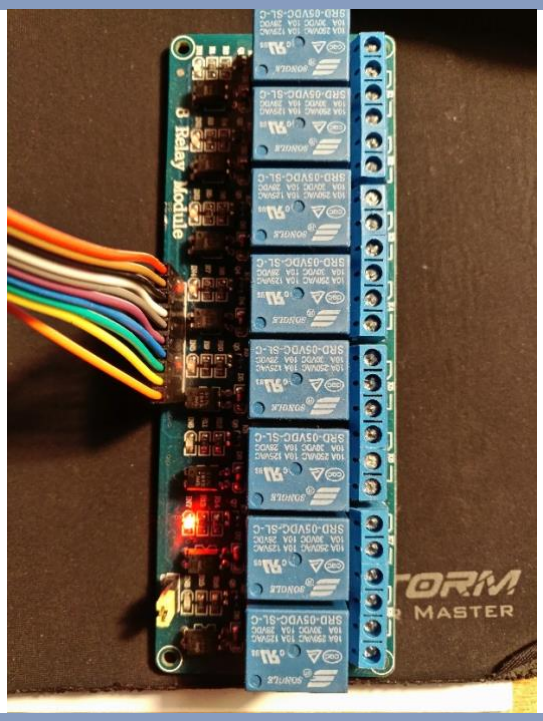
Παρακάτω, μπορείτε να δείτε κάποιες εικόνες που τράβηξα κατά τη δοκιμή διαφορετικών σεναρίων.





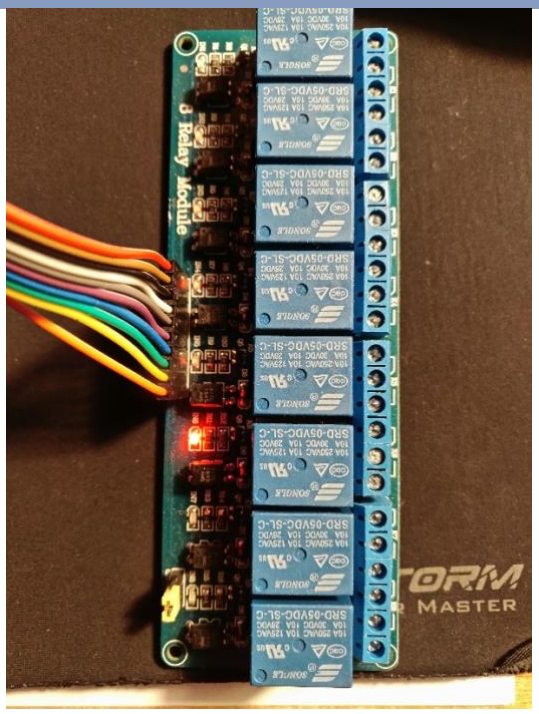
One Two Three Four

Five Six Seven Eight



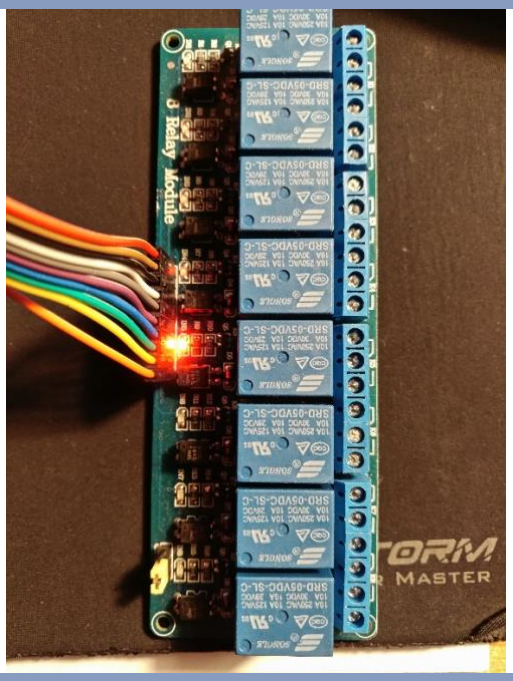
One Two Three Four

Five Six Seven Eight



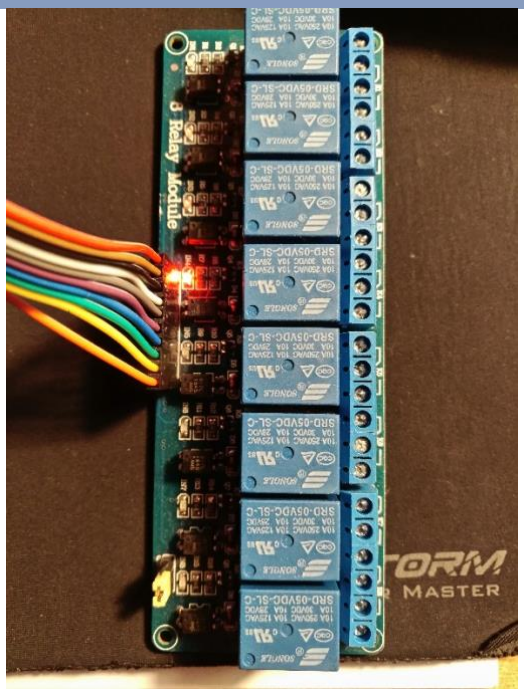
One Two Three Four

Five Six Seven Eight



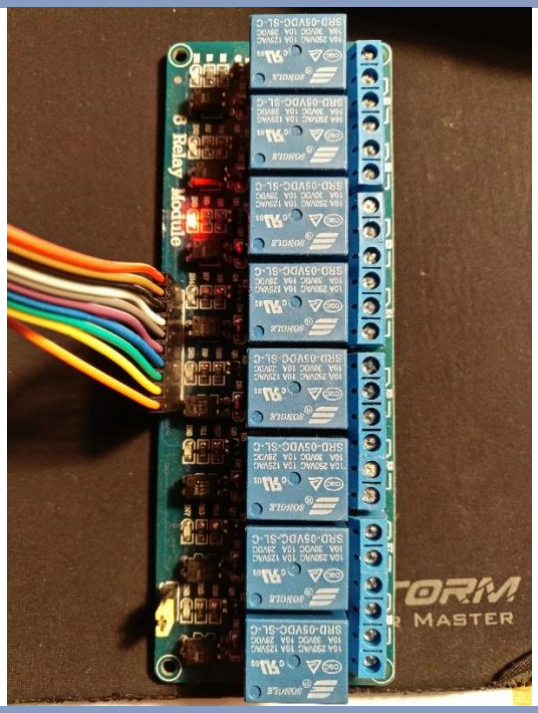
One Two Three Four

Five Six Seven Eight



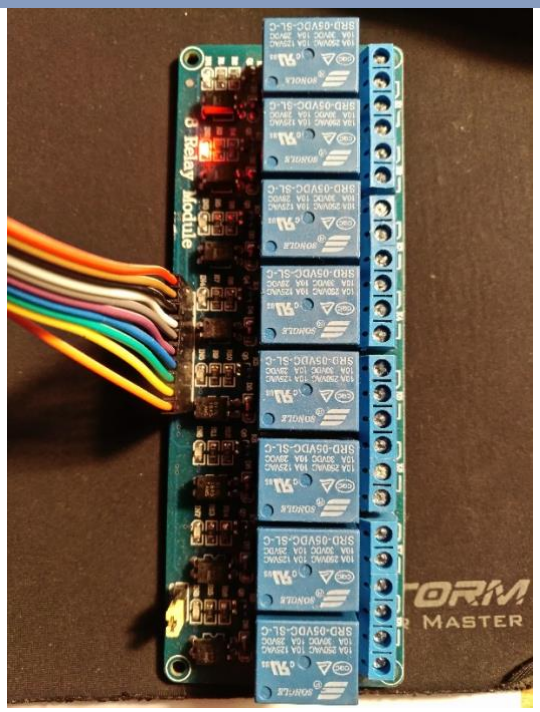
One Two Three Four

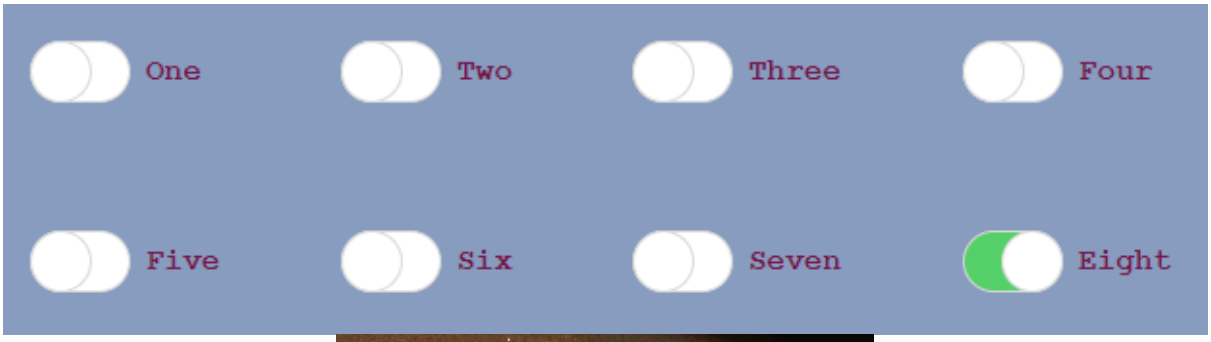
Five Six Seven Eight

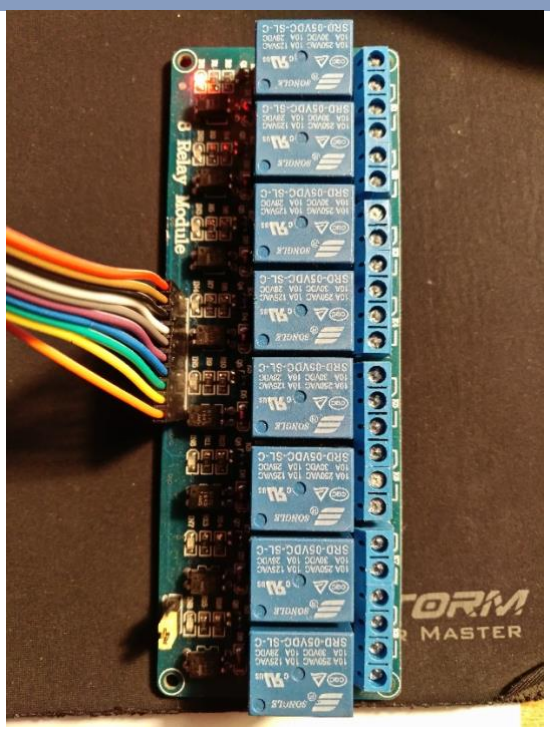


One Two Three Four

Five Six Seven Eight







7. Εφαρμοσμένη εφαρμογή Android (χρησιμοποιώντας πακέτα UDP)

Ο λόγος για τον οποίο επιλέξαμε να δημιουργήσουμε μια εφαρμογή Android ήταν κυρίως επειδή υπήρξε αξιοσημείωτη καθυστέρηση στην επικοινωνία μεταξύ της PHP Web Interface και του Raspberry. Αυτό συμβαίνει επειδή η εφαρμογή PHP στέλνει πακέτα TCP ενώ η εφαρμογή Android χρησιμοποιεί πακέτα UDP κατά τη διάρκεια της επικοινωνίας.

7.1 *TCP vs UDP*

Το πρωτόκολλο **TCP** λειτουργεί με τη δημιουργία δεσμών μεταξύ του αποστολέα και του δέκτη των πακέτων. Μόλις ολοκληρωθεί επιτυχώς μια σύνδεση, όλα τα δεδομένα αποστέλλονται από τον έναν υπολογιστή στον άλλο με τη μορφή πακέτων χρησιμοποιώντας αυτή τη σύνδεση. Τα κύρια χαρακτηριστικά του **TCP** είναι:

Αξιοπιστία: Το TCP χρησιμοποιεί διάφορους μηχανισμούς για να διασφαλίσει ότι τα πακέτα που μεταδίδονται από τον αποστολέα θα φτάσουν σίγουρα στον παραλήπτη με τη σωστή σειρά. Αυτοί οι μηχανισμοί περιλαμβάνουν την επιβεβαίωση της λήψης ενός πακέτου από τον παραλήπτη, την επαναπροώθηση χαμένων πακέτων και τον καθορισμό ενός ελάχιστου χρονικού διαστήματος εντός του οποίου πρέπει να λαμβάνεται κάθε πακέτο (timeout). Σε περίπτωση απώλειας ενός πακέτου, ο αποστολέας προσπαθεί να το ξαναστείλει. Επίσης, αν ο παραλήπτης διαπιστώσει ότι ένα πακέτο δεν έχει φτάσει, τότε θα ζητήσει από τον αποστολέα να το στείλει ξανά.

Packet Series: Εάν δύο πακέτα αποστέλλονται σε μία σύνδεση το ένα μετά το άλλο, τότε το πρωτόκολλο TCP εγγυάται ότι θα φτάσουν στον παραλήπτη με την ίδια σειρά με την αποστολή τους.

Σε περίπτωση που ένα πακέτο λείπει και άλλα πακέτα έρχονται πρώτα, τότε κρατούνται στο buffer μέχρι να φτάσει το πακέτο που λείπει. Στη συνέχεια αναδιατάσσονται και εμφανίζονται με τη σωστή σειρά στον παραλήπτη.

Πολλές πληροφορίες: Το πρωτόκολλο TCP θεωρείται ιδιαίτερα "βαρύ", δεδομένου ότι απαιτούνται τουλάχιστον 3 πακέτα για να δημιουργηθεί η σύνδεση πριν από τη μετάδοση οποιουδήποτε πακέτου δεδομένων. Επίσης, οι μηχανισμοί αξιοπιστίας που την εφαρμόζουν καθιστούν ακόμα πιο "βαρύ", γεγονός που φυσικά έχει σημαντικό αντίκτυπο στην ταχύτητα μετάδοσης δεδομένων. Το UDP είναι ένα απλούστερο και ευκολότερο πρωτόκολλο, στο οποίο δεν υπάρχει έννοια σύνδεσης. Κάθε πακέτο UDP διασχίζει το δίκτυο ως ξεχωριστή αυτόνομη μονάδα αντί μιας σειράς πακέτων σε μια σύνδεση, όπως το TCP.

Τα κύρια χαρακτηριστικά του **UDP** είναι:

Αναξιόπιστο: Κατά την αποστολή ενός πακέτου, ο αποστολέας δεν είναι σε θέση να γνωρίζει εάν το πακέτο θα φτάσει στον προορισμό του σωστά ή θα χαθεί στο δίκτυο. Δεν υπάρχει πρόβλεψη για επιβεβαίωση λήψης πακέτου από τον παραλήπτη ούτε για αναμετάδοση χαμένου πακέτου.

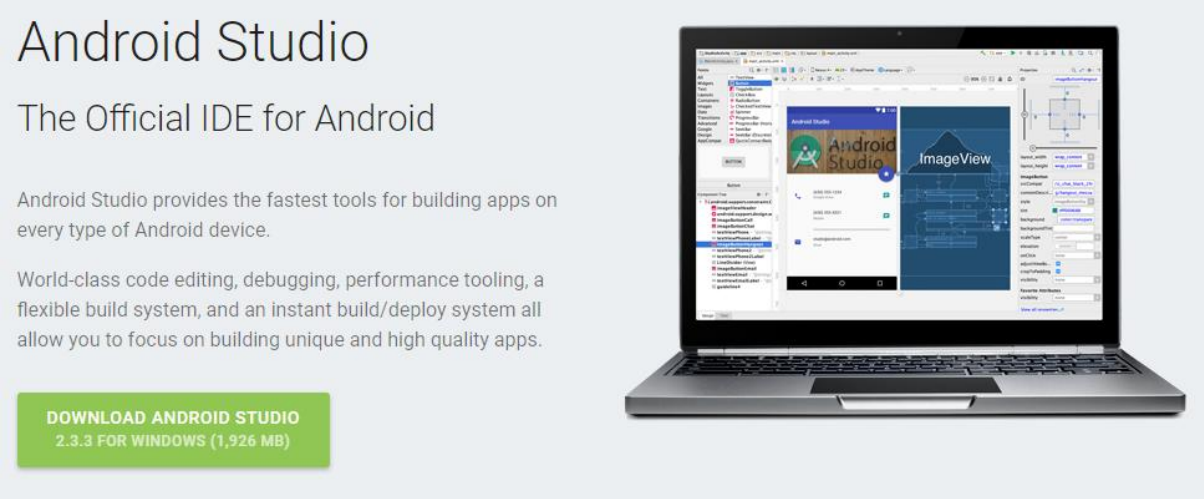
Δεν υπάρχει σειρά: Τα πακέτα UDP, σε αντίθεση με το TCP, δεν είναι αριθμημένα και επομένως δεν υπάρχει συγκεκριμένη σειρά με την οποία θα πρέπει να φτάσουν στον παραλήπτη.

Ελαφρύ για το δίκτυο: Το πρωτόκολλο αυτό είναι πολύ ελαφρύ σε σχέση με το TCP επειδή δεν εφαρμόζει όλους τους αξιόπιστους μηχανισμούς επικοινωνίας. Αυτό έχει ως αποτέλεσμα να είναι πολύ πιο γρήγορο.

Datagrams: Κάθε πακέτο UDP ονομάζεται επίσης "datagram" και θεωρείται ως μία ενιαία οντότητα που πρέπει να μεταδοθεί στο σύνολό της. Κατά συνέπεια, δεν υπάρχει η ιδέα της διακλάδωσης πακέτων μέσα σε ένα κανάλι / σύνδεση.

7.2 *Ανάπτυξη της εφαρμογής Android*

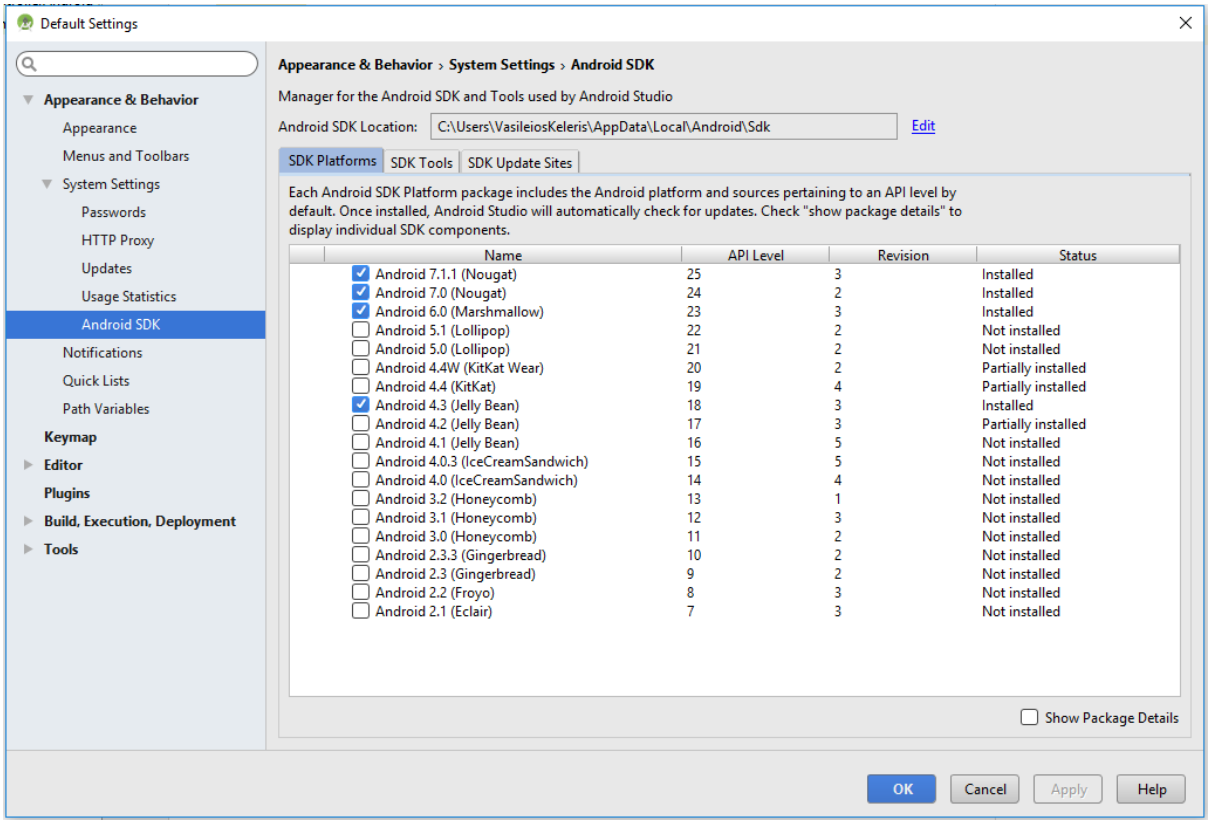
Θα χρησιμοποιήσουμε το Android Studio για την ανάπτυξη της εφαρμογής μας. Το Android Studio είναι ένα ολοκληρωμένο περιβάλλον προγραμματισμού (IDE) για την ανάπτυξη εφαρμογών στην πλατφόρμα Android. Για να κατεβάσετε την πιο πρόσφατη έκδοση, μπορούμε να επισκεφτούμε τον ακόλουθο σύνδεσμο και στη συνέχεια μπορούμε να πατήσουμε εδώ.



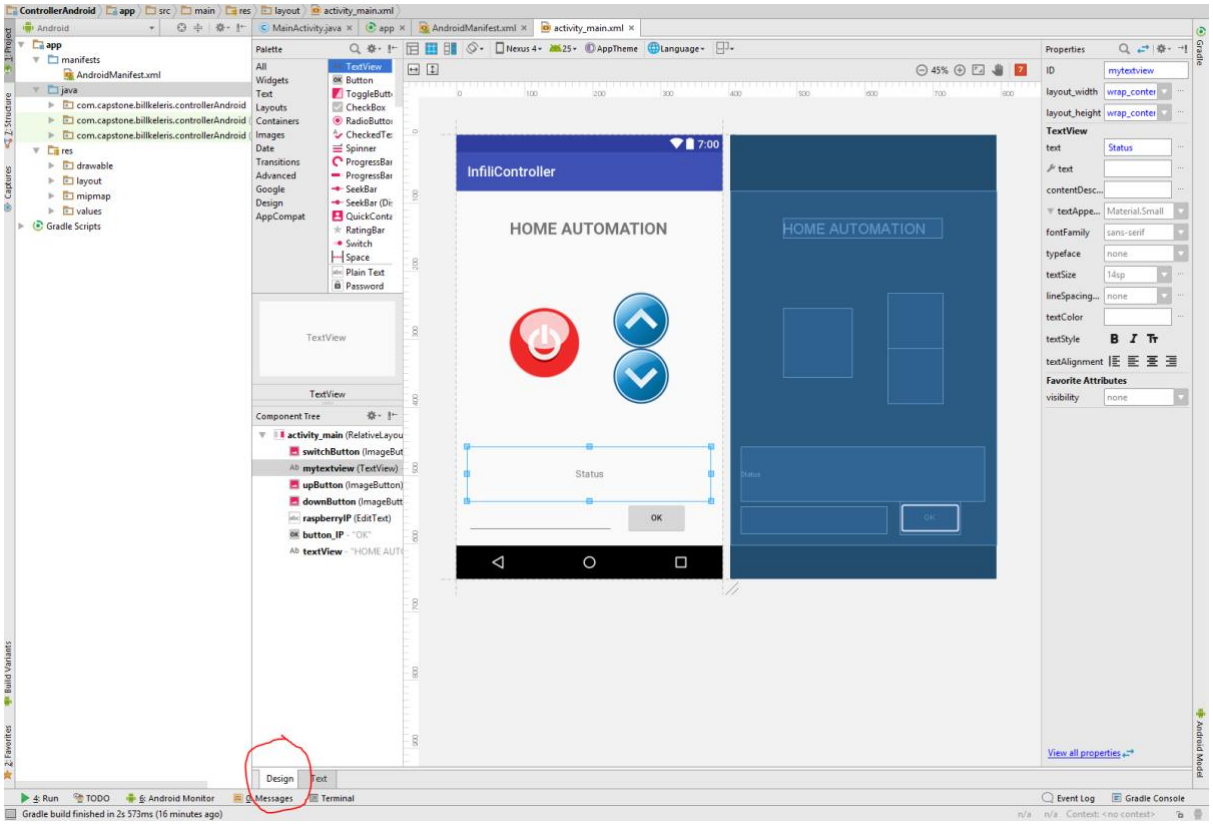
Μόλις κατεβάσαμε επιτυχώς, εγκαταστήσαμε και τρέξαμε το Android Studio, δημιουργήσαμε μία νέα εφαρμογή Android και της δώσαμε το όνομα της επιλογής μας. Στην περίπτωσή μας, το όνομα της εφαρμογής είναι "**ControllerAndroid**" και ο τομέας της εταιρείας είναι "**com.capstone.billkeleris**"

Υπάρχουν δύο τρόποι για να εκτελέσετε την εφαρμογή μας. Η πρώτη δημιουργεί και χρησιμοποιεί μια εικονική κινητή συσκευή και η δεύτερη, που προτείνω, χρησιμοποιεί το δικό μας κινητό τηλέφωνο Android μετά τη σύνδεσή του με τον υπολογιστή μας με ένα καλώδιο USB.

Στη συνέχεια, στο μενού της επάνω γραμμής, κάνουμε κλικ στο **Εργαλεία -> Android -> Διαχειριστής SDK** και εγκαθιστούμε την τρέχουσα έκδοση Android του κινητού μας τηλεφώνου.



Το Android Studio προσφέρει ένα πολύ χρήσιμο περιβάλλον για να σχεδιάσετε την εφαρμογή σας χρησιμοποιώντας εργαλεία μεταφοράς και απόθεσης. Για να γίνει αυτό, ανοίγουμε το αρχείο **activity_main.xml** και κάνουμε κλικ στην καρτέλα **Σχεδίαση**



Σε αυτό το σημείο μπορούμε να χρησιμοποιήσουμε Κουμπιά, Κουτιά Ελέγχου, TextViews, RadioButtons και οποιοδήποτε άλλο στοιχείο που επιθυμούμε από το αριστερό αναπτυσσόμενο μενού. Τα στοιχεία που χρησιμοποιήσαμε στην εφαρμογή μας είναι:

TextView → Τίτλος της εφαρμογής: **HOME AUTOMATION**

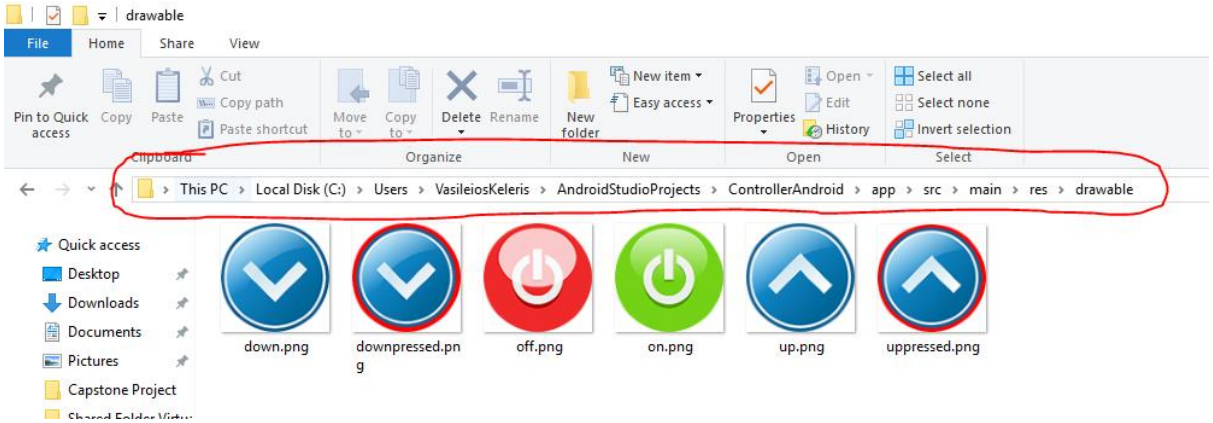
ImageButtons → **Διακόπτης ενεργοποίησης / απενεργοποίησης** και **βέλη πάνω / κάτω**

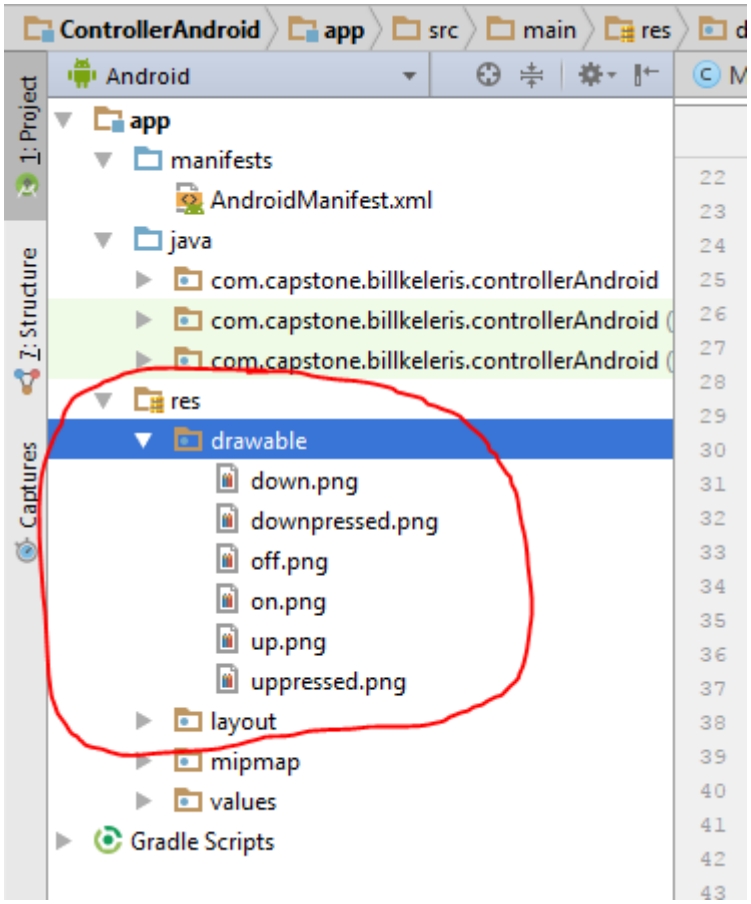
TextView → **Τρέχουσα κατάσταση** των κουμπιών που έχουν πατηθεί (π.χ. Ενεργοποίηση, Απενεργοποίηση, πατημένο, μη πατημένο)

EditText → Το πεδίο στο κάτω μέρος της οθόνης, όπου ο χρήστης εισάγει **τη διεύθυνση IP του Raspberry Pi.**

Κουμπί → Το **κουμπί OK** δίπλα στο EditText που επιβεβαιώνει την εισαγωγή του χρήστη.

Όλες οι εικόνες που χρησιμοποιήσαμε για το διακόπτη ενεργοποίησης / απενεργοποίησης και τα βέλη "Επάνω / Κάτω" πρέπει να τοποθετηθούν μέσα στο **ανασυρόμενο** φάκελο που βρίσκεται κάτω από το αρχείο **res** που βρίσκεται στον ακόλουθο κατάλογο των Windows:





Όλες οι εικόνες έχουν συμπιεστεί για να μειωθεί η καθυστέρηση και όπως μπορούμε να συμπεράνουμε από τα παραπάνω screenshots:

- Όταν πιέσετε το πλήκτρο **off.png**, αντικαθίσταται από το πλήκτρο **on.png**
- Όταν πατηθεί το κουμπί **on.png**, αντικαθίσταται από το πλήκτρο **off.png**
- Όταν πατήσετε το πλήκτρο **up.png**, αντικαθίσταται από το κουμπί **uppressed.png**
- Όταν πιέσετε το κουμπί **down.png**, αντικαθίσταται από το κουμπί **downpressed.png**

Με αυτόν τον τρόπο καταφέρουμε να δημιουργήσουμε μερικά πολύ ωραία εφέ που βοηθούν τον χρήστη να καταλάβει ποιο κουμπί πιέζεται κάθε φορά.

The **final code** of the user interface can be found below:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

```
android:id="@+id/activity_main"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context="com.capstone.billkeleris.controllerAndroid.MainActivity">


<ImageButton
   android:id="@+id/switchButton"
   android:layout_width="100dp"
   android:layout_height="100dp"
   android:background="#0000"
   android:scaleType="fitXY"
   android:src="@drawable/off"
   android:layout_alignTop="@+id/upButton"
   android:layout_alignLeft="@+id/textView"
   android:layout_alignStart="@+id/textView"
   android:layout_marginTop="22dp" />


<TextView
   android:id="@+id/mytextview"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Status"
   android:gravity="center"
   android:layout_below="@+id/downButton"
   android:layout_marginTop="62dp"
   android:layout_alignParentLeft="true"
   android:layout_alignParentStart="true"
   android:layout_above="@+id/button_IP"
   android:layout_alignParentRight="true"
   android:layout_alignParentEnd="true" />


<ImageButton
   android:id="@+id/upButton"
   android:layout_width="80dp"
   android:layout_height="80dp"
   android:src="@drawable/up"
   android:scaleType="fitXY"
   android:background="#0000"
   android:layout_marginTop="78dp"
   android:layout_below="@+id/textView"
   android:layout_alignRight="@+id/textView"
   android:layout_alignEnd="@+id/textView" />


<ImageButton
   android:id="@+id/downButton"
   android:layout_width="80dp"
   android:layout_height="80dp"
   android:src="@drawable/down"
   android:scaleType="fitXY"
   android:background="#0000"
   android:layout_below="@+id/upButton"
```

```
    android:layout_alignLeft="@+id/upButton"
    android:layout_alignStart="@+id/upButton" />

  <EditText
    android:id="@+id/raspberryIP"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_toLeftOf="@+id/downButton"
    android:layout_toStartOf="@+id/downButton"
    android:textAlignment="viewStart"
    android:textSize="16sp" />

  <Button
    android:id="@+id/button_IP"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="OK"
    android:textSize="12sp"
    android:layout_alignParentBottom="true"
    android:layout_toRightOf="@+id/raspberryIP"
    android:layout_toEndOf="@+id/raspberryIP"
    android:layout_marginLeft="18dp"
    android:layout_marginStart="18dp" />

  <TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="24dp"
    android:lineSpacingExtra="18sp"
    android:text="HOME AUTOMATION"
    android:textSize="24sp"
    android:textStyle="bold" />

</RelativeLayout>
```

Κάθε εφαρμογή Android πρέπει να έχει ένα αρχείο AndroidManifest.xml. Αυτό το αρχείο παρέχει κρίσιμες και ουσιώδεις πληροφορίες, τις οποίες το σύστημα πρέπει να γνωρίζει πριν να μπορέσει να εκτελέσει οποιοδήποτε κώδικα. Ο κώδικας που χρησιμοποιήσαμε για το AndroidManifest.xml παρουσιάζεται παρακάτω. Όπως μπορείτε να δείτε, περιγράφουμε τα στοιχεία της εφαρμογής μας, αναθέτουμε τα δικαιώματα, ονομάζουμε το πακέτο Java και δηλώνουμε ότι η εφαρμογή μας θα εκτελείται μόνο σε πορτραίτο.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.capstone.billkeleris.controllerAndroid">

  <application
    android:largeHeap="true"
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
```

```
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">

    <activity
      android:name="com.capstone.billkeleris.controllerAndroid.MainActivity"
      android:screenOrientation="portrait" >
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>

  <uses-permission android:name="android.permission.INTERNET" />

</manifest>
```

Προχωρώντας, ανοίγουμε και επεξεργαζόμαστε το αρχείο **MainActivity.java**. Αυτό το αρχείο περιλαμβάνει όλο τον κώδικα Java της εφαρμογής μας Android. Πρώτον, εισάγουμε όλες τις κατάλληλες βιβλιοθήκες στο Έργο μας

```
package com.capstone.billkeleris.controllerAndroid;

import android.content.Context;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.TextView;

import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.net.UnknownHostException;
```

Στη συνέχεια, δημιουργούμε την κλάση MainActivity που επεκτείνει την AppCompatActivity και δηλώνουμε όλες τις μεταβλητές που πρόκειται να χρησιμοποιήσουμε.

```
public class MainActivity extends AppCompatActivity {

  boolean switchButton_isPressed = true;

  ImageButton switchButton, up, down;
  TextView tv;

  String python_server;
```

```
Button button_IP;
EditText raspberry_IP;
```

Τώρα, ήρθε η ώρα να αντιστοιχίσουμε τις αντίστοιχες τιμές στις μεταβλητές που μόλις δημιουργήσαμε. Η εντολή **findViewById (R.id.componentId)** αναζητά την ταυτότητα των στοιχείων που δημιουργήσαμε νωρίτερα στο Graphical User Interface.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);

   switchButton = (ImageButton) findViewById(R.id.switchButton);
   up = (ImageButton) findViewById(R.id.upButton);
   down = (ImageButton) findViewById(R.id.downButton);

   button_IP = (Button)findViewById(R.id.button_IP);
   raspberry_IP   = (EditText)findViewById(R.id.raspberryIP);
   raspberry_IP.setHint("Raspberry IP");
```

Το ακόλουθο κομμάτι κώδικα αντιστοιχίζει τη διεύθυνση IP που έχει εισάγει ο χρήστης μέσα στο πεδίο κειμένου στη μεταβλητή **raspberry_IP.** Με αυτόν τον τρόπο, η εφαρμογή μας Android γνωρίζει τη διεύθυνση IP του Raspberry Pi, στο οποίο θα στείλει όλα τα πακέτα.

```
button_IP.setOnClickListener(
      new View.OnClickListener()
      {
         public void onClick(View view)
         {
            python_server = raspberry_IP.getText().toString();
            tv = (TextView)findViewById(R.id.mytextview);
            tv.setText("IP was successfully assigned");

            InputMethodManager inputManager = (InputMethodManager)
                  getSystemService(Context.INPUT_METHOD_SERVICE);

            inputManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(),
                  InputMethodManager.HIDE_NOT_ALWAYS);
         }
      });
```

Στην παρακάτω ενότητα, εφαρμόζουμε τη λειτουργία του διακόπτη λυχνιών On / Off. Η συνάρτηση **setOnClickListener ()** μας δίνει τη δυνατότητα να καθορίσουμε μια συγκεκριμένη ενέργεια όταν πιέσουμε το κουμπί. Ο boolean μεταβλητός διακόπτης **switchButton_isPressed** (true ή false) χρησιμοποιείται μέσα στην εντολή if για να προσδιοριστεί αν η λάμπα είναι ενεργοποιημένη ή απενεργοποιημένη. Εάν ο χρήστης κάνει κλικ στο πράσινο κουμπί (ON), αυτή η μεταβλητή θα γίνει αληθής. Από την άλλη πλευρά, αν ο χρήστης κάνει κλικ στο κόκκινο κουμπί (OFF), αυτή η μεταβλητή θα γίνει ψευδής. Η συνάρτηση setImageResource (). Στη συνέχεια, εκτυπώνουμε το αντίστοιχο μήνυμα στην οθόνη ανάλογα με την κατάσταση του διακόπτη χρησιμοποιώντας τη συνάρτηση setText (). Τέλος, δημιουργούμε ένα Datagram μέσω του οποίου στέλνουμε ένα πακέτο UDP που περιέχει το κείμενο "Ο λαμπτήρας είναι τώρα αναμμένος" ή "Ο λαμπτήρας είναι τώρα απενεργοποιημένος". Αυτό το κείμενο αποθηκεύεται στη μεταβλητή messageStr και στην περίπτωσή μας η θύρα του διακομιστή όπου στέλνουμε το πακέτο είναι: 12345

```
switchButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view)
    {

        if(switchButton_isPressed){
            switchButton.setImageResource(R.drawable.on);
            tv = (TextView)findViewById(R.id.mytextview);
            tv.setText("Lamp is now ON");

            new Thread() {
                @Override
                public void run() {
                    String messageStr="Lamp is now ON";
                    int server_port = 12345;
                    DatagramSocket s = null;
                    try {
                        s = new DatagramSocket();
                    } catch (SocketException e) {
                        e.printStackTrace();
                    }
                    InetAddress local = null;
                    try {
                        local = InetAddress.getByName(python_server);
                    } catch (UnknownHostException e) {
                        e.printStackTrace();
                    }
                    int msg_length=messageStr.length();
                    byte[] message = messageStr.getBytes();
                    DatagramPacket p = new DatagramPacket(message, msg_length,local,server_port);
                    try {
                        s.send(p);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    Log.d("myTag", "This is my message");
                }
            }.start();

        }
        else{
            switchButton.setImageResource(R.drawable.off);
            tv = (TextView)findViewById(R.id.mytextview);
            tv.setText("Lamp is now OFF");

            new Thread() {
                @Override
                public void run() {
                    String messageStr="Lamp is now OFF";
                    int server_port = 12345;
                    DatagramSocket s = null;
                    try {
                        s = new DatagramSocket();
                    } catch (SocketException e) {
                        e.printStackTrace();
                    }
                    InetAddress local = null;
                    try {
                        local = InetAddress.getByName(python_server);
```

```
                } catch (UnknownHostException e) {
                    e.printStackTrace();
                }
                int msg_length=messageStr.length();
                byte[] message = messageStr.getBytes();
                DatagramPacket p = new DatagramPacket(message, msg_length,local,server_port);
                try {
                    s.send(p);
                } catch (IOException e) {
                    e.printStackTrace();
                }
                Log.d("myTag", "This is my message");
            }
        }.start();

    }

    switchButton_isPressed = !switchButton_isPressed;

  }
});
```

Η ίδια λογική χρησιμοποιείται για την εφαρμογή των βέλη πάνω και κάτω. Αλλά σε αυτή την περίπτωση, αντί να χρησιμοποιήσουμε τη λειτουργία setOnClickListener (), χρησιμοποιούμε setOnTouchListener () επειδή πρέπει να εκτελέσουμε μια συγκεκριμένη ενέργεια για ολόκληρη την περίοδο που πατάμε το πλήκτρο. Η ενέργεια θα σταματήσει μόνο όταν ο χρήστης απελευθερώσει το δάχτυλό του από το βέλος. Έτσι, αυτή τη φορά, τα μηνύματα που στέλνουμε στο διακομιστή είναι: Το κουμπί "Πάνω" είναι πατημένο, το κουμπί "Πάνω" δεν είναι πατημένο, το κουμπί "Κάτω" πατηθεί και το κουμπί "Κάτω" δεν πατηθεί. Συνεπώς, εκτυπώνουμε το κατάλληλο κείμενο στην οθόνη Android όπως: Άνοιγμα πόρτας γκαράζ, κλείσιμο πόρτας γκαράζ και διακοπή.

UP ARROW

```
up.setOnTouchListener(new View.OnTouchListener() {
   @Override
   public boolean onTouch(View view, MotionEvent event)
   {
      if(event.getAction() == MotionEvent.ACTION_DOWN) {
         up.setImageResource(R.drawable.uppressed);
         tv = (TextView) findViewById(R.id.mytextview);
         tv.setText("Opening Garage Door");

         new Thread() {
            @Override
            public void run() {
               String messageStr="Up Button is Pressed";
               int server_port = 12345;
               DatagramSocket s = null;
               try {
                  s = new DatagramSocket();
               } catch (SocketException e) {
                  e.printStackTrace();
               }
               InetAddress local = null;
               try {
                  local = InetAddress.getByName(python_server);
               } catch (UnknownHostException e) {
```

```
                e.printStackTrace();
            }
            int msg_length=messageStr.length();
            byte[] message = messageStr.getBytes();
            DatagramPacket p = new DatagramPacket(message, msg_length,local,server_port);
            try {
                s.send(p);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }.start();

}
else if (event.getAction() == MotionEvent.ACTION_UP) {
    up.setImageResource(R.drawable.up);
    tv = (TextView) findViewById(R.id.mytextview);
    tv.setText("Stopped");

    new Thread() {
        @Override
        public void run() {
            String messageStr="Up Button is not Pressed";
            int server_port = 12345;
            DatagramSocket s = null;
            try {
                s = new DatagramSocket();
            } catch (SocketException e) {
                e.printStackTrace();
            }
            InetAddress local = null;
            try {
                local = InetAddress.getByName(python_server);
            } catch (UnknownHostException e) {
                e.printStackTrace();
            }
            int msg_length=messageStr.length();
            byte[] message = messageStr.getBytes();
            DatagramPacket p = new DatagramPacket(message, msg_length,local,server_port);
            try {
                s.send(p);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }.start();

}
    return false;
}
});
```

7.3 Python Server

Από την πλευρά του διακομιστή, πρέπει να εκτελέσουμε ένα script python που θα ακούει όλη την ώρα στην υποδοχή που περιμένει να λάβει τα μηνύματα που εστάλησαν νωρίτερα από την εφαρμογή Android. Οι καρφίτσες GPIO που χρησιμοποιήσαμε στο παράδειγμά μας είναι: pinList = [19, 6, 13].

- GPIO pin 19 – Relay Input 1 - corresponds to the On/Off Switch
- GPIO pin 6 – Relay Input 2 - corresponds to the Up Arrow
- GPIO pin 13 – Relay Input 3 - corresponds to the Down Arrow

Η μεταβλητή server_address αποθηκεύει τη διεύθυνση IP του Raspberry Pi και τη θύρα που ακούει τις συνδέσεις του πελάτη

server_address = ('192.168.2.2', 12345)

Χρησιμοποιώντας κάποιες δηλώσεις εάν ελέγξουμε το περιεχόμενο των ληφθέντων μηνυμάτων και έτσι ενεργοποιούμε την αντίστοιχη ενέργεια GPIO (GPIO.HIGH ή GPIO.LOW)

```
#!/usr/bin/python

import socket

import sys

import os


import RPi.GPIO as GPIO

import time


GPIO.setmode(GPIO.BCM)


# init list with pin numbers

pinList = [19, 6, 13]


# loop through pins and set mode and state to 'low'


for i in pinList:

    GPIO.setup(i, GPIO.OUT)

    GPIO.output(i, GPIO.HIGH)


# time to sleep between operations in the main loop


# SleepTimeL = 2
```

```
# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Bind the socket to the port
server_address = ('192.168.2.2', 12345)
print >>sys.stderr, 'starting up on %s port %s' % server_address
sock.bind(server_address)

while True:
    print >> sys.stderr, '...'
    data, address = sock.recvfrom(4096)

    # print >> sys.stderr, 'received %s bytes from %s' % (len(data), address)
    # print >> sys.stderr, data

    # if data:
        # sent = sock.sendto(data, address)
        # print >> sys.stderr, 'sent %s bytes back to %s' % (sent, address)

    # First Switch

    if data=="Lamp is now ON":
        print >> sys.stderr, "Lamp is now ON"
        # GPIO: 19 - IN1
        GPIO.output(19, GPIO.LOW)
        # time.sleep(SleepTimeL);

    if data=="Lamp is now OFF":
        print >> sys.stderr, "Lamp is now OFF"
        # GPIO: 19 - IN1
```

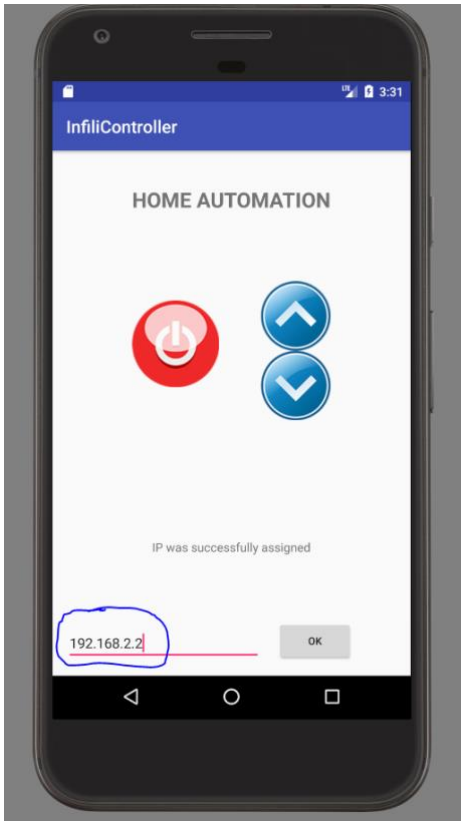
```
    GPIO.output(19, GPIO.HIGH)
    # time.sleep(SleepTimeL);




# UP BUTTON
if data=="Up Button is Pressed":
    print >> sys.stderr, "Opening Garage Door"
    # GPIO: 6 - IN3
    GPIO.output(6, GPIO.LOW)
    # time.sleep(SleepTimeL);



if data=="Up Button is not Pressed":
    print >> sys.stderr, "Stopped"
    # GPIO: 6 - IN3
    GPIO.output(6, GPIO.HIGH)
    # time.sleep(SleepTimeL);




# DOWN BUTTON
if data=="Down Button is pressed":
    print >> sys.stderr, "Closing Garage Door"
    # GPIO: 13 - IN2
    GPIO.output(13, GPIO.LOW)
    # time.sleep(SleepTimeL);



if data=="Down Button is not Pressed":
```

```
print >> sys.stderr, "Stopped"

# GPIO: 13 - IN2

GPIO.output(13, GPIO.HIGH)

# time.sleep(SleepTimeL);
```

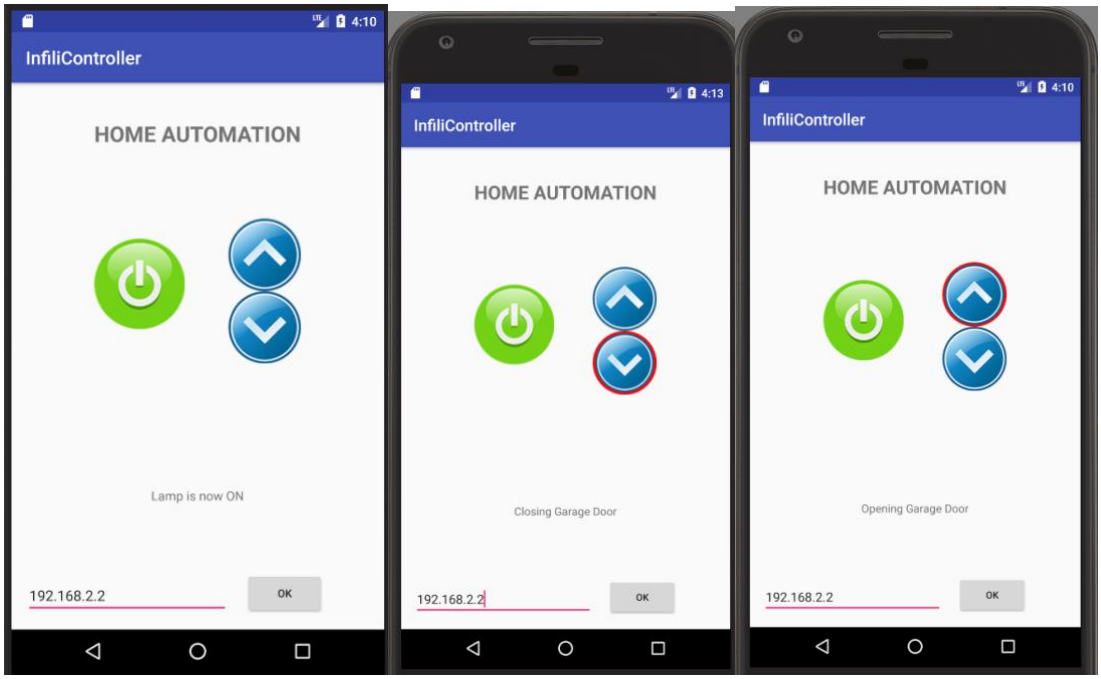
7.4 Δοκιμή της εφαρμογής Android

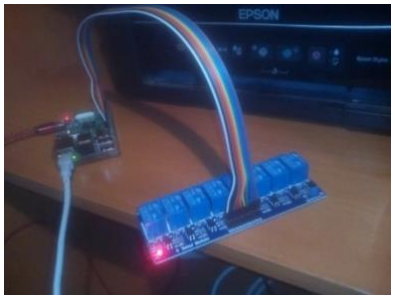
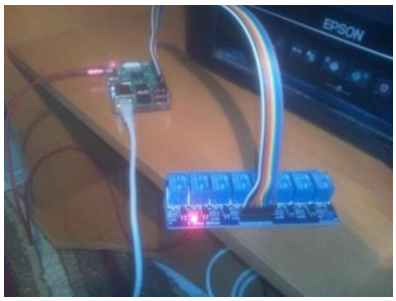
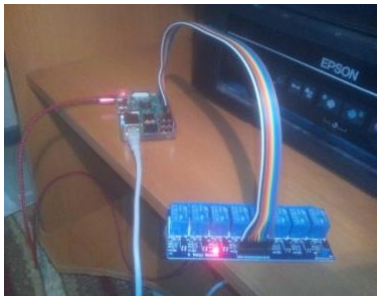
Για να δοκιμάσουμε την εφαρμογή Android, συνδέουμε το κινητό μας τηλέφωνο με τον υπολογιστή, κάνουμε κλικ στο πράσινο κουμπί λειτουργίας στην επάνω μπάρα του Android Studio και περιμένουμε την ολοκλήρωση της εγκατάστασης. Εάν όλα λειτουργούν καλά, η εφαρμογή πρέπει να ξεκινήσει αυτόματα μετά από μερικά δευτερόλεπτα και θα πρέπει να δούμε την παρακάτω οθόνη.

Τώρα, εισάγουμε τη διεύθυνση IP του Raspberry Pi στο πεδίο στο κάτω μέρος του παραθύρου και κάνουμε κλικ στο OK. Θα πρέπει να δείτε το μήνυμα "Η διεύθυνση IP έχει εκχωρηθεί με επιτυχία"



Τώρα, είμαστε έτοιμοι να δοκιμάσουμε τη λειτουργικότητα της εφαρμογής μας. Κάνοντας κλικ στον διακόπτη On / Off, θα πρέπει να ενεργοποιήσουμε και να απενεργοποιήσουμε το πρώτο κανάλι ρελέ. Ομοίως, θα πρέπει να παρατηρήσουμε ότι οι λαμπτήρες των δεύτερων και τρίτων καναλιών αναμετάδοσης ενεργοποιούνται και απενεργοποιούνται όταν κρατάμε τα κουμπιά βέλους πάνω και κάτω.



8 Λογισμικό αναγνώρισης φωνής χρησιμοποιώντας το Sphinx CMU

Το τελευταίο μέρος του Έργου μας είναι να προσθέσουμε φωνητική αναγνώριση στο λογισμικό μας ώστε να μπορούμε να ελέγξουμε τα φώτα και την πόρτα του γκαράζ χρησιμοποιώντας φωνητικές εντολές. Αυτή η λειτουργικότητα καθιστά τη ζωή μας πολύ πιο εύκολη και επιτρέπει μια ταχύτερη αλληλεπίδραση με τις έξυπνες συσκευές μας. Εκτός αυτού, ένα τέτοιο λογισμικό είναι πραγματικά χρήσιμο και απαραίτητο για τα άτομα με ειδικές ανάγκες, καθώς μερικές φορές η φωνή τους είναι ο μόνος τρόπος για να ολοκληρώσουν επιτυχώς πολλές από τις καθημερινές τους δραστηριότητες.

8.1 Εγκατάσταση και διαμόρφωση

Το λογισμικό που θα χρησιμοποιήσουμε για τη δημιουργία του συστήματος αναγνώρισης φωνής μας είναι ένα πλαίσιο ανοιχτού κώδικα που ονομάζεται Sphinx CMU. Αυτό το λογισμικό χρησιμοποιεί κάποιους αλγόριθμους αναγνώρισης ομιλίας για την παροχή αποτελεσματικής και ακριβούς αναγνώρισης ομιλίας. Το μεγάλο πλεονέκτημα της CMF Sphinx είναι ότι έχει σχεδιαστεί για πλατφόρμα χαμηλών πόρων, γι 'αυτό είναι το τέλειο εργαλείο για χρήση με το Raspberry Pi.

8.1.1 SphinxBase

Πρώτον, πρέπει να εγκαταστήσουμε όλες τις απαραίτητες εξαρτήσεις, έτσι τρέχουμε την εντολή:

```
sudo apt-get install gcc automake autoconf libtool bison swig python-dev libpulse-dev
```

Then we navigate to /home/pi/Desktop and we create a folder named: **speech_recognition**

```
mkdir speech_recognition
```

We cd (change directory) inside this newly created folder:

```
cd speech_recogntion
```

Now we need to install **sphinxbase** so we clone the appropriate repository

```
git clone https://github.com/cmusphinx/sphinxbase.git
```

We cd (change directory) inside the downloaded folder

```
cd sphinxbase
```

Now, we execute the script **autogen.sh** which is going to create all the files we will need for the compilation

```
./autogen.sh
```

After running successfully the previous command, we will notice that the file **configure** has been created to our directory. We execute this file by typing:

```
./configure
```

Now we are ready to run:

```
make clean all
```

and then:

```
make check
```

If there is no error so far, we can proceed by typing:

```
sudo make install
```

To continue, we declare the following path by typing:

```
export LD_LIBRARY_PATH=/usr/local/lib
```

and we use the nano editor to open and edit the file **ld.so.conf** inside the **etc** folder

```
sudo nano /etc/ld.so.conf
```

It is **very important** to add the following line/path to the end of the file: **/usr/local/lib**

So overall, the final contents of the file will be:

```
include /etc/ld.so.conf.d/*.conf

/usr/local/lib
```

Then we save and exit the file and we execute the following command in order for the changes to take effect:

```
sudo ldconfig
```

7.2.1 PocketSphinx ??

Moving on we will need to download and install **PocketSphinx** which is lightweight speech recognition engine. PocketSphinx is a library that depends on SphinxBase and it is a very important part of the Open Source CMU Sphinx Toolkit offering great capabilities in speech recognition.

Firstly, we clone the necessary repository inside the speech_recognition directory using git:

```
git clone https://github.com/cmusphinx/pocketsphinx.git
```

Then we need to cd (change directory) inside the download folder named pocketsphinx

```
cd pocketsphinx
```

We run the executable file **autogen.sh** to create automatically all the necessary files

```
./autogen.sh
```

Then, we execute the file **configure** which is going to look for every dependency one by one that we need to have in our system.

```
./configure
```

Once it is done, we move on with the compilation of PocketSphinx typing:

```
make clean all
```

and

```
make check
```

Finally, we make the installation by typing:

```
sudo make install
```

8.2 Δημιουργία του Μοντέλου Γλώσσας

Το γλωσσικό μοντέλο αποτελείται από όλες τις φωνητικές εντολές που πρόκειται να χρησιμοποιήσουμε. Πρώτον, πρέπει να δημιουργήσουμε ένα αρχείο κειμένου (π.χ. corpus.txt) όπου θα αποθηκεύσουμε όλες τις φράσεις-κλειδιά / φωνητικές εντολές. Το παρακάτω στιγμιότυπο οθόνης εμφανίζει τις λέξεις-κλειδιά που συμπεριλάβαμε στο αρχείο μας.

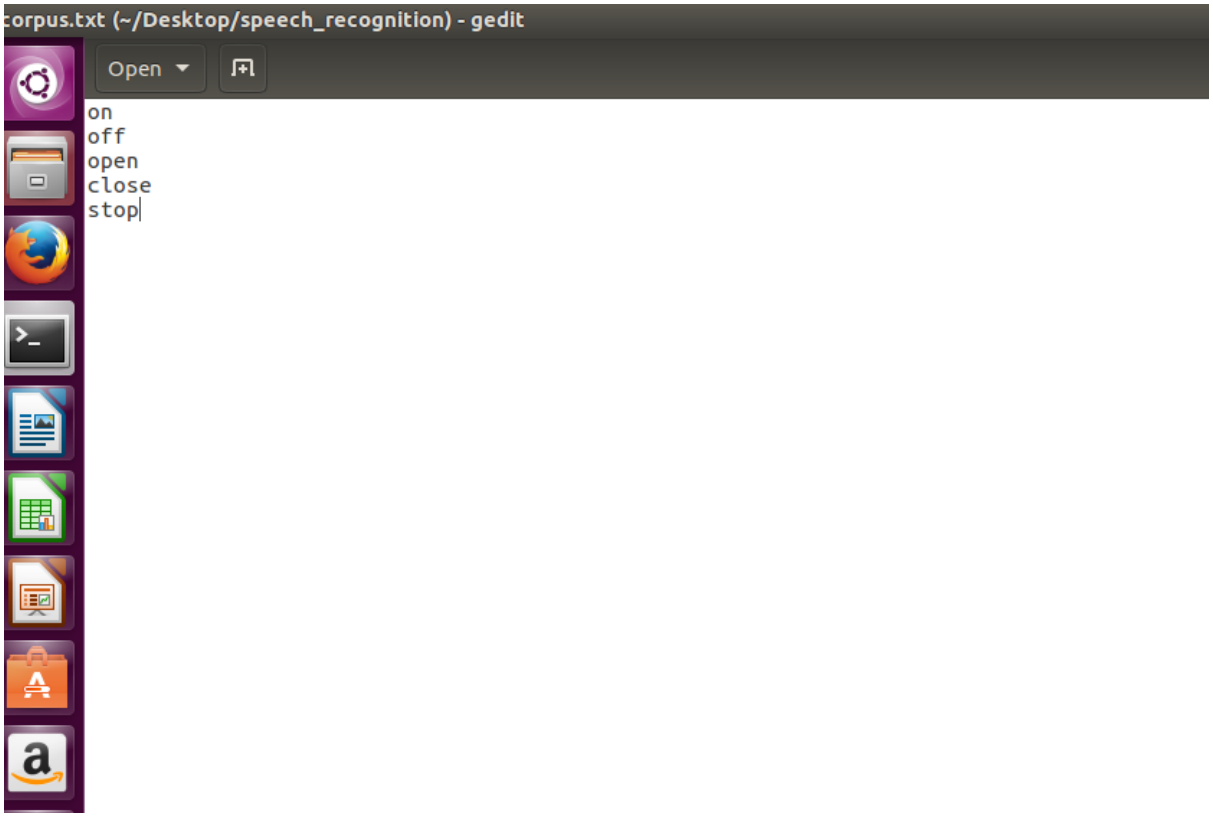
On → TURNS ON THE LAMP

Off → TURNS OFF THE LAMP

Open → OPENING THE GARAGE DOOR

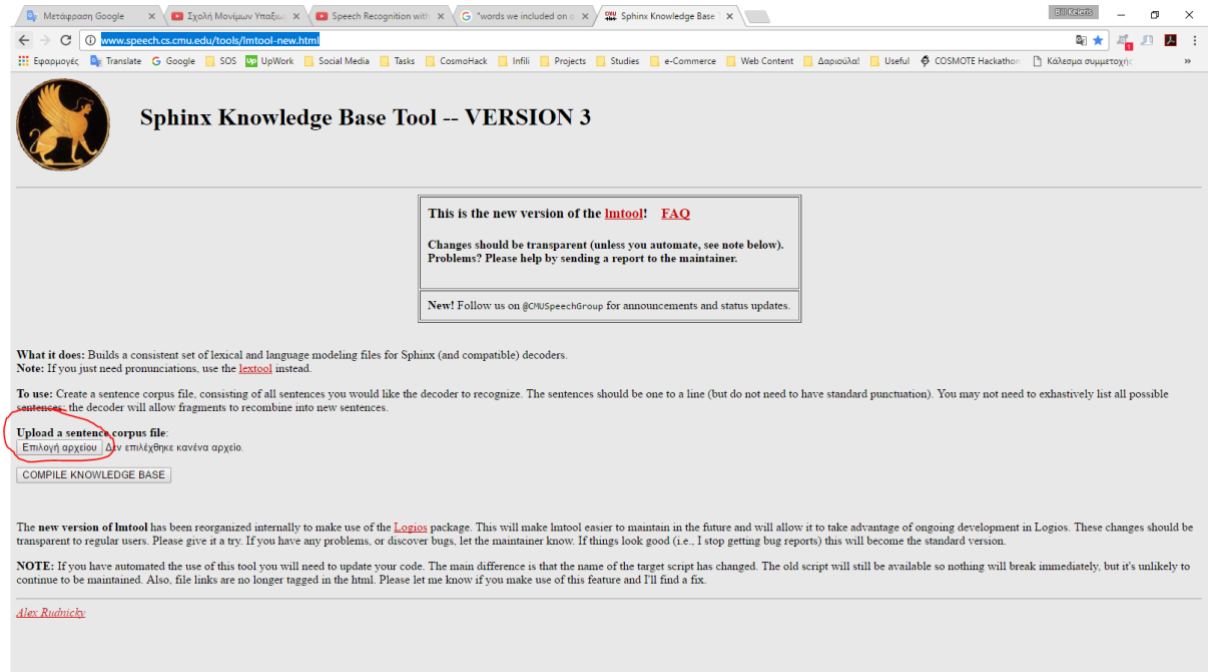
Close → CLOSING THE GARAGE DOOR

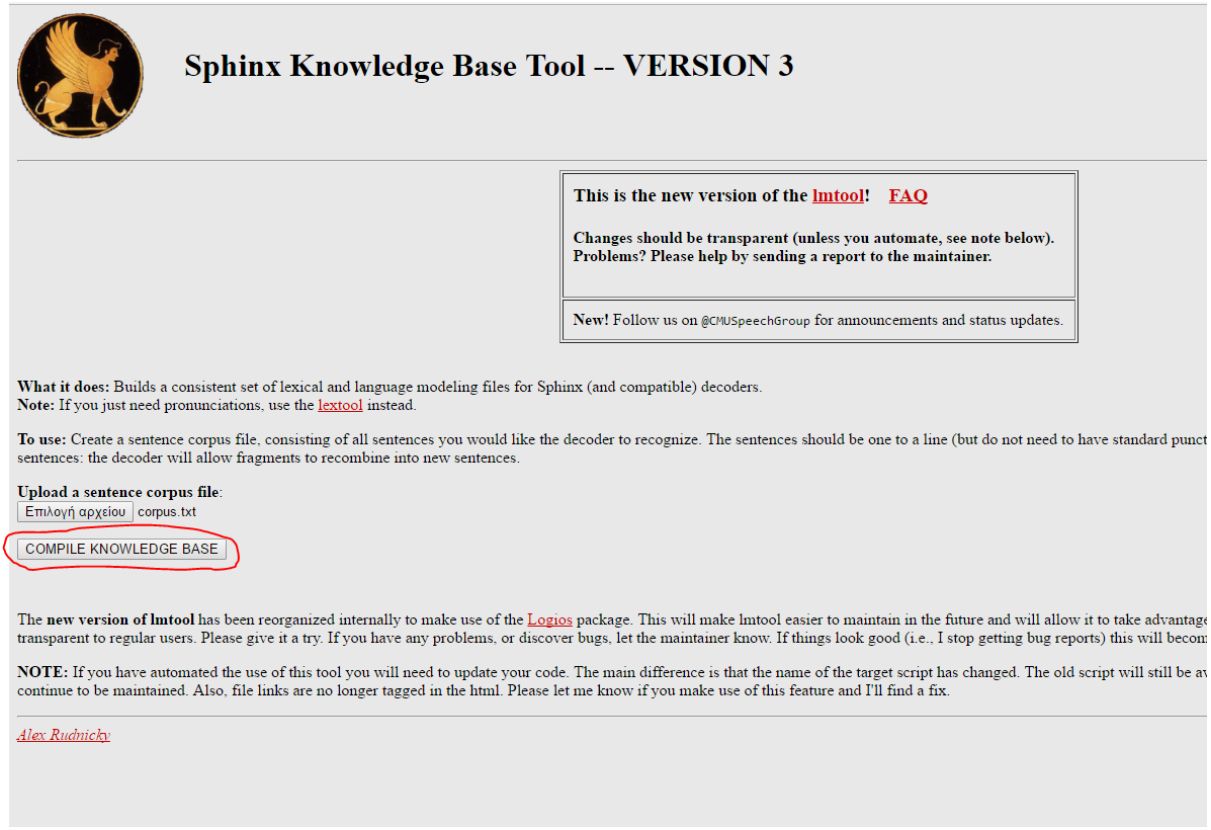
Stop → STOPS OPENING/CLOSING THE GARAGE DOOR



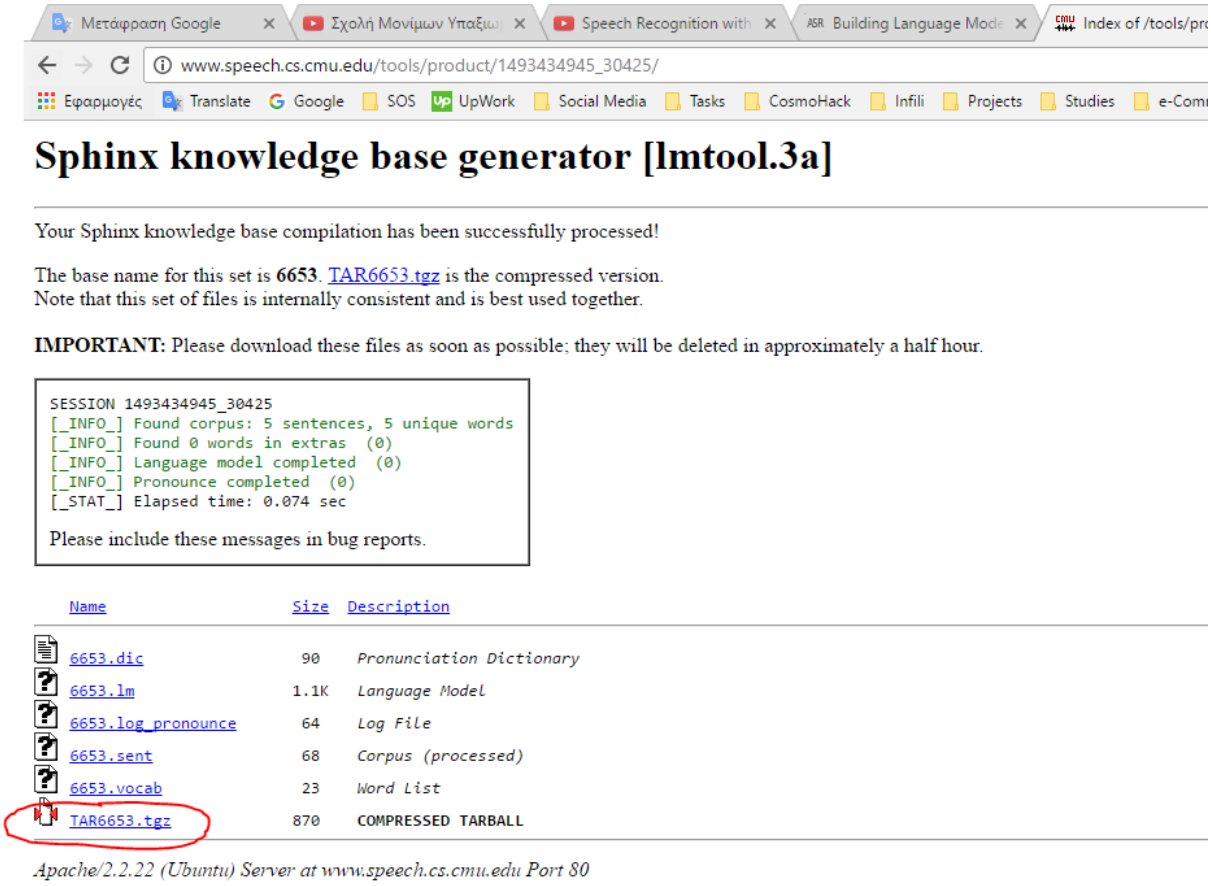
Στη συνέχεια, πρέπει να επισκεφτούμε τον ακόλουθο ιστότοπο και να ανεβάσουμε το αρχείο corpus.txt για να δημιουργήσουμε ένα ακριβές σύνολο αρχείων γλώσσας και λεξικών μοντέλων για αποκωδικοποιητές Sphinx (παρακάτω εικόνα)

http://www.speech.cs.cmu.edu/tools/lmtool-new.html





Μόλις μεταφορτώσουμε με επιτυχία το αρχείο coprpus.txt, θα μεταφερθούμε σε μια ιστοσελίδα όπου μπορούμε να κατεβάσουμε το αρχείο TAR το οποίο περιλαμβάνει όλα τα αρχεία που χρειαζόμαστε



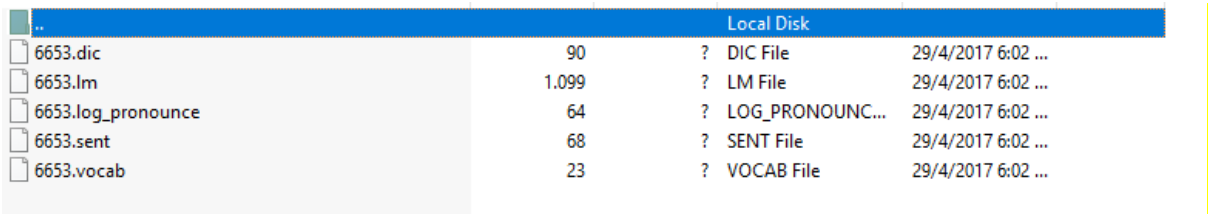
After downloading the TAR6653.tgz file to our computer, we type:

```
gunzip TAR6653.tgz
```

This will create a tar file so in order to untar it we need to type:

```
tar -xvf TAR6653.tar
```

Παρακάτω υπάρχει ένα στιγμιότυπο οθόνης με τα περιεχόμενα του αρχείου Tar. Ο αριθμός μπορεί να είναι διαφορετικός για εσάς



Τώρα είναι η ώρα να κάνουμε την πρώτη μας δοκιμασία να μιλάμε στο μικρόφωνο και να προφέρουμε τις φράσεις-κλειδιά που ορίσαμε προηγουμένως.
Για να γίνει αυτό, θα επισκεφτούμε τον κατάλογο με όλα τα εξαγόμενα αρχεία που δίνουμε στην εντολή:

```
pocketsphinx_continuous -inmic yes -lm 6653.lm -dict 6653.dic
```

Σε αυτό το χρονικό σημείο, το PocketSphinx πρέπει να είναι σε θέση να καταλάβει κάθε φράση που προφέρεται και να την εμφανίσει στο τερματικό σου.

8.3 Ανάπτυξη και δοκιμή του συστήματος αναγνώρισης φωνής

8.3.1 LiveSpeech

Το LiveSpeech είναι ένα πολύ χρήσιμο και σημαντικό πακέτο που παρέχει μια διεπαφή Python στις βιβλιοθήκες PocketSphinx και CMU Sphinxbase. Με αυτόν τον τρόπο, είμαστε σε θέση να καταγράψουμε και να αποθηκεύσουμε τη φωνητική εντολή που δίνεται από το χρήστη σε μια μεταβλητή python και στη συνέχεια να την στείλουμε ως ένα datagram στο Raspberry Pi. Για να εγκαταστήσετε το LiveSpeech, πρέπει να πληκτρολογήσετε τις ακόλουθες εντολές:

```
sudo apt-get install python-pip
```

```
pip install --upgrade pip setuptools wheel
```

```
pip install --upgrade pocketsphinx
```

Τώρα, μπορούμε να γράψουμε ένα script python (script.py) που θα χρησιμοποιήσει το μοντέλο γλώσσας που δημιουργήσαμε νωρίτερα. Όπως μπορούμε να συμπεράνουμε από τον παρακάτω κώδικα, πρώτα εισάγουμε όλες τις κατάλληλες ενότητες και στη συνέχεια δηλώνουμε το ρυθμό δειγματοληψίας και τις διαδρομές των αρχείων γλωσσικών μοντέλων. Προχωρώντας, δημιουργούμε 2 μεταβλητές (UDP_IP και UDP_PORT) οι οποίες αποθηκεύουν τη διεύθυνση IP και τη θύρα του διακομιστή python (που τρέχει στο Raspberry). Η φωνητική εντολή που δίνεται από το χρήστη αποθηκεύεται στη μεταβλητή "φράση" και στη συνέχεια χρησιμοποιούμε τη συνάρτηση str () για να την μετατρέψουμε σε μια συμβολοσειρά και να τη αποθηκεύσουμε στη μεταβλητή "εντολή". Στη συνέχεια, χρησιμοποιούμε πολλαπλές δηλώσεις για να ελέγξουμε την είσοδο (φωνητική εντολή) του χρήστη και κατά συνέπεια να στείλουμε ένα datagram που περιλαμβάνει το αντίστοιχο μήνυμα. Αυτό επιτυγχάνεται με το άνοιγμα μιας υποδοχής που επιτρέπει τη μεταφορά πακέτων UDP. Για να εκτελέσετε αυτή τη δέσμη ενεργειών Python, πρέπει να πληκτρολογήσετε την εντολή python script.py

```
import os

import socket

from pocketsphinx import LiveSpeech, get_model_path



model_path = get_model_path()



speech = LiveSpeech(
```

```
    sampling_rate=16000,
    hmm=os.path.join(model_path, 'en-us'),
    lm='/home/bill/Desktop/speech_recognition/6653.lm',
    dic='/home/bill/Desktop/speech_recognition/6653.dic'
)

UDP_IP = "192.168.2.2"
UDP_PORT = 12345

for phrase in speech:
    command = str(phrase)

    if command=="ON":
        print("LAMP IS NOW TURNED ON")
        MESSAGE = "Lamp is now ON"
        sock = socket.socket(socket.AF_INET, # Internet
        socket.SOCK_DGRAM) # UDP
        sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))

    if command=="OFF":
        print("LAMP IS NOW TURNED OFF")
        MESSAGE = "Lamp is now OFF"
        sock = socket.socket(socket.AF_INET, # Internet
        socket.SOCK_DGRAM) # UDP
        sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))

    if command=="OPEN":
        print("OPENING GARAGE DOOR...")
        MESSAGE = "Opening Garage Door"
        sock = socket.socket(socket.AF_INET, # Internet
```

```
    socket.SOCK_DGRAM) # UDP
    sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))


if command=="CLOSE":
    print("CLOSING GARAGE DOOR...")
    MESSAGE = "Closing Garage Door"
    sock = socket.socket(socket.AF_INET, # Internet
    socket.SOCK_DGRAM) # UDP
    sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))


if command=="STOP":
    print("STOPPED")
    MESSAGE = "Stop"
    sock = socket.socket(socket.AF_INET, # Internet
    socket.SOCK_DGRAM) # UDP
    sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
```

8.3.2 Server Python

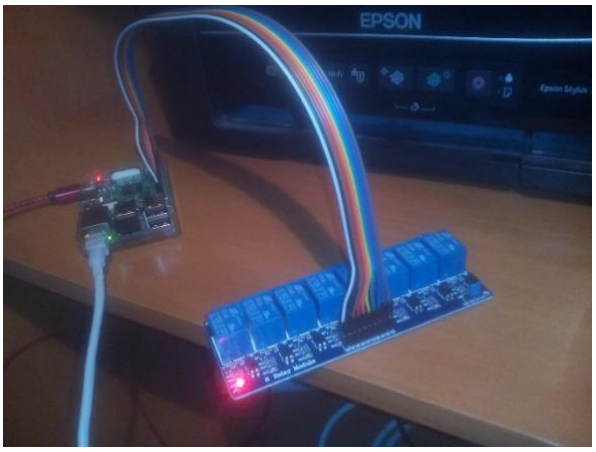
Από την άλλη πλευρά, πρέπει να δημιουργήσουμε ένα διακομιστή Python που θα τρέχει στο Raspberry Pi, το οποίο θα δεχτεί και θα χειριστεί αυτά τα datagrams. Για το λόγο αυτό, περιηγείστε σε φάκελο της επιλογής μας και δημιουργούμε το αρχείο **voiceRecognition.py**. Μετά την εισαγωγή όλων των κατάλληλων βιβλιοθηκών Python, καθορίζοντας τη διεύθυνση IP και τη θύρα του Raspberry Pi και δηλώνοντας όλους τους ακροδέκτες GPIO που πρόκειται να χρησιμοποιήσουμε, συνεχίζουμε να γράφουμε κάποιες **if statements** που θα ελέγχουν τις φωνητικές εντολές που λαμβάνονται μέσω της υποδοχής. Υπάρχουν διαφορετικές ενέργειες που εκτελούνται κάθε φορά ανάλογα με τη φωνητική εντολή που λάβατε:

VOICE COMMAND	**ACTION**	**REAL LIFE SCENARIO**
ON	Turns on the led light of the first relay channel	Turns on the Lamp
OFF	Turns off the led light of the first relay channel	Turns off the lamp
OPEN	Turns on the led light of the third relay channel. Turns off the led light of the second relay channel	Opening the Garage Door until we give the voice command: STOP

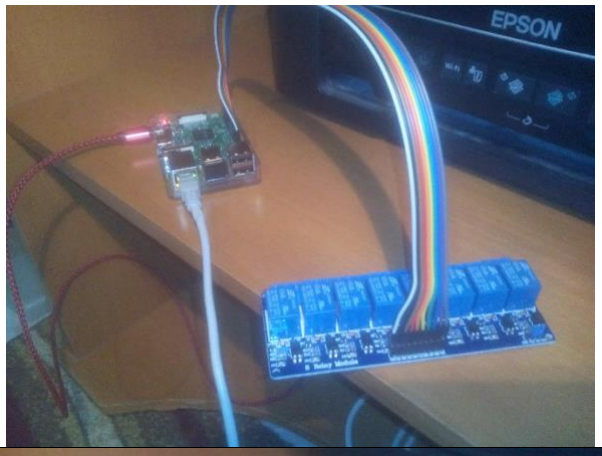
CLOSE	Turns on the led light of the second relay channel. Turns off the led light of the third relay channel	Closing the Garage Door until we give the voice command: STOP
STOP	Turns off the led light of both second and third relay channel.	Stops opening or closing the garage door

VOICE COMMAND **RELAY CHANNEL**

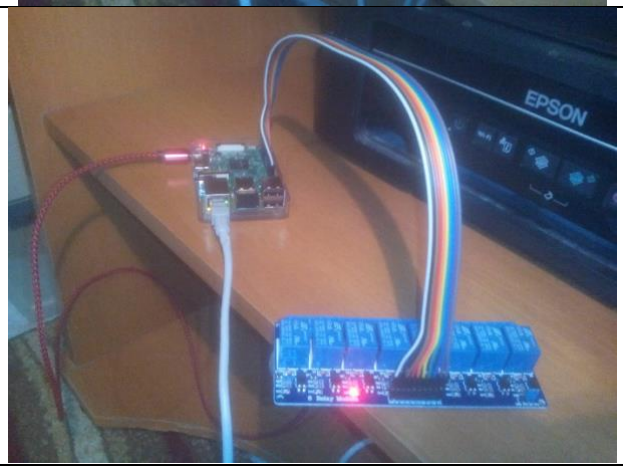
ON →

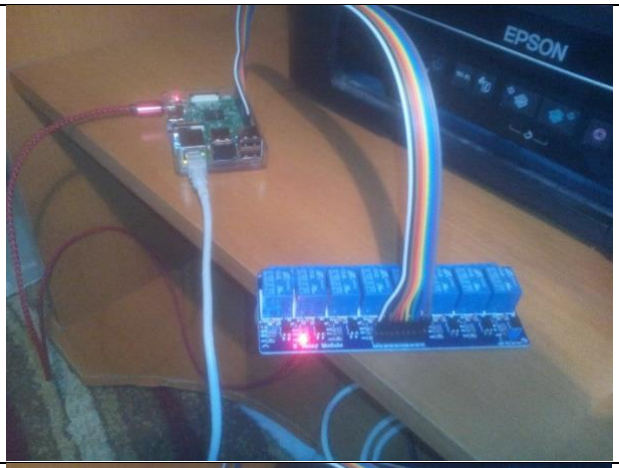
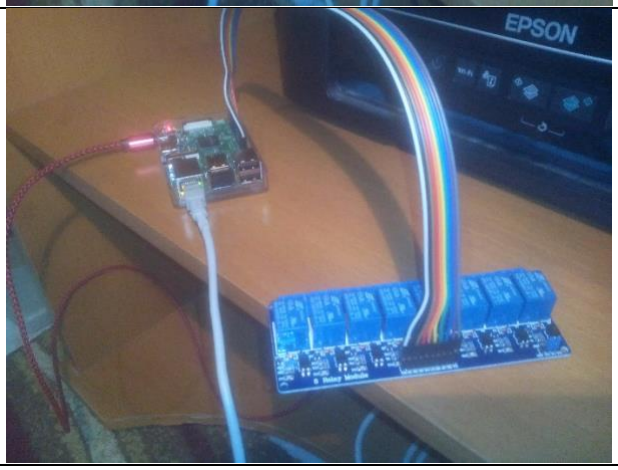


OFF →



OPEN →



CLOSE →		
STOP →		

```
#!/usr/bin/python

import socket

import sys

import os


import RPi.GPIO as GPIO

import time


GPIO.setmode(GPIO.BCM)


# init list with pin numbers

pinList = [19, 6, 13]


# loop through pins and set mode and state to 'low'
```

```
for i in pinList:
    GPIO.setup(i, GPIO.OUT)
    GPIO.output(i, GPIO.HIGH)

# time to sleep between operations in the main loop

# SleepTimeL = 2

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Bind the socket to the port
server_address = ('192.168.2.2', 12345)
print >>sys.stderr, 'starting up on %s port %s' % server_address
sock.bind(server_address)

while True:
    print >> sys.stderr, '...'
    data, address = sock.recvfrom(4096)



    if data=="Lamp is now ON":
        print >> sys.stderr, "Lamp is now ON"
        # GPIO: 19 - IN1
        GPIO.output(19, GPIO.LOW)
        # time.sleep(SleepTimeL);

    if data=="Lamp is now OFF":
        print >> sys.stderr, "Lamp is now OFF"
```

```
    # GPIO: 19 - IN1
    GPIO.output(19, GPIO.HIGH)
    # time.sleep(SleepTimeL);



# Opening Garage Door
if data=="Opening Garage Door":
    print >> sys.stderr, "Opening Garage Door..."
    # GPIO: 13 - IN3
    GPIO.output(13, GPIO.HIGH)
    # GPIO: 6 - IN3
    GPIO.output(6, GPIO.LOW)
    # time.sleep(SleepTimeL);



# Closing Garage Door
if data=="Closing Garage Door":
    print >> sys.stderr, "Closing Garage Door..."
    # GPIO: 6 - IN2
    GPIO.output(6, GPIO.HIGH)
    # GPIO: 13 - IN2
    GPIO.output(13, GPIO.LOW)
    # time.sleep(SleepTimeL);



if data=="Stop":
    print >> sys.stderr, "Stopped"
    # GPIO: 6 - IN3
    GPIO.output(6, GPIO.HIGH)
    # GPIO: 13 - IN2
```

```
GPIO.output(13, GPIO.HIGH)

# time.sleep(SleepTimeL);
```

9 Συμπέρασμα

Το έξυπνο σπίτι είναι αναμφισβήτητα το σπίτι του μέλλοντος, καθιστώντας όλες τις βασικές καθημερινές μας δραστηριότητες ικανοποιημένες με το πάτημα ενός κουμπιού ή χρησιμοποιώντας τη φωνή μας. Αυτή η διπλωματική εργασία αποσκοπούσε στην παροχή του τρόπου και των μέσων που απαιτούνται για την αυτοματοποίηση κάποιων από τις καθημερινές σας εργασίες χρησιμοποιώντας φθηνό υλικό και τεχνολογίες ανοιχτού κώδικα. Η ενεργοποίηση και απενεργοποίηση της λάμπας ή το άνοιγμα και το κλείσιμο της πόρτας γκαράζ ήταν μόνο δύο παραδείγματα των απεριόριστων δυνατοτήτων που διαθέτουμε. Ο ίδιος συλλογισμός μπορεί να εφαρμοστεί και σε άλλες συσκευές όπως έξυπνη τηλεόραση, ηχητικό σύστημα, θερμοστάτη ή ακόμα και στα παράθυρα τυφλών. Μπορείτε να έχετε πρόσβαση στο γραφικό περιβάλλον χρήστη (GUI) είτε από ένα πρόγραμμα περιήγησης (π.χ. Google Chrome) είτε από τη συσκευή smartphone (π.χ. εγκατάσταση μιας εφαρμογής Android) ή ακόμα και χρησιμοποιώντας ένα σύστημα φωνητικής αναγνώρισης (π.χ. Sphinx). Εάν ενδιαφέρουμε περισσότερο για την ασφάλεια και την αξιοπιστία, τότε θα προτιμούσαμε μια λύση όπου η επικοινωνία δημιουργείται μέσω πακέτων TCP, αλλά αν ενδιαφέρεστε περισσότερο για την ταχύτητα και το μικρό φορτίο του δικτύου τότε θα πάμε με τη λύση πακέτου UDP. Εφαρμόσαμε και τις δύο εκδόσεις, δημιουργώντας μια εφαρμογή PHP που έστειλε πακέτα TCP και μια εγγενή εφαρμογή Android που έστειλε πακέτα UDP. Προφανώς, η εφαρμογή Android είχε πολύ μικρότερη καθυστέρηση και η επικοινωνία με το Raspberry Pi ήταν στιγμιαία. Τα ίδια γρήγορα και εντυπωσιακά αποτελέσματα επιτεύχθηκαν επίσης χρησιμοποιώντας το σύστημα φωνητικών εντολών. Η Python διαδραμάτισε βασικό ρόλο κατά τη διάρκεια της διαδικασίας ανάπτυξης, καθώς μας επέτρεψε να δημιουργήσουμε ένα αποδοτικό και αξιόπιστο διακομιστή Raspberry Pi που θα επιτρέπει την ταχεία μετάδοση των πακέτων UDP (datagrams). Επιπλέον, η Python μας επέτρεψε να ενσωματώσουμε την αναγνώριση φωνής σε μια δέσμη ενεργειών μέσω της βιβλιοθήκης LiveSpeech καθώς και να καθορίσουμε τη διέλευση του ρεύματος μέσω του σωστού καναλιού αναμετάδοσης χρησιμοποιώντας πολλές διαφορετικές "αν δηλώσεις". Οι δυνατότητες αυτών των συστημάτων είναι απεριόριστες και πολύ καινοτόμες. Η ευρεία διαθεσιμότητα και η δυνατότητα μεταφοράς του Διαδικτύου σε συνδυασμό με το Διαδίκτυο των πραγμάτων (IoT) μας έφεραν στην ψηφιακή εποχή, όπου μπορούμε να αλληλεπιδράσουμε με οποιαδήποτε συσκευή, χωρίς να χρειάζονται πολλά διαφορετικά τηλεχειριστήρια, χωρίς πολύπλοκες εγκαταστάσεις και με την την ίδια ευκολία μιλάμε ή χειριζόμαστε ένα smartphone για οποιονδήποτε άλλο σκοπό.

10 Αναφορές

1. Samuel Greengard (2015). The Internet of Things. Retrieved from https://books.google.gr/books?id=oyyyBwAAQBAJ&printsec=frontcover&dq=internet+of+things&hl=el&sa=X&ved=0ahUKEwjZpoDxgtDTAhXBohQKHZoGBt4Q6AEIITAA#v=onepage&q=internet%20of%20things&f=false

2. Agus Kurniawan (2016). Getting Started with Raspberry Pi 3. Retrieved from https://books.google.gr/books?id=uhK7CwAAQBAJ&printsec=frontcover&dq=raspberry+pi

+3&hl=el&sa=X&ved=0ahUKEwiOkquHhNDTAhUSfFAKHR7oA88Q6AEIITAA#v=onepage&q=raspberry%20pi%203&f=false

3. CMUSphinx Guide for Developers. (n.d.). Retrieved May 01, 2017, from http://cmusphinx.sourceforge.net/wiki/tutorial

4. Raspbian Operating System. (n.d.). Retrieved May 01, 2017, from https://www.raspbian.org/RaspbianDocumentation

5. Adam Gerber, Clifton Craig (2015). Learn Android Studio Build Android Apps Quickly and Effectively. Retrieved from https://books.google.gr/books?id=yFEnCgAAQBAJ&printsec=frontcover&dq=android+studio&hl=el&sa=X&ved=0ahUKEwjipofNhNDTAhWNLVAKHWuFDCAQ6AEIUjAG#v=onepage&q=android%20studio&f=false

6. GPIO Raspberry Pi Hardware. (n.d.). Retrieved May 01, 2017, from https://www.raspberrypi.org/documentation/usage/gpio/

7. Raspberry Pi Basics. (n.d.). Retrieved May 01, 2017, from https://www.raspberrypi.org/documentation/

8. Kai-Fu Lee (1989). Automatic Speech Recognition, The Development of the SPHINX System. Retrieved from https://books.google.gr/books?id=KVwFCAAAQBAJ&printsec=frontcover&dq=voice+recognition&hl=el&sa=X&ved=0ahUKEwjdx6CthtDTAhWHJFAKHWkCBHcQ6AEIKDAB#v=onepage&q=voice%20recognition&f=false

9. Meet Android Studio. (n.d.). Retrieved May 01, 2017, from https://developer.android.com/studio/intro/index.html

10. Brandon Rhodes and John Goerzen (2014). Foundations of Python Network Programming. Retrieved from https://books.google.gr/books?id=oAkZBQAAQBAJ&pg=PA53&dq=python+socket&hl=el&sa=X&ved=0ahUKEwiu57e6rtDTAhVFLFAKHaLwAFwQ6AEIITAA#v=onepage&q=python%20socket&f=false

11. Wikipedia (2017). Internet of things. Retrieved from https://en.wikipedia.org/wiki/Internet_of_things

12. Alessandro Bassi, Martin Bauer, Martin Fiedler, Thorsten Kramp, Rob van Kranenburg, Sebastian Lange, Stefan Meissner (2013). Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model. Retrieved from https://link.springer.com/book/10.1007%2F978-3-642-40403-0

13. Friedemann Mattern and Christian Floerkemeier (2015). From the Internet of Computers to the Internet of Things. Retrieved from http://vs.inf.ethz.ch/publ/papers/Internet-of-things.pdf

14. Ovidiu Vermesan and Peter Friess (2014). Internet of Things – From Research and Innovation to Market Deployment. Retrieved from http://internet-of-things-research.eu/pdf/IoT-From%20Research%20and%20Innovation%20to%20Market%20Deployment_IERC_Cluster_eBook_978-87-93102-95-8_P.pdf

15. Dave Evans (2011). The Internet of Things, How the Next Evolution of the Internet Is Changing Everything. Retrieved from http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

16. Hannelore Marginean, Daniel Karzel, Tuan-Si Tran (2016). A Reference Architecture for the Internet of Things. Retrieved from https://www.infoq.com/articles/internet-of-things-reference-architecture

17. Hakima Chaouchi (2010). The Internet of Things Connecting Objects. Retrieved from https://books.google.gr/books?id=EGNm4iT8TC8C&printsec=frontcover&dq=internet+of+things&hl=el&sa=X&redir_esc=y#v=onepage&q=internet%20of%20things&f=false

18. Adrian McEwen & Hakim Cassimally (2014). Designing the Internet of Things. Retrieved from https://books.google.gr/books?id=iYkKAgAAQBAJ&printsec=frontcover&dq=internet+of+things&hl=el&sa=X&redir_esc=y#v=onepage&q=internet%20of%20things&f=false

19. Sergey Balandin, Yevgeni Koucheryavi, Sergey Andreev (2015). Internet of Things, Smart Spaces and Next Generation Networks and Systems. Retrieved from

https://books.google.gr/books?id=XBNcCgAAQBAJ&printsec=frontcover&dq=internet+of+things&hl=el&sa=X&redir_esc=y#v=onepage&q=internet%20of%20things&f=false

20. Timothy L. Warner (2014). Hacking Raspberry Pi. Retrieved from https://books.google.gr/books?id=3TUkAQAAQBAJ&printsec=frontcover&dq=raspberry+pi&hl=el&sa=X&redir_esc=y#v=onepage&q=raspberry%20pi&f=false

21. Peter Membrey and David Hows (2013). Learn Raspberry Pi with Linux. Retrieved from https://books.google.gr/books?id=N2QTLjo0NSUC&pg=PA79&dq=raspberry+pi&hl=el&sa=X&redir_esc=y#v=onepage&q=raspberry%20pi&f=false

22. Milan Sigmund (2003). Voice Recognition by Computer. Retrieved from https://books.google.gr/books?id=B9VuCBBYzJ4C&printsec=frontcover&dq=voice+recognition&hl=el&sa=X&redir_esc=y#v=onepage&q=voice%20recognition&f=false

23. Marziah Karch (2010). Android for Work Productivity for Professionals. Retrieved from https://books.google.gr/books?id=6tLAyQLSzG0C&printsec=frontcover&dq=android&hl=el&sa=X&redir_esc=y#v=onepage&q=android&f=false

24. Zheng-Hua Tan and Borge Lindberg (2008). Automatic Speech Recognition on Mobile Devices and over Communication Networks. Retrieved from https://books.google.gr/books?id=IftV7maWhSMC&printsec=frontcover&dq=voice+recognition&hl=el&sa=X&redir_esc=y#v=onepage&q=voice%20recognition&f=false

25. Christian Benvenuti (2006). Understanding Linux Network Internals. Retrieved from https://books.google.gr/books?id=ALapr7CvAKkC&printsec=frontcover&dq=linux+networking&hl=el&sa=X&redir_esc=y#v=onepage&q=linux%20networking&f=false

26. Tony Bautts, Terry Dawson, Gregor N. Purdy (2005). Linux Network Administrator's Guide. Retrieved from https://books.google.gr/books?id=T8V02u2jDP0C&printsec=frontcover&dq=linux+networking&hl=el&sa=X&redir_esc=y#v=onepage&q=linux%20networking&f=false

27. Shane Conder and Lauren Darcey (2010). Android Wireless Application Development. Retrieved from

http://jfod.cnam.fr/seja/supports/biblio/Android%20Wireless%20Application%20Development%202nd.pdf

28. Agus Kurniawan (2014). Raspberry Pi I/O Programming Using Python. Retrieved from https://books.google.gr/books?id=3bWjBAAAQBAJ&printsec=frontcover&dq=raspberry+pi&hl=el&sa=X&redir_esc=y#v=onepage&q=raspberry%20pi&f=false

29. Matt Anniss (2014). How does Voice Recognition Work?. Retrieved from https://books.google.gr/books?id=Jc7nAgAAQBAJ&printsec=frontcover&dq=voice+recognition&hl=el&sa=X&redir_esc=y#v=onepage&q=voice%20recognition&f=false

30. Mohammed M. Alani (2014). Guide to OSI and TCP/IP Models. Retrieved from https://books.google.gr/books?id=PRi5BQAAQBAJ&printsec=frontcover&dq=osi+model&hl=el&sa=X&redir_esc=y#v=onepage&q=osi%20model&f=false

31. Budi Kurniawan (2015). Java for Android. Retrieved from https://books.google.gr/books?id=QjZ7CgAAQBAJ&printsec=frontcover&dq=java+android&hl=el&sa=X&redir_esc=y#v=onepage&q=java%20android&f=false

32. Jay Lacroix (2015). Mastering Linux Network Administration. Retrieved from https://books.google.gr/books?id=a_OoCwAAQBAJ&printsec=frontcover&dq=linux+networking&hl=el&sa=X&redir_esc=y#v=onepage&q=linux%20networking&f=false

33. Gokhan Kurt (2015). Raspberry Pi Android Projects. Retrieved from https://books.google.gr/books?id=O9hOCwAAQBAJ&printsec=frontcover&dq=raspberry+pi+android&hl=el&sa=X&redir_esc=y#v=onepage&q=raspberry%20pi%20android&f=false

34. Levi Balling, Dario Bosnjak, Christopher Johson and Todd Rogers (2012). Smart Home Computer Engineering Final Project. Retrieved from http://www.eng.utah.edu/~cs3992/archive/2012/SmartHomeFinalReport.pdf

35. Steven Goodwin (2015). Smart Home Automation with Linux and Raspberry Pi. Retrieved from https://books.google.gr/books?id=Zls0TZcopvEC&printsec=frontcover&dq=home+automation+raspberry+pi&hl=el&sa=X&redir_esc=y#v=onepage&q=home%20automation%20raspberry%20pi&f=false