Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Τίτλος: **Ανάπτυξη Android εφαρμογής για έλεγχο μέσω internet ενός microcontroller που θα ελέγχει και θα αλλάζει την κατάσταση λειτουργίας (state) ηλεκτρικής συσκευής.**

Ροδολάκης Κωνσταντίνος (ΑΜ:1348)

Επιβλέπων καθηγητής: Βλησίδης Ανδρέας

Επιτροπή αξιολόγησης: Παναγιωτάκης Σπύρος, Στρατάκης Δημήτρης

Ημερομηνία παρουσίασης: 6 Νοεμβρίου 2014

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστώ τον καθηγητή μου Δρ. Ανδρέα Βλησίδη, ο οποίος ήταν και ο επιβλέπων της πρακτικής μου καθώς και ο εισηγητής της πτυχιακής μου εργασίας , με του οποίου την υποστήριξη και καθοδήγηση κατάφερα να την διεκπεραιώσω.

Έπειτα ευχαριστώ όσους με βοήθησαν πνευματικά και υλικά παρέχοντάς μου τα απαραίτητα για την ολοκλήρωση της πτυχιακής.

Τέλος θα ήθελα να ευχαριστήσω το εργαστήριο Βιομηχανικών Αυτοματισμών και Πληροφοριακών Δικτύων για την παραχώρηση του απαραίτητου υλικού για την εκπόνηση της πτυχιακής.

Abstract

In the present thesis, an application will be created using the **Arduino IDE**, which will then run in a server consisted of an **Arduino UNO R3**, and a compatible **Ethernet Shield** (based on the **W5100 chipset)**, with wireless internet access via **Wi-Fi** using the wireless router **TL-WR703N** in client mode. The way to make the appropriate adjustments in the domestic router's firmware as well as in the **TL-WR703N,** will be analyzed. Later on, an application will be designed and written, using the **ADT** (**Android Developing Tools**) plugin for Eclipse **IDE** and the latest **Android SDK 4.2.2** (API level: 17), where the user will be able to **check** the state (on/off) of all the electric appliances connected on the Arduino UNO server, as well as **altering** their state.

Σύνοψη

Στην παρούσα πτυχιακή, θα δημιουργηθεί μια εφαρμογή κάνοντας χρήση του **Arduino IDE**, η οποία θα τρέχει σε ένα server αποτελούμενο από ένα **Arduino UNO R3** και ένα συμβατό **Ethernet Shield** (βασισμένο στο **W5100 chipset**), με ασύρματη πρόσβαση στο internet **μέσω Wi-F**i κάνοντας χρήση του ασύρματου δρομολογητή **TL-WR703N** σε λειτουργία client. Θα αναλυθεί πως θα γίνουν οι κατάλληλες ρυθμίσεις από τον οικιακό δρομολογητή καθώς και στο **TL-WR703N** ώστε να μπορούμε να έχουμε πρόσβαση στον server μας ανά πάση στιγμή από παντού. Κατόπιν κάνοντας χρήση του **ADT** (**Android Developing Tools**) plugin για το **Eclipse IDE** θα σχεδιαστεί και θα γραφτεί μια **εφαρμογή** για Android, κάνοντας χρήση του τελευταίου **Android SDK 4.2.2** (API level: 17), από την οποία θα μπορεί να γίνει **έλεγχος** σε τι κατάσταση (ανοιχτές/κλειστές) βρίσκονται οι ηλεκτρικές συσκευές που βρίσκονται πάνω στον server **Arduino UNO R3**, καθώς και να **μεταβληθεί** η κατάστασή τους.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

1. Εισαγωγή

Η εξέλιξη της τεχνολογίας σε συνδυασμό με την εκτενή εξάπλωση των **Open Source Software** και **Hardware platforms** έχουν κάνει προσιτή την ανάπτυξη εφαρμογών και συσκευών στο ευρύ κοινό και μάλιστα με πολύ χαμηλό κόστος. Στην παρούσα πτυχιακή χρησιμοποιώντας έναν **Microcontroller Arduino UNO** και κάποια περιφερειακά, θα κατασκευάσουμε μια συσκευή η οποία έχοντας ασύρματη πρόσβαση στο Internet μέσω Wi-Fi, θα μπορεί να ελέγχει ακόμα και ολόκληρο το ηλεκτρικό δίκτυο ενός σπιτιού από την άλλη άκρη της γης με τη βοήθεια μιας εφαρμογής **Android** για κινητές συσκευές η οποία κατασκευάστηκε με ελεύθερο λογισμικό.

1.1 Περίληψη

Η ανοιχτού κώδικα ηλεκτρονική πλατφόρμα πρωτότυπων κυκλωμάτων **Arduino** με το ευέλικτο λογισμικό και υλικό της, έχει δώσει τη δυνατότητα σε σχεδιαστές, χομπίστες και σε οποιονδήποτε έχει την ικανότητα, την γνώση και την όρεξη, να δημιουργήσει διαδραστικά αντικείμενα και περιβάλλοντα. Το Arduino μπορεί να συνδυαστεί με πλήθος αισθητήρων καθώς και με άλλες υπομονάδες επέκτασης όπως αυτή που θα χρησιμοποιηθεί στην υπάρχουσα πτυχιακή (**Ethernet Shield**), σε συνδυασμό με άλλου τύπου ηλεκτρονικό και μη υλικό, ώστε να υλοποιηθεί επιτυχώς το τελικό προϊόν.

Το **Android**, ένα ανοιχτού κώδικα λογισμικό το οποίο δίνει ζωή σε εκατοντάδες εκατομμύρια κινητές συσκευές σε περισσότερες από 190 χώρες , είναι η πιο ευρέως εγκατεστημένη πλατφόρμα και συνεχίζει να επεκτείνεται καθημερινά με εκατομμύρια ενεργοποιήσεις κινητών συσκευών. Το Android παρέχει μια εξαιρετική πλατφόρμα για τον σχεδιασμό και τη δημιουργία εφαρμογών, καθώς επίσης και έναν εύκολο τρόπο διανομής τους μέσα από το δικό του ηλεκτρονικό κατάστημα (Play Store).

Στην παρούσα πτυχιακή, κάνοντας χρήση του **Arduino IDE**, θα γραφτεί μια εφαρμογή κάνοντας χρήση των κατάλληλων βιβλιοθηκών, η οποία θα τρέχει μέσα στον βασισμένο σε **Arduino UNO** Server. Ο Server αποτελείται από ένα **Arduino Uno** και ένα συμβατό **Ethernet Shield** βασισμένο στο **Chipset W5100**, το οποίο είναι και το κομμάτι που δίνει τις βασικές δικτυακές δυνατότητες στον server μας. Κατόπιν χρησιμοποιώντας τον ασύρματο δρομολογητή **TL-WR703N** σε λειτουργία **client**, θα αναλυθεί πως θα δώσουμε στον server μας τη δυνατότητα να συνδέεται **ασύρματα στο Internet**, χωρίς να χρειάζεται να είναι κοντά σε κάποιο ενσύρματο δρομολογητή καθώς και όλες τις απαραίτητες ρυθμίσεις που πρέπει να γίνουν στο οικιακό μας δίκτυο για να μπορεί ο server μας να είναι μονίμως προσπελάσιμος από παντού, χρησιμοποιώντας υπηρεσίες **DDNS**.

Τέλος, χρησιμοποιώντας το **ADT** (**Android Developing Tools**) plugin για το **Eclipse IDE**, θα σχεδιαστεί και θα γραφτεί μια εφαρμογή για κινητές συσκευές Android, κάνοντας χρήση του τελευταίου **Android SDK 4.2.2** (**API level: 17**), από την οποία θα μπορεί να γίνει **έλεγχος** σε τι κατάσταση (ανοιχτές / κλειστές) βρίσκονται οι ηλεκτρικές συσκευές που βρίσκονται πάνω στον server Arduino server Arduino UNO, καθώς και να **μεταβληθεί η κατάστασή τους.**

1.2 Κίνητρο για την Διεξαγωγή της Εργασίας

Παρακολουθώντας τις τεχνολογικές εξελίξεις στον τομέα των έξυπνων σπιτιών καθώς και όλες τις ακριβές και ασύμφορες προτάσεις που κυκλοφορούν στην αγορά στον συγκεκριμένο τομέα, κρίθηκε σωστό να βρεθεί μια πολύ πιο οικονομική και προσιτή για τον μέσο καταναλωτή λύση.

1.3 Σκοπός και Στόχοι Εργασίας

- Δημιουργία αξιόπιστου κώδικα για τον **Server**.
- Ευκρινής και αναλυτική περιγραφή διαδικασίας συνδεσιμότητας συσκευών και δικτύου.
- Αναλυτική περιγραφή όλου του απαραίτητου υλικού και λογισμικού.
- Σωστή σχεδιαστική προσέγγιση δημιουργίας του **Interface** για την **Android Εφαρμογή**, βασισμένη στα **Holo Guidelines**.
- Δημιουργία Android Εφαρμογής η οποία θα είναι συμβατή με μεγάλο εύρος συσκευών.
- Απρόσκοπτη λειτουργία Εφαρμογής Android..
- Σταθερή λειτουργία του συστήματος (**Arduino Server, Ethernet Shield, TL-WR703N**).
- Απόλυτος έλεγχος της λειτουργίας του **Server** μέσα από την **Android Εφαρμογή**.

1.4 Δομή Εργασίας

Στα κεφάλαια που ακολουθούν θα δούμε αναλυτικά το υλικό, το λογισμικό και την προσέγγιση που υπήρξε για την επίτευξη του τελικού αποτελέσματος. Πιο συγκεκριμένα, στο **Κεφάλαιο 2** θα δούμε μια μικρή περίληψη της μεθοδολογίας υλοποίησης του project, ενώ στο **Κεφάλαιο 3** θα κάνουμε μια αναφορά στο τι συμβαίνει στα τεχνολογικά δρώμενα, όσον αφορά το κομμάτι με το οποίο ασχολείται αυτή η διπλωματική. Συνεχίζοντας, στο **Κεφάλαιο 4** θα γίνει μια συνοπτική ανάλυση των απαιτήσεων του Συστήματος. Προχωρώντας στο **Κεφάλαιο 5**, θα δούμε πρώτα αναλυτικά όλο το απαραίτητο υλικό που χρησιμοποιήθηκε, ενώ στη συνέχεια θα δούμε το λογισμικό του οποίου έγινε χρήση. Στο **Κεφάλαιο 6** θα ακολουθήσει η μέθοδος σύνδεσης του υπολογιστή με τον microcontroller σε συνδυασμό με το Ethernet Shield, καθώς και ποια βήματα ακολουθήθηκαν για τον επιτυχή προγραμματισμό του. Ακόμα θα γίνει ανάλυση στο πρόγραμμα που δημιουργήθηκε για να τρέξει στον Server/Microcontroller, καθώς και τις λειτουργίες του. Έπειτα στο **Κεφάλαιο 7**, θα γίνει αναλυτική περιγραφή των ρυθμίσεων που έγιναν στο TL-WR703N, στο WRT54G αλλά και στο TG585, ώστε να υπάρχει απρόσκοπτη σύνδεση με τον Arduino Server και σωστή δρομολόγηση των αιτήσεων προς αυτόν. Επιπλέον θα γίνει αναφορά στο πως χρησιμοποιήθηκε η υπηρεσία DDNS ώστε να γίνει προσπελάσιμος ο Server μέσω Internet με ευκολία. Στο **Κεφάλαιο 8** που ακολουθεί, θα δειχθούν αναλυτικά τα βήματα που χρειάστηκαν για να στηθεί το προγραμματιστικό περιβάλλον Eclipse. Επίσης θα αναλυθεί η δημιουργία του Project, εκτενής παρουσίαση των Building Blocks που χρησιμοποιήθηκαν και της λειτουργίας τους. Ακόμα θα αναλυθούν τα Activities και το Interface της εφαρμογής καθώς και πως αυτή λειτουργεί αλληλεπιδρώντας με τον Arduino Server. Στο **τελευταίο Κεφάλαιο**, θα γίνει αναφορά σε πιθανές μελλοντικές επεκτάσεις καθώς και στη χρησιμότητα του συστήματος που δημιουργήθηκε.

2. Μεθοδολογία Υλοποίησης

Το πρώτο μέλημα ξεκινώντας την επίλυση του ζητούμενου της παρούσας πτυχιακής, ήταν ο έλεγχος του **Arduino UNO R3**. Έπρεπε να υπάρξει πριν από όλα τα αλλά έντονη ενασχόληση με το πώς προγραμματίζεται ο microcontroller μέσα από το **Arduino IDE 1.0.4**, και πως μπορεί να γίνει ο έλεγχος των εξόδων του σειριακά, πριν περάσουμε στο δικτυακό κομμάτι. Έπειτα έγινε ο πειραματισμός με sensors όπως το LM35 για μέτρηση της θερμοκρασίας, για τη δημιουργία λογικών συνθηκών ελέγχου των εξόδων του **Arduino UNO**, βάσει της εισόδου που έπαιρνε από τον αισθητήρα.

Κατόπιν έγινε ο συνδυασμός του Arduino UNO με το **Ethernet Shield HanRun HR911105A 10/49**, περνώντας τη σύνδεση στο φυσικό επίπεδο (**Ethernet**). Πλέον η σύνδεση για να δοθούν οι εντολές στο **Arduino UNO** θα βασιζόταν σε ένα καλώδιο **Ethernet** που συνέδεε, μέσω θυρών **RJ45**, το Laptop με το βασισμένο στο **W5100 chipset**, Ethernet Shield. Διαβάζοντας εκτενώς τα απαραίτητα για τη χρήση της βιβλιοθήκης του chipset, δημιουργήθηκε ο ελάχιστος κατάλληλος κώδικας για τον απευθείας έλεγχο, μιας μόνο εξόδου του Arduino UNO, μέσω εντολών **GET** κάνοντας χρήση του **HTTP** (Hypertext Transfer Protocol) και του browser **Mozilla Firefox v.23**. Το θέμα που δημιουργήθηκε αρχικά, ήταν ότι το Laptop δεν μπορούσε να επικοινωνήσει με τον Server. Το πρόβλημα λύθηκε αφαιρώντας την αυτόματη απόδοση **IP** μέσω **DHCP** στο laptop, και με την απόδοση **Static IP** στην κάρτα δικτύου του.

Έπειτα, έγινε κατάλληλη μετατροπή στον κώδικα ώστε να υποστηρίζει πλέον 8 εξόδους, δύο από τις οποίες παρέχουν και αναλογική λειτουργία, καθώς και η απαραίτητη προσθήκη για την απεικόνιση της κατάστασης της κάθε εξόδου στο server ώστε να μπορέσει να διαβαστεί αργότερα από την εφαρμογή.

Φτάνοντας στο σημείο αυτό είχε έρθει η στιγμή της απόφασης της τελικής σύνδεσης με το internet. Έπειτα από ώριμη σκέψη κρίθηκε ακατάλληλη για το σκοπό της εφαρμογής η απευθείας ενσύρματη σύνδεση σε δρομολογητή/modem, καθώς αυτό θα απαιτούσε ύπαρξη τηλεφωνικής εγκατάστασης πολύ κοντά στο σημείο της εγκατάστασης του εξοπλισμού και θα δυσχέραινε την χρήση του όπως θα μείωνε και τη χρησιμότητά του. Έπειτα από έρευνα αγοράς, βρέθηκε η λύση του πολύ φτηνού **TP-LINK TL-WR703N version 1.6**, ενός δρομολογητή με δυνατότητες ασύρματης δικτύωσης σε λειτουργία client, ο οποίος συμπεριλήφθηκε εν τέλει στο σύνολο του διαθέσιμου εξοπλισμού.

Συνδέοντας για πρώτη φορά το TL-WR703N, και εισερχόμενοι στο μενού της συσκευής της συσκευής, βλέπουμε ότι η γλώσσα χρήσης είναι στα κινέζικα και δεν υπάρχει εναλλακτική επιλογή. Κάνοντας "mouse over" πάνω από τους συνδέσμους του menu, βλέπουμε κάτω στον browser μας τον σύνδεσμο τον οποίο ανοίγει ο οποίος μας προϊδεάζει σε τι μενού θα πάμε καθώς η γλώσσα του url είναι στα αγγλικά. Επειδή το γλωσσικό εμπόδιο θα δυσκόλευε πολύ τη συνέχεια της πτυχιακής καθώς και της κατανόησης από τον αναγνώστη, έγινε έρευνα και βρέθηκε από παρόμοιο μοντέλο – το **TL-MR3020** - το **firmware 3.12.11 Build 120320 Rel.42138n** το οποίο μας παρείχε αγγλικό μενού όμοιο με το κινέζικο της συσκευής. Με το ρίσκο να καταστραφεί η συσκευή αποφασίσθηκε η εγκατάσταση του νέου firmware στο μοντέλο του εξοπλισμού. Αφού η έκβαση ήταν επιτυχής και με ένα μενού πλέον στην αγγλική γλώσσα, ξεκίνησε η ρύθμιση του TL-WR703N ώστε να δουλέψει συνδυαστικά με το Ethernet Shield.

Αφού τέθηκε σε λειτουργία client το TL-WR703N, συνδέθηκε μέσω **Wi-Fi** με **WPA2** μέθοδο κρυπτογράφησης, στο τοπικό οικιακό δίκτυο και στο router **Linksys WRT54G v.5**. Το WRT54G τρέχει **custom** firmware DD-WRT και συγκεκριμένα το **firmware**: **DD-WRT v24-sp2 (10/10/09) micro**. Στο WRT54G έγιναν επίσης οι απαραίτητες ρυθμίσεις για την σωστή διευθυνσιοδότηση του WR703N καθώς και το άνοιγμα των απαραίτητων θυρών. Επίσης έγιναν οι απαραίτητες ρυθμίσεις για μια δωρεάν **DDNS** υπηρεσία ώστε να είναι προσπελάσιμος ανά πάσα στιγμή ο server χωρίς να γνωρίζουμε την IP του. Τέλος το WRT54G συνδέεται και αυτό με τη σειρά του μέσω Ethernet καλωδίου σε έναν δρομολογητή **Thomson TG585 v7** με **έκδοση firmware 7.4.2.7** το οποίο είναι και η τελευταία συσκευή του συστήματος, και είναι η συσκευή που συνδέει το σύστημα με το internet **μέσω ADSL 2+ γραμμής**. Όπως και στο WRT54G, έτσι και εδώ, έπρεπε να γίνουν οι απαραίτητες ρυθμίσεις και να ανοίξουν οι κατάλληλες θύρες ώστε να γίνει προσπελάσιμος ο server μέσω internet.

Τέλος κάνοντας χρήση του **Eclipse v.4.2.1** σε συνδυασμό με το **Android Development Toolkit v.22.0.0** σχεδιάστηκε και γράφτηκε η εφαρμογή Android που θα ελέγχει το server του συστήματος.

3. State of the Art in Home Automation

Εδώ και πολλά χρόνια βλέπουμε τις εταιρείες να προσπαθούν να εμπορευματοποιήσουν την ιδέα ενός smart home control system, με διάφορες ιδέες και προσεγγίσεις.

Μόλις πρόσφατα η Google εξαγόρασε τη Nest, κατασκευαστή ενός έξυπνου θερμοστάτη, για ένα υπέρογκο ποσό. Η εξαγορά φυσικά δεν αφορά μόνο τα δικαιώματα εκμετάλλευσης του προϊόντος, αλλά και την εξαγορά πατεντών και προσωπικού, σε μια αγορά που αρχίζει πλέον να καλπάζει. Ο θερμοστάτης αυτός έχει την ικανότητα να ρυθμίζεται από το κινητό του χρήστη, όπως επίσης με τον καιρό μαθαίνει τα γούστα του ιδιοκτήτη και αυτορυθμίζεται χωρίς να υπάρχει ανάγκη για έλεγχο από τη μεριά του χρήστη.

Η εταιρεία Savant σε συνεργασία με την Apple, έχουν δημιουργήσει το δικό τους σύστημα Home Automation με ολοκληρωμένες λύσεις για έλεγχο όλου του σπιτιού, είτε αυτό αφορά τον φωτισμό, τη μουσική, τη θέρμανση, την ασφάλεια, ακόμα και τον έλεγχο των κουρτινών, όλα ελεγχόμενα από το iPad ή το iPhone του χρήστη μέσα από μια εύχρηστη εφαρμογή.

Η Samsung, παρουσίασε στην CES που έλαβε χώρα στο Las Vegas στις αρχές Γενάρη, τα πλάνα της για το Home Automation. Σκοπεύει και εκείνη με τη σειρά της, μέσα από το δικό της κλειστό οικοσύστημα φυσικά, να παρέχει μέσα από ένα οικιακό server, έλεγχο σε όλες τις συσκευές τις, μέσα από το smartphone ή ακόμα και το smartwatch του χρήστη. Ο έλεγχος θα περιλαμβάνει φυσικά το φωτισμό του σπιτιού, όπως επίσης το έξυπνο πλυντήριο ή και το έξυπνο ψυγείο της κατασκευάστριας εταιρείας.

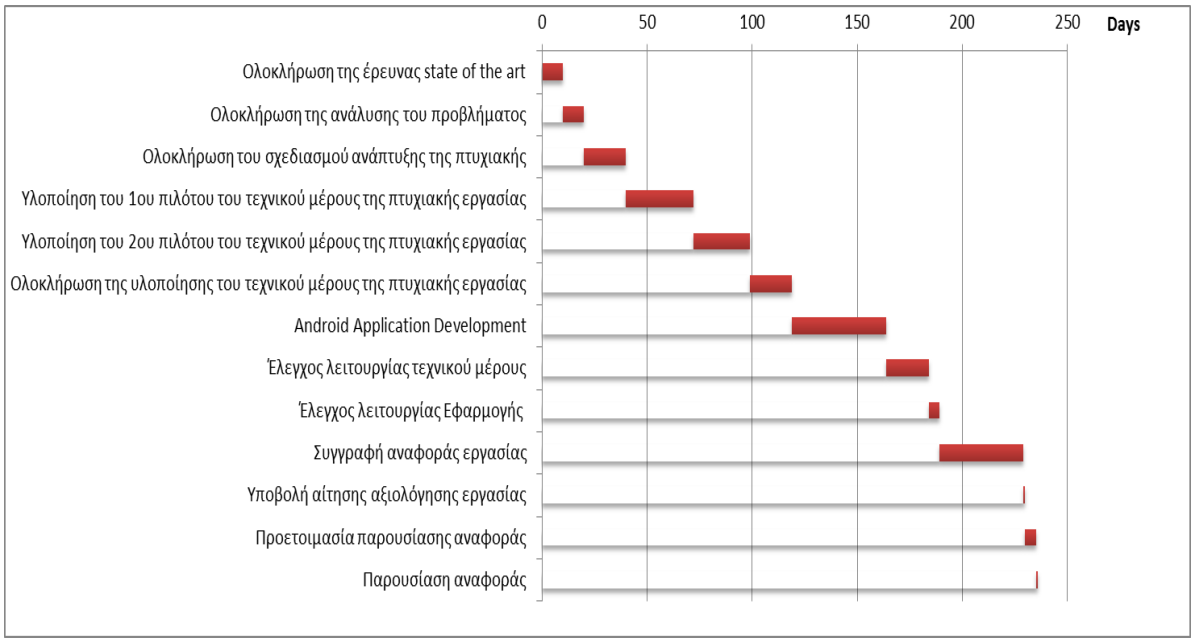
Όσο υπέροχες και αν ακούγονται αυτές οι παροχές, το κόστος απόκτησης των έξυπνων αυτών συσκευών καθώς και το σύστημα ελέγχου τους είναι πάρα πολύ μεγάλο. Επίσης η κάθε εταιρεία κρατάει κλειστό το δικό της ecosystem για εμπορικούς λόγους. Όπως είναι λογικό λοιπόν, μια συσκευή της Savant, δεν πρόκειται να δουλέψει με το ecosystem της Samsung, κοκ.

3.1 Σημαντικοί Στόχοι για την Ολοκλήρωση της Πτυχιακής

- Ολοκλήρωση της έρευνας state of the art 10
- Ολοκλήρωση της ανάλυσης του προβλήματος 10
- Ολοκλήρωση του σχεδιασμού ανάπτυξης της πτυχιακής 20
- Υλοποίηση του 1^{ου} πιλότου του τεχνικού μέρους της πτυχιακής εργασίας 32
- Υλοποίηση του 2^{ου} πιλότου του τεχνικού μέρους της πτυχιακής εργασίας 27
- Ολοκλήρωση της υλοποίησης του τεχνικού μέρους της πτυχιακής εργασίας 20
- Android Application Development 45
- Έλεγχος λειτουργίας τεχνικού μέρους 20
- Έλεγχος λειτουργίας Εφαρμογής 5
- Συγγραφή αναφοράς εργασίας 40
- Υποβολή αίτησης αξιολόγησης εργασίας 1
- Προετοιμασία παρουσίασης αναφοράς 5

- Παρουσίαση αναφοράς 1

3.2 Χρονοδιάγραμμα Εργασίας (Gantt Chart)



Εικόνα 1: Χρονοδιάγραμμα Εργασίας (Gantt Chart) [1]

4. Ανάλυση Απαιτήσεων Συστήματος

Όπως σε κάθε πολύπλοκο σύστημα που περιλαμβάνει την δημιουργία λογισμικού καθώς και την σωστή σύνδεση και απρόσκοπτη λειτουργία τμημάτων υλικού, πρέπει να προσδιοριστούν οι απαραίτητες τελικές απαιτήσεις που θα έχουμε από το τελικό προϊόν.

Σκοπός μας είναι ο καταρτισμένος χρήστης με τον απαραίτητο εξοπλισμό και την εφαρμογή να μπορεί να στήσει και να χρησιμοποιήσει το σύστημα χωρίς να συναντήσει ιδιαίτερη δυσκολία.

Το σύστημα **Arduino – Ethernet Shield – Router – Android Εφαρμογή** θα πρέπει να παρέχει στο χρήστη την δυνατότητα να συνδεθεί ασύρματα μέσω της Εφαρμογής στο Android κινητό του, στον Arduino Server δίνοντας τα σωστά στοιχεία του (IP και Port). Όταν γίνεται η σύνδεση, ο χρήστης θα είναι ικανός να ελέγχει τις εξόδους του server καθώς και να βλέπει την κατάστασή τους μέσα από την εφαρμογή.

Όσον αφορά το Σύστημα, θα πρέπει να πληροί τις απαραίτητες προϋποθέσεις:

- Ο Arduino Server θα πρέπει να έχει χαμηλή κατανάλωση ρεύματος
- Να υπάρχει σταθερή και αξιόπιστη λειτουργία του Arduino Server
- Εύκολη υλική εγκατάσταση συστήματος Arduino λόγω παροχής ασύρματης σύνδεσης
- Ασύρματη πρόσβαση μέσω εφαρμογής Android από smartphones και tablets
- Σταθερή και αξιόπιστη λειτουργία εφαρμογής
- Συμβατότητα με πληθώρα Android συσκευών
- Intuitive Interface εφαρμογής και ευκολία χρήσης της
- Λειτουργία συστήματος με χαμηλή κατανάλωση δεδομένων κινητής τηλεφωνίας και λειτουργία ακόμα και σε αδύναμα δίκτυα (Edge)
- Χρήση switches και sliders στην εφαρμογή ως visual elements
- Λειτουργία εφαρμογής σε δίκτυα WLAN , WAN και GSM

5. Περιβάλλον Σχεδίασης, Υλικό – Λογισμικό

Στο κεφάλαιο αυτό θα αναλύσουμε όλα τα επιμέρους στοιχεία (υλικό, λογισμικό, περιβάλλον σχεδίασης) που χρειάστηκαν για την δημιουργία του τελικού συστήματος.

Το λειτουργικό σύστημα που θα χρησιμοποιηθεί για την εκτέλεση των προγραμμάτων που θα βοηθήσουν την εκπόνηση της εργασίας είναι τα Windows 7.

Το **υλικό** που απαιτήθηκε είναι το εξής:

- Ηλεκτρονικός Υπολογιστής
- Microcontroller Arduino Uno R3
- Ethernet Shield HanRun HR911105A 10/49 βασισμένο στο W5100 chipset
- Router-Client TP-LINK TL-WR703N version 1.6
- Linksys WRT54G v.5
- Thomson TG585 v7
- Relay Module τάσης λειτουργίας 5 Volt DC
- Καλώδια για Arduino
- Φορτιστής 5 Volt microUSB
- Καλώδιο RJ45
- LEDs
- Breadboard
- Πολύμπριζο
- Κινητό Τηλέφωνο με Android, LG Optimus G E975
- Tablet με Android, Barnes & Noble Nook HD+

Το **λογισμικό** που απαιτήθηκε είναι το εξής:

- Windows 7 Ultimate
- Arduino IDE 1.0.5-r2
- Android OS
- Eclipse v.4.2.1 + Android Development Toolkit v.22.0.0
- Firmware DD-WRT v24-sp2 (10/10/09) micro

- Mozilla Firefox
- Chrome Mobile
- Firmware 3.12.11 Build 120320 Rel.42138n από το TL-MR3020 για το TL-WR703N

Οι **γλώσσες προγραμματισμού** που χρησιμοποιήθηκαν ήταν οι εξής:

- HTML
- JAVA
- Wiring
- C++

Τέλος χρησιμοποιήθηκε η **υπηρεσία** Dynamic DNS, No-IP [2]

5.1 Υλικό Μέρος

Παρακάτω θα ακολουθήσει περιγραφή του σημαντικότερου υλικού με χρήσιμα σχόλια που αφορούν τον στόχο της παρούσας διπλωματικής.

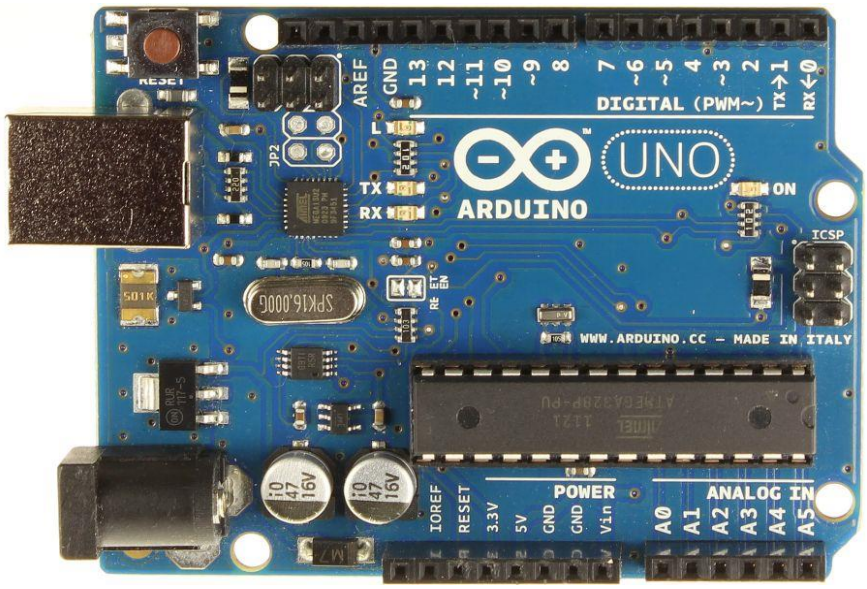
5.1.1 Arduino Uno R3

Το Arduino [3] είναι μια υπολογιστική πλατφόρμα βασισμένη σε μια απλή μητρική πλακέτα ανοικτού κώδικα, με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, και η οποία μπορεί να προγραμματιστεί με τη γλώσσα **Wiring** (ουσιαστικά πρόκειται για τη γλώσσα προγραμματισμού C++ και ένα σύνολο από βιβλιοθήκες, υλοποιημένες επίσης στην C++).

Το Arduino μπορεί να χρησιμοποιηθεί για την ανάπτυξη ανεξάρτητων διαδραστικών αντικειμένων αλλά και να συνδεθεί με υπολογιστή μέσω προγραμμάτων σε Processing, Max/MSP, Pure Data, SuperCollider.

Οι περισσότερες εκδόσεις του Arduino μπορούν να αγοραστούν προ-συναρμολογημένες· το διάγραμμα και πληροφορίες για το υλικό είναι ελεύθερα διαθέσιμα για αυτούς που θέλουν να συναρμολογήσουν το Arduino μόνοι τους.

Μία πλακέτα Arduino αποτελείται από ένα μικροελεγκτή **Atmel AVR** (ATmega328 και ATmega168 στις νεότερες εκδόσεις, ATmega8 στις παλαιότερες) και συμπληρωματικά εξαρτήματα για την διευκόλυνση του χρήστη στον προγραμματισμό και την ενσωμάτωση του σε άλλα κυκλώματα. Όλες οι πλακέτες περιλαμβάνουν ένα γραμμικό ρυθμιστή τάσης **5V** και έναν **κρυσταλλικό ταλαντωτή** 16MHz. Ο μικροελεγκτής είναι από κατασκευής προγραμματισμένος με έναν **bootloader**, έτσι ώστε να μην χρειάζεται εξωτερικός προγραμματιστής.

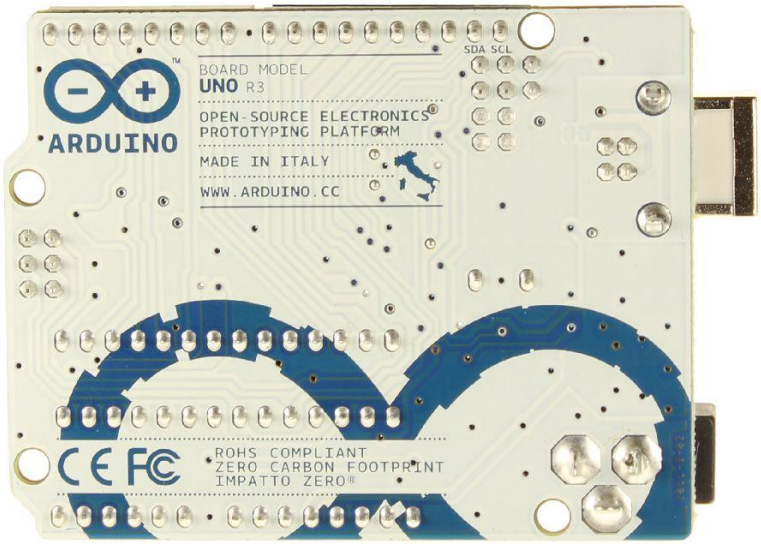


Εικόνα 2: Επάνω Όψη Arduino UNO R3

Σε εννοιολογικό επίπεδο, στην χρήση του Arduino **software stack**, όλα τα boards προγραμματίζονται με μία **RS-232** σειριακή σύνδεση, αλλά ο τρόπος που επιτυγχάνεται αυτό διαφέρει σε κάθε hardware εκδοχή. Οι σειριακές πλάκες Arduino περιέχουν ένα απλό **level shifter** κύκλωμα για να μετατρέπει μεταξύ σήματος επιπέδου **RS-232** και **TTL**. Τα τωρινά Arduino προγραμματίζονται μέσω **USB**, αυτό καθίσταται δυνατό μέσω της εφαρμογής προσαρμοστικών chip **USB-to-Serial** όπως το **FTDI FT232**.

Ο πίνακας Arduino εκθέτει τα περισσότερα **microcontroller I/O pins** για χρήση από άλλα κυκλώματα. Το **Arduino Uno** παρέχει 14 ψηφιακά **I/O pins**, έξι από τα οποία μπορούν να παράγουν **pulse-width** διαμορφωμένα σήματα (PWM modulation), και έξι αναλογικά δεδομένα. Αυτά τα pins βρίσκονται στην κορυφή του πίνακα μέσω **female headers** 0.1 ιντσών (2,2mm).

Το **operating Voltage** του Arduino είναι **5V**. Για κάθε **I/O Pin** που χρησιμοποιείται χρειάζεται **40mA DC** ρεύματος.



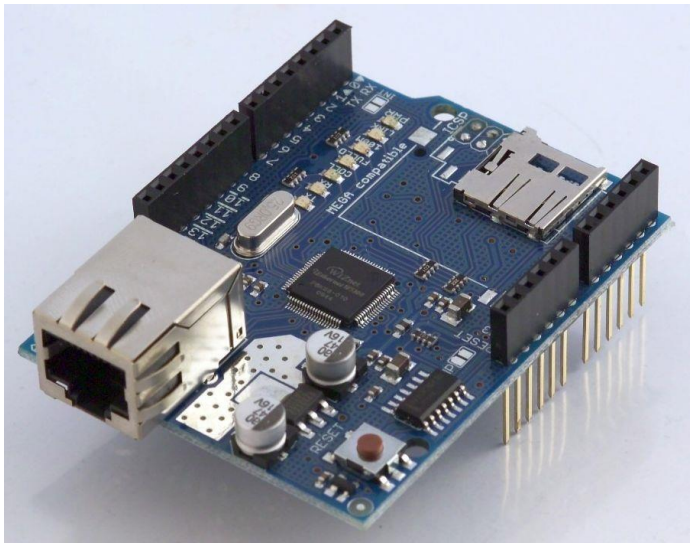
Εικόνα 3: Κάτω Όψη Arduino UNO R3

5.1.2 Ethernet Shield HanRun HR911105A 10/49 βασισμένο στο W5100 chipset

Το **Arduino Ethernet Shield** [4] είναι ένα module που συνδέεται πάνω στην πλακέτα Arduino και μπορεί να συνδέσει το Arduino μέσω ενός καλωδίου **RJ45** με το δίκτυό μας. Το ρεύμα που χρειάζεται για να λειτουργήσει το module είναι 5V τα οποία τα παίρνει από το Arduino εφόσον κάθεται ολόκληρο πάνω στα Pins του Arduino. Το συγκεκριμένο Shield που βασίζεται πάνω στο **Chipset WIZnet W5100**, έχει **16kb buffer** και ταχύτητα **10/100 Mbps**. Το **W5100** μπορεί να παρέχει ένα network stack ικανό για **TCP** και **UDP**.

Το **HanRun HR911105A 10/49** έχει επίσης μια onboard υποδοχή για **micro-SD** card η οποία μπορεί να χρησιμοποιηθεί για την αποθήκευση αρχείων και για την διανομή τους μέσω δικτύου.

Το Shield διαθέτει ακόμα έναν **Reset Controller**, ο οποίος διασφαλίζει ότι το W5100 module γίνεται reset όπως χρειάζεται με κάθε εκκίνηση του συστήματος.



Εικόνα 4: Ethernet Shield HanRun HR911105A 10/49 βασισμένο στο WIZnet W5100

Το Arduino επικοινωνεί με το W5100 και την SD card κάνοντας χρήση της **SPI bus**. Αυτό γίνεται στα digital pins **10**, **11**, **12** και **13** στο Arduino UNO. Αυτά τα pins δεν μπορούν να χρησιμοποιηθούν για γενικό I/O.

Τέλος το shield περιλαμβάνει αρκετές ενδεικτικές λυχνίες που μας ενημερώνουν για την ύπαρξη ρεύματος, την συνδεσιμότητα, την ταχύτητα του δικτύου, το αν υπάρχει λήψη και αποστολή δεδομένων καθώς και αν υπάρχουν συγκρούσεις πακέτων στο δίκτυο.

5.1.3 Client-Router TP-LINK TL-WR703N

Το **TP-Link TL-WR703N** είναι ένα φορητό ασύρματο Router με δυνατότητες **Client**, **Access Point**, **Repeater** και **Bridge**. Υποστηρίζει ασύρματα δίκτυα με πρότυπο δικτύωσης 802.11b/g/n και ταχύτητες έως 150 Mbps. Επίσης υποστηρίζει τα πρότυπα κρυπτογράφησης WEP, WPA/WPA2, WPA-PSK/WPA2-PSK, ενώ μπορεί να έχει το ρόλο Router δίνοντας ασύρματη πρόσβαση στο internet σε συσκευές με τη λειτουργία 3G USB Modem Host που διαθέτει.

Στο εσωτερικό του κρύβεται μια **CPU Atheros AR7240** συχνότητας **400MHz**, ένα chipset **Atheros AR9331** με ενσωματωμένο **wireless** ισχύος **20dBm – 100mW**, **4 MB** ενσωματωμένη **flash μνήμη**, **32 MB μνήμη RAM**. Ακόμα έχει μια εξωτερική **USB 2.0 High-Speed** θύρα ενώ η

τροφοδοσία του γίνεται μέσω της υποδοχής **micro-USB** που είναι ίδια με αυτή που συναντάμε και στις κινητές συσκευές.

Για να δουλέψει χρειάζεται ρεύμα τάσης **5V DC**. Η κατανάλωσή του κυμαίνεται στα **100mAh** [5] με την κεραία Wi-Fi **ανοιχτή**, ενώ στα **80mAh** με την κεραία Wi-Fi **κλειστή**. Η μέση κατανάλωσή του είναι περίπου **0.5W** που είναι εξαιρετικά μικρή. Ακόμα διαθέτει μια θύρα **Ethernet RJ45** για σύνδεση με καλώδιο, κάτι που μας είναι απαραίτητο για το συστημά μας. Τέλος διαθέτει ένα **power button** και ένα **reset button**.

Εικόνα 5: TP-LINK TL-WR703N

5.1.4 Linksys WRT54G v5

Το **Linksys WRT54G v5** είναι ένα **Router/Access Point**, με το chipset **BCM5352** της **Broadcom**. Η συχνότητα [6] της **CPU** είναι **200 MHz**, ενώ φέρει **8MB RAM** και **2MB flash memory**. Έχει **4+1 θύρες Ethernet RJ45** για την λειτουργία του ως **Network Switch** (η θύρα WAN για το Internet είναι μέρος του εσωτερικού Network Switch, αλλά ανήκει σε διαφορετικό VLAN – δηλαδή στο ίδιο υποδίκτυο αλλά σε διαφορετικό τμήμα του). Η συσκευή έχει **δύο εξωτερικές αποσπώμενες κεραίες** ενώ υποστηρίζει ασύρματα δίκτυα με πρότυπο δικτύωσης **802.11b/g** με ταχύτητες έως **54Mbps** στα **2.4GHz**, ενώ ενσύρματα υποστηρίζει ταχύτητες **10/100Mbps**. Επίσης υποστηρίζει όλα τα σύγχρονα πρότυπα κρυπτογράφησης WEP, WPA/WPA2, WPA-PSK/WPA2-PSK. Τέλος το Router φέρει **9 LEDs** τα οποία μας ενημερώνουν για την κατάσταση του δικτύου, τη συνδεσιμότητα του και για την λειτουργία της συσκευής, ενώ στο πίσω μέρος έχει ένα κουμπί "**Reset**". Για να βελτιώσουμε τις δικτυακές ικανότητες της συσκευής, έχει περαστεί το **custom firmware: DD-WRT v24-sp2 (10/10/09) micro**.

Εικόνα 6: Linksys WRT54G v5

5.1.5 Thomson TG585 v7

Το **Thomson TG585 v7** είναι ένα **Router/Access Point**, με το chipset **BCM6338** της **Broadcom**. Η συχνότητα [7] της **CPU** είναι **240MHz**, ενώ φέρει **16MB RAM** και **4MB flash memory**.

Έχει **4 θύρες Ethernet RJ45** για την λειτουργία του ως **Network Switch** ενώ φέρει και μια θύρα **RJ11** για ενσύρματη σύνδεση με το τηλεφωνικό δίκτυο και κατ' επέκταση για πρόσβαση στο Internet μέσω **ADSL2+** γραμμής. Η συσκευή έχει μία εξωτερική κεραία ενώ υποστηρίζει ασύρματα δίκτυα με πρότυπο δικτύωσης **802.11b/g** με ταχύτητες έως **54Mbps** στα **2.4GHz**, ενώ ενσύρματα υποστηρίζει ταχύτητες **10/100Mbps**. Επίσης υποστηρίζει όλα τα σύγχρονα πρότυπα κρυπτογράφησης WEP, WPA/WPA2, WPA-PSK/WPA2-PSK αν και στην παρούσα πτυχιακή δεν θα χρειαστούμε το ασύρματο Interface της συσκευής καθώς θα έχει καθαρά δράση ως **Internet Gateway**. Τέλος φέρει ένα **power button** και κάποια **LEDs** ένδειξης της συνδεσιμότητας της συσκευής.

Εικόνα 7: Thomson TG585 v7

5.1.6 Relay Module Τάσης Λειτουργίας 5V DC

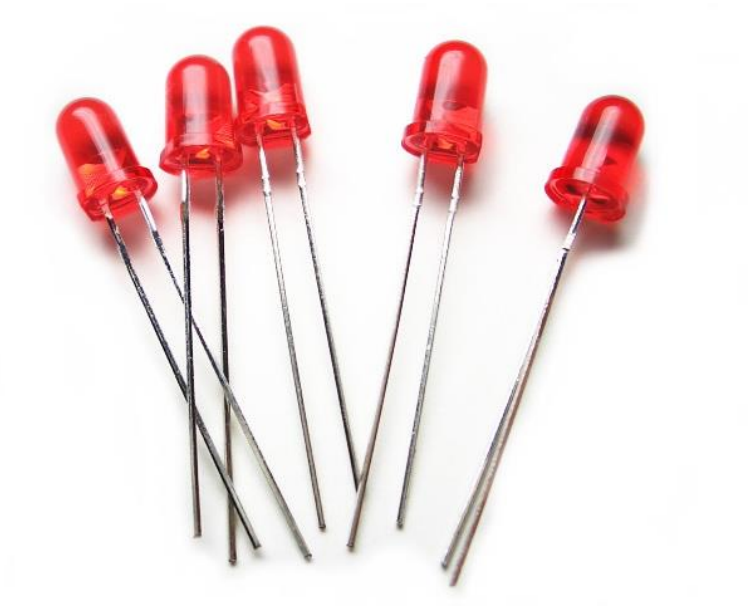
Το **Ρελέ** (**Relay**) που θα χρησιμοποιήσουμε, είναι ένας ηλεκτρικός διακόπτης που ανοίγει και κλείνει ένα ηλεκτρικό κύκλωμα κάτω από τον έλεγχο ενός άλλου ηλεκτρικού κυκλώματος. Το ρελέ είναι επίσης ικανό να ελέγχει ένα κύκλωμα εξόδου υψηλότερης ισχύος από το κύκλωμα εισόδου. Κάθε του επαφή μπορεί να είναι Κανονικά-Ανοικτή, δηλαδή το κύκλωμα να αποσυνδέεται όταν το ρελέ είναι ανενεργό και να συνδέεται όταν είναι ενεργό, Κανονικά-Κλειστή, δηλαδή να αποσυνδέεται το κύκλωμα όταν το ρελέ είναι ενεργό και να συνδέεται όταν το ρελέ είναι ανενεργό, ή Μεταγωγική, δηλαδή να ελέγχει δύο κυκλώματα ως μια επαφή κανονικά-ανοικτή και μια επαφή κανονικά-κλειστή που έχουν ένα κοινό ακροδέκτη. Στο κύκλωμά μας θα χρησιμοποιήσουμε ένα Ρελέ στατικού τύπου [8], που ουσιαστικά αποτελείται από ηλεκτρονικά κυκλώματα για να αναπτύξει όλα εκείνα τα χαρακτηριστικά που επιτυγχάνονται με τα κινούμενα μέρη ενός ηλεκτρομαγνητικού ρελέ. Το **Relay Module** (μοντέλο **SRD-05VDC-SL-C**) της **Songle** που θα χρησιμοποιήσουμε, είναι ένα ολοκληρωμένο κύκλωμα με ρελέ, το οποίο έχει τάση λειτουργίας **5V DC**, ενώ μπορεί να ελέγξει ρεύμα έως **10A** στα **250V AC**, στα **125V AC**, **30V DC** και στα **28V DC**. Αν είχαμε μόνο το ρελέ και όχι την τυποποιημένη μονάδα, θα έπρεπε να το συνδυάσουμε με το απαραίτητο ηλεκτρονικό υλικό (αντιστάσεις, διόδους κτλ) για να έχουμε το ίδιο κύκλωμα.

Εικόνα 8: Relay Module SRD-05VDC-SL-C

5.1.7 LEDs (Light-Emitting Diode)

Το **LED** (**Light-Emitting Diode**) [9], είναι ένας ημιαγωγός ο οποίος εκπέμπει φωτεινή ακτινοβολία στενού φάσματος όταν του παρέχεται μια ηλεκτρική τάση κατά τη φορά ορθής πόλωσης. Το χρώμα του φωτός που εκπέμπεται, εξαρτάται από την χημική σύσταση του ημιαγώγιμου υλικού που χρησιμοποιείται και μπορεί να είναι υπεριώδες, ορατό ή υπέρυθρο. Το μήκος κύματος του φωτός που εκπέμπεται και κατά συνέπεια το χρώμα του, εξαρτάται από το χάσμα ενέργειας των υλικών τα οποία χρησιμοποιούνται για την δημιουργία του περάσματος **p-n**.

Μερικά πολύ σημαντικά πλεονεκτήματα των LEDs είναι η πολύ μεγαλύτερη παραγωγή φωτός ανά watt συγκριτικά με τις λάμπες πυράκτωσης, το μικρός τους μέγεθος, η γρήγορη απόκριση τους (ON/OFF) που είναι της τάξεως του **microsecond**, η πολύ μικρή εκπομπή θερμότητας, ο μεγάλος χρόνος ζωής, η αντίσταση τους σε κραδασμούς και τέλος η χαμηλή τους τοξικότητα, καθώς δεν περιέχουν **υδράργυρο** όπως οι λάμπες **φθορισμού**.



Εικόνα 9: LEDs

5.1.8 LG Optimus G E975

Το **Optimus G** της **LG** [10], είναι ένα κινητό τηλέφωνο με λογισμικό **Android 4.1.2** Jelly Bean που κυκλοφόρησε τον Αύγουστο του 2012. Υποστηρίζει δίκτυα **2G** (GSM 850/900/1800/1900), δίκτυα **3G** (HSDPA 900/2100) με **Download Speed** έως **42Mbps** και **Upload Speed** έως **5.76Mbps** καθώς και δίκτυα **4G** (LTE 800/900/1800/2100/2600) με **Download Speed** έως **100Mbps** και **Upload Speed** έως **50Mbps**. Έχει υποδοχή micro SIM, οθόνη αφής capacitive, τύπου **True HD-IPS+,** ανάλυσης **768x1280**, 16 εκατομμυρίων χρωμάτων, **4.7 ιντσών**, με πυκνότητα **pixel per inch** ~**318**. Η συσκευή έχει **32GB** αποθηκευτικό χώρο και **2GB** μνήμης **RAM**. Ακόμα, ενσωματώνει το **SOC** της **Qualcomm MDM9615/APQ8064** το οποίο περιλαμβάνει ένα τετραπύρηνο επεξεργαστή **Krait** συχνότητας **1.5GHz** και της **Adreno 320** που είναι υπεύθυνη για την επεξεργασία των γραφικών της συσκευής. Η οπίσθια camera της συσκευής είναι στα **13MP** και η εμπρόσθια στα **1.2MP**, ενώ έχει και **πληθώρα αισθητήρων** όπως γυροσκόπιο, βαρόμετρο, πυξίδα κτλ. Η συσκευή έχει φυσικά κεραία για υποστήριξη δικτύων **Wi-Fi 802.11 a/b/g/n στα 2.4 και 5GHz**, καθώς και Bluetooth v4.0 με υποστήριξη A2DP. Τέλος έχει ραδιόφωνο **FM** με υποστήριξη RDS, υποδοχή 3.5mm για έξοδο ήχου, **GPS** (με A-GPS και Glonass), ενώ λειτουργεί με μια μη αφαιρούμενη μπαταρία τύπου Li-Po (Lithium-Polymer) χωρητικότητας **2100mAh**.

Θα πρέπει να τονίσουμε ότι η εφαρμογή που θα φτιάξουμε δεν χρειάζεται ιδιαίτερη επεξεργαστική ισχύ, οπότε ακόμα και παλαιότερα κινητά δεν θα έχουν κανένα απολύτως πρόβλημα να την τρέξουν απρόσκοπτα.

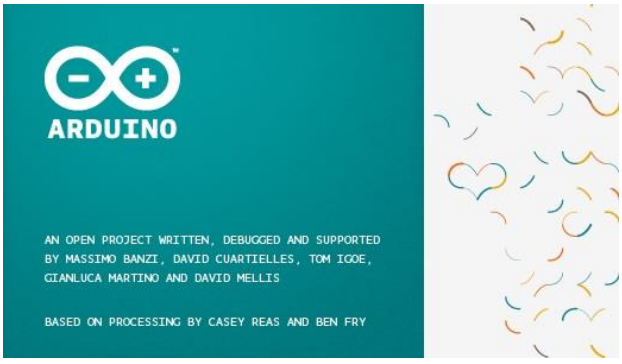
Εικόνα 10: LG Optimus G E975

5.2 Λογισμικό

Παρακάτω θα ακολουθήσει η περιγραφή του σημαντικότερου λογισμικού.

5.2.1 Arduino IDE 1.0.5-r2

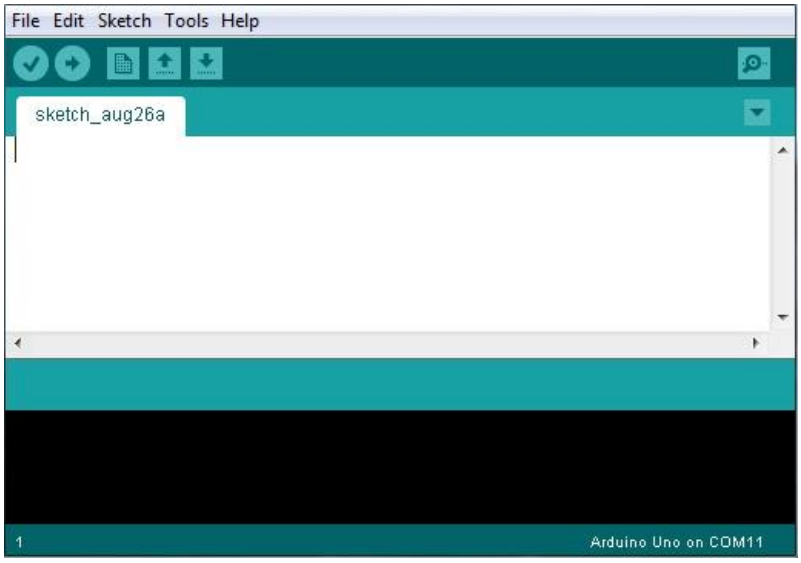
Το Arduino IDE είναι ένα περιβάλλον προγραμματισμού ανοιχτού κώδικα [11]. Μπορεί να εγκατασταθεί στα λειτουργικά συστήματα των Windows, Mac OS X και Linux, ενώ η γλώσσα προγραμματισμού που χρησιμοποιεί ονομάζεται **Wiring** (ουσιαστικά πρόκειται για τη C++ με κάποιες μετατροπές).



Εικόνα 11: Arduino IDE

Το περιβάλλον ανάπτυξης περιλαμβάνει έναν επεξεργαστή κειμένου για τη συγγραφή του κώδικα, μια περιοχή μηνυμάτων (αποσφαλμάτωσης), μια κονσόλα κειμένου, μια μπάρα εργαλείων με κουμπιά με τις πιο κοινές λειτουργίες, καθώς και μια σειρά από μενού. Συνδέεται με το Arduino για την φόρτωση των προγραμμάτων στη συσκευή και για να επικοινωνήσει μαζί της, κάτι στο οποίο βοηθάει αποτελεσματικά το **Serial Monitor** του IDE, με τη βοήθεια του κατάλληλου κώδικα.

Το λογισμικό που γράφεται χρησιμοποιώντας το **Arduino IDE** ονομάζεται **sketch**. Αυτά γράφονται στον επεξεργαστή κειμένου του προγραμματιστικού περιβάλλοντος και αποθηκεύονται με την επέκταση ***.ino**. Πέραν των βασικών λειτουργιών (copy/paste), παρέχει επίσης λειτουργία αναζήτησης και αντικατάστασης κώδικα. Η περιοχή μηνυμάτων δίνει χρήσιμες πληροφορίες αποσφαλμάτωσης κατά την αποθήκευση και την εξαγωγή του προγράμματος. Στην κάτω δεξιά πλευρά του παραθύρου προβάλλονται πληροφορίες για το είδος του Arduino που έχουμε συνδέσει (UNO, Mega etc) καθώς και σε ποια **Serial Port** έχει γίνει η σύνδεση. Η μπάρα εργαλείων μας δίνει τη δυνατότητα να κάνουμε "**Verify**" και "**Upload**" τα προγράμματα, καθώς και να δημιουργήσουμε, να ανοίξουμε κάποιο υπάρχον, ή να αποθηκεύσουμε κάποιο **sketch**, καθώς και τη δυνατότητα να ανοίξουμε το **Serial Monitor**.



Εικόνα 12: Περιβάλλον Ανάπτυξης Κώδικα Arduino

Απαραίτητες για να μας δώσουν επιπλέον λειτουργικότητα και για την υποστήριξη **modules**, είναι η βιβλιοθήκες. Για να συμπεριλάβουμε μια βιβλιοθήκη στο πρόγραμμά μας πρέπει να την κάνουμε **import** το οποίο θα φανεί στο πρόγραμμά μας με τη μορφή της δήλωσης "**#include**" στην αρχή του κώδικα, ακολουθούμενη από το όνομα της βιβλιοθήκης. Επειδή οι βιβλιοθήκες γίνονται **upload** στο **board** μαζί με το **sketch** μας, καταλαμβάνουν και αυτές πολύτιμο χώρο στον ήδη περιορισμένο της συσκευής, οπότε η χρήση τους πρέπει να γίνεται με σύνεση. Κάποιες βιβλιοθήκες περιλαμβάνονται στο **Arduino IDE**, όμως πολλές θα χρειαστεί να τις κατεβάσουμε από διάφορες πηγές.

Στην παρούσα πτυχιακή χρειάστηκε να κατεβάσουμε, να κάνουμε **import** και να δουλέψουμε με την βιβλιοθήκη που είναι απαραίτητη για τη χρήση του **Ethernet Shield** που είναι βασισμένο στο **W5100 chipset.**

5.2.2 Android OS

Το **Android** είναι ένα πολύ δημοφιλές λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας βασισμένο στον πυρήνα του **Linux**, ενώ στην παρούσα φάση αναπτύσσεται από την **Google**. Το Android είναι πρωτίστως κατασκευασμένο για χρήση σε συσκευές με οθόνες αφής όπως τα **smartphones** και τα **tablets**, ενώ υπάρχουν παραλλαγές του με ειδικά σχεδιασμένο **interface** για τηλεοράσεις (**Android TV**), αυτοκίνητα (**Android Auto**) και έξυπνα ρολόγια (**Android Wear**), ενώ αυτή τη στιγμή μετράει πάνω από ένα δις ενεργούς χρήστες , όπως ανακοινώθηκε στο Google I/O 2014 [12].

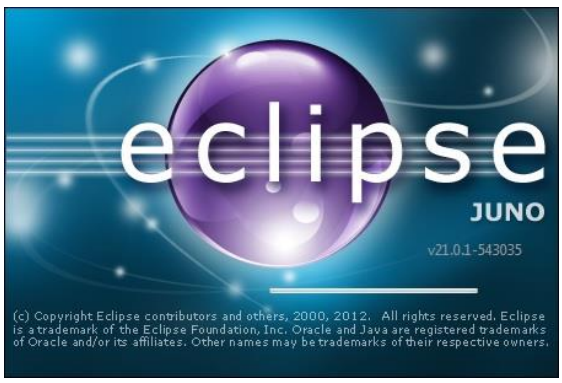
Εικόνα 13: Android Logo

Ο κώδικας του Android διανέμεται από την Google κάτω από την άδεια του ανοιχτού λογισμικού, αν και πλέον οι περισσότερες Android συσκευές διανέμονται στο εμπόριο με ένα συνδυασμό ανοιχτού λογισμικού και ιδιόκτητου λογισμικού, το οποίο περιλαμβάνει συνήθως την εκάστοτε έκδοση Android, τις υπηρεσίες και τις κλειστού κώδικα εφαρμογές της Google (όπως ο Chrome ή το Play Store), καθώς και το έξτρα λογισμικό ή και υπηρεσίες του εκάστοτε κατασκευαστή.

Το Android έχει μια τεράστια γκάμα εφαρμογών που διανέμονται κυρίως μέσα από το Play Store της Google και μπορούν να είναι είτε δωρεάν είτε επί πληρωμή. Οι εφαρμογές αναπτύσσονται κυρίως σε **Java**, χρησιμοποιώντας το **Android Software Development kit** (**SDK**) [13]. Το SDK περιλαμβάνει ένα μεγάλο εύρος από προγραμματιστικά εργαλεία, περιλαμβανομένου ενός debugger (για αποσφαλμάτωση), βιβλιοθηκών λογισμικού, έναν εξομοιωτή smartphone βασισμένο στον **QEMU**, εκτενή βιβλιογραφία, υποδείγματα κώδικα καθώς και εκπαιδευτικού τύπου υλικό. Το επίσημα υποστηριζόμενο IDE (**integrated development environment**) είναι το **Eclipse**, κάνοντας χρήση του **ADT** (**Android Development Tools**) **plugin**. Φυσικά υπάρχουν και άλλα εργαλεία ανάπτυξης (όπως το **Android Studio** που διανέμεται από τη Google, είναι σε φάση **beta** και αναμένεται να αντικαταστήσει το Eclipse στο μέλλον όσον αφορά την ανάπτυξη εφαρμογών Android), αλλά το **Eclipse** θα μας απασχολήσει στην παρούσα διπλωματική.

5.2.3 Eclipse v.4.2.1 + Android Development Toolkit v.22.0.0

Το **Eclipse** [14]είναι ένα IDE (**integrated development environment**) το οποίο αποτελείται από ένα βασικό **workspace** και ένα επεκτεινόμενο σύστημα από **plugins** για την παραμετροποίηση του προγραμματιστικού περιβάλλοντος. Γραμμένο κυρίως σε **Java**, το **Eclipse** μπορεί να χρησιμοποιηθεί για τη δημιουργία εφαρμογών της ίδιας γλώσσας, αλλά με τη βοήθεια των plugins μπορεί να χρησιμοποιηθεί και για την ανάπτυξη εφαρμογών σε άλλες γλώσσες όπως οι **C**, **C++**, **Fortran**, **PHP**, **Python**, **Ruby** κτλ).



Εικόνα 14: Eclipse Splash Screen

Εικόνα 15: ADT Splash Screen

Στην παρούσα πτυχιακή θα κάνουμε χρήση του plugin **Android Development Toolkit v.22.0.0** (**ADT**) [15], το οποίο περιέχει όλες τις απαραίτητες βιβλιοθήκες και τα **APIs** για την ανάπτυξη της διεπαφής και των λειτουργιών μιας εφαρμογής **Android**, μέχρι και την έκδοση **4.2.2** του λογισμικού.

5.2.4 Firmware DD-WRT micro

Το **DD-WRT** είναι ένα Open Source firmware βασισμένο σε **Linux**, κατάλληλο για μια μεγάλη ποικιλία **WLAN** δρομολογητών και ενσωματωμένων συστημάτων. Το χαρακτηριστικό που το κάνει να ξεχωρίζει, είναι ότι προσφέρει τον ευκολότερο δυνατό χειρισμό (ο οποίος όμως απαιτεί πολύ καλές γνώσεις δικτύων), ενώ ταυτόχρονα υποστηρίζει ένα τεράστιο αριθμό λειτουργιών που περιορίζονται μόνο από το υλικό της εκάστοτε συσκευής.

Εικόνα 16: DD-WRT Logo

Η διεπαφή του DD-WRT έχει λογική δόμηση και η διαχείρισή του γίνεται μέσα από ένα web interface στο οποίο έχουμε πρόσβαση με τη χρήση ενός Internet Browser. Συγκριτικά με το λογισμικό που είναι προεγκατεστημένο στα περισσότερα WLAN routers, το DD-WRT επιτρέπει μια απρόσκοπτη λειτουργία του συστήματος, με εμφανώς μεγαλύτερη λειτουργικότητα που καλύπτει ακόμα και απαιτήσεις σε επαγγελματικό επίπεδο.

Λόγω της μεγάλης κοινότητας του DD-WRT και τις υποστήριξης που παρέχεται από προγραμματιστές και χρήστες, πιθανά ελαττώματα του συστήματος εντοπίζονται πολύ γρήγορα και έτσι μπορούν να διορθωθούν χωρίς να υπάρχει καθυστέρηση.

Για συσκευές που προορίζονται κυρίως για προσωπική χρήση, το DD-WRT είναι διαθέσιμο δωρεάν. Για πλατφόρμες που χρησιμοποιούνται για εμπορική χρήση, απαιτείται μια άδεια επί πληρωμή. Σε σύγκριση με την δωρεάν διαθέσιμη έκδοση, η επαγγελματική έκδοση επιτρέπει επιπλέον ρυθμίσεις των παραμέτρων του WLAN, δημιουργώντας έτσι την δυνατότητα κατασκευής σταθερών και αξιόπιστων δικτυακών υποδομών. Ειδικές απαιτήσεις μπορούν να εκπληρωθούν μέσω ειδικά σχεδιασμένων εκδόσεων του DD-WRT.

Θα δούμε αναλυτικά κάποια από τα μενού της micro έκδοσης του DD-WRT και στο πως τα χρησιμοποιήσαμε για να επιτύχουμε τους στόχους της παρούσας διπλωματικής.

5.2.5 Dynamic DNS (DDNS) Service “No-IP”

Το **DDNS** (Dynamic Domain Name System) [16] είναι μια μέθοδος η οποία μας επιτρέπει την αυτόματη ενημέρωση ενός διακομιστή ονομάτων στο **Domain Name System**, ο οποίος είναι διασυνδεμένος με μια δυναμική (μεταβαλλόμενη) IP διεύθυνση.

Τυπικά, μόλις ένας χρήστης συνδεθεί με το Internet, ο πάροχος (ISP) του εκάστοτε χρήστη του εκχωρεί μια IP διεύθυνση από την “πισίνα” των διαθέσιμων IP διευθύνσεων που δεν χρησιμοποιούνται από κάποιον, μόνο κατά τη διάρκεια της συγκεκριμένης σύνδεσης. Των περισσότερων παρόχων οι δυναμικές IP έχουν κάποιο timeout και αλλάζουν μετά από κάποιο χρονικό διάστημα (αποσυνδέοντας προσωρινά το χρήστη). Αυτή η μέθοδος της δυναμικής εκχώρησης διευθύνσεων, επεκτείνει την διαθεσιμότητα των διευθύνσεων IP από τον εκάστοτε πάροχο.

Ένας πάροχος DDNS και συγκεκριμένα ο **No-IP** στην περίπτωσή μας, κάνει χρήση ενός ειδικού προγράμματος που τρέχει στη συσκευή (υπολογιστή, δρομολογητή κτλ) του εκάστοτε χρήστη, το οποίο επικοινωνεί με την DNS υπηρεσία κάθε φορά που IP διεύθυνση που έχει δοθεί από τον ISP

αλλάζει. Με τη σειρά του ο πάροχος DDNS ενημερώνει την DNS βάση δεδομένων του για να αντικατοπτρίσει την αλλαγή IP διεύθυνσης. Έτσι, ακόμα και αν ενός **domain name** η IP διεύθυνση αλλάζει συχνά, οι υπόλοιποι χρήστες δεν χρειάζεται να ξέρουν την καινούργια IP για να συνδεθούν με το απομακρυσμένο σύστημα.

Εικόνα 17: No-IP Logo

Για παράδειγμα: Εκχωρείται σε έναν δρομολογητή η διεύθυνση 174.153.23.22 από τον ISP πάροχο του χρήστη. Κάνοντας χρήση μιας υπηρεσίας DDNS, ο χρήστης, μπορεί να αντιστοιχίσει αυτή την IP σε ένα domain name, όπως το http://myself.noip.me. Με τις απαραίτητες ρυθμίσεις οποιοσδήποτε θα μπορούσε να δει τον υπολογιστή της διεύθυνσης 174.153.23.22, αλλά πλέον αρκεί να δώσει το παραπάνω domain name. Η συγκεκριμένη IP θα αλλάξει σύντομα από τον ISP πάροχο του χρήστη, όμως με την υπηρεσία DDNS που τρέχει στον δρομολογητή του χρήστη, θα ενημερωθεί ο DDNS πάροχος και θα αντιστοιχήσει την νέα IP στο ίδιο domain name καθιστώντας μονίμως προσπελάσιμο τον υπολογιστή, παρά την αλλαγή της IP του.

Στην παρούσα διπλωματική θα χρησιμοποιήσουμε την υπηρεσία για να κάνουμε μονίμως προσπελάσιμο το συστημά μας μέσω Web, ώστε να μπορούμε να αλληλεπιδρούμε μαζί του παρά τις αλλαγές στην IP διεύθυνσή του.

6. Σύνδεση, Επικοινωνία και Προγραμματισμός “Arduino UNO R3 + Ethernet Shield HanRun HR911105A 10/49”

Σε αυτό το κεφάλαιο θα αναλύσουμε πως συνδέσαμε τον υπολογιστή μας με τον microcontroller σε συνδυασμό με το Ethernet Shield και ποια βήματα κάναμε για τον επιτυχή προγραμματισμό του για τον τελικό μας στόχο. Επίσης θα δούμε αναλυτικά το πρόγραμμα που δημιουργήθηκε για να τρέξει στον server/microcontroller και τις λειτουργίες του.

6.1 Σύνδεση Arduino Uno R3 με Ethernet Shield

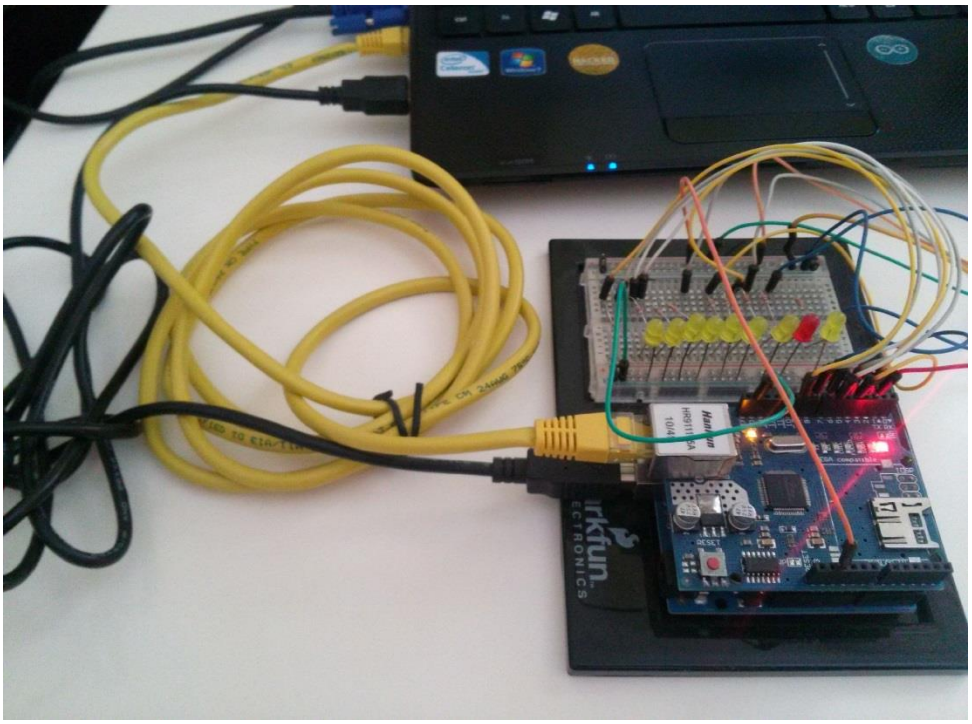
Η σύνδεση του **Arduino Uno R3** με το Ethernet Shield είναι πάρα πολύ απλή. Το μόνο που έχουμε να κάνουμε είναι να “κουμπώσουμε” το Shield πάνω στο Arduino, αντιστοιχίζοντας τα pins του Shield ένα προς ένα με τις υποδοχές του Arduino.

Εικόνα 18: Arduino Uno R3 + Ethernet Shield

Από εδώ και στο εξής το Arduino αποκτάει **ενσύρματες** δικτυακές ικανότητες χάρις στην νέα θύρα **RJ45**. Το Shield παίρνει ρεύμα κατευθείαν από το Arduino Uno, ενώ τα pins που είχε το Arduino, πλέον αντιστοιχούνται ένα προς ένα σε αυτά του Shield. Επίσης το Arduino απέκτησε και δυνατότητα αποθήκευσης αρχείων χάρις στην υποδοχή micro SD που φέρει το Ethernet Shield. Στην παρούσα πτυχιακή δεν θα γίνει χρήση αυτής της δυνατότητας.

6.2 Καλωδιακή Σύνδεση Server με Υπολογιστή

Όπως αναφέρθηκε στο Κεφάλαιο 2, αρχικά θέλαμε να επιτευχθεί απευθείας σύνδεση μεταξύ του Laptop και του Server (**Arduino+Ethernet**). Έπρεπε λοιπόν να ασχοληθούμε και με τη ρύθμιση της κάρτας δικτύου από την πλευρά του **Laptop**, αλλά και με την δημιουργία του κατάλληλου προγράμματος στον **Server** μας ώστε να υπάρχει επικοινωνία μεταξύ τους.



Εικόνα 19: Καλωδιακή σύνδεση Server με το Laptop

Το Laptop στο ενσύρματο δικτυακό του κομμάτι έτρεχε με τη λειτουργία του **DHCP**, το οποίο προσδίδει δυναμικά τα χαρακτηριστικά του δικτύου (όπως η IP) στο Laptop όταν αυτό συνδέεται σε συσκευές όπως **routers** ή **hubs**. Θεωρώντας ότι από τη μεριά του Laptop δεν χρειαζόταν κάποια επιπλέον ρύθμιση για να υπάρξει επικοινωνία, πέρασα στο κομμάτι του Server.

6.3 Δημιουργία Τμήματος Κώδικα του Arduino Server Υπεύθυνο για τα Δικτυακά Χαρακτηριστικά της Συσκευής

Το Arduino, όπως αναφέρθηκε και νωρίτερα, για να λειτουργήσει χρειάζεται ένα κομμάτι κώδικα το οποίο ονομάζεται "**Sketch**". Έπειτα από έρευνα και μέσα από trial and error, εισάγοντας τις **απαραίτητες βιβλιοθήκες**, φορτώθηκε το Sketch "**Web Server**" από τα "**Examples**" του Arduino IDE. Στη συνέχεια, παραμετροποιήθηκε το τμήμα του κώδικα το οποίο έδινε στον server τα απαραίτητα χαρακτηριστικά που πρέπει να έχει οποιαδήποτε συσκευή με δυνατότητες δικτύωσης και είναι τα ακόλουθα:

- Η **MAC Address** [17]. Ένας δεκαεξαδικός σειριακός αριθμός ο οποίος είναι μοναδικός για κάθε συσκευή και χρησιμοποιείται για την επικοινωνία μεταξύ δικτυακών συσκευών εντός ενός τοπικού δικτύου. Σε κάθε επικοινωνία οποιασδήποτε δικτυακής συσκευής με μια άλλη, ο αριθμός αυτός αποκαλύπτεται από τον αποστολέα (**source**) στον παραλήπτη (**destination**). Στην περίπτωσή μας όμως ο μοναδικός αυτός αριθμός δεν δόθηκε από το εργοστάσιο στη συσκευή μας. Έτσι ήμασταν αναγκασμένοι να δημιουργήσουμε εμείς έναν. Το μόνο που έπρεπε να προσέξουμε για να μην υπάρχουν προβλήματα, ήταν να μην υπάρχει συσκευή στο δίκτυό μας με την ίδια MAC με αυτήν που θα δημιουργούσαμε [1].

[1] Οι πιθανότητες να συμβεί αυτό στη συγκεκριμένη περίπτωση είναι απειροελάχιστες και συγκεκριμένα 1 στις 2^{48} εφόσον κάθε αλφαριθμητικό μπορεί να πάρει $2^4 = 16$ τιμές επί 12 που είναι στο σύνολο. Άρα έχουμε $(2^4)^{12}$.

- Η **IP Address** [18]. Ένας μοναδικός αριθμός που χρησιμοποιείται από συσκευές για τη μεταξύ τους αναγνώριση και συνεννόηση σε ένα δίκτυο υπολογιστών που χρησιμοποιεί το Internet Protocol standard. Κάθε συσκευή που ανήκει στο δίκτυο πρέπει να έχει τη δική της μοναδική διεύθυνση.

- Η **Gateway**. Η Πύλη Δικτύου είναι το υλικό ή το λογισμικό που χρησιμοποιείται για τη σύνδεση ανάμεσα σε διαφορετικά δικτυακά περιβάλλοντα. Οι Πύλες Δικτύου μπορούν να λειτουργήσουν σε αρκετά από τα ανώτερα στρώματα του μοντέλου OSI[2], κυρίως στα στρώματα συνόδου, παρουσίασης και εφαρμογών.

- Το **Subnet**. Μια λογική υποδιαίρεση ενός IP δικτύου. Όλοι οι υπολογιστές που ανήκουν σε ένα υποδίκτυο, διευθυνσιοδοτούνται με ένα κοινό, πανομοιότυπο, πιο σημαντικό ψηφίο-ομάδα στην IP διεύθυνσή τους.

- Η **Port**. Κάθε Server κάνει τις υπηρεσίες του διαθέσιμες κάνοντας χρήση αριθμημένων θυρών, μια για κάθε υπηρεσία που είναι διαθέσιμη στον Server. Για παράδειγμα ένας Web Server θα είναι συνήθως διαθέσιμος στην θύρα 80, ενώ ένας FTP Server στην θύρα 21.

Οι Clients συνδέονται σε μια υπηρεσία, σε μια συγκεκριμένη IP Address, και σε μια συγκεκριμένη θύρα. Αυτό είναι το **τρίπτυχο λειτουργίας** και του δικού μας συστήματος, στο οποίο θα αναφερθούμε εκτενέστερα παρακάτω.

```
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0D, 0x78, 0xE0 };
byte ip[] = { 192, 168, 0, 101 };
byte gateway[] = { 192, 168, 0, 254 };
byte subnet[] = { 255, 255, 255, 0 };
EthernetServer server(5881);
```

Εικόνα 20: Δήλωση Χαρακτηριστικών Δικτύου Server στο Arduino Sketch

Στην "Εικόνα 20" βλέπουμε το κομμάτι του κώδικα στο **sketch** μας, στο οποίο γίνεται η δήλωση των χαρακτηριστικών του server. Αργότερα θα δούμε διεξοδικά όλο το πρόγραμμα και τις βιβλιοθήκες που χρησιμοποιήθηκαν.

byte mac[] = { 0x90, 0xA2, 0xDA, 0x0D, 0x78, 0xE0 }; → είναι το κομμάτι του κώδικα που δηλώνει ότι η MAC Address της συσκευής θα είναι η 90:A2:DA:0D:78:E0.

byte ip[] = { 192, 168, 0, 101 }; → είναι το κομμάτι του κώδικα που δηλώνει ότι η IP του Server στο τοπικό δίκτυο θα είναι η 192.168.0.101.

byte gateway[] = { 192, 168, 0, 254 }; → είναι το κομμάτι του κώδικα που δηλώνει ότι η IP της Πύλης Δικτύου μας θα είναι η 192.168.0.254 και θα μας χρειαστεί αργότερα, στην σύνδεση μας με το TL-WR703N.

byte subnet[] = { 255, 255, 255, 0 }; → είναι το κομμάτι του κώδικα που δηλώνει το Subnet στο οποίο ανήκει ο Server.

EthernetServer server(5881); → είναι το κομμάτι του κώδικα που δηλώνει σε ποια Port θα ακούει ο Server. Επιλέχθηκε τυχαία η θύρα 5881.

[2] Το μοντέλο OSI είναι μια ιεραρχική δομή επτά επιπέδων που καθορίζει τις προδιαγραφές επικοινωνίας μεταξύ δύο υπολογιστών, ορίζοντας επακριβώς τον σκοπό κάθε επιπέδου αλλά και τα χρησιμοποιούμενα πρωτόκολλα. Τα επίπεδα από το κατώτερο προς το ανώτερο είναι τα: Φυσικό, Ζεύξης Δεδομένων, Δικτύου, Μεταφοράς, Συνόδου, Παρουσίασης, Εφαρμογών.

Όπως βλέπουμε λοιπόν, τα χαρακτηριστικά εδώ **δεν δόθηκαν** μέσω της λειτουργίας **DHCP**. Αντιθέτως δώσαμε όλα τα χαρακτηριστικά της συσκευής **χειροκίνητα** για να παραμένουν σταθερά στον Server μας, ώστε να μπορούμε να τον χρησιμοποιούμε απροβλημάτιστα ξέροντας ότι δεν θα αλλάξουν αυτόματα δημιουργώντας μας προβλήματα στην συνέχεια. Επίσης, αν χρησιμοποιούσαμε στο πρόγραμμά μας την λειτουργία DHCP της βιβλιοθήκης, το μέγεθος του Sketch μας θα αυξάνονταν σημαντικά.

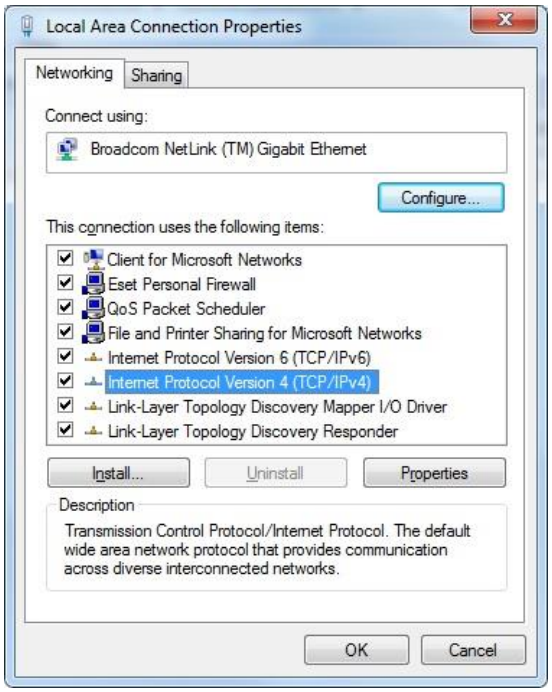
Δυστυχώς με αυτές τις ρυθμίσεις στον Server μας και με το DHCP ενεργοποιημένο στην κάρτα δικτύου του Laptop, υπήρχε θέμα συνδεσιμότητας στο δίκτυο. Έτσι χρειάστηκε να βάλουμε και στο Laptop τις ρυθμίσεις χειροκίνητα ώστε να μπορέσουμε να προχωρήσουμε.

6.4 Ρυθμίσεις Ενσύρματης Κάρτας Δικτύου Υπολογιστή

Στα Windows 7 Ultimate λοιπόν, χρειάστηκε να ρυθμίσουμε χειροκίνητα την κάρτα δικτύου μας απενεργοποιώντας τον DHCP. Για να γίνει αυτό, πήγαμε στο Μενού "**Έναρξη**" και τρέξαμε την εντολή "***ncpa.cpl*** ". Αυτό άνοιξε αμέσως τις συνδέσεις δικτύου του υπολογιστή μας. Επιλέξαμε το "**Local Area Connection**", κάναμε δεξί κλικ πάνω του και επιλέξαμε το "**Properties".**

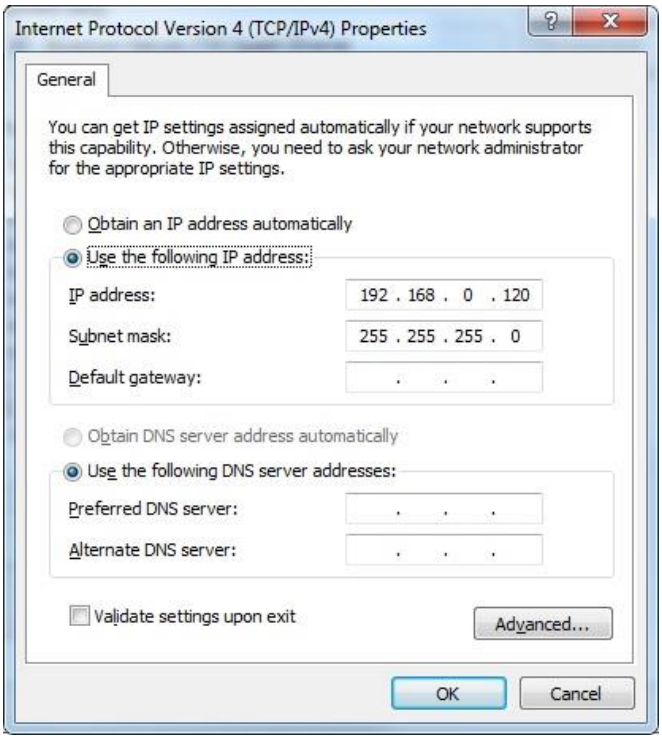
Εικόνα 21: Συνδέσεις Δικτύου – Καλωδιακή Σύνδεση

Στην συνέχεια κάνουμε διπλό κλικ στο "**Internet Protocol Version 4 (TCP/IPv4)**", ώστε να περάσουμε στην παραμετροποίηση των χαρακτηριστικών τις κάρτας δικτύου.



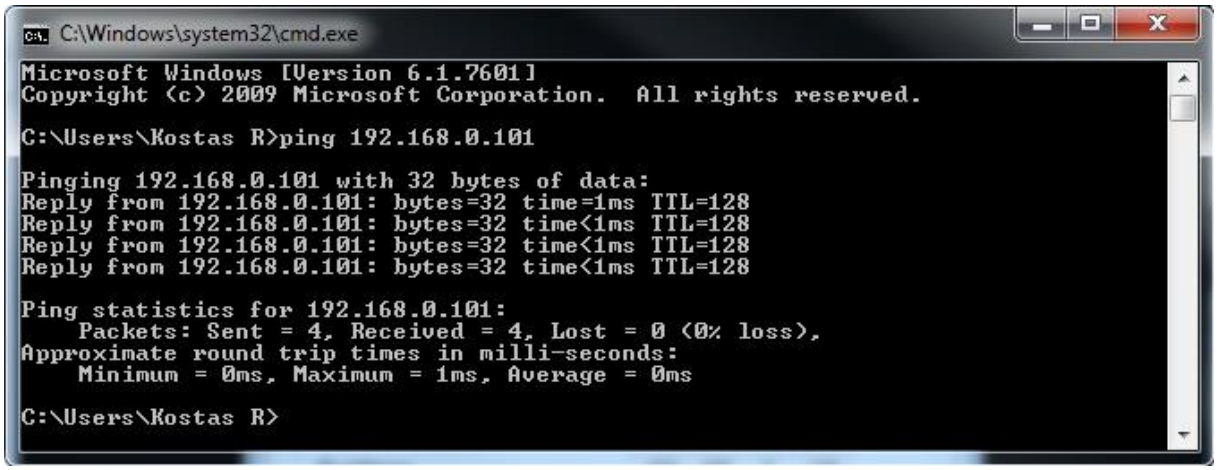
Εικόνα 22: Επιλογή Πρωτοκόλλου που θα Παραμετροποιηθεί

Στη συνέχεια, εκχωρούμε την **στατική IP** "192.168.0.120" στην κάρτα δικτύου όπως και το απαραίτητο **Subnet Mask** "255.255.255.0". Βλέπουμε ότι εκχωρήσαμε μια IP η οποία είναι στο **ίδιο υποδίκτυο** με την IP "192.168.0.1" που δώσαμε στον Arduino Server, ώστε να μπορέσει να "δει" η μια συσκευή την άλλη.



Εικόνα 23: Εκχώρηση Στατικής IP στην Κάρτα Δικτύου του Υπολογιστή

Για να κάνουμε τον τελικό έλεγχο και να βεβαιωθούμε ότι οι δύο συσκευές επικοινωνούν, ανοίγουμε το **Command Prompt** των Windows 7 Ultimate, τρέχοντας την εντολή "**cmd**". Από το **Terminal** εκτελούμε την εντολή "**Ping**" [19] στην IP του Server μας και περιμένουμε να δούμε αν υπάρχει απάντηση.



Εικόνα 24: Απάντηση του Server στο Ping του Υπολογιστή.

Όπως βλέπουμε από την "Εικόνα 24", η απάντηση του Server μας ήταν άμεση, με **Round Trip Time** [3] μικρότερο ή ίσο του 1 ms, ενώ δεν υπήρχαν χαμένα πακέτα κατά την επικοινωνία. Επομένως έχουμε πλέον ένα λειτουργικό δίκτυο και μπορούμε να περάσουμε στο κομμάτι του προγραμματισμού του Server που αφορά το κομμάτι των εντολών που δέχεται ο Server από τους Clients.

6.5 Προγραμματισμός του Server/Microcontroller με τη Χρήση του Arduino IDE

Έχοντας πλέον ένα λειτουργικό δίκτυο, ήρθε πλέον η ώρα της δημιουργίας του sketch, το οποίο θα είναι υπεύθυνο από τη μια να συνδέει το Server στο εκάστοτε δίκτυο, και από την άλλη να είναι ικανό να δέχεται απομακρυσμένες εντολές και να τις εκτελεί στις εξόδους του Microcontroller.

Εκκινούμε λοιπόν την εφαρμογή Arduino IDE. Πριν ξεκινήσουμε να γράψουμε το **Sketch**, θα πρέπει να πάμε στην μπάρα που υπάρχει πάνω από το φύλλο εργασίας και να επιλέξουμε: **Tools → Board → Arduino Uno**, για να επιλέξουμε την πλακέτα με την οποία θα δουλέψουμε. Έπειτα επιλέγουμε πάλι **Tools → Serial Port**, και διαλέγουμε την Σειριακή Θύρα όπου έχουμε συνδέσει το καλώδιο USB από την ηλεκτρονική πλακέτα στον υπολογιστή. Στην συνέχεια ξεκινάμε να γράφουμε το Sketch στην γλώσσα προγραμματισμού Wiring στο φύλλο εργασίας (Εικόνα 12).

Ξεκινάμε πάντα τον κώδικα κάνοντας **Import** τις απαραίτητες **βιβλιοθήκες** για το πρόγραμμά μας.

[3] Ο χρόνος που χρειάζεται ένα σήμα για να σταλεί συν το χρόνο που χρειάζεται για να επιστρέψει πίσω η επιβεβαίωση ότι έφτασε στον προορισμό του εξ' αρχής.

```
#include <SPI.h>
#include <Ethernet.h>


byte mac[] = { 0x90, 0xA2, 0xDA, 0x0D, 0x78, 0xE0 };
byte ip[] = { 192, 168, 0, 101 }; //
byte gateway[] = { 192, 168, 0, 254 }; //
byte subnet[] = { 255, 255, 255, 0 }; //
EthernetServer server(5881);  //

int range;
int analog [] = { 2, 4, 5, 8};

String readString;
```

Εικόνα 25: Εισαγωγή Βιβλιοθηκών και Δήλωση Χαρακτηριστικών Δικτύου και Μεταβλητών

- **#include <SPI.h>** → Εισαγωγή της βιβλιοθήκης SPI (Serial Peripheral Interface) [20]. Είναι η βιβλιοθήκη που επιτρέπει σε SPI συσκευές να επικοινωνούν μεταξύ τους, με το Arduino να είναι η κύρια συσκευή.

- **#include <Ethernet.h>** → Εισαγωγή της βιβλιοθήκης Ethernet [21]. Είναι η βιβλιοθήκη που επιτρέπει στο Arduino να συνδεθεί με το Internet.

Αμέσως μετά ακολουθεί η δήλωση των χαρακτηριστικών του Server μας όπως τα είδαμε στην "Εικόνα 20" και τα αναλύσαμε παραπάνω. Στη συνέχεια έχουμε τη δήλωση κάποιων μεταβλητών καθώς και τη δήλωση των Pins του Arduino που θα χρησιμοποιηθούν και ως αναλογικά.

- **int range;** → Δήλωση της ακέραιης μεταβλητής range.

- **int analog [] = { 2, 4, 5, 8};** → Δήλωση πίνακα που θα χρειαστεί για να κάνουμε χρήση κάποιων Pins ως αναλογικά.

- **String readString;** → Η μέθοδος για να διαβάσουμε τα Requests που θα γίνονται στο Server

Ξεκινάμε με την συνάρτηση Setup() η οποία εκτελείται όταν το Sketch ξεκινάει και χρησιμοποιείται για την τοποθέτηση αρχικών τιμών σε μεταβλητές, Pins λειτουργίας, βιβλιοθηκών κτλ. Στην συνέχεια περνάμε στην ανάθεση των Pins που θα χρησιμοποιήσουμε. Στο παρόν πρόγραμμα θα δουλέψουμε με **8 Pins** και συγκεκριμένα με τα Pins από **2** έως και **9**. Μετά ξεκινάμε την **βιβλιοθήκη του Ethernet** δίνοντας ταυτόχρονα και τα χαρακτηριστικά της συσκευής μας. Έπειτα δίνουμε εντολή στον Server να ξεκινήσει να ακούει για εισερχόμενες συνδέσεις. Όσες γραμμές κώδικα στο Sketch μας κάνουν χρήση του "**Serial**", είναι για λόγους **debugging** μέσω του Serial Monitor που διαθέτει το Arduino IDE, ενώ θα μπορούσαμε μετά την ολοκλήρωση του προγράμματος να τις βάζαμε ως σχόλια χωρίς να επηρέαζε τη λειτουργικότητα του προγράμματός μας.

```
void setup()
{


  for (int i = 2; i < 10; i++)
  {
    pinMode(i, OUTPUT);
  }

  //start Ethernet

  Ethernet.begin(mac, ip, gateway, subnet);
  server.begin();

  //enable serial data print

  Serial.begin(9600);
  Serial.println("Arduino Remote server 1.0");
}
```

Εικόνα 26: Ανάθεση Λειτουργικών Pins και Εκκίνηση του Server

- **for (int i = 2; i < 10; i++)**
  {
  pinMode(i, OUTPUT);
  }
  → Ανάθεση των Pins (2 έως 9) μέσω των οποίων θα ελέγχουμε τις συσκευές μας.

- **Ethernet.begin(mac, ip, gateway, subnet);** → Εκκίνηση βιβλιοθήκης Ethernet και απόδοση χαρακτηριστικών συσκευής.

- **server.begin();** → Εκκίνηση του Server ώστε να μπορεί να αρχίσει να ακούει για εισερχόμενες συνδέσεις.

Στη συνέχεια μπαίνουμε στο κομμάτι του προγράμματος που θα εκτελείται κάθε φορά που θα υπάρχει ένα νέο Request από κάποιον Client.

```
void loop()
{
  EthernetClient client = server.available();
  if (client) {
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
```

Εικόνα 27: Συνάρτηση Loop και Έλεγχος για Clients

- **void loop()** → Επαναλαμβάνει διαδοχικά ότι βρίσκεται μέσα της και επιτρέπει στο πρόγραμμά μας να αλλάζει και να αντιδρά σε εξωτερικές αλλαγές [22].

Το υπόλοιπο κομμάτι του κώδικα ελέγχει για εισερχόμενες συνδέσεις από Clients.

Στην συνέχεια περνάμε στο κομμάτι του κώδικα που διαβάζει **χαρακτήρα προς χαρακτήρα** το HTTP Request που κάνει ο Client και το αποθηκεύει. Όταν δει ότι τελειώνει το request ξεκινάει να τυπώνει δεδομένα στον Client που έκανε το Request.

```
if (readString.length() < 50) {
  readString += c;
}

if (c == '\n') {
  ////////////////
  Serial.println(readString);

  client.println("HTTP/1.1 200 OK"); //send new page
  client.println("Content-Type: text/html");

  client.println();
```

Εικόνα 28: Ανάγνωση HTTP Request και Αποθήκευση σε String

- **if (readString.length() < 50) {**
  readString += c;
  }
  → Ανάγνωση χαρακτήρα προς χαρακτήρα του HTTP Request και αποθήκευσή του.

- **if (c == '\n') {** → Ελέγχει που τελείωσε το HTTP Request.

Το υπόλοιπο κομμάτι κώδικα ξεκινάει την αποστολή μια νέας σελίδα μέσω HTTP με τη μορφή HTML ώστε να μπορέσει να την διαβάσει ο Client.

Συνεχίζουμε με το τμήμα του Sketch που πλέον είναι υπεύθυνο για την **απόδοση τιμών** στα Pins του Server. Η "**for**" επανάληψη εκτελείται 8 φορές, όσα είναι και τα **ψηφιακά** Pins ελέγχου του Server στο Sketch μας. Σε κάθε επανάληψη διαβάζει αν έχει δοθεί εντολή μέσω κάποιου request στο Server και ανάλογα θέτει την έξοδο του αντίστοιχου Pin **ενεργή** ή **όχι**. Τα **Pins 2 έως και 9** έχουν όλα ψηφιακή έξοδο.

```
for (int p=1; p < 9; p++)
{
  if(readString.indexOf("n"+String(p)) >0)
  {
    digitalWrite(p+1, HIGH);
  }
  else {
    if(readString.indexOf("f"+String(p)) >0)
    {
      digitalWrite(p+1, LOW);
    }
  }
}
```

Εικόνα 29: Απόδοση Τιμής Βάσει HTTP Request στα Ψηφιακά Pins 2 έως και 9 του Server

- **for (int p=1; p < 9; p++)** → Η επανάληψη θα εκτελεστεί **8 φορές**, όσα και τα **Digital Pins** του Arduino Server που θα **ελέγχουμε**.

- **if(readString.indexOf("n"+String(p)) >0)** → Το κομμάτι αυτό είναι πολύ σημαντικό. Είναι η στιγμή που ο Server κάνει **Parsing** [23] στο HTTP Request. Στην προκειμένη περίπτωση, **ελέγχει** να δει αν στο Request (μετά την "/" που ακολουθεί την διεύθυνση και τη θύρα του Server) υπάρχει το **γράμμα** "**n**" ακολουθούμενο από τον **αριθμό 1**[4] **έως και 8** [24].

- **digitalWrite(p+1, HIGH);** → Αν υπάρχει το "n1" ή το "n2"…ή το "n8" στο Request, τότε ο Server **ενεργοποιεί** το αντίστοιχο Pin (p+1). Άρα για την πρώτη επανάληψη p=1, δηλαδή για το "n1" θα ενεργοποιηθεί το Pin 2 (p+1=1+1=2). Ομοίως για τα υπόλοιπα Pins έως το 9.

- **if(readString.indexOf("f"+String(p)) >0)** → Όπως και με το "n" έτσι και εδώ γίνεται **Parsing** του Request. Εδώ ο Server όμως **ελέγχει** για το **γράμμα "f"**.

- **digitalWrite(p+1, LOW);** → Αν εδώ αντίστοιχα υπάρχει το "f1" ή το "f2"… ή το "f8" στο Request, τότε ο Server **απενεργοποιεί** το αντίστοιχο Pin.

Επομένως αν κάνουμε ένα HTTP Request της μορφής "**ServerAddress:Port/n1n4n6**", τότε ο Server θα δώσει εντολή να ενεργοποιηθούν τα **Pins 2, 5 και 7**.

Ακολουθεί το κομμάτι του Sketch που είναι υπεύθυνο στο να τυπώσει σε HTML κείμενο την κατάσταση στην οποία θα είναι το κάθε ένα Pin από τα 8 (το οποίο με τη σειρά του αντιστοιχεί σε ένα Device), ώστε να μπορεί να τα διαβάσει ο Client.

```
int w = 2;
while(w < 10)
{
  if (digitalRead(w)==0)
  {
    client.print("{ Device ");
    client.print(w-1);
    client.print(" off }");
  }
  else
  {
    client.print("{ Device ");
    client.print(w-1);
    client.print(" on }");
  }
  w = w+1;
}
```

Εικόνα 30: Τύπωση Κατάστασης των Pins Εξόδου (Devices) του Server

- **int w = 2;**
  while(w < 10)
  {
  → Αρχικοποίηση του "**w**" και εισαγωγή στον **Βρόγχο** ο οποίος θα εκτελεστεί **8 φορές**.

[4] Εδώ η String(p) μετατρέπει τον αριθμό που διαβάζει σε string, ώστε να μπορέσει να διαβαστεί από την indexOf(), η οποία ψάχνει από την αρχή του string (δηλαδή του κομματιού που κάνει το request) για να εντοπίσει το n1,n2 κτλ.

- ```
  if (digitalRead(w)==0)
  {
    client.print("{ Device ");
    client.print(w-1);
    client.print(" off }");
  }
  ```
  → Κάνοντας χρήση της client.print() [25], μπορούμε να τυπώσουμε ως απάντηση HTML κείμενο εύκολα, ώστε να είναι άμεσα και γρήγορα προσπελάσιμο από τον Client που μας έκανε το Request. Εδώ γίνεται έλεγχος στα ψηφιακά Pins από 2 έως 9 και **τυπώνει το State** του καθενός με αρίθμηση από το 1 έως το 8. Αν το **Pin 2 είναι ανενεργό** για παράδειγμα, τότε τυπώνει **{ Device 1 off }**, ενώ αυτό επαναλαμβάνεται διαδοχικά για όλα τα Pins.

- ```
  else
  {
    client.print("{ Device ");
    client.print(w-1);
    client.print(" on }");
  }
  ```
  → Εδώ ομοίως με πριν, **ελέγχει αν είναι ενεργά** τα Pins στο Server και τυπώνει αντίστοιχα για το Pin 2 την φράση **{ Device 1 on }**, ενώ κάνει το ίδιο και για τα υπόλοιπα Pins έως το 9.

Έχοντας αναλύσει πλέον το κομμάτι που αφορά τα ψηφιακά Pins εξόδου, το πώς ένας Client θα μπορεί να ελέγχει απομακρυσμένα τις εξόδους του Server, καθώς και το πως ο Server θα στέλνει στον εκάστοτε client την κατάστασή τους, θα αναφερθούμε σε ένα ολοκληρωμένο παράδειγμα.

Έστω λοιπόν ότι έχουμε 8 ρελέ τα οποία είναι συνδεδεμένα το κάθε ένα σε μια έξοδο ελέγχου οι οποίες αντιστοιχούν στα **Pins** από ένα **2 έως και 9**. Ο Server μας βρίσκεται τοπικά στη διεύθυνση **192.168.0.101** και ακούει στην θύρα **5881**. Η **μόνη ενεργή** έξοδος αυτήν τη στιγμή στον Server μας αντιστοιχεί στο **Pin 9**, δηλαδή στην **συσκευή 8**. Θέλουμε λοιπόν να στείλουμε ένα μήνυμα από τον Client στον Server μας, στο οποίο θα του λέμε να ενεργοποιήσει τις 4 πρώτες συσκευές που αντιστοιχούν στα Pins 2 έως και 5, ενώ θέλουμε να κλείσει την $8^{η}$ συσκευή που αντιστοιχεί στο Pin 9.

Για να γίνει αυτό θα πρέπει να **στείλουμε ένα Get Request**. Ο πιο γρήγορος τρόπος για να γίνει αυτό είναι να ανοίξουμε τον Internet Browser της επιλογής μας και να εκτελέσουμε την εντολή μας από την URL bar του. Στην προκειμένη περίπτωση θα εκτελέσουμε την εντολή "**192.168.0.101:5881/n1n2n3n4f8**". Τα "**n1n2n3n4**" μετά την IP και την Port που ακούει ο Server, θα δώσουν την εντολή στις τέσσερις πρώτες συσκευές να **ανοίξουν** (δηλαδή τα Pins 2 έως και 5 να γίνουν ενεργά), ενώ το "**f8**" θα δώσει την εντολή στην $8^{η}$ συσκευή (η οποία αντιστοιχεί στο Pin 9) που ήταν ανοιχτή να **κλείσει**. Με την ίδια λογική μπορούμε να ανοίγουμε και να κλείνουμε όλα τα Pins εξόδου.

Πλέον περνάμε στο κομμάτι του κώδικα που είναι υπεύθυνο για την αναλογική λειτουργία κάποιων από τα Pins εξόδου. Στο Arduino Uno R3, τα ψηφιακά Pins **3, 5, 6 και 9** από το εύρος των Pins που χρησιμοποιούμε εμείς, μπορούν να χρησιμοποιηθούν και ως **αναλογικές έξοδοι**. Οι τιμές που μπορεί να πάρει το κάθε Pin από αυτά, κυμαίνονται από 0 έως και 255, με το 0 να σημαίνει ότι η έξοδος είναι ανενεργή, και με το 255 να σημαίνει ότι είναι ενεργή. Δηλαδή εδώ η αναλογική τιμή **255**, είναι ισοδύναμη με την **αντίστοιχη ψηφιακή** τιμή του **1**.

```
for (int x = 0; x < 4; x++)
{
  for (range = 0; range < 256; range++)
  {
    if(readString.indexOf("s" + String(analog[x])  + "-" + String(range)) >0)
    {
      analogWrite(analog[x]+1, range);
    }
  }
}
```

Εικόνα 31: Χρήση των Pins 3, 5, 6 και 9 ως Αναλογικών Εξόδων

- **for (int x = 0; x < 4; x++)** → Εκτέλεση του εξωτερικού βρόγχου 4 φορές.

- **for (range = 0; range < 256; range++)** → Εκτέλεση του εσωτερικού βρόγχου 256 φορές, όσο και το πλήθος των αναλογικών τιμών που μπορούν να πάρουν τα Pins 3, 5, 6 και 9.

- **if(readString.indexOf("s" + String(analog[x]) + "-" + String(range)) >0)** → Είναι η στιγμή που ο Server κάνει **Parsing** στο HTTP Request. Στην προκειμένη περίπτωση, **ελέγχει** να δει αν στο Request (μετά την "/" που ακολουθεί την διεύθυνση και τη θύρα του Server) υπάρχει το **γράμμα** "**s**" ακολουθούμενο από τους **αριθμούς 3, 5, 6 ή 9**, ακολουθούμενο από μια παύλα **"-"**, ακολουθούμενο από μια τιμή από το **0** έως το **255**. Εδώ γίνεται χρήση του πίνακα **analog[x]** ο οποίος είναι ο τρόπος για να δηλώσουμε τα Pins που θα έχουν και αναλογική χρήση στο πρόγραμμά μας.

- **analogWrite(analog[x]+1, range);** → Αν υπάρχει για παράδειγμα το "s2-128" στο Request, τότε ο Server **ενεργοποιεί** την αντίστοιχη αναλογική τιμή του Pin του analog[x]+1 η οποία αντιστοιχεί στην προκειμένη περίπτωση στο Pin 3. Ομοίως για τις υπόλοιπες τρεις τιμές του πίνακα analog[x].

Ακολουθεί το τελευταίο κομμάτι του Sketch που θα τρέχει στο Server μας.

```
readString="";

delay(10);

client.stop();
```

Εικόνα 32: Προετοιμασία για το Επόμενο Loop και Τέλος Sketch

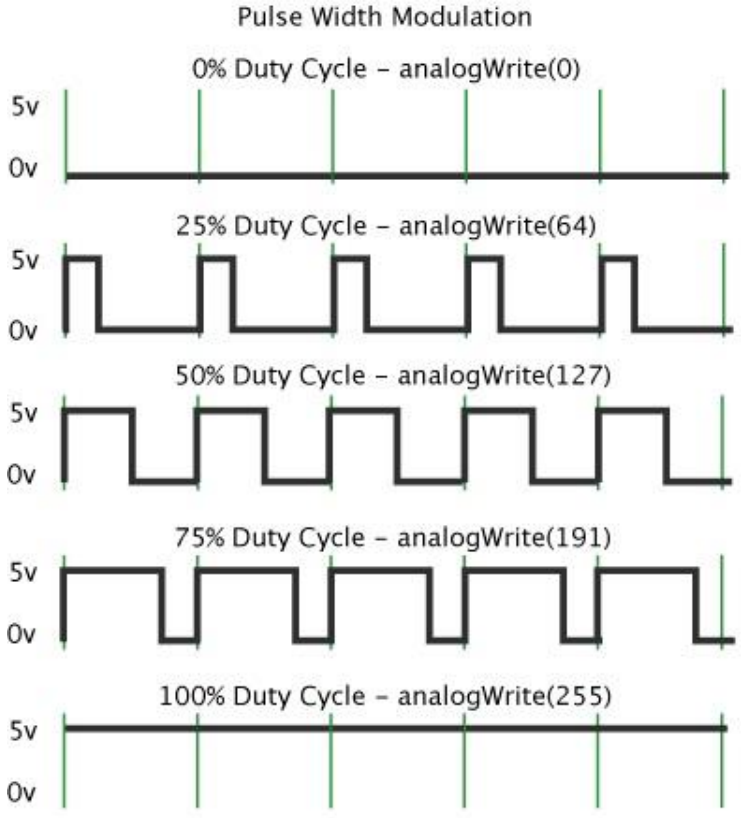
- readString=""; → Η εντολή αυτή καθαρίζει το String ώστε να είναι έτοιμο για το επόμενο Parsing, εάν αυτό υπάρξει στο επόμενο Loop.

- client.stop(); → Αποσύνδεση του Client από τον Server.

Με αυτή την εντολή τελειώνει το Sketch που θα τρέχει στον Server μας. Όταν γράψουμε τον κώδικα επιλέγουμε πάλι στην μπάρα επάνω από το φύλλο εργασίας **Sketch → Verify / Compile** για να ελέγξει εάν στο πρόγραμμα υπάρχουν συντακτικά ή λειτουργικά λάθη. Εάν υπάρχουν μας τα επισημαίνει με έντονο χρώμα. Εάν δεν υπάρχουν επιλέγουμε πάλι **File → Upload** και αρχίζει το φόρτωμα του προγράμματος που γράψαμε.

Πριν περάσουμε στο επόμενο κεφάλαιο, αξίζει να γίνουν κάποιες διευκρινήσεις όσον αφορά την αναλογική λειτουργία της συσκευής μας. Όπως είδαμε στο κομμάτι κώδικα της "**Εικόνας 31**", γίνεται χρήση της εντολής **analogWrite()** [26]. Αυτό που κάνει η **analogWrite()**, είναι να δίνει μια αναλογική τιμή σε ένα Pin, της οποία το **εύρος** κυμαίνεται από **0 έως 255**, με το 0 να αντιστοιχεί σε τάση **0 Volts** και το 255 σε τάση **5 Volts** (το αντίστοιχο της ψηφιακής εξόδου τιμής 1), το οποίο μας επιτρέπει να φωτίσουμε ένα LED σε **διαφορετικά επίπεδα φωτεινότητας** ή να κινήσουμε ένα κινητήρα σε διαφορετικές ταχύτητες.

Ουσιαστικά, αυτό που γίνεται, είναι να δίνεται ένας **PWM** (Pulse Width Modulation) [27] στο εκάστοτε Pin. Το PWM, είναι μια τεχνική η οποία μας επιτρέπει να παίρνουμε "αναλογικά" αποτελέσματα χρησιμοποιώντας ψηφιακά μέσα. Η χρήση της ψηφιακής λειτουργίας γίνεται για να δημιουργηθεί ένας τετραγωνικός παλμός, δηλαδή ένα σήμα που εναλλάσσεται μεταξύ των καταστάσεων on και off. Αυτό το μοτίβο εναλλασσόμενων καταστάσεων μπορεί να εξομοιώσει τάσεις μεταξύ των **5 Volts** (on) και των **0 Volts** (off), με το να **αλλάζει την αναλογία του χρόνου** στον οποίο το σήμα παραμένει στο on, και που στο σήμα παραμένει στο off. Η διάρκεια που περνάει το σήμα στη φάση "on" ονομάζεται **πλάτος του παλμού**. Για να πάρουμε διαφορετικές αναλογικές τιμές, αλλάζουμε ή διαμορφώνουμε το πλάτος του παλμού. Εάν επαναλάβουμε αυτό το μοτίβο μεταξύ on-off αρκετά γρήγορα, κάνοντας χρήση ενός LED για παράδειγμα, το αποτέλεσμα θα είναι σαν αυτό ενός σήματος με κάποια σταθερή τάση μεταξύ των 0 και 5 Volts, ελέγχοντας έτσι την φωτεινότητα του LED. Έτσι λοιπόν ανάλογα την τιμή που θα δώσουμε από το 0 έως το 255, το Arduino κανονίζει βάσει της συχνότητας που λειτουργεί το κάθε Pin, τι διάρκεια θα έχει το πλάτος του παλμού ώστε να δώσει το επιθυμητό αποτέλεσμα (Εικόνα 33). Στο Arduino Uno που χρησιμοποιούμε εμείς για παράδειγμα, τα Pins 5 και 6 έχουν συχνότητα περίπου **980 Hz**, ενώ συνήθως το PWM σε άλλα Pins έχει συχνότητα περίπου ίση με **490 Hz**.



Εικόνα 33: PWM στο Arduino στις Αναλογικές Τιμές από 0 έως 255

Τελειώνοντας το Sketch μας και έχοντας πλέον τον Server έτοιμο και ικανό να ανταποκριθεί σε Client Requests, θα αναλύσουμε την μέθοδο καθώς και τις κατάλληλες ρυθμίσεις των συσκευών του δικτύου μας, με τις οποίες θα κάνουμε τον **Server / Microntroller** μας **προσπελάσιμο** από οποιοδήποτε σημείο του πλανήτη υπάρχει **δίκτυο**, χρησιμοποιώντας ταυτόχρονα μια DDNS υπηρεσία.

7. Ρύθμιση του TL-WR703N, των Routers του Τοπικού Δικτύου, Καθώς και Αξιοποίηση της DDNS Υπηρεσίας.

Σε αυτό το κεφάλαιο θα γίνει αναλυτική περιγραφή των ρυθμίσεων που έγιναν στο **TL-WR703N** ώστε να δουλέψει σε **Client mode**, καθώς και τις επιπλέον ρυθμίσεις που χρειάστηκε ώστε να λειτουργεί απρόσκοπτα και άμεσα, κάθε φορά που θα συνδέεται στο Arduino αλλά και στις υπόλοιπες συσκευές του δικτύου. Επίσης θα γίνει περιγραφή στις ρυθμίσεις που έγιναν στα Routers του τοπικού δικτύου, καθώς και πως αξιοποιήθηκε η **DDNS** υπηρεσία για να γίνει προσπελάσιμος ο Server μας από τον έξω κόσμο.

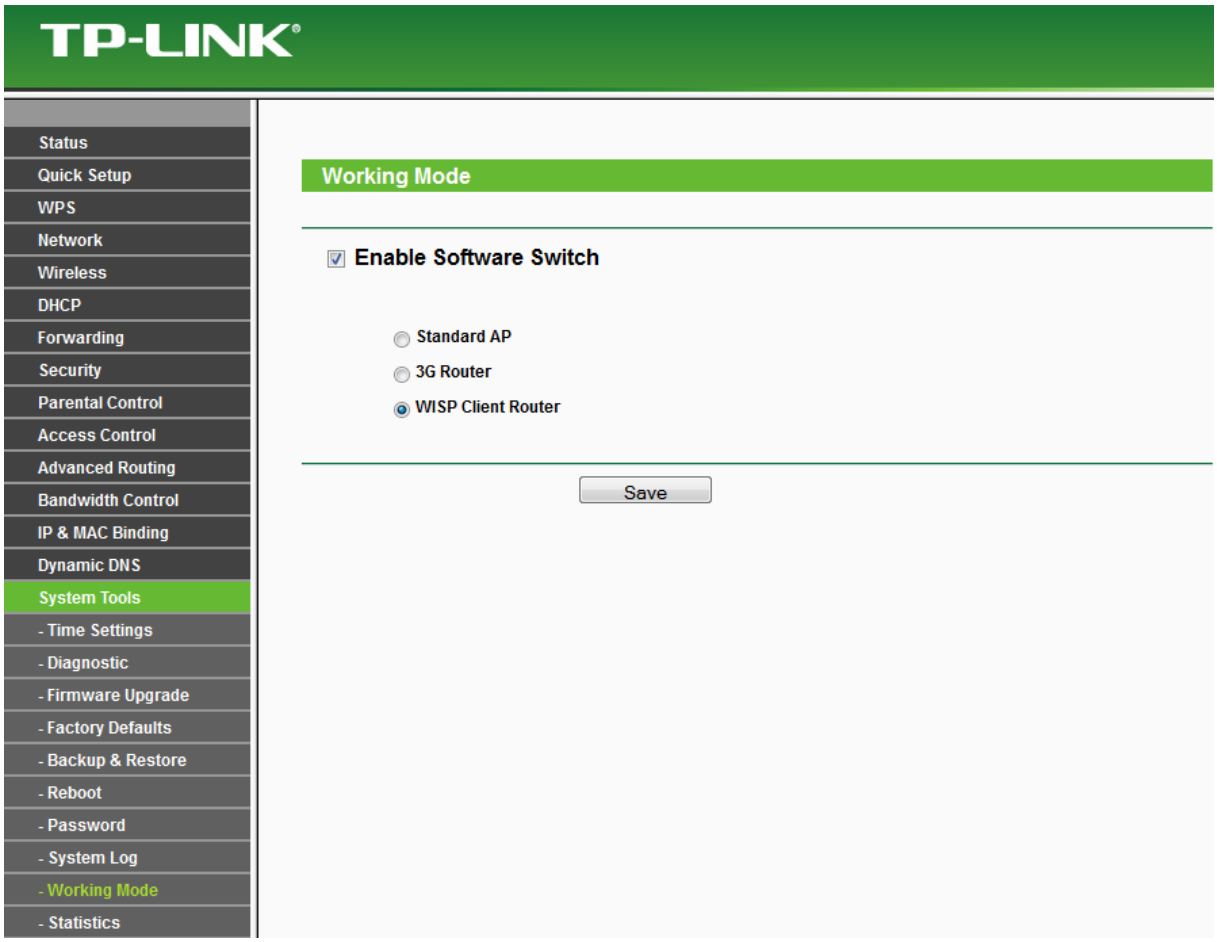
7.1 Ρυθμίσεις του TL-WR703N

Για να έχουμε πρόσβαση στο μενού του TL-WR703N θα πρέπει να συνδεθούμε με το Laptop απευθείας πάνω στη συσκευή μέσω Wi-Fi. Την επιλέγουμε λοιπόν από την λίστα με τις διαθέσιμες συσκευές που κάνουν broadcast και εισάγουμε τον εργοστασιακό κωδικό ασφαλείας δικτύου για να συνδεθούμε.

Εικόνα 34: Σύνδεση στο TL-WR703N

Στην συνέχεια εισάγουμε την **192.168.0.254** που είναι και η IP της συσκευής μας στην URL bar του Firefox, εισάγουμε τα απαραίτητα **Username** και **Password** και εισερχόμαστε στο μενού της συσκευής.

Αρχικά θα πρέπει να θέσουμε την συσκευή μας σε λειτουργία κατάλληλη ώστε να μπορεί να συμπεριφερθεί ως **κάρτα για ασύρματο δίκτυο**. Επιλέγουμε λοιπόν από το μενού στα αριστερά **System Tools → Working Mode**. Έπειτα ενεργοποιούμε το **Software Switch**, και επιλέγουμε το "**WISP Client Router**".



Εικόνα 35: Επιλογή Τύπου Λειτουργίας TL-WR703N

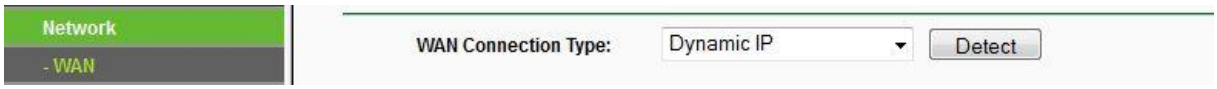
Εδώ ενεργοποιήσαμε το Software Switch, διότι το μοντέλου του οποίου το firmware χρησιμοποιούμε έχει εξωτερικό διακόπτη εναλλαγής λειτουργίας πάνω στην συσκευή, επομένως ήταν απαραίτητο να γίνει για να μην υπάρξουν προβλήματα στην λειτουργία της συσκευής.

Το **WISP Client Router**[5] [28] που διαλέξαμε, χρησιμοποιεί την θύρα **Ethernet** της συσκευής μας ως θύρα για το **LAN** κομμάτι του δικτύου που δημιουργούμε (TL-WR703N και Arduino Server), ενώ η **Ασύρματη** κεραία της συσκευής μας, θα έχει λειτουργία θύρας **WLAN** , η οποία θα μας συνδέσει ασύρματα με τον επόμενο κόμβο του δικτύου μας που είναι το WRT54, ενώ ταυτόχρονα θα δημιουργήσει μια γέφυρα δύο κατευθύνσεων μεταξύ τους, ενώνοντας ουσιαστικά δύο τοπικά δίκτυα μεταξύ τους. Το πώς ακριβώς θα γίνει αυτό θα το αναλύσουμε στην επόμενη ενότητα (7.2).

Οι θύρες ενός Router χωρίζονται σε δύο κύριες κατηγορίες, τις **LAN** και τις **WAN**. Οι θύρες WAN χρησιμοποιούνται πάντα από ένα Router για να προωθήσουν την κίνηση του δικτύου προς άλλα Routers, ενώ οι θύρες LAN χρησιμοποιούνται για να επικοινωνούν μεταξύ τους οι εσωτερικοί Clients ενός δικτύου. Εάν ένας υπολογιστής συνδεόταν στην εξωτερική WAN θύρα ενός Router αντί σε μια από τις LAN θύρες, τότε το Router θα προωθούσε την κίνηση προς αυτόν τον υπολογιστή νομίζοντας έτσι ότι τα δεδομένα "αφήνουν" το δίκτυο. Χωρίς κάποιον τρόπο λοιπόν ο υπολογιστής να προωθήσει τα δεδομένα, αυτά δεν θα έφταναν ποτέ στον προορισμό τους.

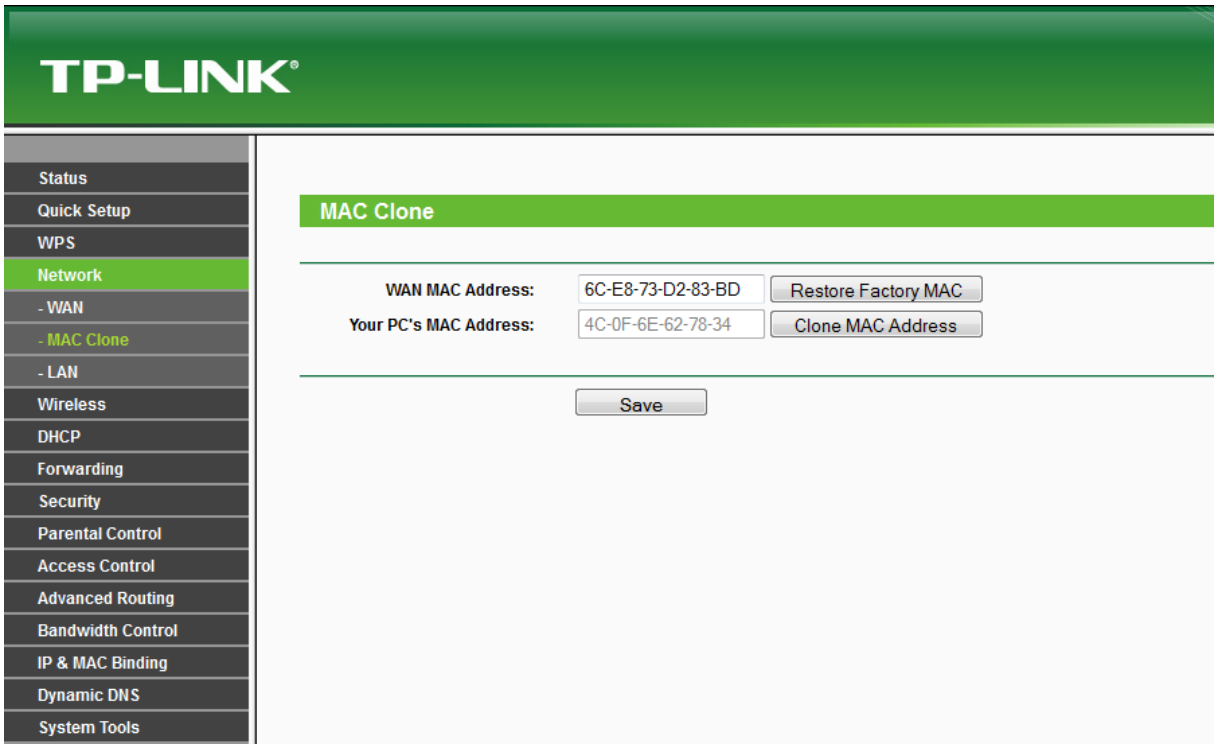
Συνεχίζοντας τις ρυθμίσεις της συσκευής μας, διαλέγουμε πάλι από το μενού στα αριστερά **Network → WAN**. Από τα δεξιά, στο "**WAN Connection Type**", επιλέγουμε το **Dynamic IP**, ώστε η συσκευής μας να παίρνει **αυτόματα IP** από το **WRT54G** που θα συνδεθεί αργότερα.

[5] WISP: Είναι τα αρχικά για το Wireless Internet Service Provider. Στην δική μας περίπτωση το ρόλο αυτό θα αναλάβει το WRT54G. Δεν χρειάζεται να υπάρχει κάποιος πάροχος που να δίνει Wireless Internet ώστε να συνδεθούμε του πάνω για να εκμεταλλευτούμε αυτή την λειτουργία. Οποιοδήποτε Router με δυνατότητα σύνδεσης στο Internet μπορεί να αναλάβει αυτόν τον ρόλο.



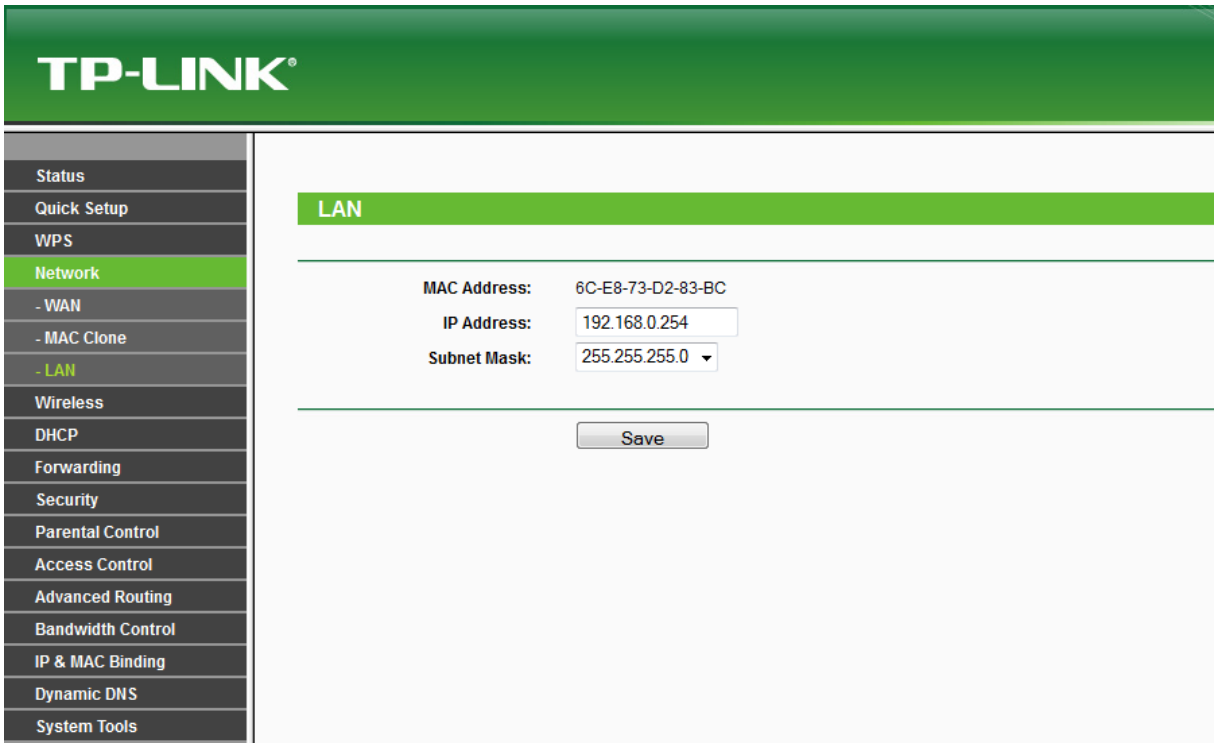
Εικόνα 36: Επιλογή Απόδοσης Δυναμικής IP για το WAN μέρος του TL-WR703N

Στην συνέχεια πάμε από το μενού στα δεξιά και διαλέγουμε το **Network → MAC Clone**. Από εκεί βλέπουμε με μια ματιά την MAC Address που αντιστοιχεί στην WAN port της συσκευής μας. Στην προκειμένη περίπτωση είναι η **6C:E8:73:D2:83:BD**. Την καταγράφουμε διότι θα μας χρειαστεί αργότερα στην ενότητα **7.2**, στις ρυθμίσεις του **WRT54G**.



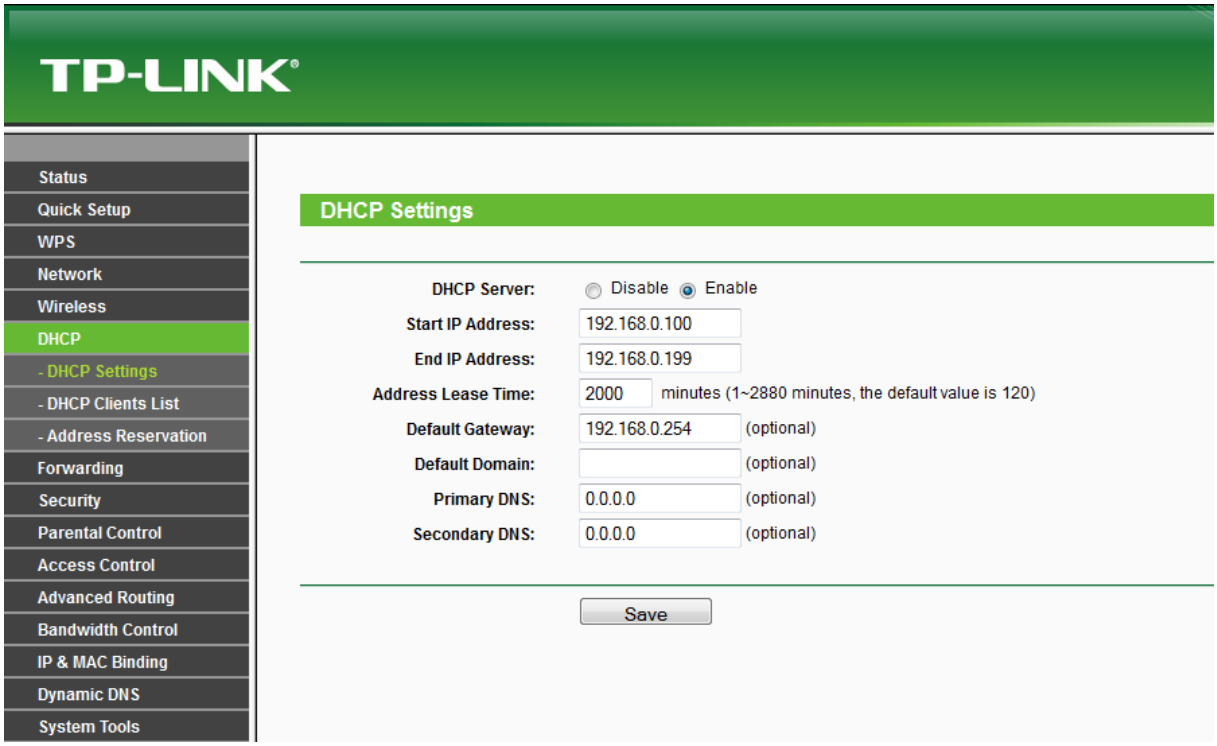
Εικόνα 37: Καταγραφή της WAN MAC Address

Στην συνέχεια επιλέγουμε από το μενού στα αριστερά **Network → LAN**. Εδώ μπορούμε να δούμε την MAC Address που αντιστοιχεί στις LAN θύρες της συσκευής και είναι η **6C:E8:73:D2:83:BD**. Εδώ παρατηρούμε ότι κάθε **physical interface** της συσκευής έχει τη δική του MAC Address, που είναι και το σωστό. Επιπλέον σε αυτό το μενού βλέπουμε την **LAN IP Address** της συσκευής μας, η οποία είναι και αυτή που χρησιμοποιήσαμε στην αρχή για να έχουμε πρόσβαση στο μενού της. Τέλος βλέπουμε το **Subnet Mask** το οποίο καθορίζει και το μέγεθος του δικτύου μας. Στην προκειμένη περίπτωση το καταχωρημένο υποδίκτυο μας είναι πολύ μεγαλύτερο από ότι χρειαζόμαστε, αλλά μιας και δεν είναι απαραίτητη η αλλαγή σε κάτι μικρότερο, το αφήνουμε ως έχει.



Εικόνα 38: LAN Mac Address, LAN IP Address και Subnet Mask

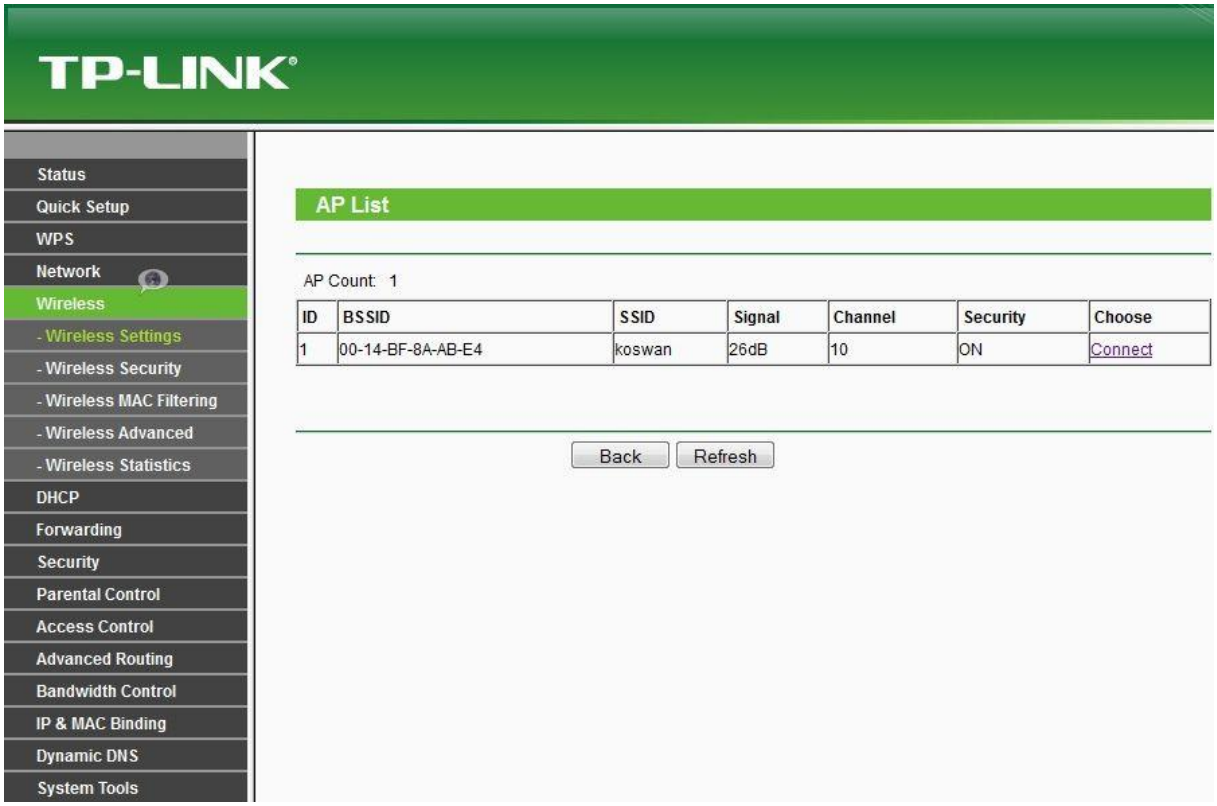
Στο επόμενο βήμα θα πρέπει να ενεργοποιήσουμε την λειτουργία **DHCP Server** για το τοπικό δίκτυο, ώστε να αποδίδονται αυτόματα IP στις συσκευές που συνδέονται τοπικά (όπως το Laptop με το οποίο έχει γίνει η σύνδεση για τη ρύθμιση). Συνήθως υπάρχει ως προεπιλογή για να διευκολύνει την σύνδεση σε νέα δίκτυα. Επιλέγουμε λοιπόν από το μενού στα αριστερά **DHCP → DHCP Settings**.



Εικόνα 39: LAN DHCP Settings

Στην συνέχεια επιλέγουμε **DHCP Server → Enable** ώστε να κάνουμε ενεργή την λειτουργία. Στο **Start** και **End IP Address** βάζουμε την πρώτη και την τελευταία IP που θέλουμε να αποδίδει ο DHCP, θέτοντας έτσι ταυτόχρονα το πλήθος των Clients που μπορούν να συνδεθούν τοπικά όπως και την πισίνα των εν δυνάμει χρησιμοποιηθέντων IPs. Στο δικό μας **Use Case**, έχουμε ανάγκη μόνο έναν Client που του αντιστοιχεί η 192.168.0.101 IP (η οποία προφανώς βρίσκεται μέσα στο δοθέν εύρος), αλλά δώσαμε μεγάλο εύρος στην προκειμένη περίπτωση, για ευκολία στον πειραματισμό. Βλέποντας λοιπόν την "Εικόνα 39", παρατηρούμε ότι η πρώτη πιθανή IP που μπορεί να πάρει ένας Client είναι η 192.168.0.100, ενώ η τελευταία είναι 192.168.0.199, άρα 100 πιθανοί Clients στο σύνολο.

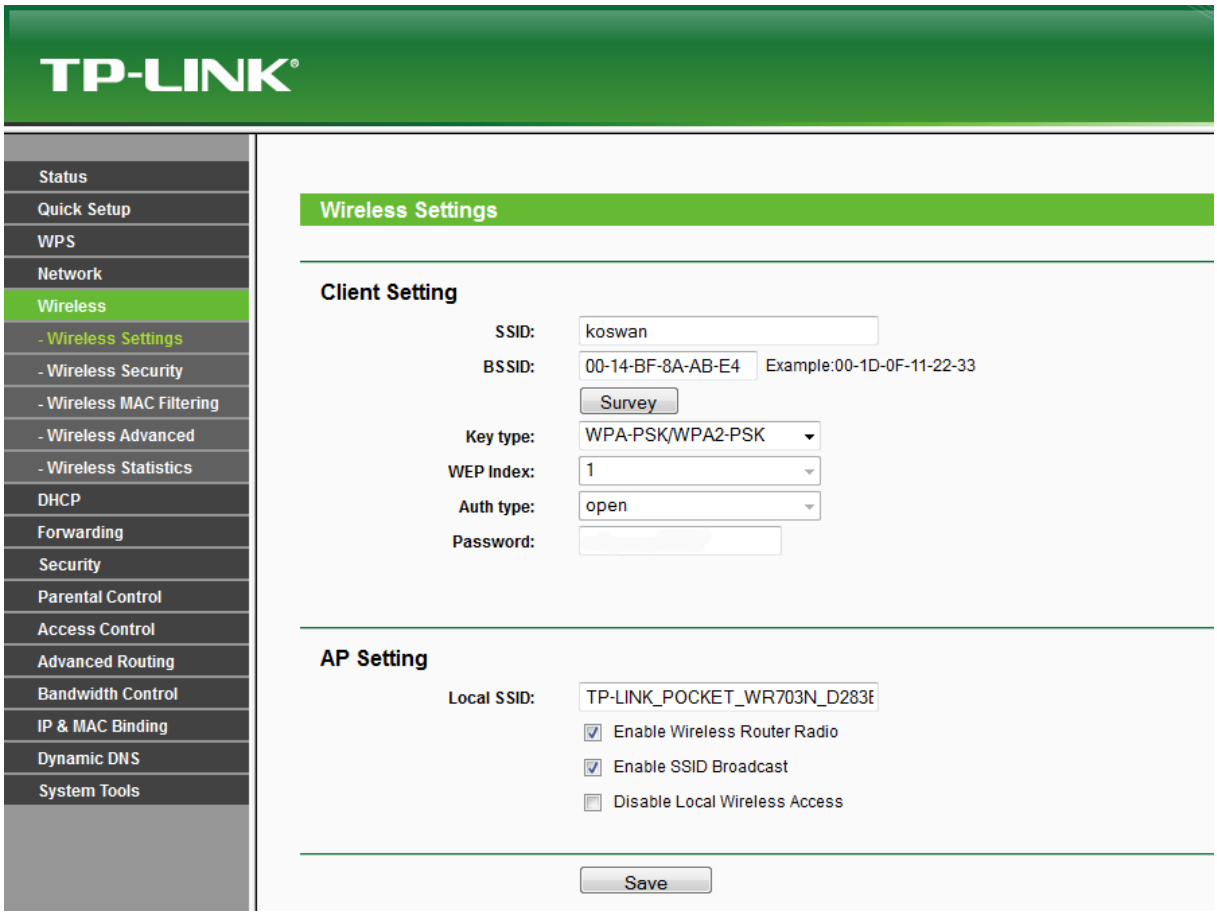
Στο επόμενο βήμα των ρυθμίσεων του TL-WR703N, θα κάνουμε τις απαραίτητες ρυθμίσεις ώστε να συνδεθούμε **ασύρματα** με επιτυχία στο WRT54G. Επιλέγουμε λοιπόν από το μενού στα αριστερά **Wireless → Wireless Settings**. Στη δεξιά μεριά πατάμε το κουμπί "**Survey**" ώστε να γίνει αναζήτηση για ανιχνεύσιμα Access Points που εκπέμπουν το **SSID** τους, ώστε να διαλέξουμε σε ποιο επιθυμούμε να συνδεθούμε.



Εικόνα 40: Λίστα Διαθέσιμων AP

Διαλέγουμε το "**koswan**", πατώντας το κουμπί "**connect**", που είναι και το **SSID**[6] του δικτύου που εκπέμπει το WRT54G και είναι αυτό στο οποίο σκοπεύουμε να συνδεθούμε. Μόλις εκτελεστεί αυτό επιστρέφουμε στην προηγούμενη καρτέλα. Εδώ συμπληρώνουμε το **Password** που αντιστοιχεί στο **Key Type** που χρησιμοποιεί το WRT54G, ώστε να μπορέσουμε να συνδεθούμε, ενώ βλέπουμε ότι στην προκειμένη περίπτωση γίνεται χρήση του πρωτοκόλλου ασφάλειας **WPA2** [29]. Αν δεν βρει αυτόματα το πρωτόκολλο ασφάλειας το διαλέγουμε εμείς από το Dropdown Menu.

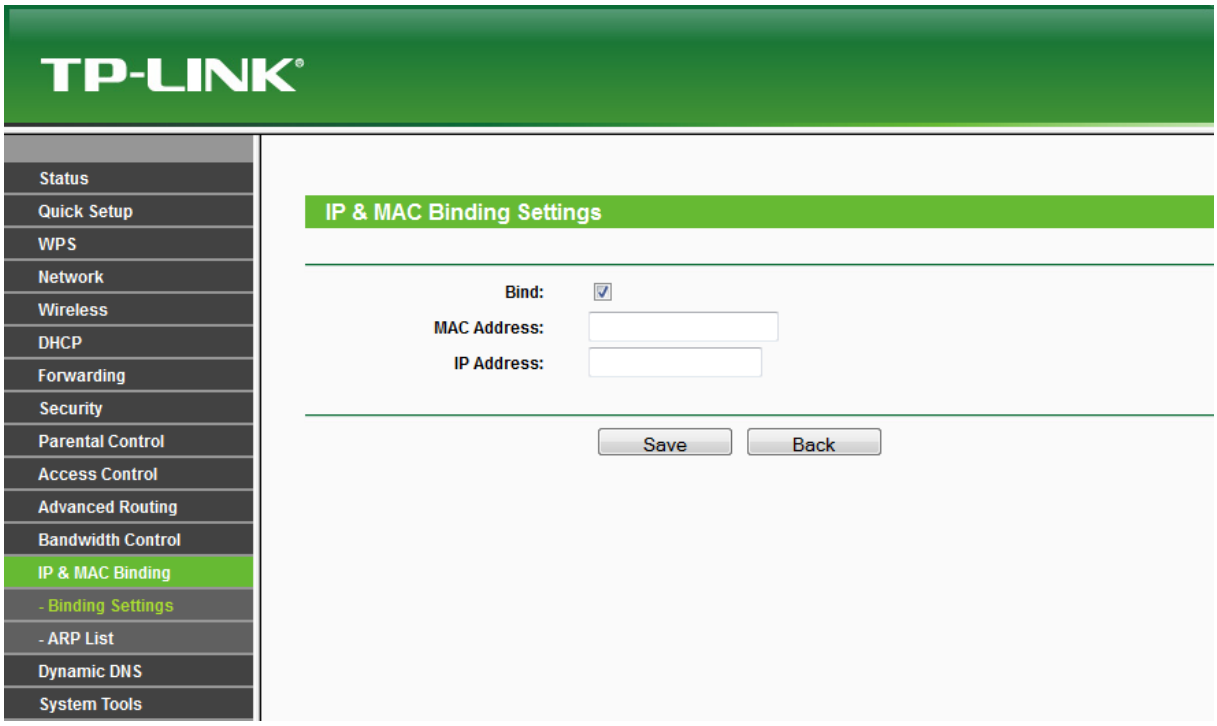
[6] Το SSID (Service Set Identifier) είναι ένα string μεγέθους 1 έως 32 byte και είναι συνήθως μια ανθρώπινα αναγνώσιμη ακολουθία χαρακτήρων, γνωστή και ως "Όνομα Δικτύου".



Εικόνα 41: Σύνδεση με WRT54G

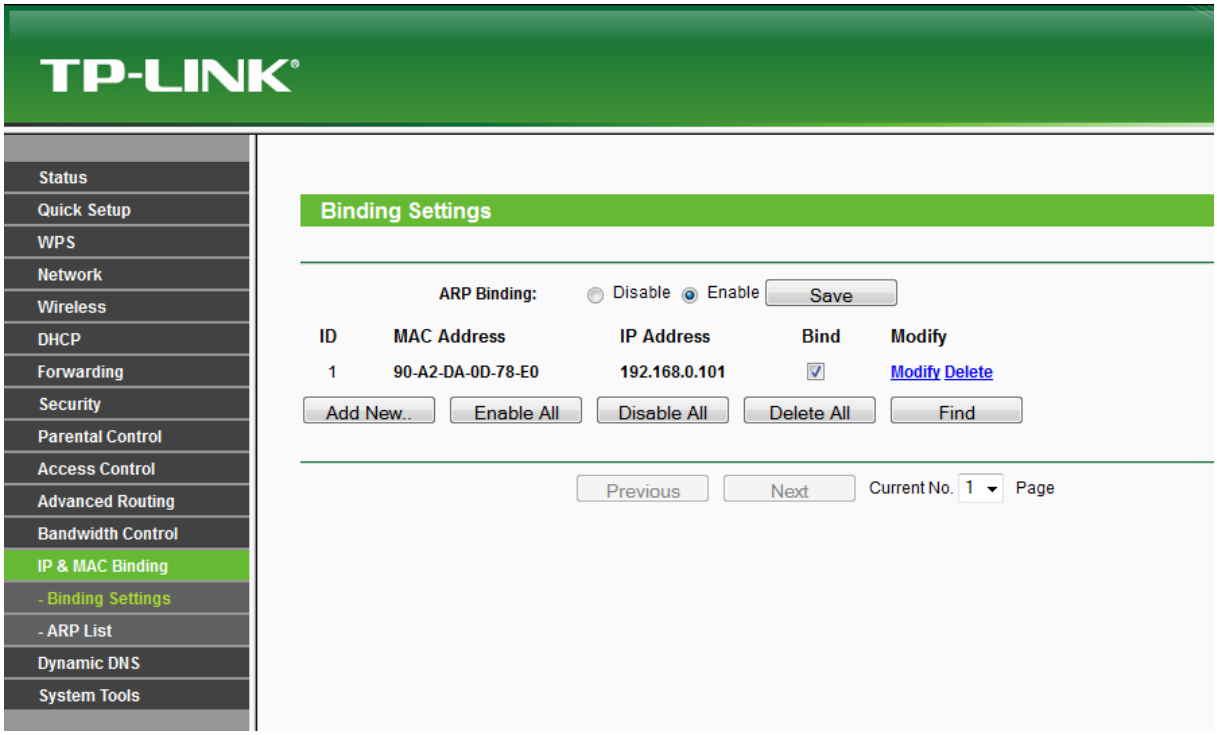
Όπως αναφέραμε στην προηγούμενη ενότητα, ο Arduino Server είναι προγραμματισμένος να παίρνει πάντα την **192.168.0.101** ως IP στο τοπικό δίκτυο που βρίσκεται. Πρέπει λοιπόν να ρυθμίσουμε το TL-WR703N κατάλληλα, ώστε να **αποδίδει την συγκεκριμένη IP μόνο στον Server** μας. Αν δεν το κάνουμε αυτό, η συσκευή μπορεί να προσπαθεί να αποδώσει διαφορετική IP στον Server μετά από κάποια επανεκκίνηση (π.χ. λόγω διακοπής ρεύματος) ή επειδή την έχει δεσμεύσει ήδη κάποια άλλη συσκευή (όπως ένα κινητό) στο τοπικό δίκτυο, καθιστώντας έτσι αδύνατη τη σύνδεση μεταξύ τους.

Επιλέγουμε λοιπόν από το μενού στα αριστερά **IP & MAC Binding → Binding Settings**. Στη συνέχεια επιλέγουμε "**Enable**" **ARP Binding** και κάνουμε **Save**. Στη συνέχεια πατάμε το κουμπί "**Add New**" για να προχωρήσουμε στην **δέσμευση** της IP.



Εικόνα 42: Δέσμευση IP Βάσει μιας MAC Address στο Τοπικό Δίκτυο

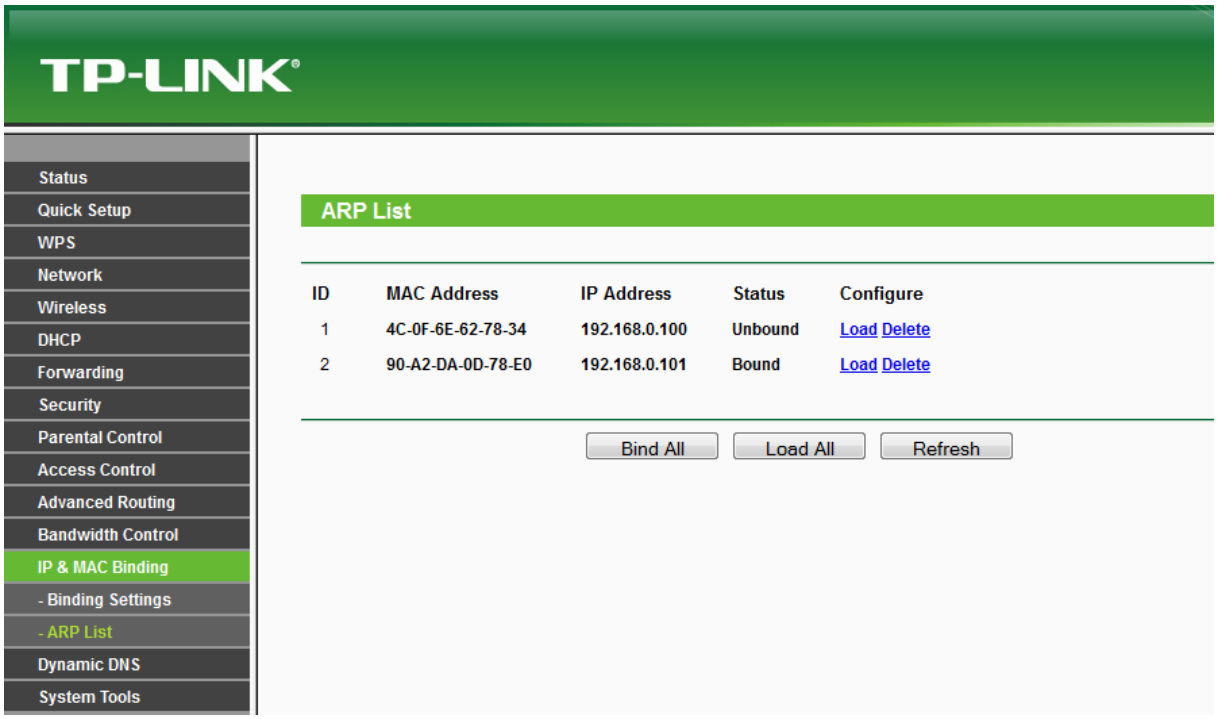
Στο πεδίο **MAC Address** συμπληρώνουμε την MAC Address που έχουμε δώσει στον Server μας όπως είδαμε στην ενότητα 6.5 και συγκεκριμένα την **90:A2:DA:0D:78:E0**. Στο πεδίο IP Address συμπληρώνουμε ποια IP θέλουμε να αποδίδεται κάθε φορά που θα συνδέεται η συσκευή με τη συγκεκριμένη MAC Address, δηλαδή ο Arduino Server. Στην προκειμένη περίπτωση λοιπόν συμπληρώνουμε την IP 192.168.0.1. Τέλος ενεργοποιούμε την ρύθμισή μας τσεκάροντας την επιλογή "**Bind**" και κάνουμε **Save**.



Εικόνα 43: Overview των Binding Settings του TL-WR703N

Πλέον κάθε φορά που θα συνδέεται ο Server μας μέσω Ethernet καλωδίου με το TL-WR703N, θα παίρνει την IP 192.168.0.101.

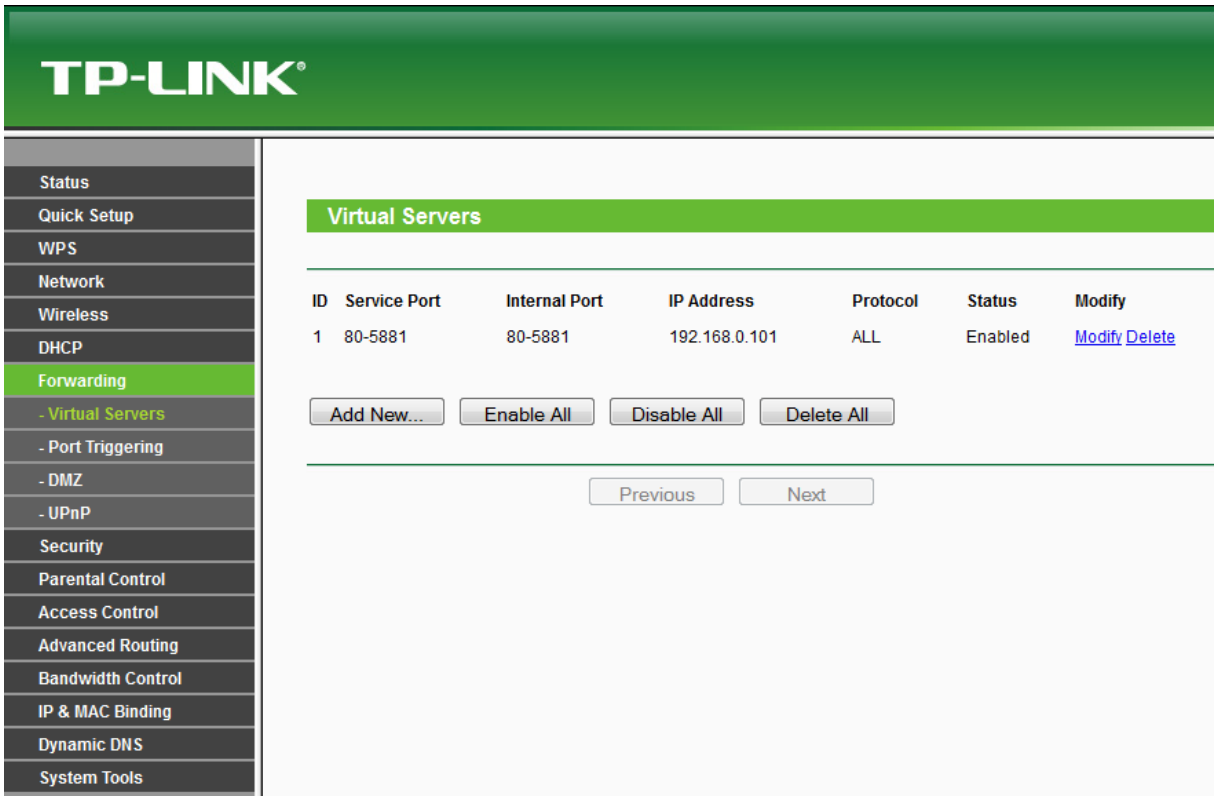
Πηγαίνοντας στο μενού **IP & MAC Binding → ARP List** (Address Resolution Protocol), μπορούμε να δούμε όλες τις IP Addresses και τις σχετιζόμενες με αυτές MAC Addresses, καθώς και να κάνουμε διαχείριση της λίστας αυτής. Το **ARP** [30] χρησιμοποιείται για να μετατρέψει μια IP Address σε μια φυσική διεύθυνση όπως είναι η MAC Address. Στην παρακάτω εικόνα βλέπουμε ότι η MAC του Server έχει αντιστοιχηθεί με την IP "192.168.0.101".



Εικόνα 44: ARP List στο TL-WR703N

Στο επόμενο βήμα θα πάμε να κάνουμε την κατάλληλη ρύθμιση ώστε κάθε φορά που γίνεται μια αίτηση από κάποιον Client σε μια συγκεκριμένη **Service Port**, το αίτημα να προωθείται στον Arduino Server. Εμείς έχουμε δηλώσει στο **Sketch** του Server ότι θα κάνουμε χρήση της θύρας 5881. Άρα λοιπόν θα πρέπει κάθε αίτημα που γίνεται από κάποιον Client προς την θύρα 5881, να δρομολογείται προς τον Arduino Server. Σε αυτό το σημείο φαίνεται ότι είναι **απαραίτητο** να έχουμε δώσει **στατική IP** στον Server μας, για να μπορέσει να γίνει η δρομολόγηση.

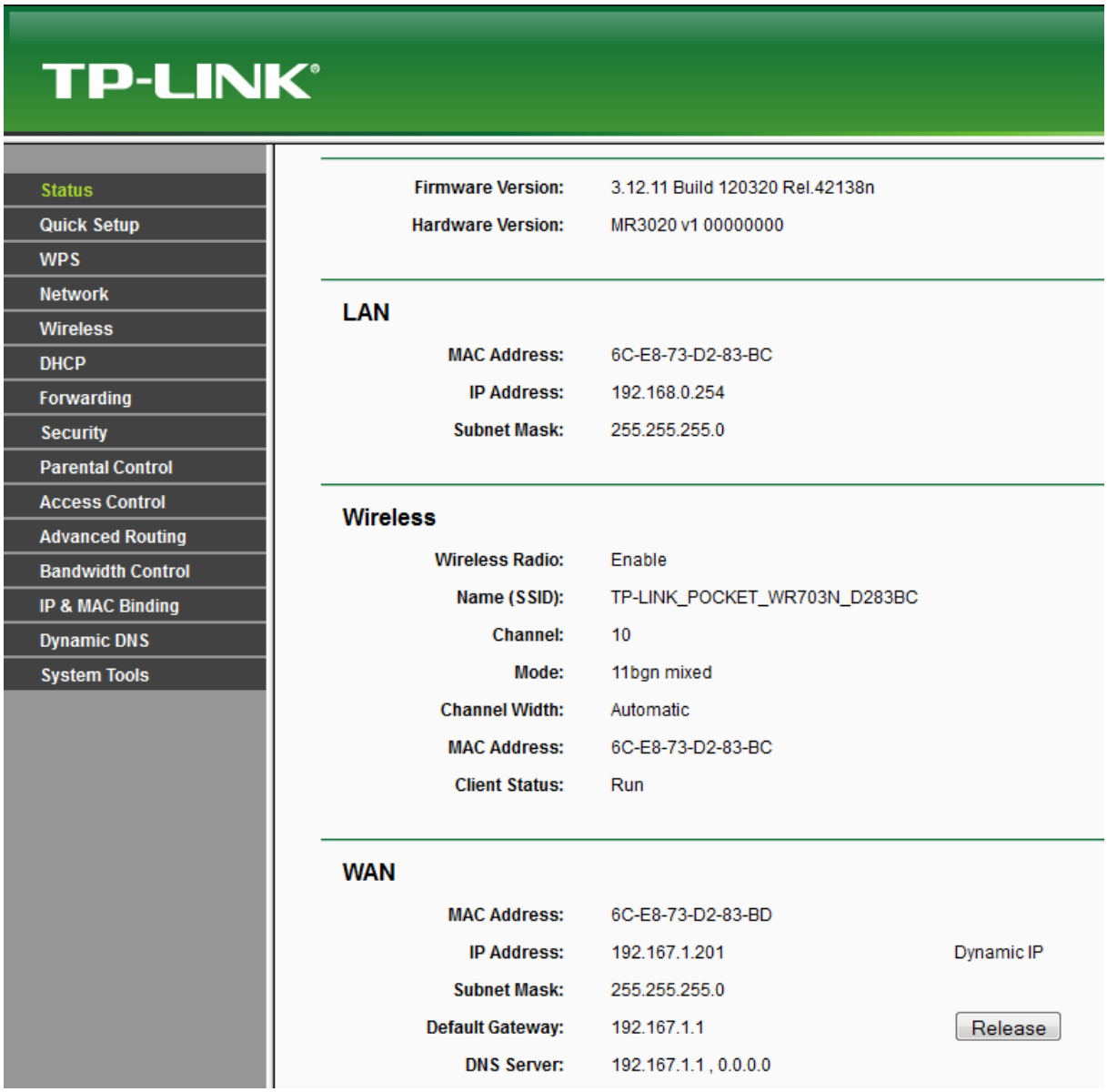
Θα κάνουμε λοιπόν χρήση μια τεχνικής **Port Forwarding** [31], η οποία επιτρέπει όλες οι αιτήσεις από το Internet προς μια συγκεκριμένη θύρα, να **ανακατευθύνονται** σε μια συγκεκριμένη IP. Έτσι θα αποστέλλονται όλα τα πακέτα από τις αιτήσεις στην θύρα 5881, προς τον Arduino Server.



Εικόνα 45: Ρυθμίσεις Port Forwarding στο TL-WR703N

Για να γίνει αυτό, επιλέγουμε από το μενού **Forwarding → Virtual Servers**. Στα δεξιά πατάμε το κουμπί **Add New** και στην καρτέλα που ανοίγει συμπληρώνουμε τα πεδία. Στο πεδίο **Service Port**, συμπληρώνουμε το εύρος θυρών ή την θύρα στην οποία θα ακούει η υπηρεσία ώστε να κάνει τη δρομολόγηση προς τον Server. Επομένως η θύρα που πρέπει να συμπληρωθεί είναι η **5881**. Για ευκολία στον πειραματισμό βάλαμε όλες οι θύρες από την **80-5881** να κάνουν δρομολόγηση προς τον Server. Στο πεδίο **IP Address** συμπληρώνουμε την εσωτερική IP του Server μας και προς την οποία θα γίνεται η δρομολόγηση. Στο πεδίο **Protocol** επιλέγουμε "**ALL**", δηλαδή θα γίνεται η δρομολόγηση είτε γίνεται χρήση του **TCP**[7], είτε του **UDP** πρωτοκόλλου. Τέλος στο πεδίο **Status** επιλέγουμε το "**Enabled**" ώστε να καθοριστεί ενεργή η ρύθμισή μας.

[7] Το TCP (Transmission Control Protocol) είναι ένα κεντρικό πρωτόκολλο από την Internet Protocol Suite, το οποίο προσφέρει μια αξιόπιστη, με σειρά και ελεγμένη από λάθη, συνεχή παροχή πληροφοριών μεταξύ προγραμμάτων που τρέχουν σε υπολογιστές οι οποίοι είναι συνδεμένοι σε ένα τοπικό δίκτυο ή στο Internet, ενώ βασίζεται στο επίπεδο μεταφοράς του μοντέλου OSI.



Εικόνα 46: Overview των Χαρακτηριστικών του TL-WR703N

Έχοντας πλέον κάνει τις απαραίτητες ρυθμίσεις στο TL-WR703N, πηγαίνουμε στο μενού **Status**, ώστε να δούμε όλες τις πληροφορίες της συσκευής. Από την παρακάτω εικόνα, παρατηρούμε ότι στο **WAN** έχει δοθεί στο TL-WR703N η IP "**192.167.1.201**". Αυτή την IP η συσκευή την πήρε δυναμικά λόγω ρυθμίσεων όπως είπαμε και προηγουμένως, αλλά η IP **δεν αποδόθηκε δυναμικά** από το WRT54G όπως θα δούμε παρακάτω.

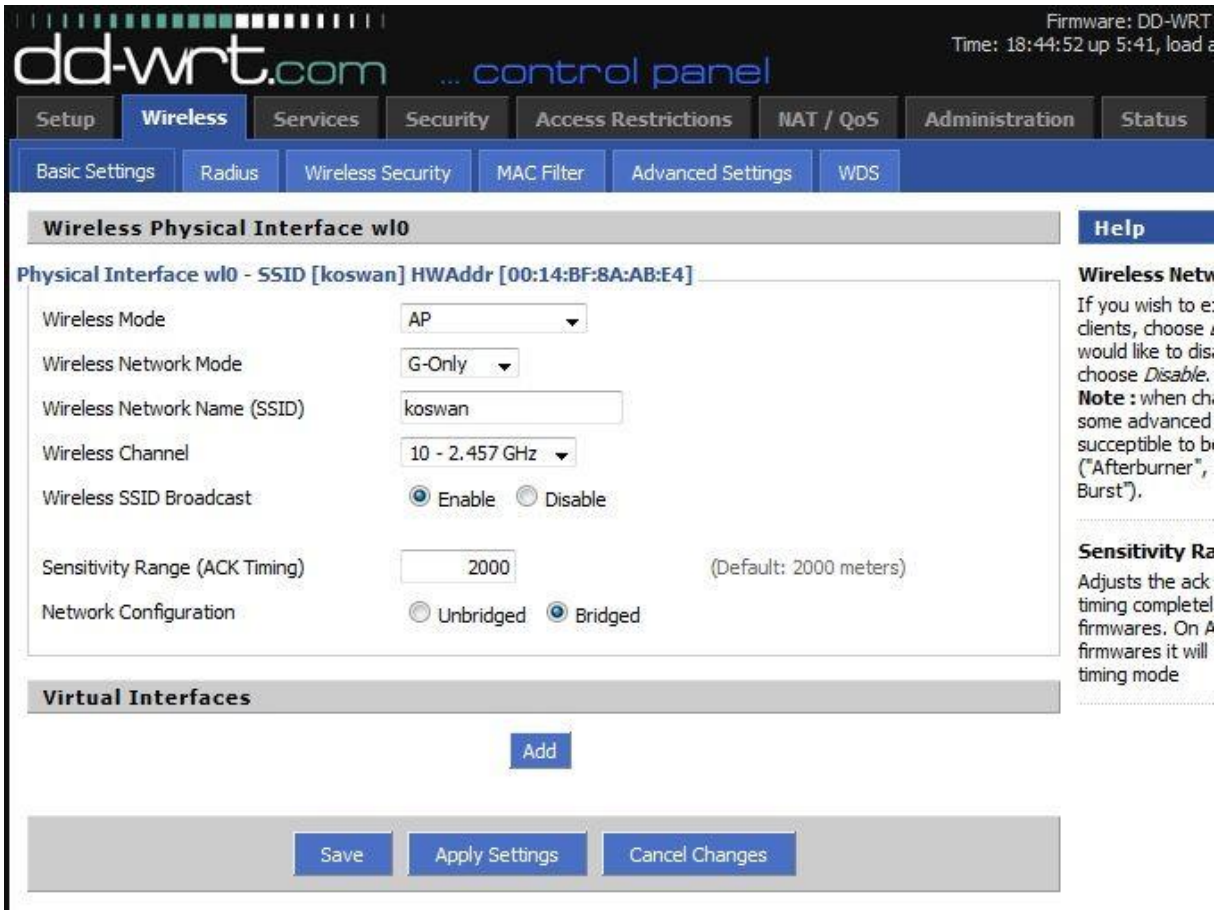
7.2 Ρυθμίσεις του Linksys WRT54G και Δημιουργία DDNS Hostname

Έχοντας πλέον δει τις απαραίτητες ρυθμίσεις στο TL-WR703N, περνάμε στον επόμενο κόμβο μας, το **WRT54G**. Για να έχουμε πρόσβαση στο μενού του WRT54G θα πρέπει να συνδεθούμε με το Laptop απευθείας πάνω στη συσκευή μέσω Wi-Fi. Την επιλέγουμε λοιπόν από την λίστα με τις διαθέσιμες συσκευές που κάνουν broadcast και εισάγουμε τον εργοστασιακό κωδικό ασφαλείας δικτύου για να συνδεθούμε.

Εικόνα 47: Σύνδεση στο Δίκτυο στα Windows 7

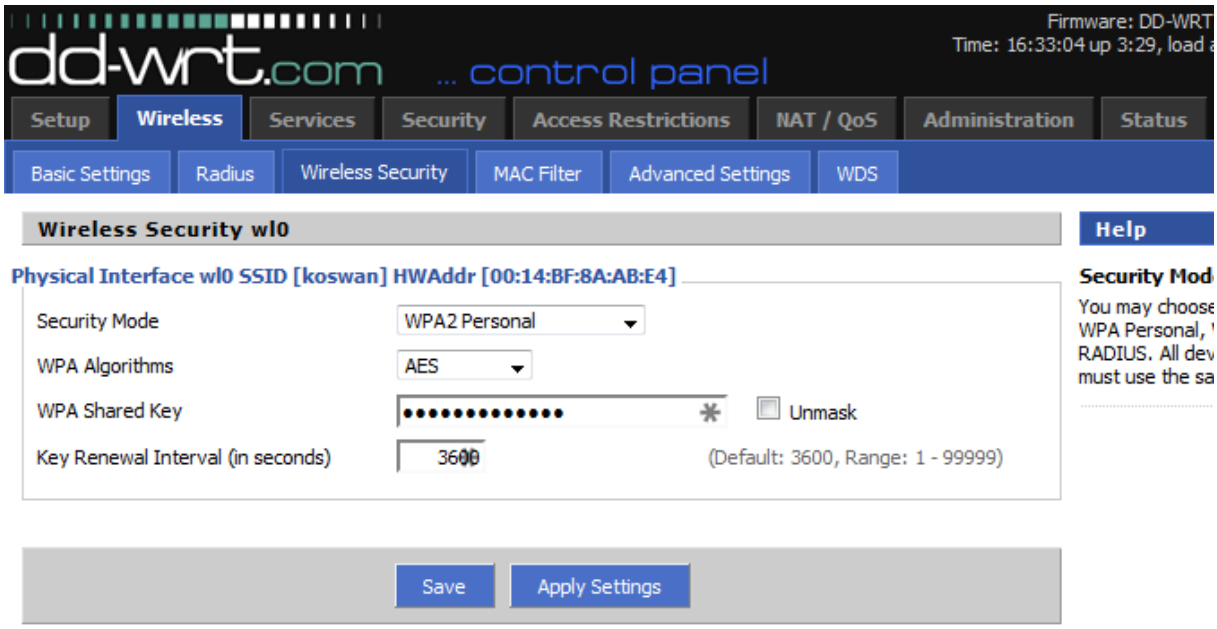
Στην συνέχεια εισάγουμε την **192.167.1.1,** που είναι και η IP της συσκευής μας στο τοπικό ασύρματο δίκτυο, στην URL bar του Firefox, εισάγουμε τα απαραίτητα **Username** και **Password** και εισερχόμαστε στο μενού της συσκευής.

Εισερχόμενοι στο μενού του **WRT54G**, επιλέγουμε **Wireless → Basic Settings** και βλέπουμε ότι η λειτουργία "**Wireless Mode**" είναι ορισμένη ως **AP** (**Access Point**). Αυτό σημαίνει ότι το WRT54G θα δρα ως ένα κεντρικό σημείο συνδέσεων, στο οποίο μπορούν να συνδεθούν Clients μέσω Wi-Fi ενώ δρα και ως γέφυρα, συνδέοντας το ενσύρματο με το ασύρματο κομμάτι ενός δικτύου. Στη συνέχεια επιλέγουμε στο "**Wireless Network Mode**" την επιλογή "**G-Only**". Αυτό σημαίνει ότι το AP θα έχει μέγιστο θεωρητικό **Throughput** στο ασύρματο **Interface** του, ίσο με 54 Mbps. Στο πεδίο "**Wireless Network Name (SSID)**", βάζουμε το όνομα του δικτύου που θέλουμε να εκπέμπει το AP. Στην επιλογή "**Wireless Channel**" διαλέγουμε το κανάλι στο οποίο θέλουμε να εκπέμπει το AP. Αν βρισκόμασταν στην Βόρεια Αμερική θα μπορούσαμε να επιλέξουμε μεταξύ των καναλιών 1 έως και 11, στην Ευρώπη από το 1 έως και το 13, ενώ στην Ιαπωνία οποιοδήποτε από τα 14 κανάλια. Επιλέχτηκε το κανάλι **10** που αντιστοιχεί στη συχνότητα **2.457GHz**. Στην επιλογή "**Wireless SSID Broadcast**" επιλέγουμε το "**Enable**", ώστε το AP να εκπέμπει το αναγνωριστικό του δικτύου και να μπορούν οι άλλες συσκευές που θέλουν να συνδεθούν να δουν το όνομά του. Τέλος, στο "**Network Configuration**" επιλέγουμε την επιλογή "**Bridged**" ώστε το AP να δρα ως γέφυρα μεταξύ των συσκευών του ασύρματου και του ενσύρματου δικτύου.



Εικόνα 48: WRT54G Wireless Basic Settings

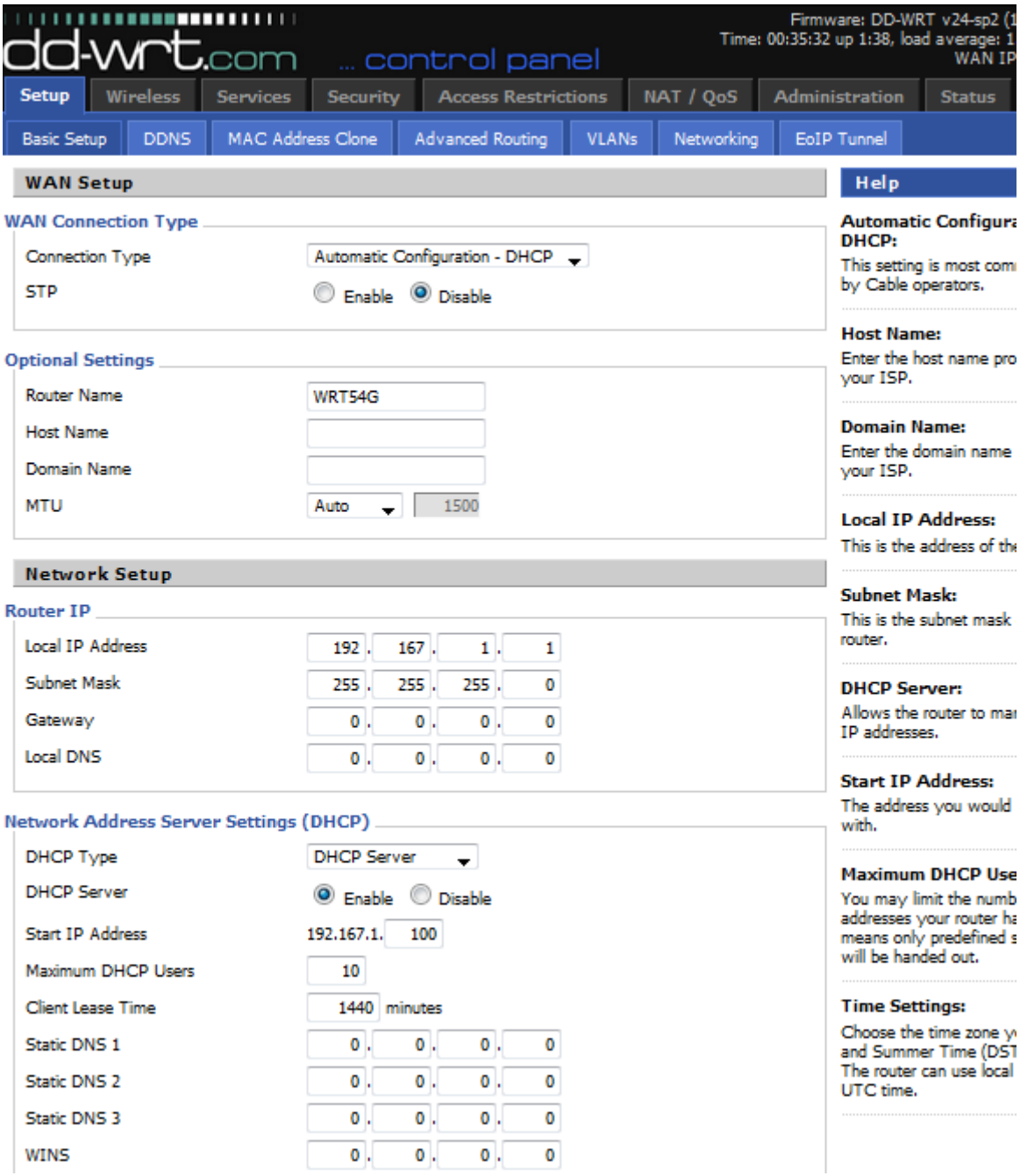
Στην συνέχεια πάμε στην καρτέλα **Wireless → Wireless Security**. Εδώ βλέπουμε ότι γίνεται χρήση του **WPA2** ως πρωτόκολλου ασφάλειας, με χρήση του αλγόριθμου κρυπτογράφησης **AES**. Στο πεδίο "**WPA Shared Key**" είναι ο κωδικός που χρησιμοποιούμε για να συνδεθούμε στο δίκτυο και φυσικά όλα τα παραπάνω είναι τα ίδια στοιχεία που χρησιμοποιήθηκαν για να συνδεθεί ασύρματα το **TL-WR703N** στο **WRT54G**, όπως περιγράφτηκε στην προηγούμενη ενότητα.



Εικόνα 49: WRT54G Wireless Security Settings

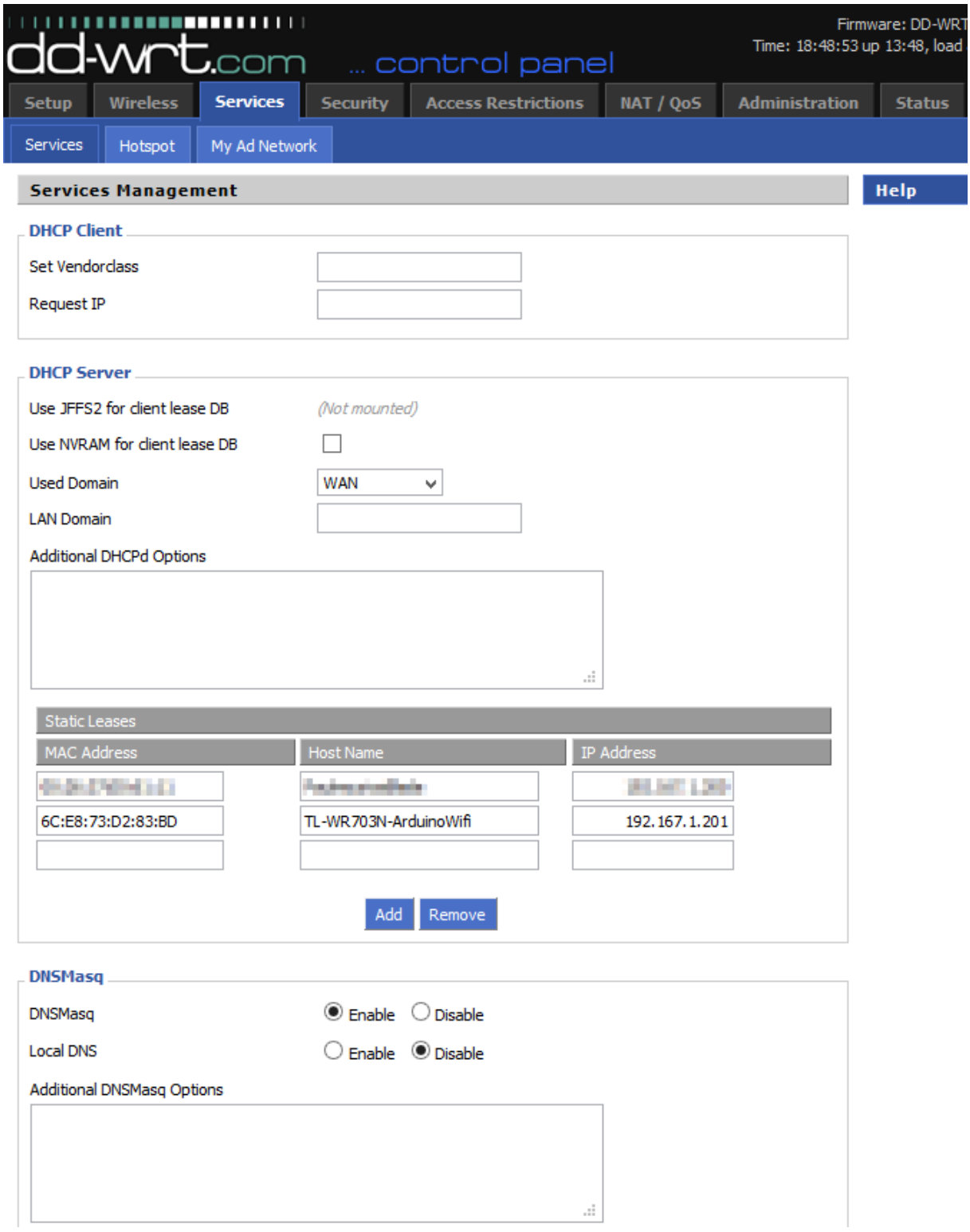
Στην συνέχεια επιστρέφουμε στην καρτέλα **Setup → Basic Setup**. Στο πεδίο "**Connection Type**" επιλέγουμε το "**Automatic Configuration – DHCP**" ώστε να αφήσουμε το WAN Interface της συσκευής να πάρει αυτόματα IP από το **Thomson TG585**. Στο πεδίο "**Router Name**", συμπληρώνουμε το όνομα του AP μας. Στη συνέχεια περνάμε στις ρυθμίσεις που αφορούν το LAN (ασύρματο και ενσύρματο) κομμάτι της συσκευής. Στο πεδίο "**Local IP Address**" δίνουμε την IP που θέλουμε να έχει το WRT54G στο τοπικό δίκτυο και συγκεκριμένα την **192.167.1.1**. Βλέπουμε λοιπόν ότι είναι η ίδια IP με αυτήν που είχε ως **Gateway** το TL-WR703N. Στο πεδίο "**Subnet Mask**" συμπληρώνουμε το **255.255.255.0** για να δηλώσουμε το υποδίκτυο, κάτι το οποίο δείχνει ότι το TL-WR703N πήρε την **192.167.1.201** δικαιολογημένα, όπως δείξαμε παραπάνω.

Κατόπιν, επιλέγουμε ως "**DHCP Type**" τον "**DHCP Server**" και τον κάνουμε "**Enable**" ώστε να υπάρχει αυτόματη απόδοση IP στις συσκευές που συνδέονται στο **LAN Interface** της συσκευής. Αυτό γίνεται για δική μας ευκολία στο δίκτυο, αλλά θα δείξουμε παρακάτω πως διαφοροποιείται όσον αφορά το δικό μας Use Case. Στο πεδίο "**Start IP Address**" βάζουμε την **192.167.1.100** ενώ στο πεδίο "**Maximum DHCP Users**" βάζουμε τον αριθμό **10**. Αυτό σημαίνει ότι οι Clients θα μπορούν να πάρουν IPs από την **192.167.1.100** έως και την **192.167.1.254** ενώ ο μέγιστος αριθμός τους δεν πρέπει να ξεπερνάει τους **10**.



Εικόνα 50: Βασικές Ρυθμίσεις για το Τοπικό Δίκτυο στο WRT54G

Στο επόμενο βήμα, θα δείξουμε πως έγινε η ρύθμιση στο **WRT54G** για να αναθέτει την ίδια IP στο **TL-WR703N** κάθε φορά που αυτό συνδέεται, ώστε να μπορεί να γίνει η δρομολόγηση των αιτημάτων προς αυτό, και κατά συνέπεια στον **Server**. Πάμε λοιπόν στην καρτέλα **Services → Services**. Στον πίνακα με τα "**Static Leases**" στο πεδίο της **MAC Address** θα βάλουμε την **φυσική διεύθυνση** που αντιστοιχεί στο **Wireless Interface** του TL-WR703N, η οποία είναι η "**6C:E8:73:D2:83:BD**" όπως είδαμε και στην προηγούμενη ενότητα. Στο πεδίο "**Host Name**" δίνουμε ένα όνομα στη συσκευή που θέλουμε αντιστοιχεί σε αυτήν την MAC Address, για δική μας διευκόλυνση κατά τη διαχείριση του δικτύου. Δόθηκε λοιπόν το όνομα "**TL-WR703N-ArduinoWifi**". Στην ίδια σειρά, στο πεδίο "**IP Address**" συμπληρώθηκε η Static IP που θέλουμε να παίρνει η συσκευή (δηλαδή το TL-WR703N) στο τοπικό ασύρματο δίκτυο, η οποία είναι η **192.167.1.201**. Άρα, ασχέτως του ότι είχαμε δηλώσει ότι η διευθυνσιοδότηση στο τοπικό δίκτυο θα γίνεται αυτόματα μέσω του DHCP, εμείς δηλώσαμε μια Static IP για το TL-WR703N.

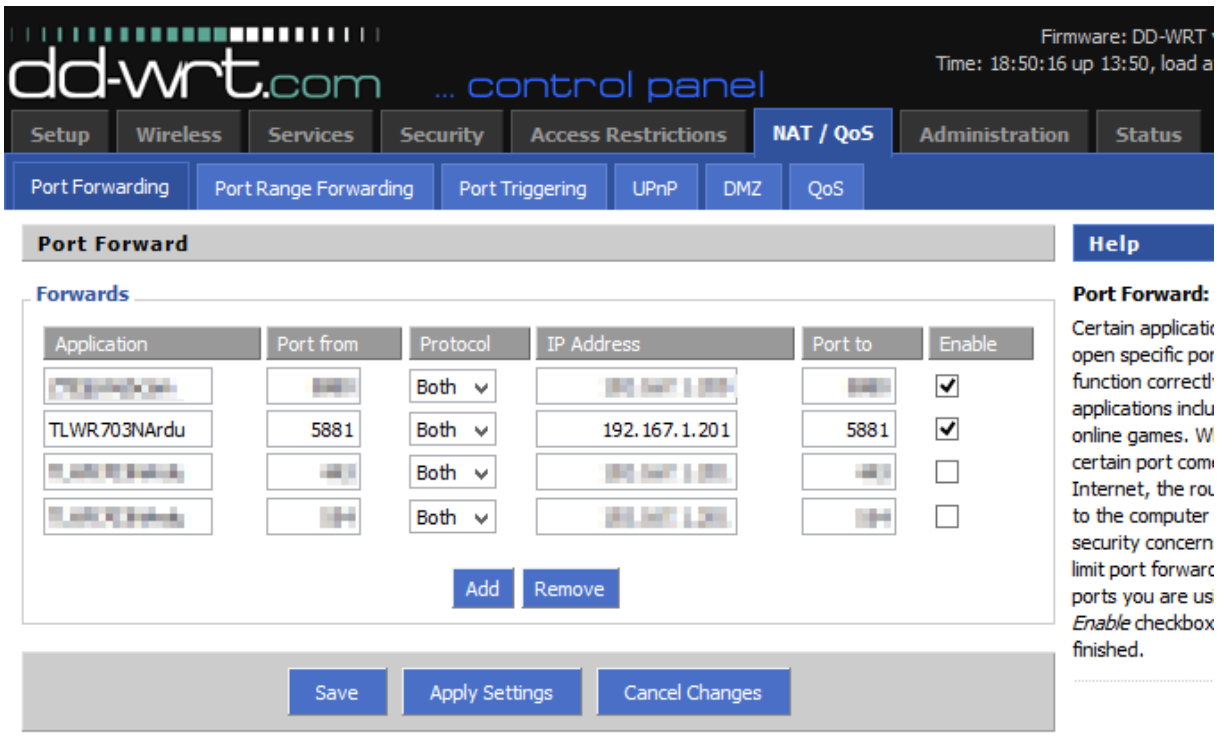


Εικόνα 51: Ανάθεση Static IP στο TL-WR703N από το WRT54G

Στο επόμενο βήμα των απαραίτητων ρυθμίσεων, θα εισάγουμε τα κατάλληλα στοιχεία, ώστε οι αιτήσεις από Clients εντός και εκτός του δικτύου μας προς μια συγκεκριμένη θύρα στο δίκτυό μας, να ανακατευθύνονται προς τον υπολογιστή που θα του υποδείξουμε. Εμείς θέλουμε όλες οι αιτήσεις προς την θύρα 5881, να ανακατευθύνονται προς την IP του TL-WR703N, το οποίο με τη σειρά του τις ανακατευθύνει στον Arduino Server. Άρα θα χρειαστεί πάλι να εφαρμόσουμε την τεχνική του Port

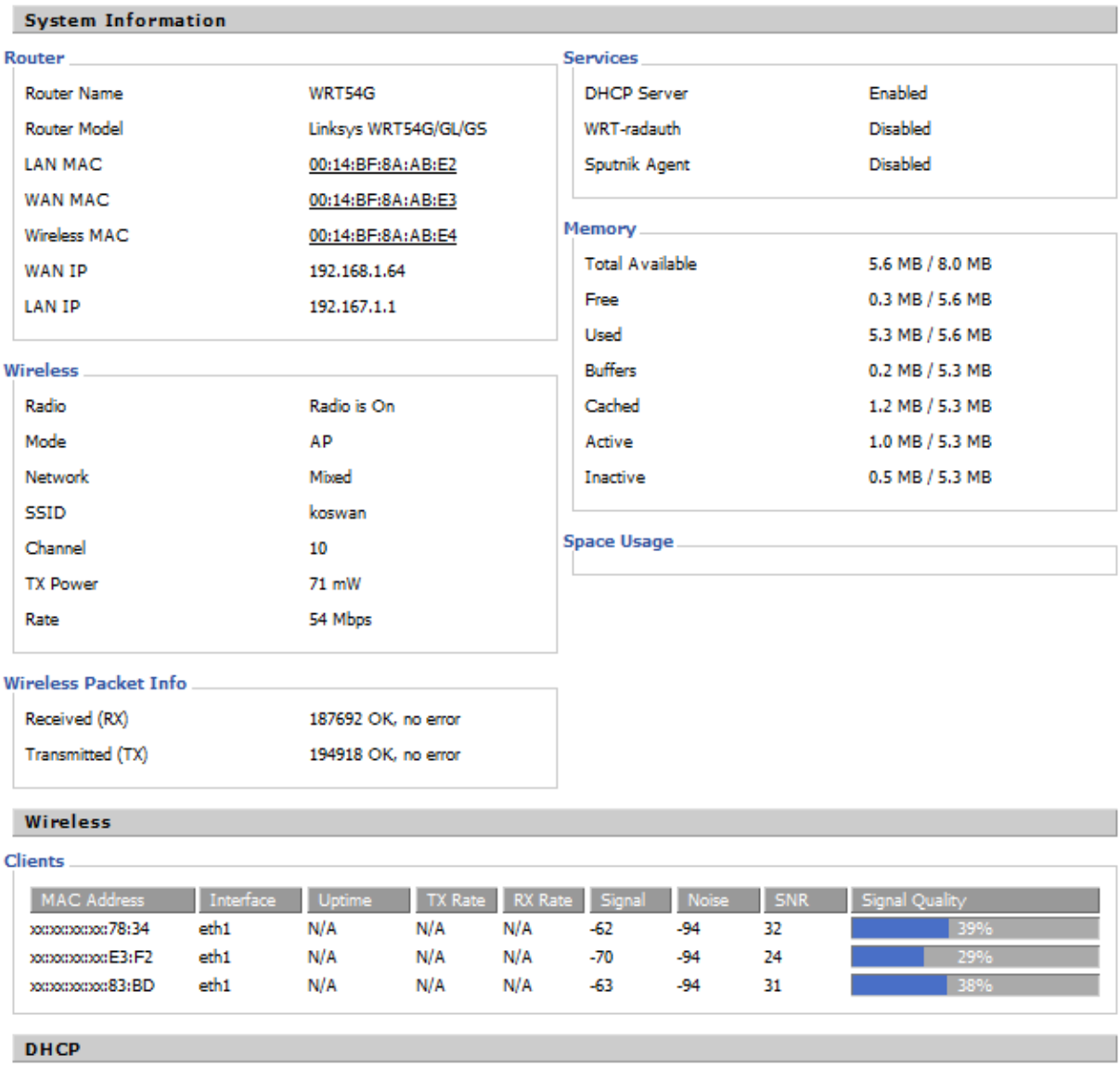
Forwarding, όπως κάναμε και στην προηγούμενη ενότητα. Φυσικά για να γίνει αυτό χρειάζεται να έχουμε αναθέσει μια στατική IP στο TL-WR703N, κάτι που δείξαμε πως έγινε στο προηγούμενο βήμα.

Επιλέγουμε λοιπόν την καρτέλα **NAT/QoS →Port Forwarding**. Στον πίνακα "**Port Forward**", στο πεδίο "**Application**" θα δώσουμε ένα όνομα στην Συσκευή/Υπολογιστή προς την οποία θέλουμε να ανακατευθύνουμε τα Requests. Όπως πριν, έτσι και εδώ, αυτό γίνεται για δική μας διευκόλυνση στη διαχείριση του δικτύου. Βάζουμε λοιπόν το όνομα "**TLWR703NArdu**". Στο πεδίο "**Port from**" θα δώσουμε τον αριθμό της θύρας στην οποία θα κάνουν τα Requests οι Clients. Στην προκειμένη περίπτωση συμπληρώνουμε την θύρα **5881**. Στο πεδίο "**Protocol**" επιλέγουμε την επιλογή "**Both**". Αυτό σημαίνει ότι θα γίνεται η δρομολόγηση είτε γίνεται χρήση του **TCP**, είτε του **UDP** πρωτοκόλλου. Στο πεδίο "**IP Address**" θα συμπληρώσουμε την IP διεύθυνση της συσκευής προς την οποία θέλουμε να προωθούνται τα Requests, δηλαδή αυτήν του TL-WR703N στο τοπικό δίκτυο, η οποία είναι η "**192.167.1.201**". Στο πεδίο "**Port to**" συμπληρώνουμε την θύρα στην οποία ακούει η συσκευή μας (Arduino Server + TL-WR703N) και τρέχει η υπηρεσία μας, δηλαδή την θύρα **5881**. Τέλος, τσεκάρουμε την επιλογή "**Enable**" στην ίδια σειρά ώστε να θέσουμε ενεργή την ρύθμισή μας και μετά κάνουμε **Save** για να την αποθηκεύσουμε.



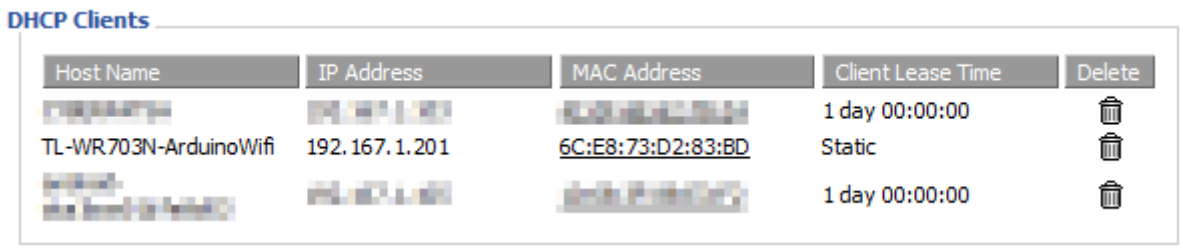
Εικόνα 52: Port Forwarding από το WRT54G προς το TL-WR703N στην Port 5881

Στην συνέχεια, πάμε στην καρτέλα **Status → Sys-Info**. Από εδώ μπορούμε να έχουμε μια καλή εποπτεία του τι γίνεται στο δίκτυό μας. Στην περιοχή Router, αξίζει να επισημανθεί ότι ως WAN IP, έχει οριστεί η **192.168.1.64**. Αυτή την IP την πήρε αυτόματα το **WRT54G** από το **TG585**, τα οποία είναι συνδεμένα ενσύρματα μεταξύ τους, κάνοντας χρήση του **DHCP**. Λίγο πιο κάτω, στην περιοχή "**Wireless – Clients**", βλέπουμε ότι το **TL-WR703N** που του αντιστοιχεί η **MAC Address** που τελειώνει σε "**83:BD**" είναι συνδεμένο με ποιότητα σήματος 38% στα -63dB.



Εικόνα 53: Εποπτεία Συστήματος στο WRT54G

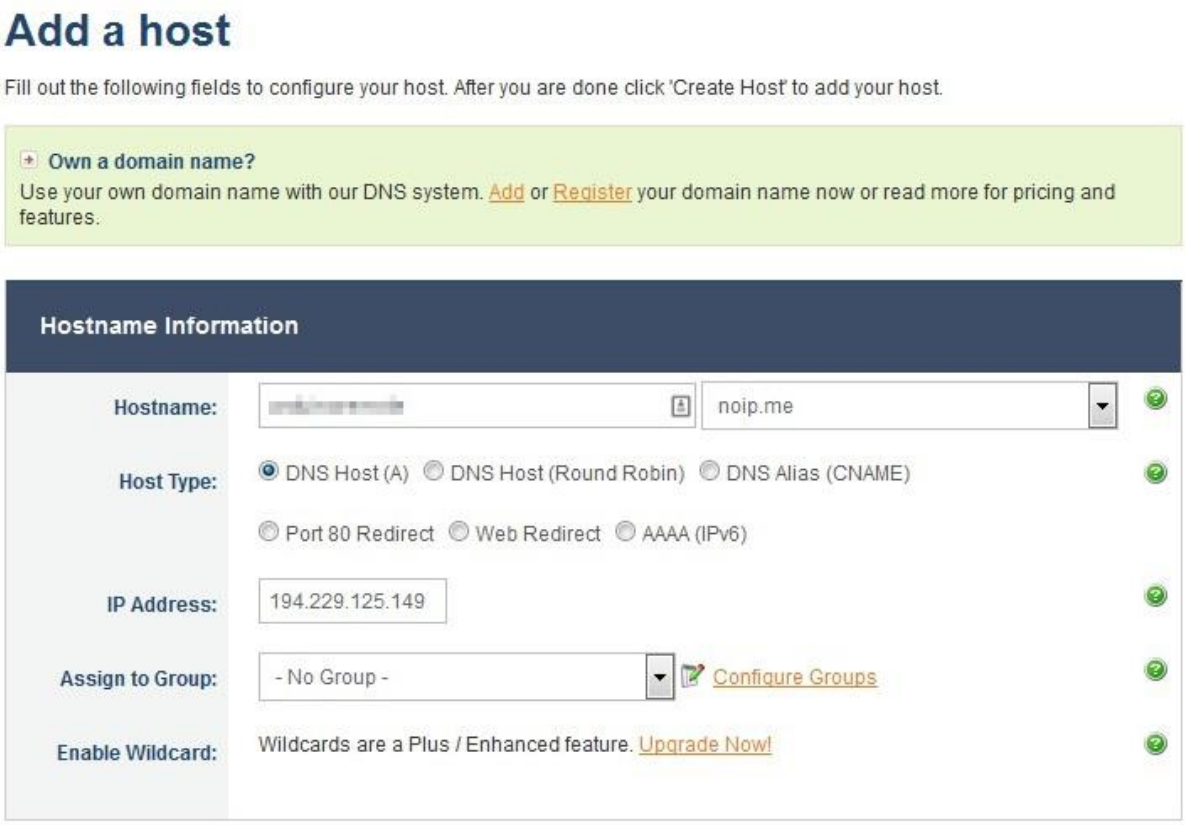
Αμέσως πιο κάτω στην περιοχή "**DHCP Clients**", επιβεβαιώνουμε ότι το TL-WR703N έχει πάρει Static IP και συγκεκριμένα την **192.167.1.201**.



Εικόνα 54: Η Static IP του TL-WR703N στους Clients του WRT54G

Η τελευταία ρύθμιση που μένει να κάνουμε στο WRT54G είναι αυτή που σχετίζεται με την DDNS υπηρεσία. Έχοντας ήδη δημιουργήσει έναν λογαριασμό στην υπηρεσία **No-IP** που παρέχει

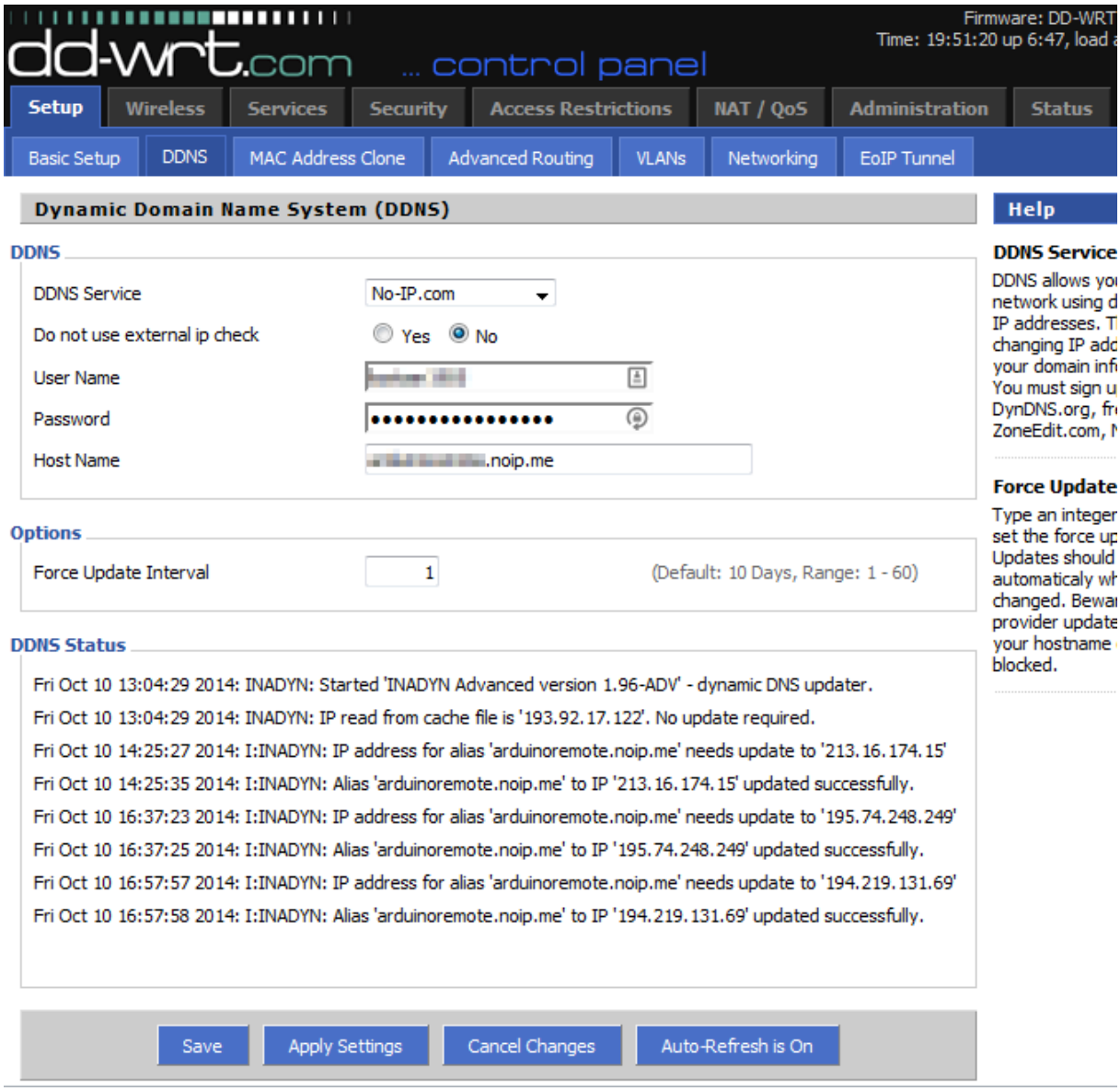
Dynamic DNS, κάνουμε **login** με τα στοιχεία μας δίνοντας το απαραίτητο **Username** και **Password**. Στη συνέχεια επιλέγουμε **Hosts / Redirects → Add Host**. Στη φόρμα που ανοίγει στα δεξιά, συμπληρώνουμε στο πεδίο "**Hostname**" το όνομα του **subdomain** στο οποίο θα ακούει η IP στην οποία αντιστοιχεί το δίκτυο μας. Ας το ονομάσουμε "**ArduinoRemoteServer**". Στο dropdown menu στην ίδια σειρά, επιλέγουμε από την λίστα το όνομα του domain που θέλουμε. Επιλέχθηκε το "**noip.me**" το οποίο είναι δωρεάν. Στο πεδίο "**Host Type**", επιλέξαμε την επιλογή "**DNS Host (A)**". Αυτή είναι η πιο συνήθης επιλογή, και αυτό που κάνει είναι να αντιστοιχεί ένα **hostname** σε μια IP διεύθυνση. Στο επόμενο πεδίο "**IP Address**" συμπληρώνεται αυτόματα η τρέχουσα IP του υπολογιστή μας. Μόλις τελειώσουμε, πατάμε το κουμπί "**Add Host**", για να δεσμεύσουμε το hostname "**ArduinoRemoteServer.noip.me**", στον λογαριασμό μας.



Εικόνα 55: Προσθήκη Hostname σε έναν Λογαριασμό στην Υπηρεσία DDNS No-IP

Στην συνέχεια, επιστρέφουμε στο μενού του WRT54G και επιλέγουμε **Setup → DDNS**. Στο πεδίο "**DDNS Service**", επιλέγουμε από το dropdown menu την επιλογή "**No-IP.com**", δηλαδή τον πάροχο της DDNS υπηρεσίας που θα χρησιμοποιήσουμε. Στην επιλογή "**Do not use external IP check**", επιλέγουμε την επιλογή "**No**". Η **WAN IP** (η οποία είναι η **192.168.1.64** όπως είδαμε λίγο πριν) του Access Point μας **δεν είναι η πραγματική εξωτερική του IP**, καθώς το AP μας βρίσκεται πίσω από ακόμα μια συσκευή που δρα ως **Gateway** και μας δίνει πρόσβαση στο Internet (δηλαδή το TG585). Άρα κρίνεται απαραίτητο να υπάρχει μια **εξωτερική δευτερεύουσα υπηρεσία**, η οποία θα **ελέγχει** την πραγματική εξωτερική IP μας και θα την **αναφέρει** πίσω στο **WRT54G**, ώστε να μπορέσει αυτό με τη σειρά του να **ενημερώσει** τον **Dynamic DNS host**.

Συνεχίζοντας, στα πεδία "**User Name**" και "**Password**" συμπληρώνουμε τα ίδια με αυτά που χρησιμοποιήσαμε για να κάνουμε **login** στο λογαριασμό μας στην στον ιστότοπο www.noip.com. Τέλος στο πεδίο "**Hostname**" συμπληρώνουμε το hostname που δεσμεύσαμε πριν στο λογαριασμό μας, δηλαδή το "**ArduinoRemoteServer.noip.me**". Στο πεδίο "**Force Update Interval**" συμπληρώνουμε "**1**". Αυτό σημαίνει ότι ακόμα και αν δεν έχει αλλάξει η IP μας ώστε να ενημερωθεί αυτόματα η υπηρεσία DDNS, κάθε μία ημέρα θα ενημερώνεται υποχρεωτικά.



Εικόνα 56: Ρύθμιση DDNS Υπηρεσίας στο WRT54G

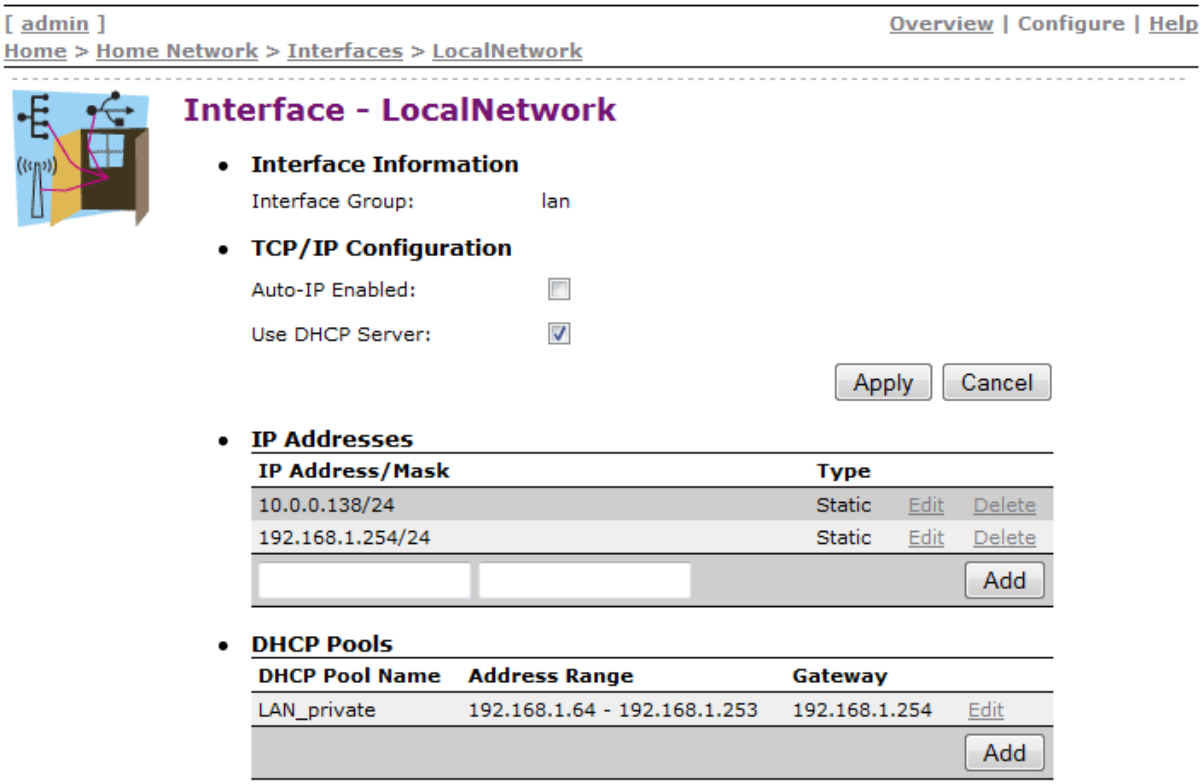
Τέλος, βλέπουμε από το "**DDNS Status**" ότι η υπηρεσία λειτουργεί κανονικά. Πλέον έχει ρυθμιστεί πλήρως το WRT54G και είμαστε έτοιμοι να δείξουμε τι ρυθμίσεις έχουν γίνει και στο TG585, ώστε το Arduino να μπορεί να είναι προσπελάσιμο από τον έξω κόσμο.

7.3 Ρυθμίσεις του THOMSON TG585 v7

Παραμένουμε συνδεμένοι στο WRT54G μέσω wireless και για να συνδεθούμε εισάγουμε την IP "**192.168.1.254**" που ανήκει στο TG585, στην URL bar του Firefox. Βάζοντας τα απαραίτητα Username και Password, εισερχόμαστε στο μενού των ρυθμίσεων του TG585 v7. Αξίζει να υπενθυμίσουμε βέβαια ότι το WRT54G συνδέεται μέσω Ethernet καλωδίου με το TG585, το οποίο έχει ρόλο Gateway.

Επιλέγουμε λοιπόν από το μενού **Home Network → Interfaces → Local Network** . Εδώ βλέπουμε ότι για την απόδοση IP στο LAN δίκτυό του γίνεται χρήση του DHCP Server. Αυτό σημαίνει ότι το WRT54G θα παίρνει αυτόματα IP στο WAN Interface του, έχοντας ως Gateway το

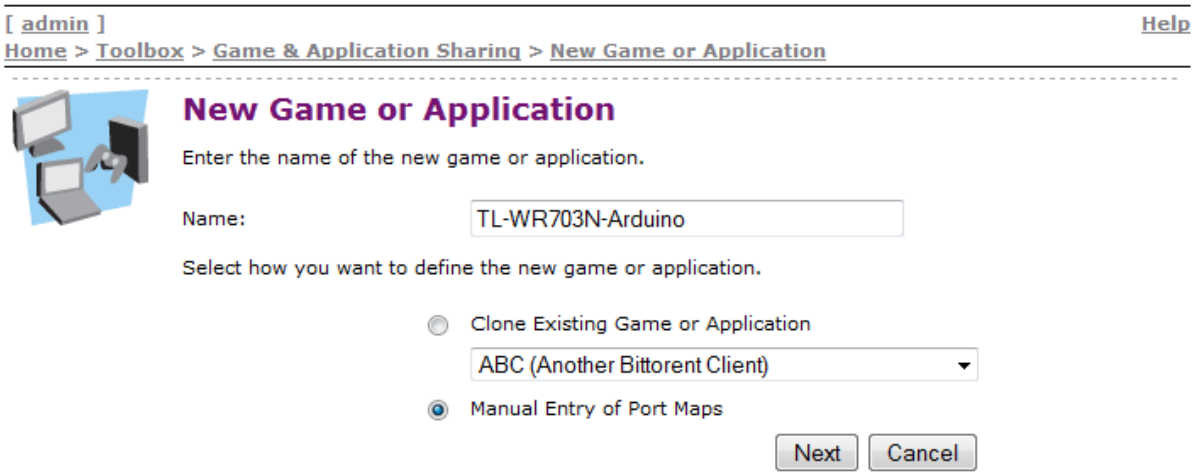
TG585. Στον πίνακα "**IP Addresses**" βλέπουμε ότι TG585 αντιστοιχεί στην Static IP **192.168.1.254** (αυτή που χρησιμοποιήθηκε για να εισέλθουμε στο μενού του) και έχει ως Subnet Mask το "**/24**", δηλαδή το "**255.255.255.0**". Στον πίνακα "**DHCP Pools**" έχουμε ρυθμίσει οι IP που θα αποδίδονται να είναι μεταξύ των **192.168.1.64** και **192.168.1.253** με **Gateway** την **192.168.1.254**. Επομένως στο **WAN Interface** του WRT54G, θα αντιστοιχεί πάντα μια IP από αυτές ενώ όπως είδαμε στην προηγούμενη ενότητα είχε την **192.168.1.64**.



Εικόνα 57: Ρύθμιση Απόδοσης IP στο Τοπικό Δίκτυο με Χρήση του DHCP Server

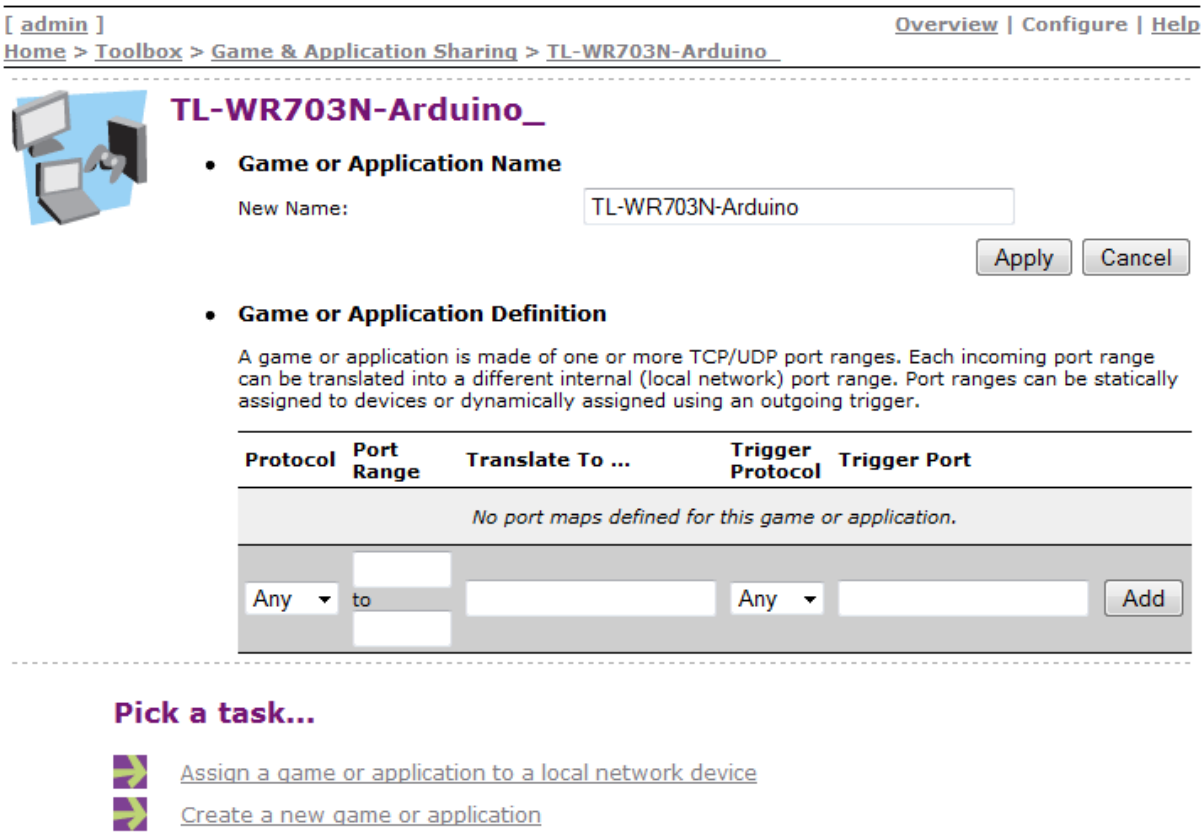
Στην συνέχεια πάμε στο μενού **Toolbox → Game & Application Sharing**. Από εκεί επιλέγουμε το "**Create a new game or application**". Από εδώ θα δημιουργήσουμε ένα Profile στο οποίο θα δηλώσουμε σε ποια εσωτερική θύρα θα προωθούνται οι αιτήσεις από το Internet προς κάποια εξωτερική θύρα.

Στο πεδίο "**Name**" δίνουμε το όνομα "**TL-WR703N-Arduino**". Ακριβώς από κάτω επιλέγουμε το "**Manual Entry of Port Maps**", κάτι που σημαίνει ότι εμείς θα διαλέξουμε πως θα γίνει η αντιστοιχία των θυρών και ότι δεν θα επιλέξουμε από κάποιο έτοιμο προφίλ από την υπάρχουσα λίστα.

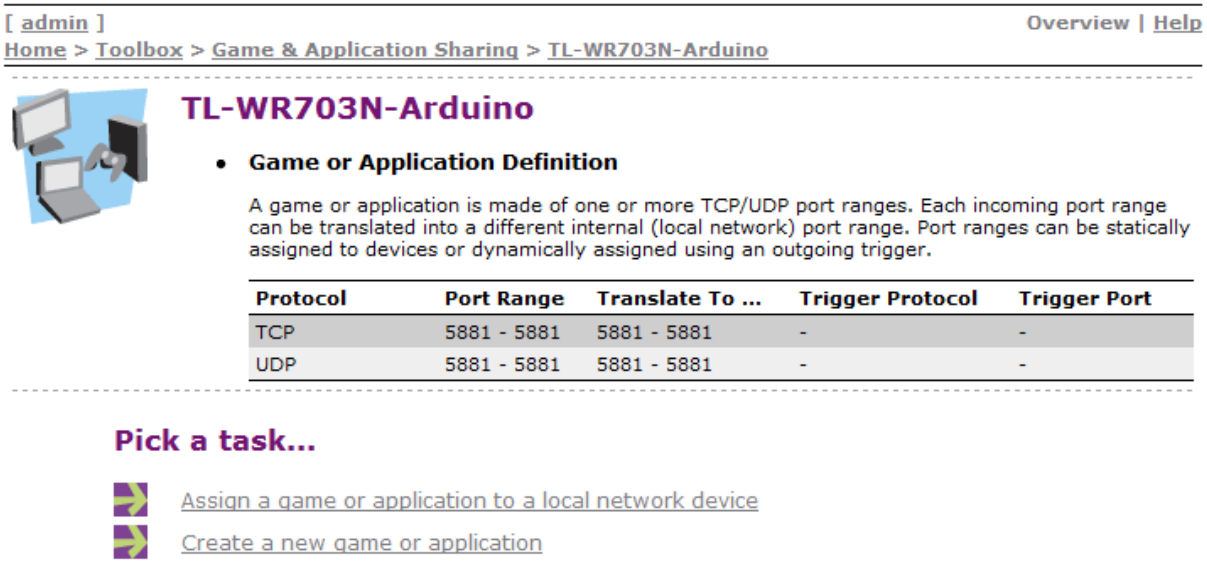


Εικόνα 58: Ονομασία Προφίλ Αντιστοιχίας Θυρών

Πατώντας "**Next**" και περνώντας στην επόμενη καρτέλα, στον Πίνακα "**Game or Application Definition**" βλέπουμε τα πεδία που πρέπει να συμπληρωθούν. Στο πεδίο "**Protocol**" επιλέγουμε το "**Any**". Αυτό σημαίνει ότι είτε γίνεται χρήση του TCP είτε του UDP πρωτοκόλλου, θα γίνεται mapping των θυρών που θα δηλώσουμε. Στα πεδία "**Port Range**" συμπληρώνουμε την θύρα 5881 και στα δύο. Αυτό σημαίνει ότι τα Requests από το Internet πρέπει να γίνονται σε αυτή την θύρα, για να προωθηθούν στη θύρα που θα συμπληρώσουμε στο πεδίο "**Translate To**". Εμείς θέλουμε τα αιτήματα στην θύρα **5881** να προωθούνται στην εσωτερική θύρα **5881**, οπότε αυτή είναι που θα συμπληρωθεί στο πεδίο. Στην συνέχεια πατάμε "**Apply"** για να ολοκληρωθούν οι ρυθμίσεις.

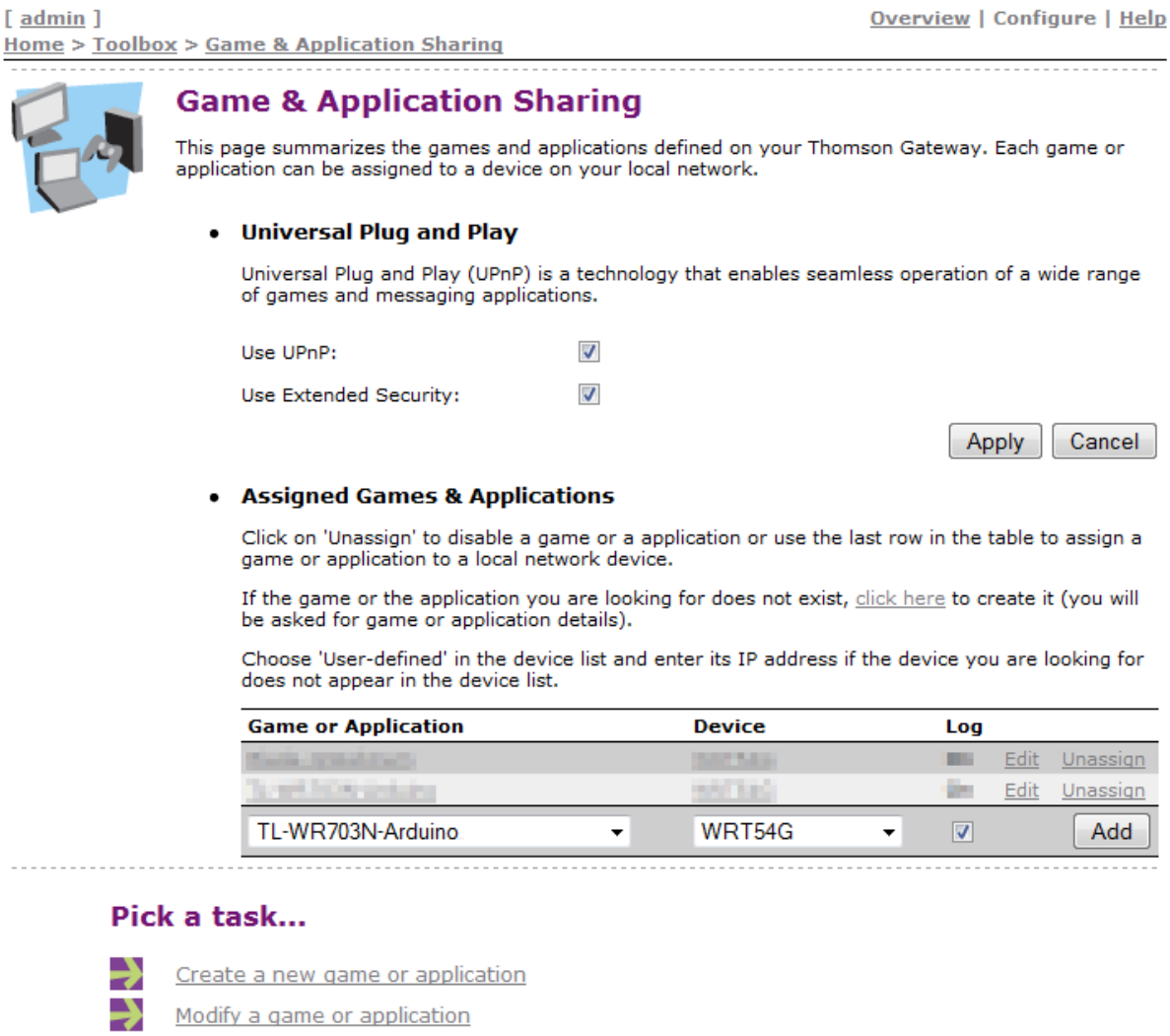


Εικόνα 59: Ορίζοντας τις Θύρες στο TG585

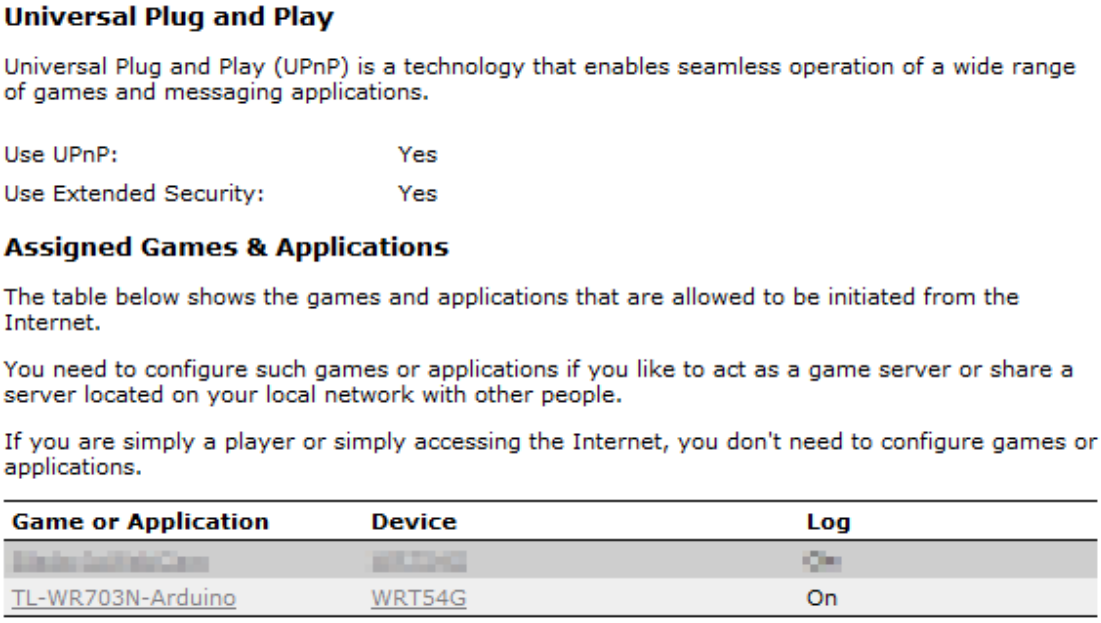


Εικόνα 60: Αντιστοιχία Θυρών στο TG585

Έπειτα, στην ίδια καρτέλα πατάμε το "**Assign a game or application to a local network device**". Με αυτόν τον τρόπο θα αντιστοιχήσουμε τον προφίλ αντιστοιχίας θυρών που θέσαμε προηγουμένως, με την συσκευή WRT54G. Ουσιαστικά με αυτόν τον τρόπο, όλες οι αιτήσεις προς την εξωτερική θύρα 5881 θα κατευθύνονται προς την εξωτερική θύρα 5881 που ακούει το WRT54G. Επιλέγουμε λοιπόν από τον πίνακα "**Assigned Games & Applications**", στην στήλη "**Game or Application**", το "**TL-WR703N-Arduino**". Στην στήλη "**Device**" επιλέγουμε το "**WRT54G**" από την λίστα. Επιλέγουμε την επιλογή "**Log**" για να ενεργοποιήσουμε το αρχείο καταγραφής για την ρύθμισή μας και τέλος πατάμε το κουμπί "**Add**" για να ολοκληρωθεί η αντιστοιχία.



Εικόνα 61: Αντιστοιχία του WRT54G με το Profile Αντιστοιχίας Θυρών



Εικόνα 62: Τελικές Ρυθμίσεις στο TG585

Με αυτήν την ρύθμιση τελειώνουν οι ρυθμίσεις όσον αφορά το κομμάτι των συσκευών που χρησιμοποιήθηκαν. Στην επόμενη ενότητα θα γίνει μια εποπτεία και έλεγχος καλής λειτουργίας του δικτύου και του Server.

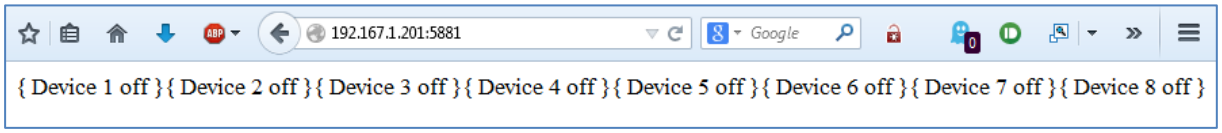
7.4 Εποπτεία και Έλεγχος Δικτύου και Arduino Server

Έχοντας πλέον ολοκληρώσει τις ρυθμίσεις του δικτύου μας, ας δούμε συνοπτικά τα στοιχεία κάθε συσκευής στο δίκτυο.

	Arduino Server	TL-WR703N	WRT54G	TG585
LAN IP	192.168.0.101	192.168.0.254	192.167.1.1	192.168.1.254
Gateway	192.168.0.254	192.167.1.1	192.168.1.254	ISP
WAN IP	Null	192.167.1.201	192.168.1.64	Public External IP
LAN Subnet Mask	255.255.255.0	255.255.255.0	255.255.255.0	255.255.255.0

Εικόνα 63: Πίνακας Στοιχείων Δικτύου

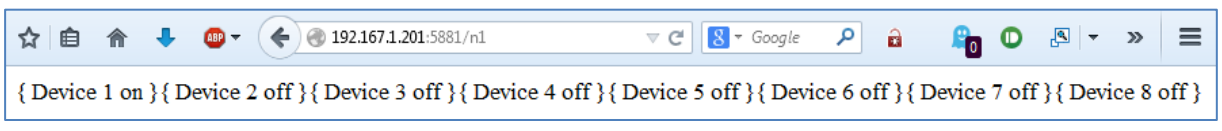
Παραμένουμε συνδεμένοι στο WRT54G μέσω wireless. Για να δούμε αν δουλεύει στο τοπικό δίκτυο ο server μας και ανταποκρίνεται, θα κάνουμε ένα **GET Request** στην **εσωτερική IP** που αντιστοιχεί στον Arduino Server και στην κατάλληλη **Port**. Πρακτικά αυτό σημαίνει, στην IP που έχει στο τοπικό δίκτυο το **TL-WR703N**. Δίνουμε λοιπόν στην URL bar του Firefox την εντολή "192.167.1.201:5881". Η απόκριση του Server αποδίδεται στην παρακάτω εικόνα.



Εικόνα 64: Response του Arduino Server στο Get Request από τον Firefox

Βλέπουμε λοιπόν ότι ο Server ανταποκρίνεται, και μας **τυπώνει ως απάντηση τα States όλων των Pins**, όπως τον είχαμε προγραμματίσει να κάνει στο Κεφάλαιο 6. Όσο είμαστε με το Laptop συνδεδεμένοι στο WRT54G, σε οποιαδήποτε WAN IP από αυτές του παραπάνω πίνακα κάνουμε GET στην Port 5881, θα είχαμε απόκριση από τον Server. Σε αυτές συμπεριλαμβάνεται και η Public IP.

Δείξαμε στο κεφάλαιο 6 από τον προγραμματισμό στο Arduino, με ποιο τρόπο θα μπορούμε να ενεργοποιήσουμε τα Pins της συσκευής. Ας ελέγξουμε λοιπόν τι απάντηση θα έχουμε από τον Server αν θέλουμε να αλλάξουμε το State της συσκευής 1, από Off σε On. Γράφουμε λοιπόν στην **URL bar** του Firefox: "**192.167.1.201:5881/n1**".



Εικόνα 65: Response του Arduino Server στο Get Request Ενεργοποίησης του Pin 2

Όπως βλέπουμε, αυτή την φορά υπάρχει πάλι απάντηση από τον Server, αλλά το **State** της συσκευής 1 που αντιστοιχεί στο **Pin 2** του Arduino Server, έχει αλλάξει σε **On**. Αυτό σημαίνει ότι το Pin 2 στο Arduino έχει οριστεί ως ενεργό. Μπορούμε να το επιβεβαιώσουμε απλά κοιτώντας το

κύκλωμα που έχει δημιουργηθεί για τις ανάγκες της παρούσας διπλωματικής, στο οποίο όταν γίνεται ενεργό το Pin 2, ανάβει ένα LED. Η διαδικασία αυτή ακολουθήθηκε όσες φορές χρειάστηκε για να επιβεβαιωθεί ότι όλα τα Pins που χρησιμοποιούνται στο Arduino ενεργοποιούνται και απενεργοποιούνται, με τον Server να τυπώνει κάθε φορά τα σωστά States.

Πλέον ήρθε η ώρα να τεστάρουμε αν οι ρυθμίσεις που έχουμε κάνει μας επιτρέπουν να δούμε τον Arduino Server εκτός του τοπικού δικτύου. Για να μπορέσουμε να το ελέγξουμε αυτό, αυτή τη φορά θα πρέπει να κάνουμε το Request χρησιμοποιώντας την Public IP που παίρνει το TG585 από τον Internet Service Provider. Όπως αναφέραμε προηγουμένως, αν το κάνουμε αυτό όντας συνδεμένοι στο τοπικό δίκτυο, η σύνδεση είναι σίγουρη. Για να το ελέγξουμε λοιπόν, θα κάνουμε χρήση του **Mobile Data** της Android συσκευής μας (LG Optimus G), το οποίο συνδέεται στο Internet κάνοντας χρήση του δικτύου κινητής τηλεφωνίας. Επειδή θέλουμε να βεβαιωθούμε ότι λειτουργεί και η DDNS υπηρεσία που έχουμε, θα κάνουμε το Request στην θύρα "5881" αλλά αυτή τη φορά κάνοντας χρήση του hostname "**ArduinoRemoteServer.noip.me**" που δεσμεύσαμε στον λογαριασμό μας και έχει ρυθμιστεί να αντιστοιχεί στην Public IP του δικτύου μας.

Ανοίγουμε τον περιηγητή **Chrome Mobile** στην Android Συσκευή και στην **Search bar** γράφουμε την εντολή "**ArduinoRemoteServer.noip.me:5881**".

Εικόνα 66: Response Arduino Server στην Port 5881 Κάνοντας Χρήση της DDNS Υπηρεσίας και του Δικτύου Κινητής Τηλεφωνίας

Όπως βλέπουμε λοιπόν στην παραπάνω εικόνα, η απάντηση που τύπωσε ο Arduino Server στον Chrome Mobile, δείχνει ότι απάντησε κανονικά και είναι πλέον **προσπελάσιμος μέσω Internet**, δίνοντάς μας μάλιστα και το State της συσκευής ένα ως On, εφόσον το Pin 2 είχε παραμείνει ενεργό από την εντολή που είχαμε δώσει από την URL bar του Firefox στο προηγούμενο βήμα.

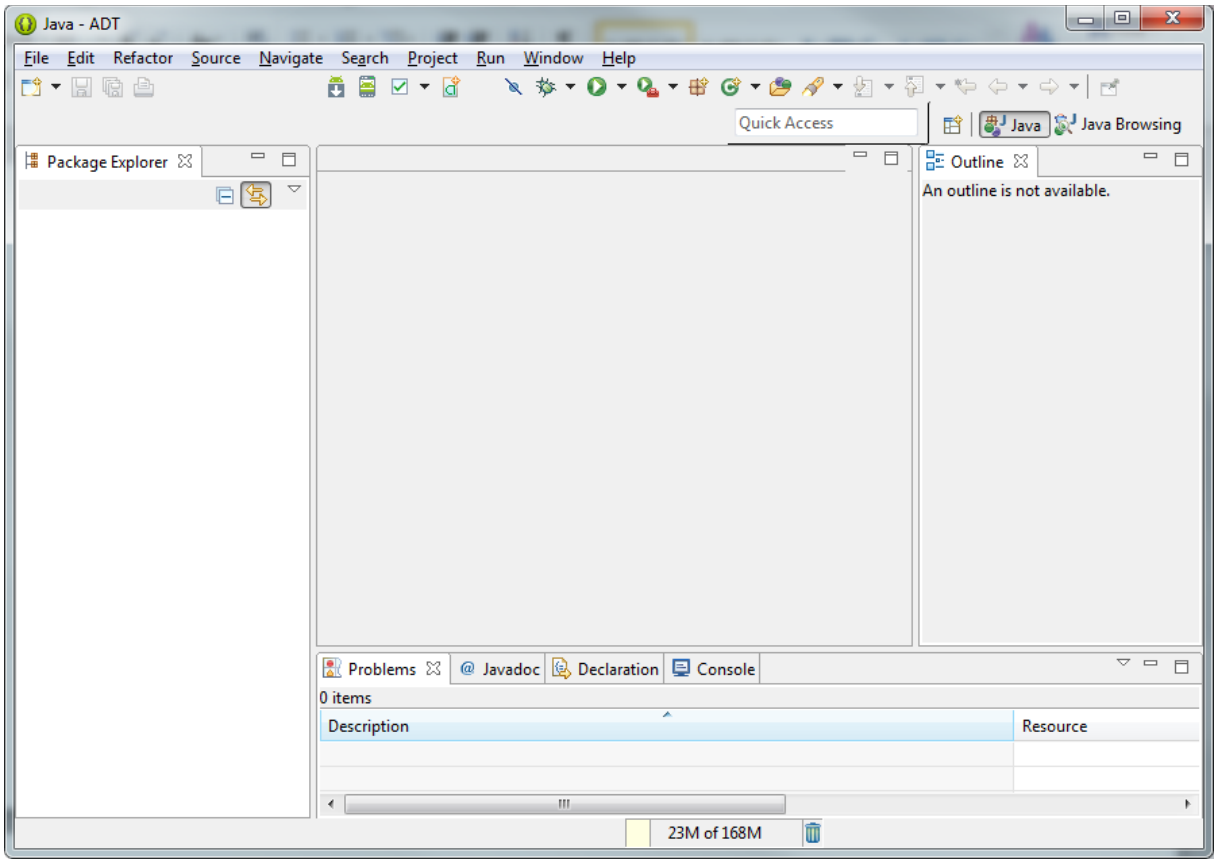
Πλέον έχοντας ένα **πλήρως λειτουργικό σύστημα**, με τον μικροελεγκτή έτοιμο να δεχτεί εντολές και να τυπώσει τα States του κάθε Pin σε οποιονδήποτε Client ακόμα και εκτός του τοπικού δικτύου, είμαστε έτοιμοι με την χρήση των κατάλληλων εργαλείων, να περάσουμε στην δημιουργία της **Android Εφαρμογής** η οποία θα **αυξήσει** την **χρηστικότητα** του συστήματος κατακόρυφα.

8. Δημιουργία, Σχεδιασμός και Λειτουργία Android Εφαρμογής

Σε αυτό το κεφάλαιο θα δούμε αναλυτικά τα βήματα που χρειάστηκαν για να στήσουμε το προγραμματιστικό περιβάλλον του **Eclipse** με τα απαραίτητα **SDK**s και **Platform Tools** για το **Android**. Θα δούμε πως δημιουργήθηκε το **Project**, ποια **Building Blocks** χρησιμοποιήθηκαν στην εφαρμογή και ποια είναι η λειτουργία τους. Ακόμα θα προβληθούν κάποια κομμάτια του κώδικα και θα γίνει παρουσίαση των **Activities** του **Interface** της εφαρμογής. Επιπλέον, θα γίνει επίδειξη στην λειτουργικότητα της εφαρμογής και το πως αυτή αλληλεπιδρά με τον **Arduino Server**.

8.1 Εγκατάσταση ADT στο Eclipse

Ανοίγοντας το **Eclipse**, δηλαδή το προγραμματιστικό περιβάλλον στο οποίο θα γράψουμε τον κώδικα της εφαρμογής για **Android**, θα βρεθούμε μπροστά σε τέσσερα βασικά panels και σε μια γρήγορη μπάρα εργαλείων, τα οποία είναι και το προεπιλεγμένο interface για προγραμματισμό σε γλώσσα **Java**.

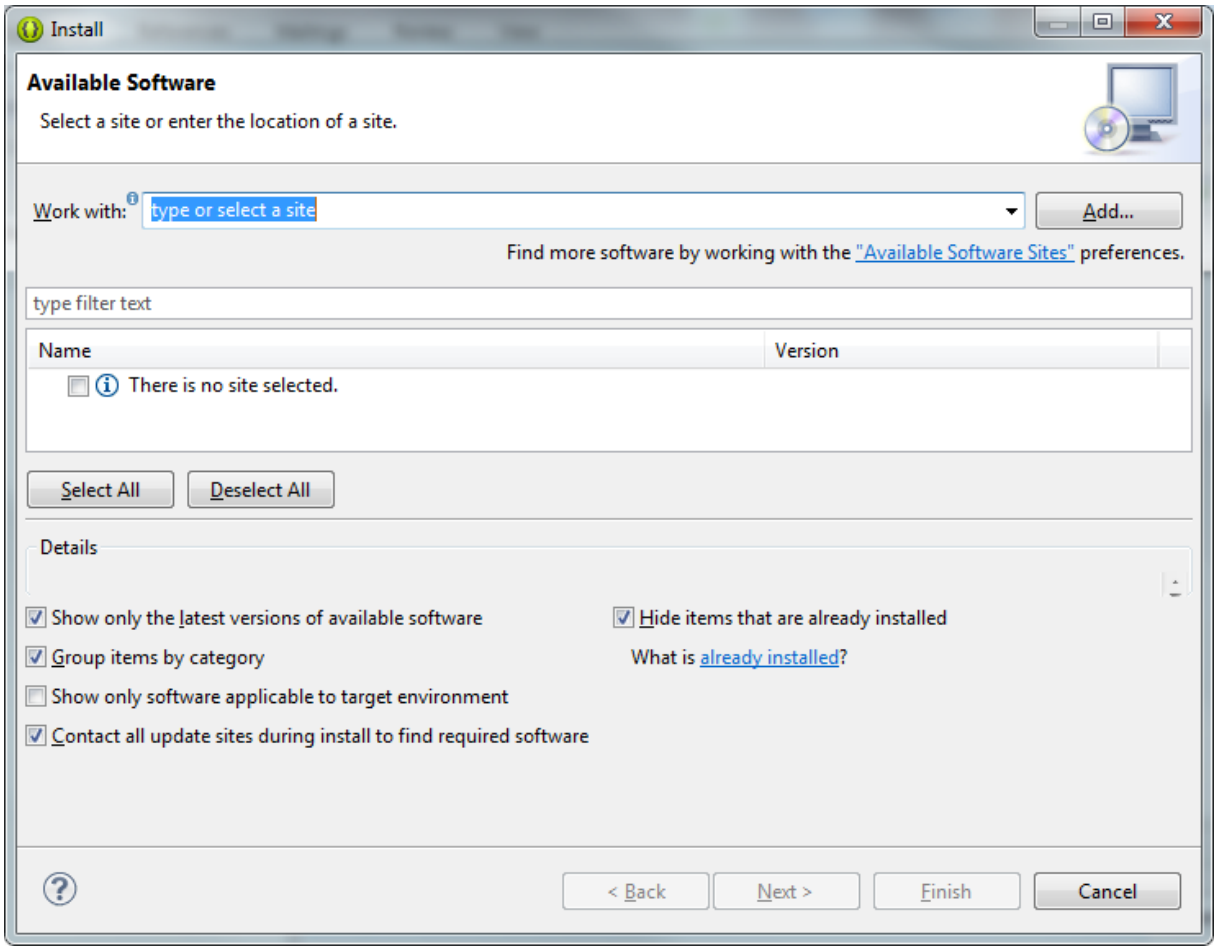


Εικόνα 67: Προγραμματιστικό Περιβάλλον Eclipse

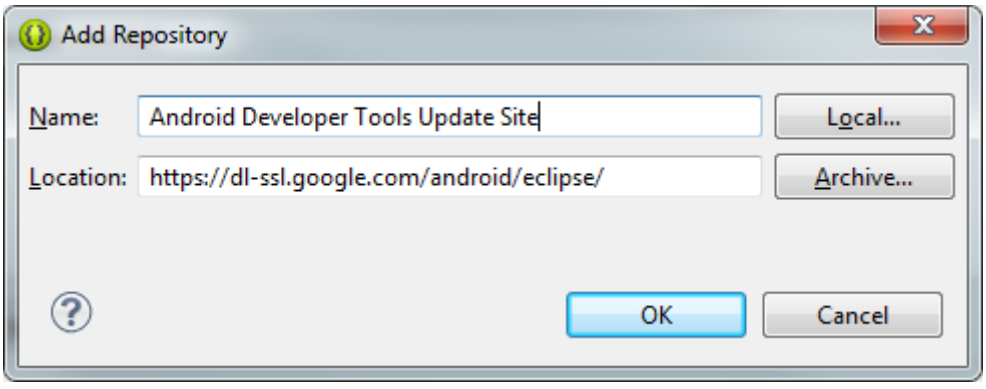
Για να μπορέσουμε να προγραμματίσουμε για συσκευές με **Android OS**, θα πρέπει να κατεβάσουμε το **ADT plugin**, το οποίο θα μας παρέχει τα απαραίτητα εργαλεία και **APIs** για την δημιουργία της εφαρμογής [32].

Επιλέγουμε λοιπόν από το μενού **Help → Install New Software**. Στην συνέχεια πατάμε "**Add**", στην πάνω δεξιά γωνία του παραθύρου που άνοιξε για να εισάγουμε την πηγή από όπου θα

κάνουμε λήψη του λογισμικού. Δίνουμε την ονομασία "**Android Developer Tools Update Site**" και βάζουμε ως πηγή την "**https://dl-ssl.google.com/android/eclipse/**" και μετά πατάμε "**OK**".

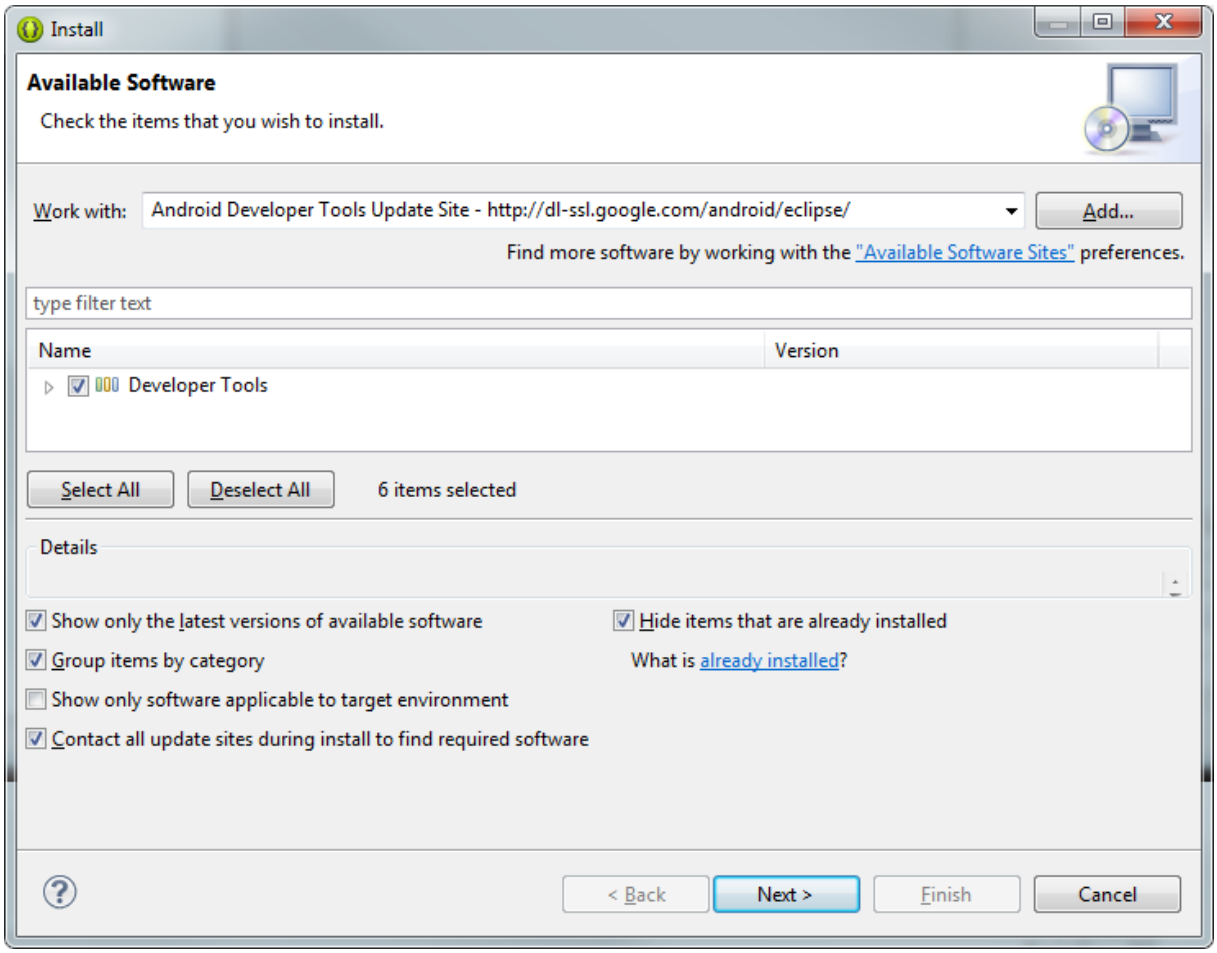


Εικόνα 68: Παράθυρο Εγκατάστασης ADT



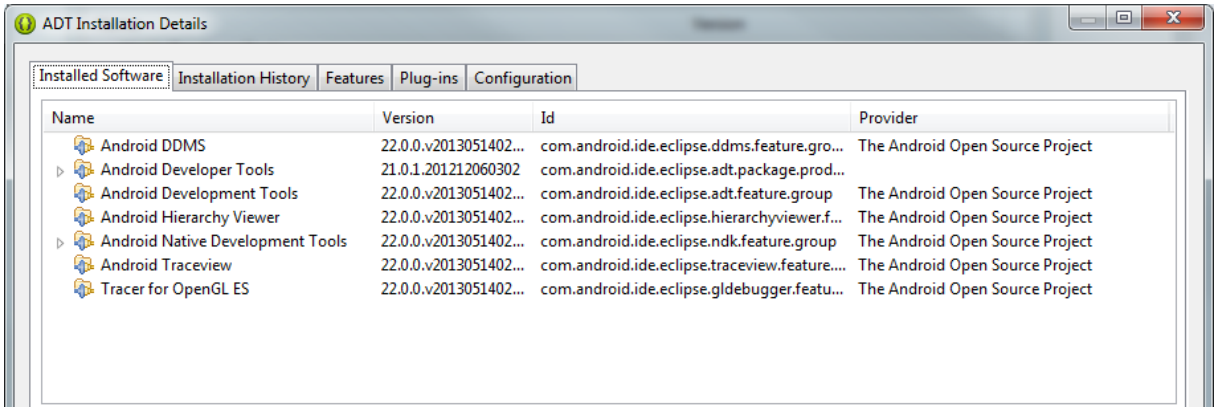
Εικόνα 69: Εισαγωγή Πηγής Λήψης του ADT Plugin

Επιστρέφουμε στο προηγούμενο παράθυρο και επιλέγουμε την πηγή που μόλις προσθέσαμε. Ακριβώς από κάτω επιλέγουμε το **checkbox** δίπλα από το "**Developer Tools**" και στην συνέχεια πατάμε "**Next**".



Εικόνα 70: Επιλογή Εγκατάστασης των Developer Tools

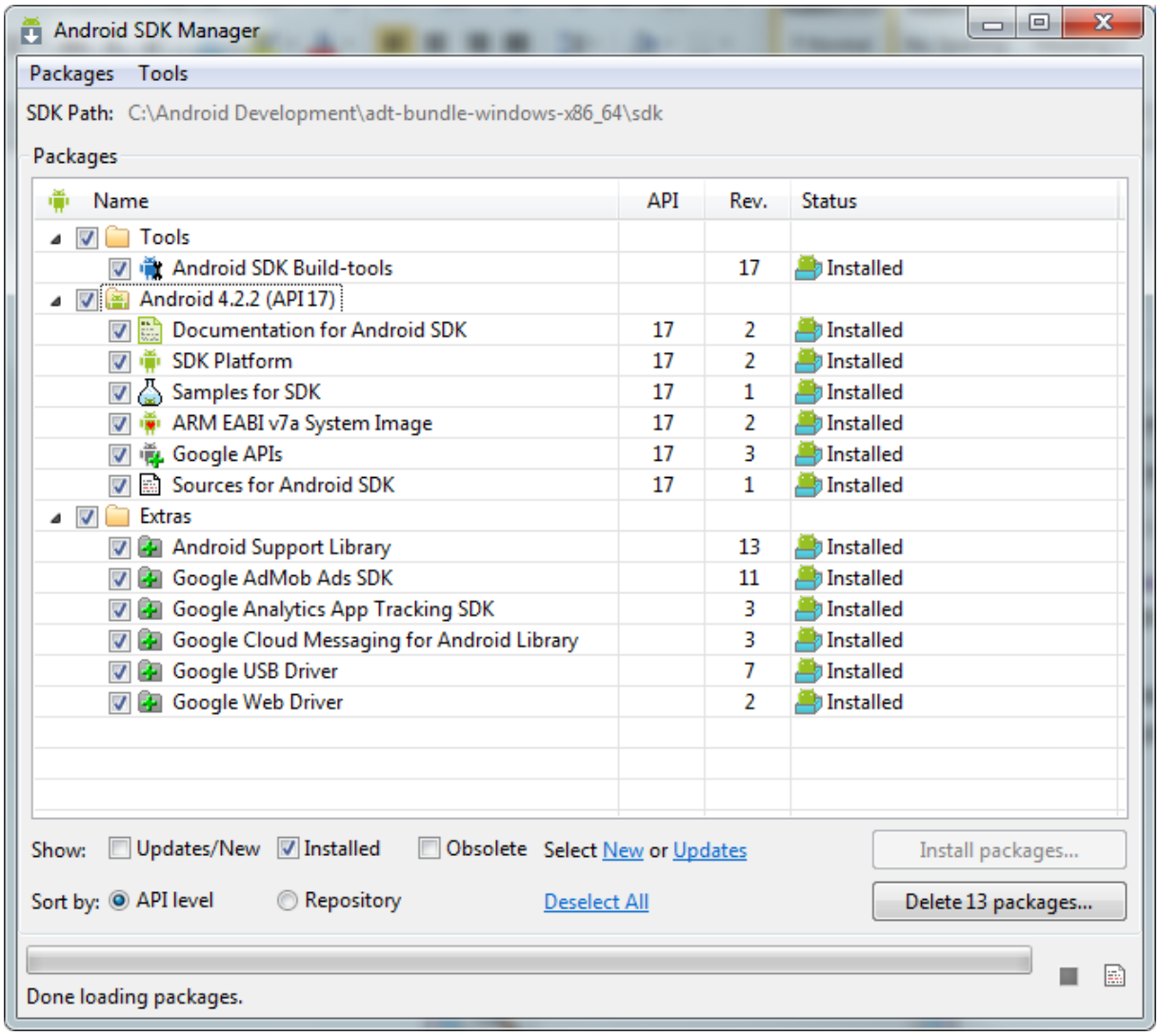
Στο επόμενο παράθυρο βλέπουμε μια λίστα από εργαλεία που είναι προς εγκατάσταση. Πατάμε "**Next**".



Εικόνα 71: Εποπτεία Εργαλείων προς Εγκατάσταση

Στην συνέχεια διαβάζουμε και αποδεχόμαστε τις άδειες χρήσης και τέλος πατάμε "**Finish**". Μόλις ολοκληρωθεί η εγκατάσταση, κάνουμε επανεκκίνηση στο **Eclipse.** Πλέον έχουμε τα εργαλεία που χρειαζόμαστε για την ανάπτυξη της εφαρμογής, αλλά όχι όλα. Θα πρέπει να προσθέσουμε λοιπόν κάποια **packages** επιπλέον.

Ξεκινάμε λοιπόν πατώντας στο εικονίδιο του **SDK Manager** στην μπάρα εργαλείων. Όταν ανοίξει ο SDK Manager για πρώτη φορά, αρκετά **packages** θα είναι επιλεγμένα από προεπιλογή. Εμείς χρειαζόμαστε τα "**Android SDK Build Tools**", όλα τα απαραίτητα για το "**Android 4.2.2 (API 17)**", καθώς και τα "**Extras**".

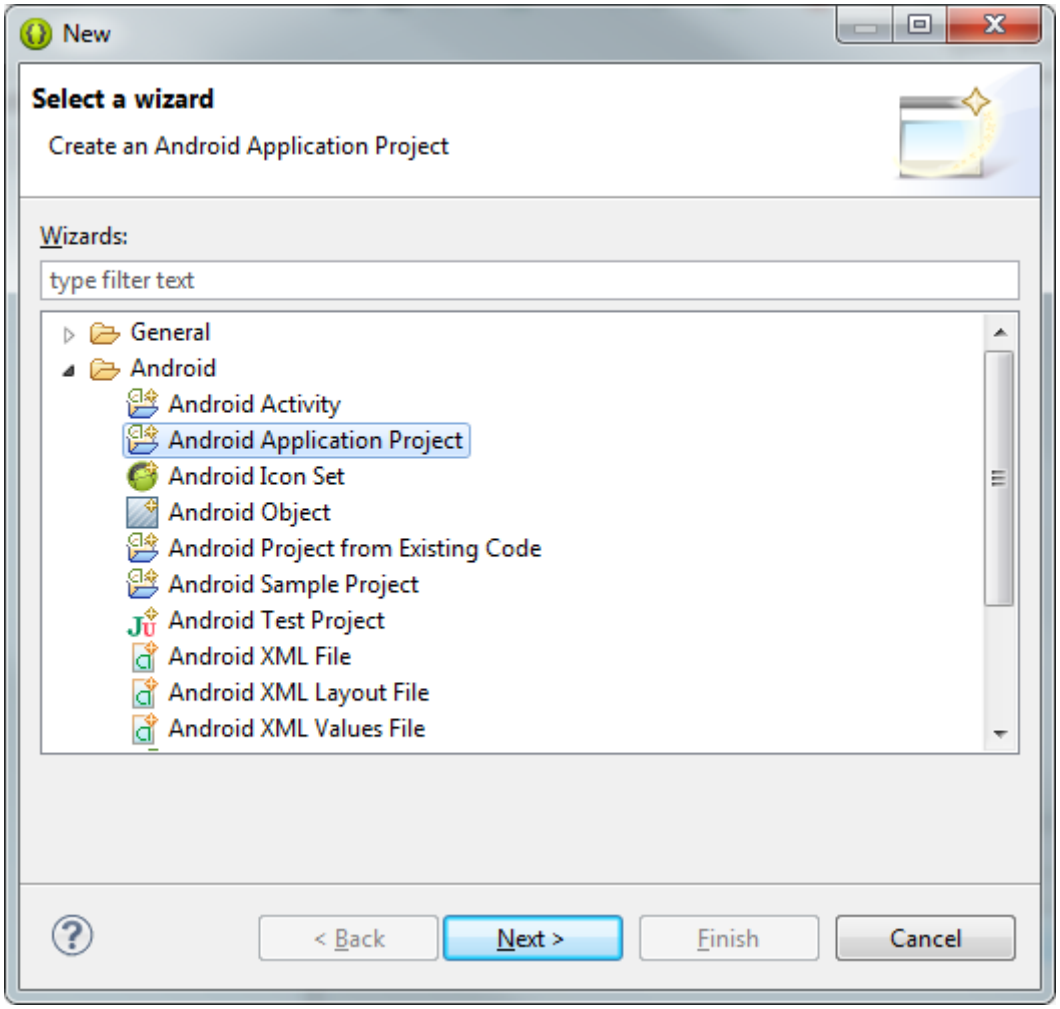


Εικόνα 72: Επιλογή Packages στον Android SDK Manager

Αφού επιλέξουμε τα παραπάνω πατάμε "**Install Packages**". Στο επόμενο παράθυρο αποδεχόμαστε όλες τις άδειες χρήσης, πατάμε "**Install**" και περιμένουμε να ολοκληρωθεί η εγκατάσταση. Πλέον το προγραμματιστικό περιβάλλον είναι έτοιμο για να ξεκινήσουμε την συγγραφή και το σχεδιασμό της εφαρμογής.

8.2 Δημιουργία Android Project

Για να ξεκινήσουμε ένα νέο Project πατάμε το κουμπί "**New**" στην μπάρα εργαλείων. Στο παράθυρο που εμφανίζεται, ανοίγουμε τον φάκελο "**Android**", επιλέγουμε το "**Android Application Project**" και έπειτα πατάμε "**Next**".

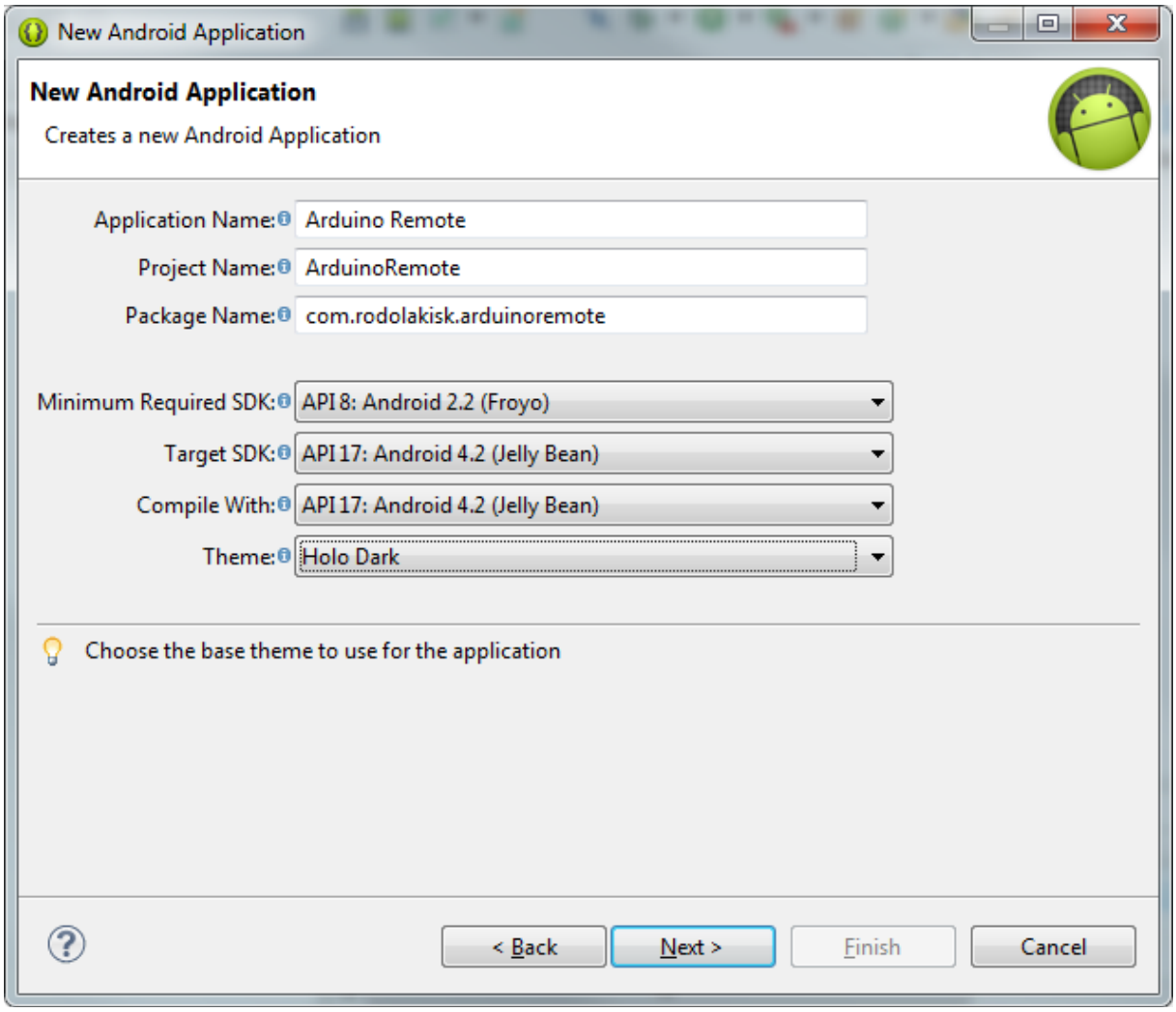


Εικόνα 73: Επιλογή Τύπου Νέου Project στο Eclipse

Στην συνέχεια συμπληρώνουμε την φόρμα που εμφανίζεται. Στο πεδίο "**Application Name**" συμπληρώνουμε το όνομα της εφαρμογής έτσι όπως θα εμφανίζεται στους χρήστες. Εδώ δόθηκε το όνομα "**Arduino Remote**". Στο πεδίο "**Project Name**" συμπληρώνουμε το όνομα του φακέλου που θα περιέχει το Project και είναι το ίδιο με αυτό που θα είναι ορατό και στο Eclipse. Δόθηκε το όνομα "**ArduinoRemote**". Στο πεδίο "**Package Name**" συμπληρώνουμε το όνομα του **package** της εφαρμογής μας ακολουθώντας τους κανόνες που ισχύουν και στην ονοματολογία τη **Java**, ενώ θα πρέπει να είναι μοναδικό σε όλο το Android σύστημά μας. Εδώ δόθηκε το όνομα "**com.rodolakisk.arduinoremote**".

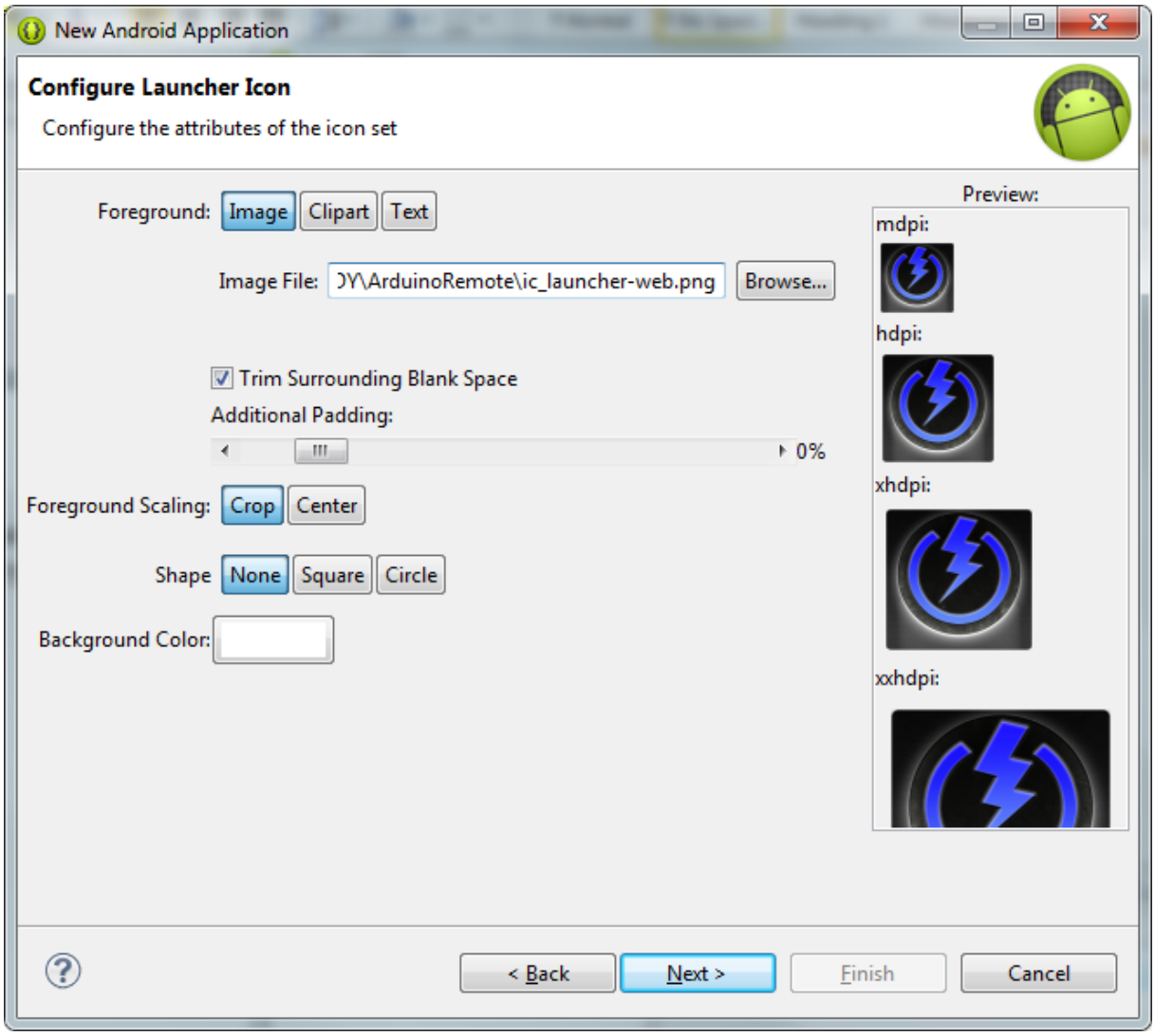
Στο "**Minimum Required SDK**" διαλέγουμε την ελάχιστη έκδοση του Android που θέλουμε να υποστηρίζει η εφαρμογή μας. Στο "**Target SDK**" διαλέγουμε την μέγιστη δυνατή έκδοση του Android που θέλουμε να υποστηρίζει η εφαρμογή και κατ' επέκταση το αντίστοιχο επίπεδο του **API**. Στο πεδίο "**Compile With**" είναι η έκδοση της πλατφόρμας βάσει της οποία θα γίνει **compile** η εφαρμογή. Προεπιλεγμένη είναι ήδη η τελευταία διαθέσιμη έκδοση που υπάρχει στο **SDK** μας,

δηλαδή στην προκειμένη περίπτωση το **Android 4.2**. Στο πεδίο "**Theme**", απλώς επιλέγουμε το αρχικό θέμα της εφαρμογής μας (σκούρο ή ανοιχτόχρωμο).



Εικόνα 74: Φόρμα Στοιχεία Νέας Εφαρμογής Android

Αφού τα συμπληρώσουμε όλα πατάμε "**Next**". Στο επόμενο παράθυρο που ανοίγει αφήνουμε τα προεπιλεγμένα και πατάμε πάλι "**Next**". Σε αυτό το παράθυρο μπορούμε να φτιάξουμε το **εικονίδιο** από το οποίο θα γίνεται εκκίνηση της **εφαρμογής** (**Launcher Icon**). Εδώ θα φορτωθεί το εικονίδιο που φτιάχτηκε αποκλειστικά για αυτή την εφαρμογή, ενώ αυτόματα θα ρυθμιστεί το μέγεθός του για όλα τα μεγέθη οθονών. Στην συνέχεια πατάμε πάλι "**Next**".



Εικόνα 75: Επιλογή του Launcher Icon

Στο επόμενο παράθυρο επιλέγουμε το "**Blank Activity**" και πατάμε "**Next**". Τέλος, δίνουμε το όνομα στο κεντρικό Activity της εφαρμογής και πατάμε "**Finish**". Εδώ δόθηκε το όνομα "**DeviceActivity**".

Πλέον το **Project** είναι έτοιμο με ένα βασικό κενό Activity, και μπορούμε να ξεκινήσουμε το **σχεδιασμό** της εφαρμογής.

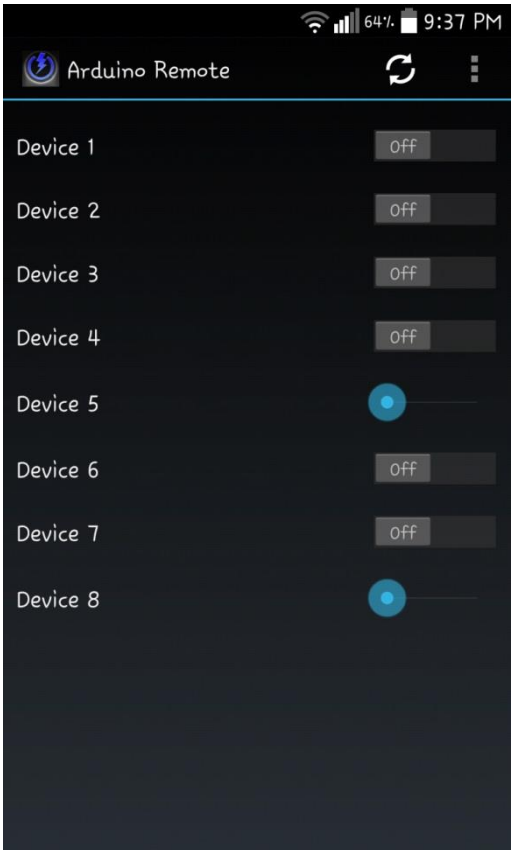
8.3 Το Interface της Android Εφαρμογής και η Λειτουργία του

Για ευκολότερη κατανόηση της λειτουργίας των κομματιών του κώδικα αργότερα, θα γίνει αναφορά πρώτα στο τελικό **Interface** της εφαρμογής και στα ***Building Blocks*** που χρησιμοποιήθηκαν.

Όλα τα **Screenshots** που θα ακολουθήσουν, είναι η διεπαφή, όπως αυτή φαίνεται από το LG Optimus G που χρησιμοποιήθηκε ως η κύρια συσκευή ανάπτυξης της εφαρμογής.

8.3.1 Το Κεντρικό Interface στο Arduino Remote

Ας ξεκινήσουμε την ανάλυση στο **Interface** της εφαρμογής από το **Main Activity** της, που είναι και η πρώτη διεπαφή που αντικρίζουμε μόλις ανοίξουμε την εφαρμογή.



Εικόνα 76: Screenshot από το Κεντρικό Interface Android Εφαρμογής

Για να ξεκινήσουμε την εφαρμογή μας από το κινητό, πατάμε το εικονίδιο που προσθέσαμε στο τελευταίο βήμα του **Project** στην προηγούμενη ενότητα. Μόλις ανοίξει η εφαρμογή, αντικρίζουμε το **Interface** που βλέπουμε στην παραπάνω εικόνα.

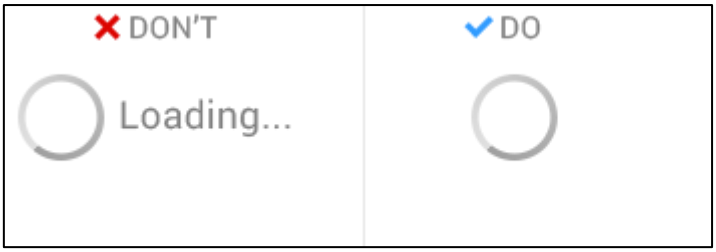
Το Interface αυτό αποτελείται από αρκετά δομικά στοιχεία. Στην πάνω μεριά βλέπουμε την "***Action Bar***" [33]. Η **Action Bar** είναι ένα χαρακτηριστικό παραθύρου το οποίο προσδιορίζει την τοποθεσία που βρίσκεται ο χρήστης. Ταυτόχρονα προσφέρει στον χρήστη την δυνατότητα να έχει αλληλεπίδραση με την εφαρμογή μέσω κουμπιών όπως επίσης και να περιηγηθεί ενώ του παρέχεται μια οικεία διεπαφή μεταξύ των εφαρμογών, η οποία προσαρμόζεται για διαφορετικές ρυθμίσεις οθονών.

Εικόνα 77: Η Action Bar στο Κεντρικό Interface της Android Εφαρμογής

Στην αριστερή μεριά βλέπουμε το λογότυπο της εφαρμογής ("***App Icon***" [33]) από το οποίο έγινε και η εκκίνησή της από τον **Launcher** του λειτουργικού. Ακριβώς δίπλα βλέπουμε το όνομα της

εφαρμογής, δηλαδή "**Arduino Remote**" μιας και βρισκόμαστε στο κεντρικό Interface της. Σε οποιοδήποτε άλλο Activity, στη θέση του ονόματος της εφαρμογής υπάρχει το όνομα του εκάστοτε Activity (όπως θα δούμε παρακάτω) το οποίο ουσιαστικά μας δηλώνει και που βρισκόμαστε ("***View Control***" [33]).

Δεξιά από το όνομα της εφαρμογής, βλέπουμε το κουμπί "**Refresh**" ("***Action Button***" [33] [34]). Όταν ο χρήστης το πατάει, δίνεται εντολή στην εφαρμογή να επικοινωνήσει με τον Server. Ανάλογα με το αν υπάρχει επικοινωνία ή όχι, η εφαρμογή πράττει ανάλογα ενημερώνοντάς μας για το αποτέλεσμα όπως θα δούμε παρακάτω. Όσο η εφαρμογή προσπαθεί να επικοινωνήσει με τον **Server**, το **Refresh Button**, μετατρέπεται σε "***Activity Circle***" [35]. Γίνεται δηλαδή ένας κινούμενος περιστρεφόμενος κύκλος, ο οποίος δηλώνει ότι υπάρχει **δραστηριότητα στο παρασκήνιο**, επομένως ο χρήστης πρέπει να περιμένει να ολοκληρωθεί η δραστηριότητα για να συνεχίσει την αλληλεπίδραση του με το γραφικό περιβάλλον. Μόλις ο κύκλος επανέλθει στην αρχική μορφή που είχε το κουμπί, σημαίνει ότι η δραστηριότητα του παρασκηνίου τελείωσε. Τα **Design Guidelines** συστήνουν ότι ο κύκλος και μόνο παρέχει αρκετό Feedback στον χρήστη, οπότε σχόλια τύπου "Loading" δίπλα από αυτόν, πρέπει να αποφεύγονται [35].



Εικόνα 78: Activity Circle στο Κεντρικό Interface της Εφαρμογής (δεξιά)

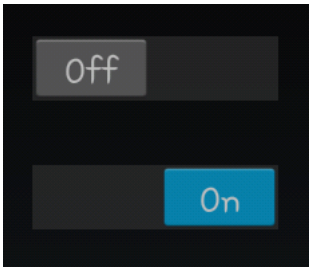
Δεξιά από το **Refresh Button** βρίσκεται το κουμπί "***Action Overflow***" [33]. Σε αυτό το κουμπί "**Menu**" προστίθενται όλες οι λειτουργίες και οι δραστηριότητες της εφαρμογής που δεν χρησιμοποιούνται τόσο συχνά, και οι οποίες θα βάραιναν οπτικά την **Action Bar** καθώς επίσης θα δυσχέραιναν τον χρήστη στην λειτουργία της εφαρμογής αν ήταν κάπου πιο ορατά.

Περνώντας ακριβώς από κάτω, βλέπουμε μια γραμμική διάταξη ("***Linear Layout***" [36]) με κάθετο προσανατολισμό, που αποτελείται από μια λίστα με "***Single Line Text View***" [37] με 8 διαφορετικές ονομασίες συσκευών. Αυτά είναι τα "***Text View Fields***". Είναι **Building Blocks** που απλώς περιέχουν πληροφορία με την μορφή κειμένου. Τα συγκεκριμένα όμως είναι δυναμικά και έχουν δράση και ως "***Text Fields***" [37]. Τα **Text Fields** επιτρέπουν στον χρήστη την εισαγωγή κειμένου μέσα στην εφαρμογή. Εδώ, τα **Text View Fields** θα βοηθήσουν τον εκάστοτε χρήστη να θυμάται ποια συσκευή ελέγχεται κάθε φορά, βάσει του ονόματος που έχει δώσει στο **Text View Field**. Από προεπιλογή τα **Text View Fields** έχουν ονομασίες "**Device 1, Device 2… Device 8**".

Αξίζει ακόμα να σημειωθεί ότι αν η οθόνη της συσκευής είναι τόσο μικρή ώστε να μην επιτρέπει την εμφάνιση και των 8 στοιχείων της λίστας, τότε δημιουργείται ένας "***Scrolling Indicator***" [38] στην δεξιά μεριά της διεπαφής, που υποδεικνύει ότι υπάρχει η δυνατότητα εμφάνισης της υπόλοιπης λίστας κάνοντας **Scrolling** κάθετα. Η ταχύτητα κύλισης είναι ανάλογη της ταχύτητας κύλισης του δακτύλου στην οθόνη κατά τη διάρκεια της αλληλεπίδρασης.

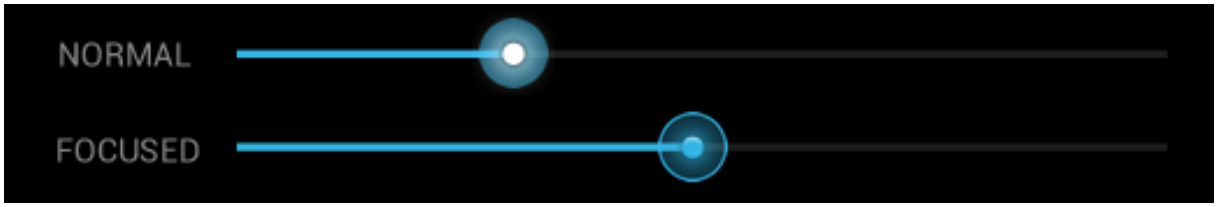
Εικόνα 79: Text View Field στο Κεντρικό Interface της Android Εφαρμογής

Δεξιά, σε κάθε ένα στα έξι από τα οχτώ **Text View Fields** στο κεντρικό **Interface** της εφαρμογής, βρίσκεται ένα "***On/Off Switch***" [39] το οποίο είναι με τη σειρά του και αυτό **γραμμικά διατεταγμένο** με **οριζόντιο προσανατολισμό** ως προς το αντίστοιχο **Text View Field** του, άρα κατ' επέκταση γραμμικά διατεταγμένα κάθετα μεταξύ τους. Αυτό το ***Building Block***, είναι ένας **διακόπτης** ο οποίος θέτει κάθε φορά μια λειτουργία στην εκάστοτε εφαρμογή **ενεργή ή ανενεργή**. Η κίνηση του διακόπτη από τα αριστερά προς τα δεξιά θέτει την λειτουργία ενεργή, ενώ το αντίθετο την απενεργοποιεί. Στην παρούσα εφαρμογή οι διακόπτες δείχνουν στον χρήστη σε τι κατάσταση βρίσκονται οι εκάστοτε συσκευές. Αν μια συσκευή είναι ανοιχτή, ο διακόπτης είναι στην θέση "**On**", ενώ αν είναι κλειστή, ο διακόπτης είναι στην θέση "**Off**". Ταυτόχρονα του δίνουν την δυνατότητα να αλληλεπιδρά μαζί τους για να **αλλάξει την κατάσταση των συσκευών που αντιστοιχούν στα Pins που ελέγχονται από τον Arduino Server με τον οποίο επικοινωνεί η εφαρμογή**. Αν ο χρήστης θέλει μια συσκευή να είναι ενεργή, θέτει τον διακόπτη που της αντιστοιχεί στην θέση "**On**", ενώ αν θέλει να είναι κλειστή τον θέτει στην θέση "**Off**". Για να γίνει ο έλεγχος και στις δύο περιπτώσεις (ανάγνωσης state και ελέγχου Pins), πρέπει σαφώς να υπάρχει σωστή επικοινωνία με τον **Arduino Server**, ενώ πρέπει να σημειωθεί ότι ανάλογα την θέση που τίθεται κάθε φορά από το χρήστη ο διακόπτης, στέλνεται και το αντίστοιχο **Get Request** που αντιστοιχεί στο εκάστοτε **Pin** στον **Arduino Server**.



Εικόνα 80: On/Off Switches στο Κεντρικό Interface της Android Εφαρμογής

Διπλά από τα εναπομείναντα δύο **Text View Fields**, βλέπουμε δύο "***Interactive Sliders***" [40] στα οποία ισχύει το ίδιο **Layout** με αυτό των **Switches**. Τα **Interactive Sliders** έχουν την ιδιότητα του να βοηθάνε τον χρήστη να επιλέξει μια τιμή από έναν συνεχή, πεπερασμένο αριθμό τιμών, απλά σύροντας τον **Slider** προς τα δεξιά ή τα αριστερά. Η **μικρότερη** τιμή είναι στην άκρως αριστερή μεριά, ενώ η μέγιστη στην άκρως δεξιά. Η διαδραστική φύση αυτού του **Building Block**, το κάνει ιδανικό για την χρήση του στην **επιλογή των αναλογικών τιμών που αντιστοιχούν στα Pins 6 και 9 του Arduino Server**. Οι τιμές που δίνει ο **Slider** όταν γίνεται χρήση του, είναι από **0** έως και **255**. Αυτές φυσικά αντιστοιχούν στα **Get Requests** που δίνουν τις αντίστοιχες τιμές από **0** έως **255** στον Arduino Server όπως αναλύθηκαν στο **Κεφάλαιο 6**. Το συγκεκριμένο **Building Block** έχει δύο χαρακτηριστικά ανάδρασης στην αλληλεπίδραση μαζί του. Η μία μορφή είναι για όταν δεν υπάρχει αλληλεπίδραση μαζί του, και η άλλη είναι όταν σύρεται από τον χρήστη προς τα αριστερά ή τα δεξιά.



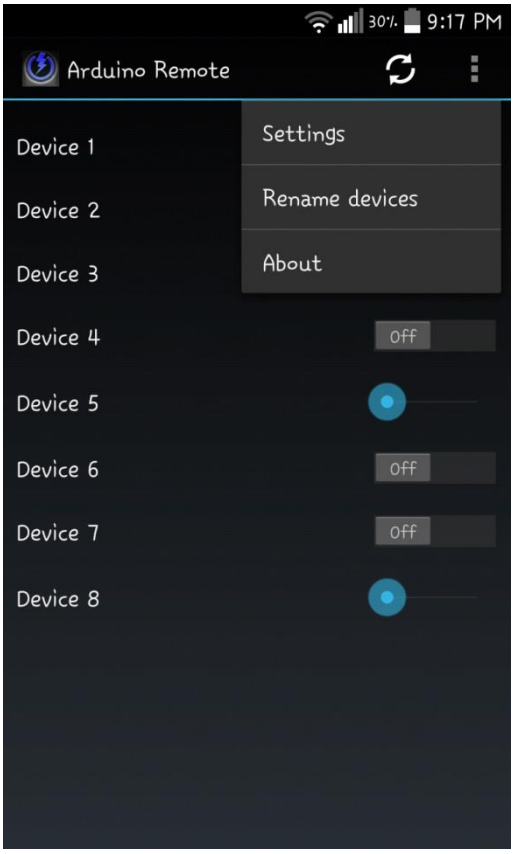
Εικόνα 81: Τα Δύο States του Interactive Slider της Android Εφαρμογής

Αποφασίστηκε στην διεπαφή να υπάρχουν μόνο δύο **Interactive Sliders**, δηλαδή μόνο δύο Pins να ελέγχονται ως αναλογικά αντί τέσσερα που έχει δυνατότητα ο Server. Αυτό έγινε διότι η αναλογική λειτουργία δεν είναι τόσο συνήθης και ο απλός On/Off διακόπτης προσφέρει μεγαλύτερη ευκολία στη χρήση.

8.3.2 Το Interface των Ρυθμίσεων στο Arduino Remote

Για να ρυθμίσουμε την εφαρμογή ώστε να μπορέσει να λειτουργήσει σωστά, πρέπει κάπου να μπορούμε να εισάγουμε και να αποθηκεύσουμε τα στοιχεία του Server, ώστε να ξέρει η εφαρμογή που να στείλει τα Get Requests. Αυτά τα δύο στοιχεία είναι η **IP Address** ή το **DNS Hostname** που του έχουμε δώσει και η **Port** που αντιστοιχεί στην υπηρεσία του Server, ενώ το μέρος στο οποίο θα πραγματοποιηθεί η καταχώρηση των στοιχείων, είναι το "**Settings**" **Interface**.

Για να ανοίξουμε το Activity των Ρυθμίσεων, πατάμε το **Action Overflow** και επιλέγουμε "**Settings**".



Εικόνα 82: Action Overflow Button

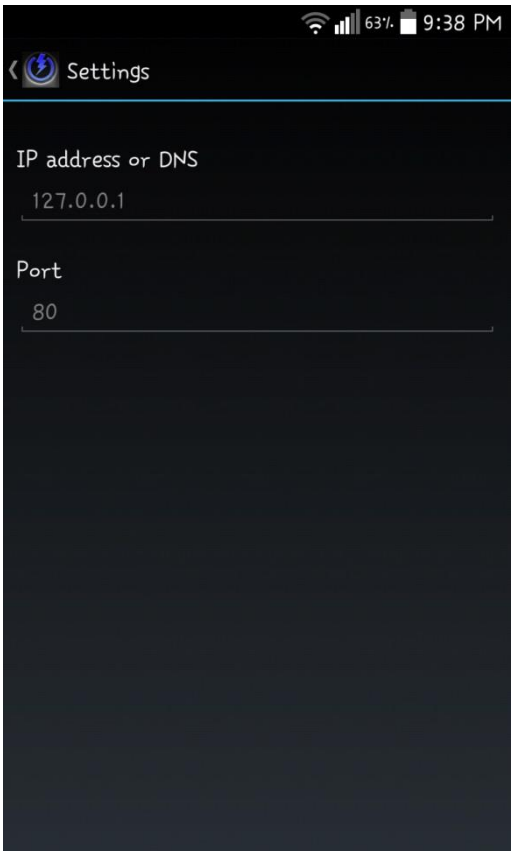
Στιγμιαία μετά την επιλογή, μεταφερόμαστε στο "**Settings**" **Interface**. Αμέσως παρατηρούμε τρία πράγματα. Το ένα είναι το "***View Control***" το οποίο πλέον γράφει "**Settings**", που είναι το όνομα του Activity στο οποίο βρισκόμαστε, αντί για"**Arduino Remote**" που έγραφε όσο ήμασταν στο Main Activity.

Εικόνα 83: Settings Interface Action Bar

Το δεύτερο πράγμα που παρατηρούμε είναι το "***Up Button***" [41] που βρίσκεται αριστερά από το **View Control** και βοηθάει στην ***πλοήγηση*** μέσα στο **Application**. Πατώντας το ο χρήστης θα

βρεθεί ένα Activity πάνω, βάσει της **ιεραρχικής δομής** των "Οθονών" της εφαρμογής, δηλαδή στο Main Activity της εφαρμογής. Όπως είδαμε στο Main Activity δεν υπήρχε **Up Button** διότι δεν υπήρχε Activity ανώτερο ιεραρχικά από το Main Activity που ήμασταν ήδη. Σε αντίθεση με το "***Back Button***" της συσκευής που μπορεί να βγάλει τον χρήστη από την εφαρμογή τελείως, ανάλογα από που προήλθε πριν βρεθεί στην εφαρμογή μέσα (βάση της **χρονολογικής** αλληλουχίας αλλαγής οθονών), το **Up Button** τον κρατάει μέσα στην εφαρμογή. ΄

Τέλος βλέπουμε ότι δεν υπάρχει το **Refresh Button** ούτε το **Action Overflow** στην συγκεκριμένη διεπαφή.

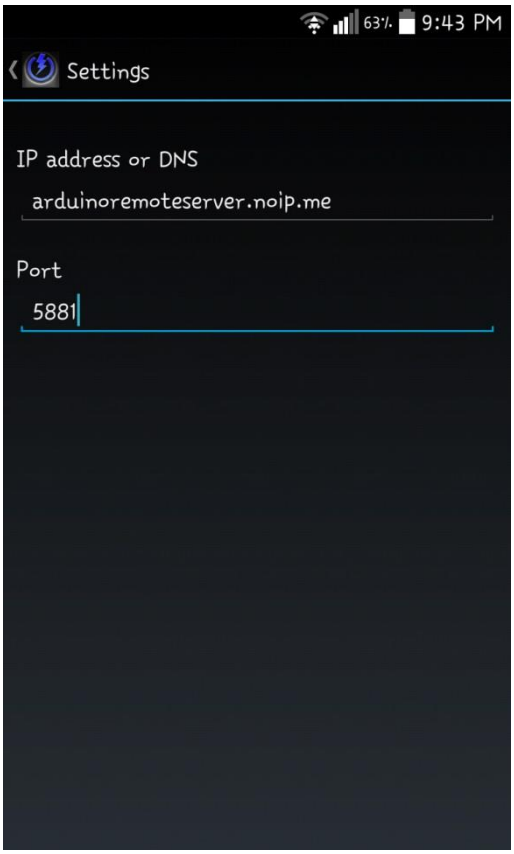


Εικόνα 84: Settings Interface της Android Εφαρμογής

Συνεχίζοντας την ανάλυση, κάτω από την **Action Bar**, βλέπουμε ένα **Linear Layout** με κάθετο προσανατολισμό, που αποτελείται από δύο συνδυασμούς "***Text View***" με "***Text Fields***".

Το πρώτο **Text View** έχει όνομα "**IP address or DNS**". Αυτό δηλώνει ότι στο **Text Field** που ακολουθεί ακριβώς από κάτω, ο χρήστης πρέπει να εισάγει τη διεύθυνση του **Arduino Server**, είτε με την μορφή μιας **IP Address**, είτε με την μορφή ενός **Hostname**. Για να διευκολυνθεί ο χρήστης στην κατανόηση του πεδίου που πρέπει να συμπληρωθεί, υπάρχει ένα "***hint***" [42] το οποίο υποδεικνύει μια πιθανή μορφή μιας IP Address που θα μπορούσε να συμπληρωθεί μέσα στο πεδίο και συγκεκριμένα την "**127.0.0.1**".

Στο δεύτερο **Text View** έχει δοθεί το όνομα "**Port**". Αυτό δηλώνει στον χρήστη ότι στο ***Text Field*** που ακολουθεί, πρέπει να συμπληρώσει σε ποια θύρα αντιστοιχεί η υπηρεσία του Arduino Server. Σε αντίθεση με το πρώτο **Text Field** το οποίο είχε τύπο "**Text**", εδώ το **Field Type** έχει τύπο "**Number**", πράγμα που σημαίνει ότι το πεδίο μπορεί να συμπληρωθεί μόνο με αριθμούς. Για ακόμα μεγαλύτερη διευκόλυνση του χρήστη, το σύστημα αναγνωρίζει ότι πεδίο δέχεται μόνο αριθμούς, με αποτέλεσμα το εικονικό πληκτρολόγιο να ανοίγει αυτόματα στο Interface εισαγωγής αριθμών όταν ο χρήστης αποφασίσει να καταχωρήσει την θύρα.



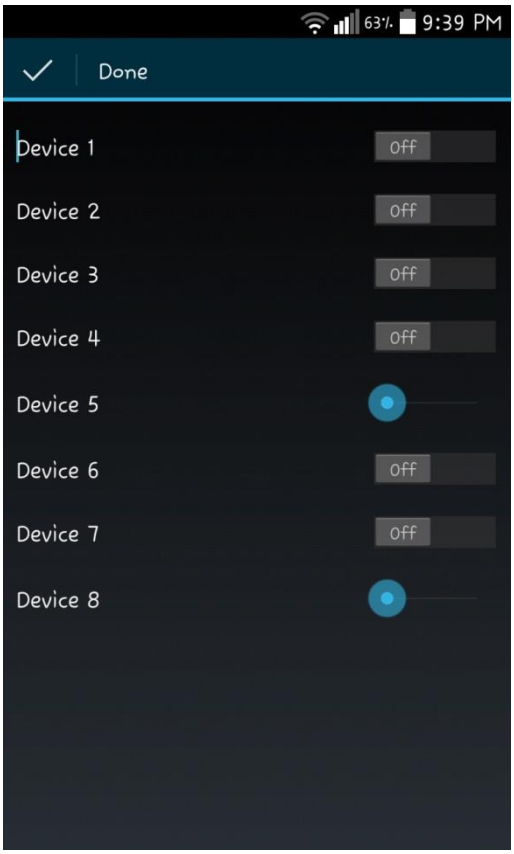
Εικόνα 85: Συμπληρώνοντας τα Στοιχεία στο Settings Interface της Android Εφαρμογής

Αν γίνει οποιαδήποτε εισαγωγή σε ένα από τα πεδία αυτά, το **hint** του πεδίου εξαφανίζεται, δίνοντας τη θέση στο κείμενο και στους αριθμούς που εισάγει ο χρήστης. Μετά από οποιαδήποτε καταχώρηση, αν ο χρήστης πατήσει το **Up Button** της διεπαφής ή το κουμπί **Back** της συσκευής του, τότε τα στοιχεία αποθηκεύονται αυτόματα και είναι έτοιμα για χρήση από την εφαρμογή.

8.3.3 Το Interface της Λειτουργίας Rename στο Arduino Remote

Όπως αναφέρθηκε παραπάνω, στο Κεντρικό Interface της Εφαρμογής, υπάρχει η δυνατότητα τα **Text View** πεδία να γίνουν **Text Fields**. Αυτό θα δώσει στον χρήστη την ικανότητα να δώσει τα δικά του ονόματα στις συσκευές που αντιστοιχούν στα **Pins** που ελέγχονται από την εφαρμογή, αντί να έχει τα προκαθορισμένα "**Device 1, Device 2…**" και φυσικά να τα αποθηκεύσει. Έτσι δεν θα χρειάζεται να θυμάται σε ποιον διακόπτη αντιστοιχεί η κάθε συσκευή, κάνοντας την χρήση της εφαρμογής πιο ευχάριστη.

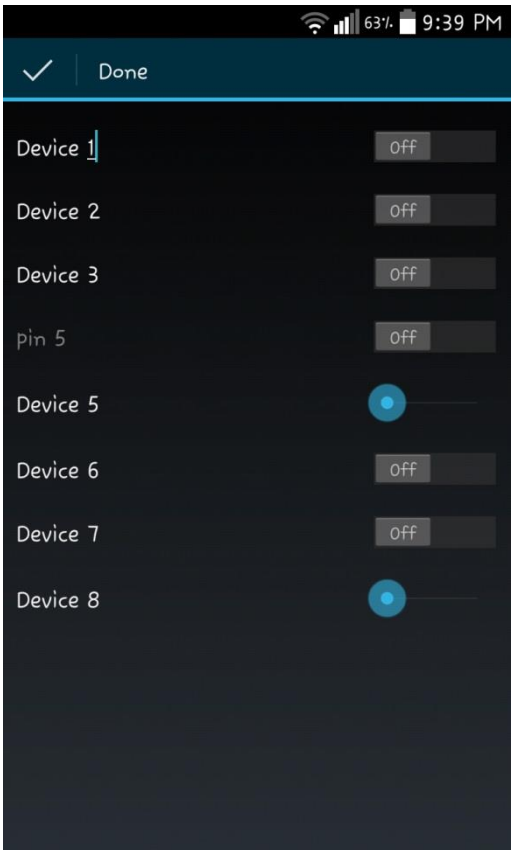
Για να κάνουμε χρήση αυτής της λειτουργίας, πατάμε το **Action Overflow** και επιλέγουμε "**Rename**". Αμέσως μετά την εντολή αυτή, αντικρίζουμε την παρακάτω εικόνα.



Εικόνα 86: Rename Interface της Android Εφαρμογής

Βλέπουμε ότι πλέον τα **Text View** πεδία είναι πλέον **επεξεργάσιμα Text Fields**. Αυτό σημαίνει ότι το περιεχόμενό τους μπορεί να σβηστεί και ο χρήστης να συμπληρώσει ότι θέλει. Αισθητή κάνει την παρουσία του ο κέρσορας στο παραπάνω Screenshot που καταδεικνύει ότι το πεδίο με το όνομα "**Device 1**" είναι στο προσκήνιο και έτοιμο για διόρθωση.

Για διευκόλυνση του χρήστη ώστε να ξέρει ποιο **Pin** του Server ελέγχεται από τον διακόπτη που αντιστοιχεί σε κάθε **Text Field**, μόλις σβηστεί τελείως η ονομασία κάποιας από τις συσκευές, εμφανίζεται ένα **hint** [42] που του το υποδεικνύει όπως φαίνεται στο παρακάτω Screenshot για το "**Device 4**" που έχει σβηστεί και υποδεικνύει ότι αντιστοιχεί στο "**Pin 5**".



Εικόνα 87: Hint του Pin του Διακόπτη στο Rename Interface της Android Εφαρμογής

Αφού αλλάξουν οι ονομασίες σε όποιες συσκευές επιθυμεί ο χρήστης, η προσοχή μας περνάει στην **Action Bar**.

Εικόνα 88: Contextual Action Bar στο Rename Interface της Android Εφαρμογής

Παρατηρούμε ότι έχει χάσει το μαύρο χρώμα που είχε αρχικά. Πλέον έχει χρωματιστεί με ένα βαθύ μπλε. Επιπλέον βλέπουμε να λείπουν όλα τα στοιχεία που την χαρακτήριζαν στο Κεντρικό Interface της εφαρμογής. Αυτό έγινε διότι κατά τη διάρκεια της εντολής "**Rename**", η **Action Bar επικαλύφθηκε** από μια προσωρινή Action Bar με την ονομασία "***Contextual Action Bar***" [43].

Η χρήση της **Contextual Action Bar** γίνεται συνήθως κατά τη διάρκεια υποεργασιών. Αυτές μπορεί να περιλαμβάνουν την επιλογή δεδομένων ή την επεξεργασία κειμένου όπως στην προκειμένη περίπτωση.

Μόλις ο χρήστης ολοκληρώσει με την ονοματολογία των συσκευών που επιθυμεί, μπορεί να πατήσει το "**Check**" **Button** δίπλα από την λέξη "**Done**" ή απλώς να πατήσει το **Back Button** της συσκευής του ώστε να αποθηκευτούν οι νέες ονομασίες. Μόλις το κάνει θα επανέλθει στο Κεντρικό Interface του Main Activity της εφαρμογής με τις νέες ονομασίες στη θέση των "**Device x**" όπου υπήρξε αλλαγή και με την **Contextual Action Bar** να έχει δώσει την θέση της στην **Action Bar** του Κεντρικού Interface.

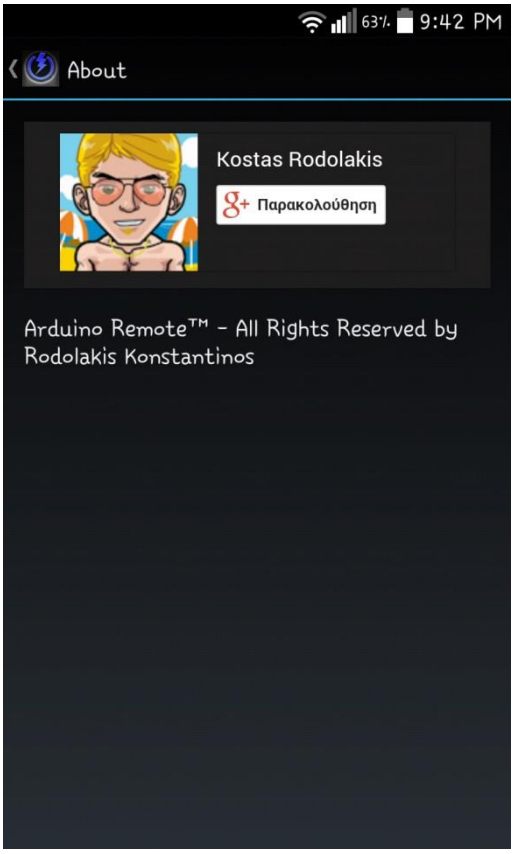
Σε περίπτωση που δοθεί κάποιο όνομα τόσο μεγάλο από τον χρήστη, ώστε να μην χωράει στο ορατό πλαίσιο ανάγνωσης, δίνεται η δυνατότητα από την εφαρμογή να γίνει ανάγνωση του κειμένου

κάνοντας οριζόντιο ***Scrolling*** [38], πατώντας και σέρνοντας προς τα αριστερά το δάχτυλο, πάνω από την επιφάνεια του κειμένου που θέλει ο χρήστης να διαβάσει.

8.3.4 Το Interface του About στο Arduino Remote

Όπως σε κάθε εφαρμογή, έτσι και στο Arduino Remote υπάρχει ένα μέρος μέσα στην εφαρμογή όπου παρέχονται πληροφορίες με στοιχεία επικοινωνίας ή πληροφορίες για τον developer.

Για να ανοίξουμε το Activity των Πληροφοριών, πατάμε το **Action Overflow** και επιλέγουμε "**About**".



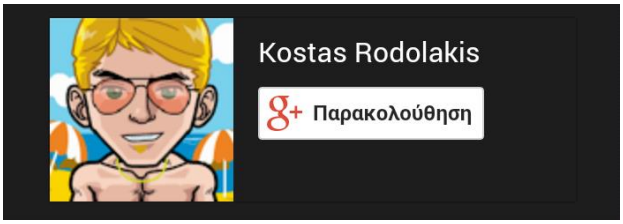
Εικόνα 89: About Interface της Android Εφαρμογής

Μόλις ολοκληρωθεί η ενέργεια, αντικρίζουμε την παραπάνω εικόνα. Όπως και στο "**Settings**" Interface, βλέπουμε στην **Action Bar** ότι το όνομα στο **View Control** πλέον γράφει "**About**" που είναι και το όνομα του Activity στο οποίο βρισκόμαστε. Φυσικά από δίπλα του δεν λείπει το **Up Button** του οποίου η χρήση είναι η ίδια με αυτή που είχε στο "**Settings**" Interface, δηλαδή να μας μεταφέρει στο Κεντρικό Interface της εφαρμογής.

Κάτω από την **Action Bar** βλέπουμε δύο Elements σε "***Relative Layout***" [44]. Αυτό σημαίνει ότι το κάθε στοιχείο προσδιορίζει την θέση του στον "**χώρο**" βάσει της θέσης που έχει ένα άλλο στοιχείο ή και βάσει των στοιχείων της οθόνης (πχ. στοίχιση στο κέντρο της οθόνης, στοίχιση ως προς το δεξί πλαίσιο κτλ). Συγκεκριμένα εδώ έχει γίνει αριστερά και από κάτω στοίχιση, του **Text View** ως προς το **Web View**.

Το "***Web View***" [45] είναι το πρώτο στοιχείο. Η χρήση του **Web View** γίνεται όταν θέλουμε να αντλήσουμε περιεχόμενο από το Web, όπως για παράδειγμα ένα Web Application. Το **Web View** δεν περιέχει τα γνωρίσματα που θα είχε ένας πλήρως ανεπτυγμένος web browser όπως βοηθήματα ελέγχου στην περιήγηση ή μια address bar. Αυτό που κάνει από προεπιλογή, είναι να δείχνει μια

ιστοσελίδα. Ένα πολύ σύνηθες σενάριο χρήσης είναι όταν θέλουμε να παρέχουμε πληροφορίες για μια εφαρμογή και αργότερα να θέλουμε να ενημερώσουμε αυτές τις πληροφορίες, όπως για παράδειγμα κάποια άδεια χρήσης. Σε μια τέτοια περίπτωση αρκεί να γίνει χρήση ενός **Web View**, το οποίο θα δείχνει σε ένα **hosted** αρχείο που θα περιέχει όλες τις απαραίτητες πληροφορίες. Όταν λοιπόν θελήσουμε να τις ενημερώσουμε, αρκεί να αλλάξουμε το περιεχόμενο αυτού του αρχείου, αντί να πρέπει να ενημερώσουμε την ίδια την εφαρμογή.



Εικόνα 90: Περιεχόμενο Web View στο About Interface της Android Εφαρμογής

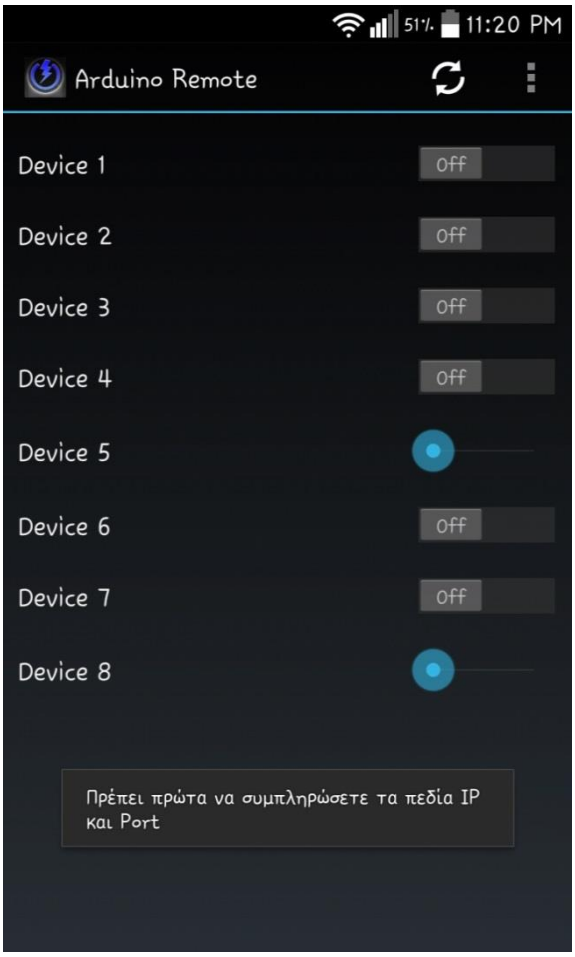
Στο τρέχον σενάριο χρήσης στο **Arduino Remote**, το **Web View** δείχνει το "***Google Plus Profile Web Badge***" [46] του **Developer,** όπως φαίνεται στην παραπάνω εικόνα. Από εκεί ο χρήστης μπορεί να ακολουθήσει τον Developer στο προσωπικό του προφίλ στο κοινωνικό δίκτυο "***Google Plus***" και να επικοινωνήσει μαζί του πατώντας το κουμπί "**Παρακολούθηση**". Το αρχείο με την **JavaScript** που περιέχει τον κατάλληλο κώδικα για την προβολή του **Badge**, φιλοξενείται σε προσωπικό χώρο στο **Cloud**. Για οποιαδήποτε αλλαγή ή προσθήκη, αρκεί να γίνει επεξεργασία του συγκεκριμένου ***.html** αρχείου με τις επιθυμητές διορθώσεις και αυτές θα φανούν αμέσως μέσα στο **Web View,** την επόμενη φορά που θα το φορτώσει ο χρήστης ανοίγοντας το "**About**" Interface.

Ακριβώς κάτω από το **Web View** υπάρχει ένα "***Text View***" το οποίο περιέχει ένα "**String**". Το κείμενο που περιέχει το Text View αναγράφει το εξής: "***Arduino Remote™ – All Rights Reserved by Rodolakis Konstantinos***".

8.3.5 Σενάρια Χρήσης, Toast Messages και Εναπομείνασες Λειτουργίες

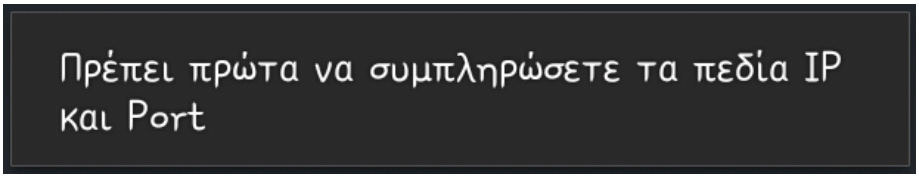
Αφού αναλύθηκαν διεξοδικά τα **Interfaces** που υπάρχουν στο **Arduino Remote**, ας περάσουμε να δούμε κάποιες λειτουργίες που κάνουν την εμφάνισή τους σε συγκεκριμένα σενάρια χρήσης.

Τι γίνεται λοιπόν όταν ο χρήστης μπαίνει στην εφαρμογή και δεν υπάρχουν τα στοιχεία του Arduino Server αποθηκευμένα; Αυτό είναι ένα σενάριο το οποίο θα συμβεί σίγουρα την πρώτη φορά που θα χρησιμοποιήσει ο χρήστης την εφαρμογή. Αφού γίνει εκκίνηση της εφαρμογής από τον χρήστη και βρεθεί στο Κεντρικό Interface της εφαρμογής, η εφαρμογή ελέγχει στο παρασκήνιο αν **υπάρχουν τα στοιχεία του Server αποθηκευμένα**. Αν δεν υπάρχουν αποθηκευμένα, ειδοποιεί τον χρήστη εμφανίζοντάς του ένα "***Toast***" [47] μήνυμα που τον ενημερώνει αναφέροντάς του το εξής: "**Πρέπει πρώτα να συμπληρώσετε τα πεδία IP και Por**t", όπως φαίνεται στην παρακάτω εικόνα.



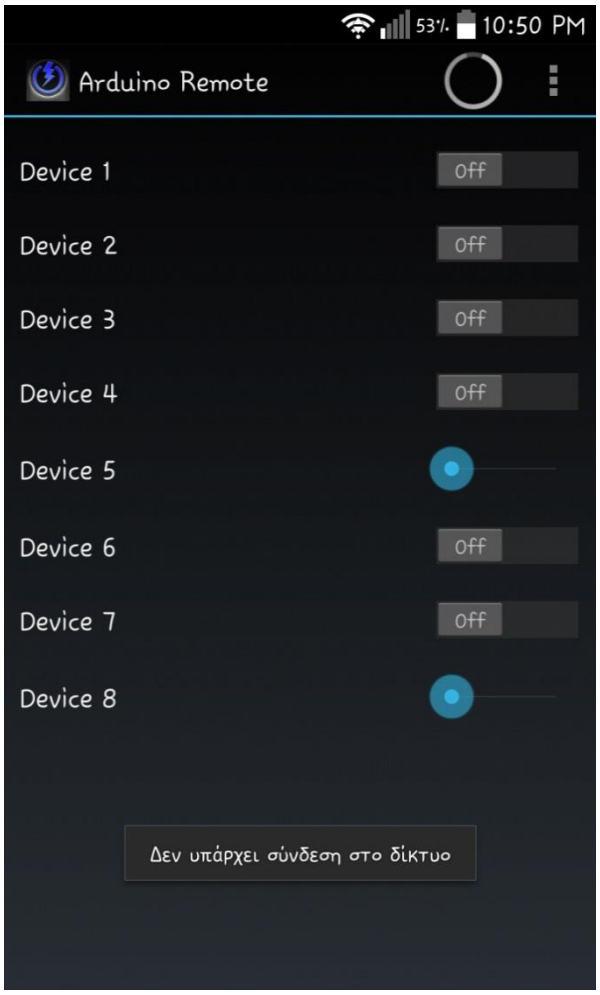
Εικόνα 91: Toast Message Ενημέρωσης του Χρήστη στην Android Εφαρμογή

Τα "**Toasts**" προσφέρουν πληροφορία με έναν ελαφρύ τρόπο. Εμφανίζουν κείμενο μέσα σε ένα μικρό διάλογο, ο οποίος έπειτα από λίγα δευτερόλεπτα κρύβεται αυτόματα χωρίς να χρειάζεται αλληλεπίδραση από τη μεριά του χρήστη.



Εικόνα 92: Toast Message

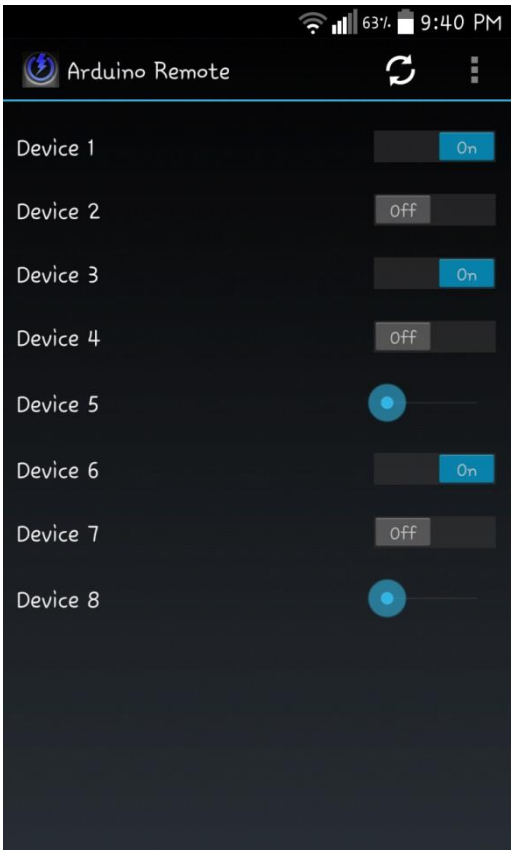
Ένα άλλο πιθανό σενάριο χρήσης είναι να υπάρχουν συμπληρωμένα τα στοιχεία του Server, αλλά να **μην υπάρχει σύνδεση στο δίκτυο**. Τότε βλέπουμε ότι κουμπί "**Refresh**" μετατρέπεται σε "**Activity Circle**" όσο η εφαρμογή προσπαθεί να συνδεθεί στο δίκτυο. Όταν επέλθει το χρονικό όριο προσπάθειας σύνδεσης που έχει τεθεί, τότε εμφανίζεται σε "**Toast**" το μήνυμα "**Δεν υπάρχει σύνδεση στο δίκτυο**".



Εικόνα 93: Toast Message Ενημέρωσης για Έλλειψη Δικτύου στην Android Εφαρμογή

Σε ένα τρίτο σενάριο χρήσης, ο χρήστης θα είχε συμπληρώσει τα πεδία **IP Address** και **Port** και θα υπήρχε πρόσβαση στο **Internet**. Όμως εάν ο Server δεν προσπελάσιμος για οποιονδήποτε λόγο, ή ο χρήστης έχει δώσει λανθασμένα στοιχεία στα πεδία της διεπαφής "**Settings**", τότε η εφαρμογή εμφανίζει το "**Toast**" μήνυμα "**Δεν βρέθηκε ο διακομιστής**", ώστε να ενημερώσει τον χρήστη για την αιτία του προβλήματος επικοινωνίας.

Εάν τελικά είναι σωστά τα στοιχεία που έχουν δοθεί στο "**Settings**" Interface, υπάρχει σύνδεση στο δίκτυο αλλά και σωστή επικοινωνία με τον Arduino Server, τότε μόλις ο χρήστης ξεκινήσει την εφαρμογή αλλά και εάν πατήσει το κουμπί το "**Refresh**" (εφόσον βρισκόταν ήδη μέσα στην εφαρμογή), μια διεργασία στο παρασκήνιο θα ανατρέξει στο **URL** του **Arduino Server**, θα κάνει **Parsing** [48] το κείμενο που αντιπροσωπεύει τα States των Pins του Arduino και θα θέσει τους διακόπτες στο Κεντρικό Interface "**On**" ή "**Off**", ανάλογα με το εκάστοτε **State**. Επομένως αν για παράδειγμα ήταν τα Pins 2, 4 και 7 ενεργά (οι συσκευές 1, 3 και 6 δηλαδή), τότε η εφαρμογή θα έθετε τα αντίστοιχα **Switches** στην θέση "**On**", ενώ τα υπόλοιπα θα τα άφηνε στην θέση "**Off**" όπως φαίνεται στο παρακάτω **Screenshot**.



Εικόνα 94: Αυτόματη Απόδοση Κατάστασης στους Διακόπτες στην Android Εφαρμογή

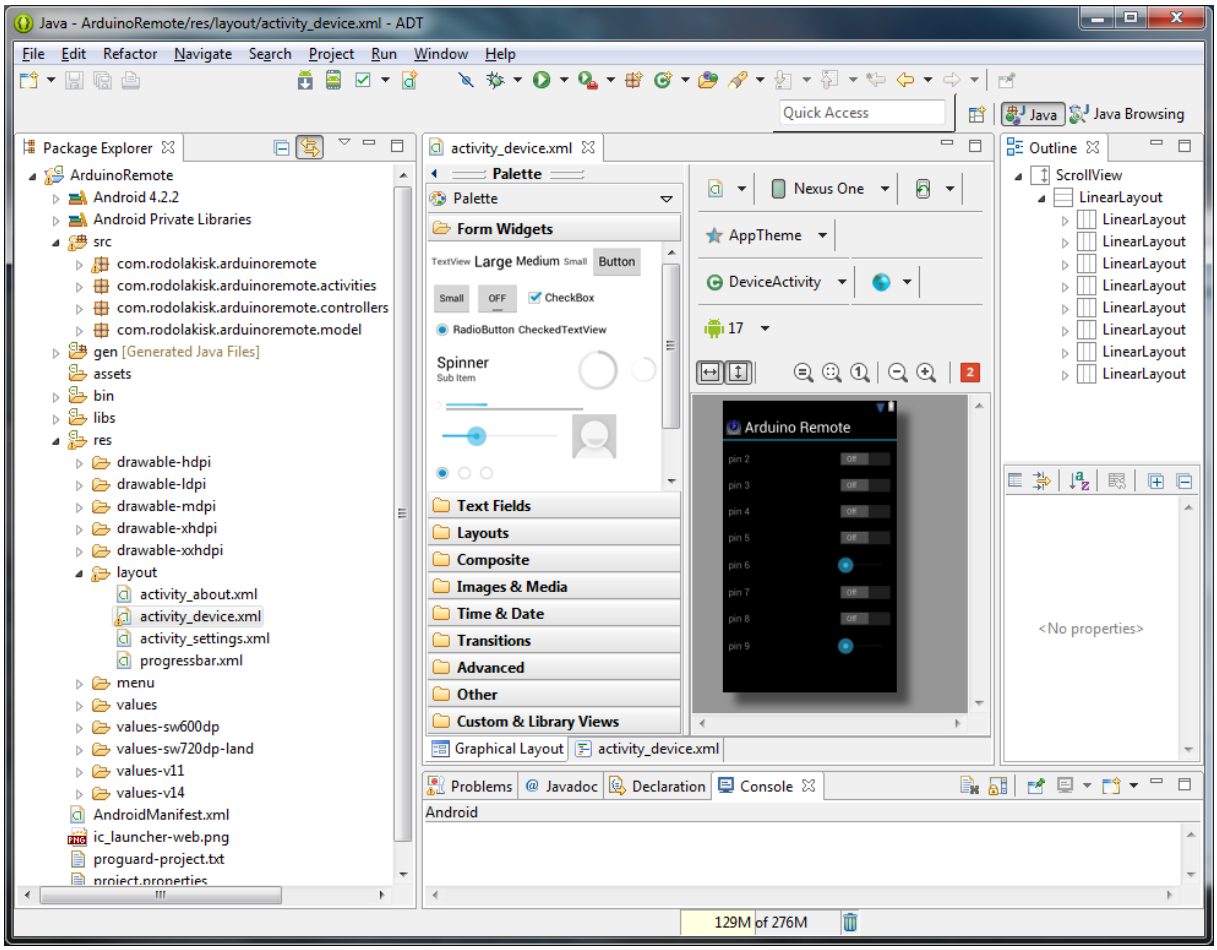
Τέλος, αξίζει να σημειωθεί ότι μόνο στην περίπτωση που υπάρχει επικοινωνία της εφαρμογής με τον Server, όταν ο χρήστης αλληλεπιδρά με κάποιο από τα **Interactive Sliders**, δίπλα σε αυτό εμφανίζεται η τρέχουσα τιμή του (με εύρος από **0 έως 255**) με τη μορφή ενός **Text View**. Μόλις σταματήσει την αλληλεπίδραση ο χρήστης, η εφαρμογή στέλνει την εντολή για αλλαγή της αναλογικής τιμής (αν υπήρξε) στο αντίστοιχο Pin και η τρέχουσα τιμή εξαφανίζεται δίπλα από τον **Slider**.

Εικόνα 95: Η Αναλογική Τιμή του Interactive Slider Κατά τη Διάρκεια της Αλληλεπίδρασης

8.4 Προβολή Τμημάτων Κώδικα Υπεύθυνου για τα Visual Elements

Ας περάσουμε να δούμε κάποια βασικά κομμάτια από τον κώδικα που γράφτηκε στο Eclipse ο οποίος είναι υπεύθυνος για την λειτουργία αλλά και την κατασκευή του παραπάνω Interface.

Έχοντας πλέον πλήρως ολοκληρωμένο το Project, βλέπουμε στην παρακάτω Εικόνα μια πλήρη εποπτεία των **Packages** καθώς και των ***.xml** των **Layouts** του κάθε Interface που φτιάχτηκε, ενώ στο κέντρο βλέπουμε το περιβάλλον σχεδίασης του Κεντρικού Interface.



Εικόνα 96: Eclipse Overview με Ολοκληρωμένο το Project της Android Εφαρμογής

Στον κώδικα που ακολουθεί, βλέπουμε την διάταξη του **activity_device.xml**, το οποίο είναι υπεύθυνο για το πως φαίνονται τα **Elements** στο Κεντρικό Interface της εφαρμογής, των οποίων η ανάλυση για το πώς στοιχίζονται στον χώρο και μεταξύ τους, έγινε στο **Κεφάλαιο 8.3.1**. Για λόγους χωροταξικούς, θα δειχθεί ο κώδικας της διάταξης μέχρι και το πρώτο "**Switch**", εφόσον από εκεί και έπειτα η στοίχιση επαναλαμβάνεται για τα υπόλοιπα στοιχεία της διεπαφής. Η στοίχιση των Interfaces "**Rename**", "**Settings**" και "**About**", των οποίων η ανάλυση έγινε παραπάνω, δεν παρουσιάζει ουσιαστικές διαφορές στον κώδικα των αντίστοιχων ***.xml** αρχείων που τους αναλογούν.

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:orientation="vertical"
        android:padding="10dp" >

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:orientation="horizontal" >

            <EditText
                android:id="@+id/et1"
                style="?android:attr/textViewStyle"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:layout_weight="0.5"
                android:background="@null"
                android:hint="@string/dev1hint"
                android:inputType="text"
                android:textAppearance="?android:attr/textAppearanceMedium" />

            <Switch
                android:id="@+id/switch1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="right"
                android:layout_marginLeft="10dp"
                android:textOff="@string/off"
                android:textOn="@string/on" />
        </LinearLayout>
```

Εικόνα 97: Activity_Device.xml Layout του Κεντρικού Interface στο Arduino Remote

Στον κώδικα που ακολουθεί βλέπουμε με ποιον τρόπο εμφανίζονται τα "**Toasts**" όταν η εφαρμογή κάνει αυτόματα **Parsing** το κείμενο στο URL που αντιστοιχεί ο Server. Με αντίστοιχο τρόπο εμφανίζονται όταν ο χρήστης πατάει το "**Refresh**" Button. Πιο πριν βρίσκονται οι συνθήκες βάσει των οποίων δημιουργούνται τα **Toasts**.

```
@Override
protected void onPostExecute(String result) {
    super.onPostExecute(result);

    if (result != null && view != null) {

        if (result.equals("-1")) {
            Toast.makeText(view.getApplicationContext(),
            "Πρέπει πρώτα να συμπληρώσετε τα πεδία IP και Port", Toast.LENGTH_LONG).show();
        } else if (result.equals("-2")) {
            Toast.makeText(view.getApplicationContext(),
            "Δεν υπάρχει σύνδεση στο δίκτυο", Toast.LENGTH_LONG).show();
        } else if (result.equals("-3")) {
            Toast.makeText(view.getApplicationContext(),
            "Ελέξτε την IP ή το DNS", Toast.LENGTH_LONG).show();
        } else if (result.equals("-4")) {
            Toast.makeText(view.getApplicationContext(),
            "Δεν βρέθηκε ο διακομιστής", Toast.LENGTH_LONG).show();
        } else {
            if (result.equals("-5")) {
                Toast.makeText(view.getApplicationContext(),
                "Η κατάσταση του διακόπτη δεν ανανεώθηκε επιτυχώς", Toast.LENGTH_LONG).show();
            }
        }
    }
}
```

Εικόνα 98: Κομμάτι Κώδικα από το UpdatePinStateTask.java Υπεύθυνο για τα Toasts

Κάθε Android Εφαρμογή πρέπει να έχει ένα ***AndroidManifest.xml*** [49] αρχείο (με ακριβώς αυτό το όνομα) στον **Root** φάκελό της. Το αρχείο αυτό παρουσιάζει τις απαραίτητες πληροφορίες σχετικά με την εφαρμογή στο λειτουργικό σύστημα **Android**. Πληροφορίες που το σύστημα πρέπει να έχει οπωσδήποτε πριν μπορέσει να τρέξει οποιοδήποτε κώδικα. Μεταξύ άλλων το αρχείο αυτό κάνει τα εξής:

- Ονομάζει το **Java Package** για την εφαρμογή. Το όνομα αυτό χρησιμεύει ως μια **μοναδική ταυτότητα** για την εφαρμογή.

- Περιγράφει τα **Components** της εφαρμογής. Τα Activities, τα Services, τους Broadcast Receivers και τους Content Provides από τα οποία αποτελείται η εφαρμογή. Δίνει ονόματα στις κλάσεις που ενσωματώνουν κάθε ένα από αυτά τα Components και εκδίδει τις ικανότητές τους. Αυτές οι δηλώσεις επιτρέπει στο Λειτουργικό Σύστημα να γνωρίζει ποια είναι αυτά τα Components και υπό ποιες προϋποθέσεις θα ενεργοποιηθούν.

- Καθορίζει ποιες **διεργασίες** θα φιλοξενήσουν τα Components της εφαρμογής

- Δηλώνει ποια **δικαιώματα** πρέπει να έχει μια εφαρμογή για να μπορέσει να έχει πρόσβαση σε συγκεκριμένα κομμάτια του API και για να μπορέσει να αλληλεπιδράσει με άλλες εφαρμογές.

- Επίσης δηλώνει ποια **δικαιώματα** πρέπει να έχουν οι άλλες εφαρμογές για να μπορέσουν να αλληλεπιδράσουν με τα Components της ίδιας της εφαρμογής.

- Κάνει μια λίστα από τις **Instrumentation κλάσεις** που προσφέρουν το προφίλ και άλλες πληροφορίες καθώς η εφαρμογή τρέχει.

- Καθορίζει το **ελάχιστο Level** του **Android API** που απαιτεί η εφαρμογή.
- Κάνει μια λίστα από τις **βιβλιοθήκες** που χρειάζεται η εφαρμογή για να τρέξει.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.rodolakisk.arduinoremote"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="17" />

    <!-- Internet Use Permissions -->
    <uses-permission android:name="android.permission.INTERNET" />
    <!-- Network State Permissions -->
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />

    <application
        android:name="com.rodolakisk.arduinoremote.ArduinoRemoteApplication"
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.rodolakisk.arduinoremote.activities.DeviceActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.rodolakisk.arduinoremote.activities.SettingsActivity"
            android:label="@string/title_activity_settings" >
        </activity>
        <activity
            android:name="com.rodolakisk.arduinoremote.activities.AboutActivity"
            android:label="@string/title_activity_about" >
        </activity>
    </application>

</manifest>
```

Εικόνα 99: AndroidManifest.xml του Arduino Remote

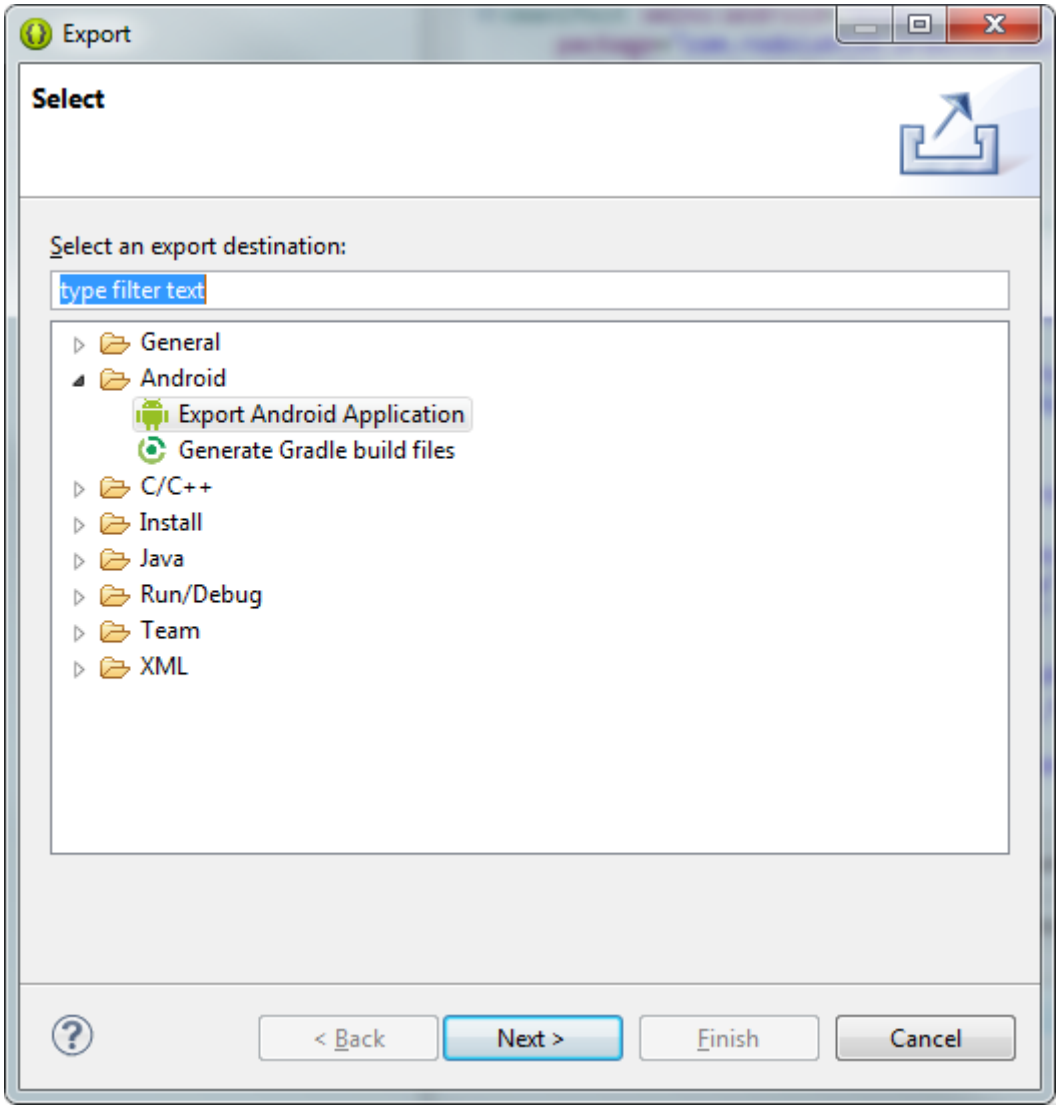
Στην παραπάνω Εικόνα βλέπουμε το **AndroidManisfest.xml** του **Arduino Remote**. Διακρίνουμε στην κορυφή το όνομα του **Package**, το οποίο είναι "**com.rodolakisk.arduinoremote**". Στην συνέχεια βλέπουμε το ελάχιστο επίπεδο του API που απαιτεί η εφαρμογή, το οποίο είναι το "**14**", ενώ αυτό βάσει του οποίου έχει γίνει Compile είναι το "**17**". Ακριβώς από κάτω βλέπουμε ποια

Permissions δηλώνει η εφαρμογή στο σύστημα ότι χρειάζεται για να τρέξει. Στην συνέχεια υπάρχουν πληροφορίες όπως το όνομα του **Icon** που έχει η εφαρμογή στον **Launcher**, καθώς και τα τέσσερα Activities με τα ονόματά τους.

8.5 Δημιουργία του APK

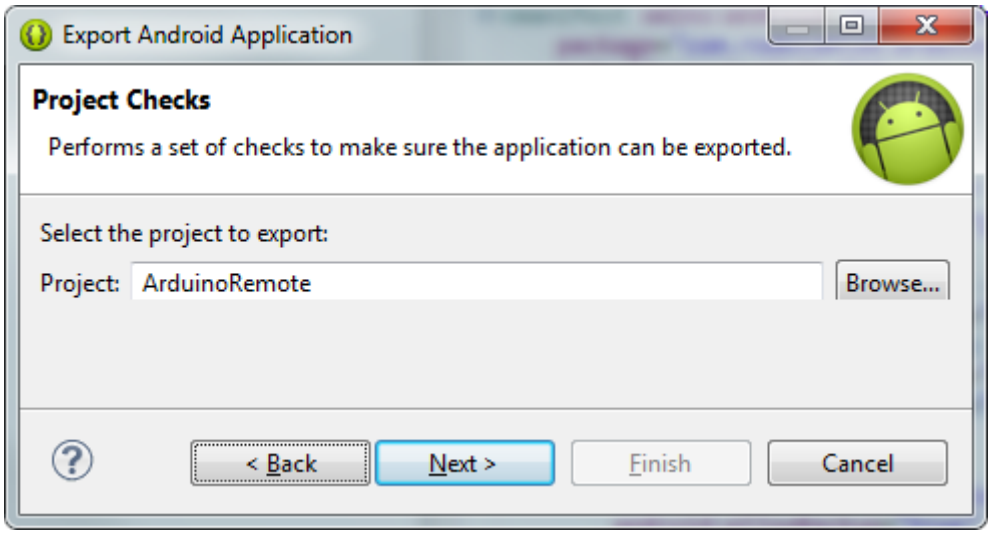
Αφού έχουμε τεστάρει την εφαρμογή μας χρησιμοποιώντας ως συσκευή **debugging** το Optimus G και ξέρουμε ότι λειτουργεί , είμαστε έτοιμοι να εξάγουμε το τελικό *.**apk**, δηλαδή το εκτελέσιμο αρχείο του **Arduino Remote**. Για να μπορεί να εγκατασταθεί το αρχείο αυτό σε οποιαδήποτε συσκευή, πρέπει να έχει μια **μοναδική ψηφιακή υπογραφή** [50].

Έχοντας επιλεγμένο το "**ArduinoRemote**" Project, επιλέγουμε **File → Export → Export Android Application** και στην συνέχεια πατάμε "**Next**".



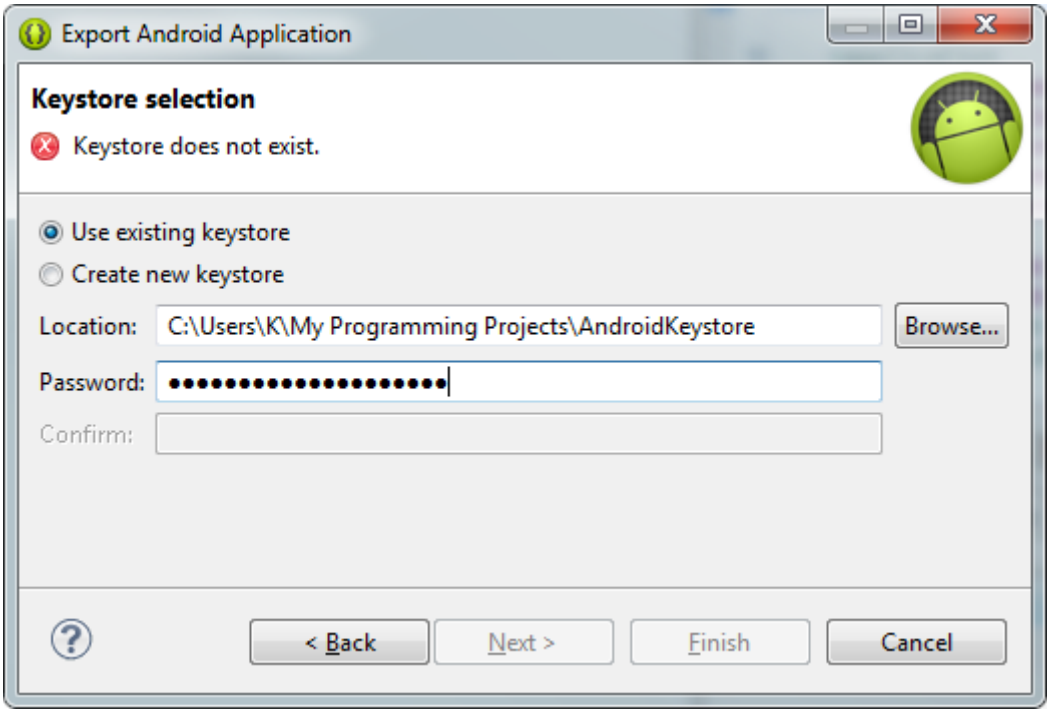
Εικόνα 100: Επιλογή Τύπου Εφαρμογής

Κατόπιν διαλέγουμε το Project το οποίο θέλουμε να υπογράψουμε ψηφιακά και στη συνέχεια πατάμε "**Next**".



Εικόνα 101: Επιλογή Project Προς Εξαγωγή

Στο επόμενο παράθυρο διαλέγουμε την τοποθεσία στην οποία βρίσκεται το **Keystore** καθώς επίσης και τον κωδικό με τον οποίο έχει δημιουργηθεί και πατάμε "**Next**".



Εικόνα 102: Επιλογή του Keystore

Στο επόμενο παράθυρο δημιουργούμε ένα μοναδικό κλειδί για την εφαρμογή, χρησιμοποιώντας ένα μοναδικό κωδικό και τα προσωπικά μας στοιχεία όπως Όνομα, Επώνυμο, Χώρα. Επίσης καθορίζουμε τι **διάρκεια ισχύος** θα έχει το κλειδί. Στη συνέχεια πατάμε "**Next**".

Στο τελευταίο παράθυρο επιλέγουμε την διαδρομή στην οποία θα γίνει **Export** το ***.apk**. Σε αυτή την διαδρομή στον σκληρό δίσκο, μπορούμε να βρούμε το τελικό αρχείο και να το διανείμουμε ως έχει σε όποιον επιθυμούμε, μέσω του **Play Store** της Google, ή με οποιοδήποτε άλλο μέσο.

9. Αποτελέσματα – Μελλοντική Εργασία και Επεκτάσεις

Έχοντας ολοκληρώσει με επιτυχία την παρούσα πτυχιακή, ας δούμε εκ του συνόλου τα αποτελέσματα, τα συμπεράσματα, καθώς και τις μελλοντικές εργασίες ή και επεκτάσεις που μπορούν να γίνουν βασιζόμενες σε αυτήν.

9.1 Αποτελέσματα

Κρίνοντας από το τελικό προϊόν, βλέπουμε ότι επιτεύχθηκε ο αρχικός στόχος που είχε τεθεί. Καταφέραμε να έχουμε ένα σύστημα ελέγχου χαμηλού κόστους και χαμηλής ενεργειακής κατανάλωσης, το οποίο ελέγχεται από οποιοδήποτε σημείο στον πλανήτη μέσω μιας Android Εφαρμογής βασισμένης στα Holo Guidelines, η οποία τρέχει απρόσκοπτα σε πληθώρα Android Συσκευών. Η χρήση της εφαρμογής από τον χρήστη είναι διαισθητική και δεν απαιτεί ιδιαίτερες γνώσεις, κάτι που την κάνει προσιτή σε όλους.

9.2 Οφέλη – Σκοπός - Συμπεράσματα

Η παρούσα πτυχιακή μπορεί να ωφελήσει οποιονδήποτε έχει ανάγκη από ένα σύστημα ελέγχου ενός σπιτιού, ενός εξοχικού, ή μεμονωμένων συσκευών από απόσταση.

Για παράδειγμα, θα μπορούσε κάποιος να το εγκαταστήσει στο εξοχικό του, συνδέοντας πάνω σε αυτό κάποιες λάμπες, ή ακόμα και ένα ραδιόφωνο, τα οποία θα άνοιγε κάποιες στιγμές της ημέρας για να δηλώσει την παρουσία προσώπου σε πιθανούς κλέφτες, ώστε να αποτρέψει κάποια εισβολή στον χώρο. Θα μπορούσε ακόμα να γίνει χρήση του σε πίνακες σπιτιών, σε συνεργασία με το κατάλληλο υλικό, για πλήρη έλεγχο ενός σπιτιού (θερμοσίφωνα, φωτισμό, θέρμανση). Ακόμα όμως και αν δεν επιθυμείται πλήρης έλεγχος όλου του σπιτιού, μπορεί να δράσει μεμονωμένα για συσκευές ως διακόπτης. Θα μπορούσε κάποιος να το εγκαταστήσει στην καφετιέρα του, ώστε κάθε πρωί που ξυπνάει, να ανοίγει το εκάστοτε Switch από την εφαρμογή ώστε ο καφές να έχει ετοιμαστεί μέχρι την στιγμή που θα είναι έτοιμος να πάει στην κουζίνα.

Όπως βλέπουμε τα σενάρια περιορίζονται μόνο από την φαντασία του καθενός και οι μελλοντικές επεκτάσεις της εφαρμογής, θα μπορούσαν να διευρύνουν σε μεγάλο βαθμό τα πολυπληθή σενάρια χρήσης.

Πέραν από τα παραπάνω οφέλη για τρίτους, προσωπικός μου στόχος μου ήταν να ασχοληθώ με τον τομέα των αυτοματισμών μέσα από την πτυχιακή αυτή, λόγω του έντονου ενδιαφέροντός μου για μάθηση πάνω στους αυτοματισμούς αλλά και στις κινητές συσκευές με λειτουργικό σύστημα Android.

9.3 Μελλοντική Εργασία και Επεκτάσεις

Το σύστημα που δημιουργήθηκε μαζί με την Android Εφαρμογή, μπορεί να γίνει η βάση για την δημιουργία κάτι περισσότερο πολύπλοκου και ακόμα περισσότερο χρήσιμου.

Θα μπορούσαν να ενσωματωθούν στον Arduino Server διάφοροι **Sensors** όπως για θερμοκρασία, υγρασία ή ακόμα και για υπεριώδη ακτινοβολία (αν αυτός βρίσκεται δίπλα σε παράθυρο). Αυτοί οι αισθητήρες θα μπορούσαν να τυπώνουν τις τιμές τους μέσα στην εφαρμογή και να ενημερώνουν τον χρήστη μέσα από την Android συσκευή του ή να εμφανίζουν **Notifications** σε αυτήν, υπενθυμίζοντάς του να βάλει αντηλιακό ή να πάρει μπουφάν.

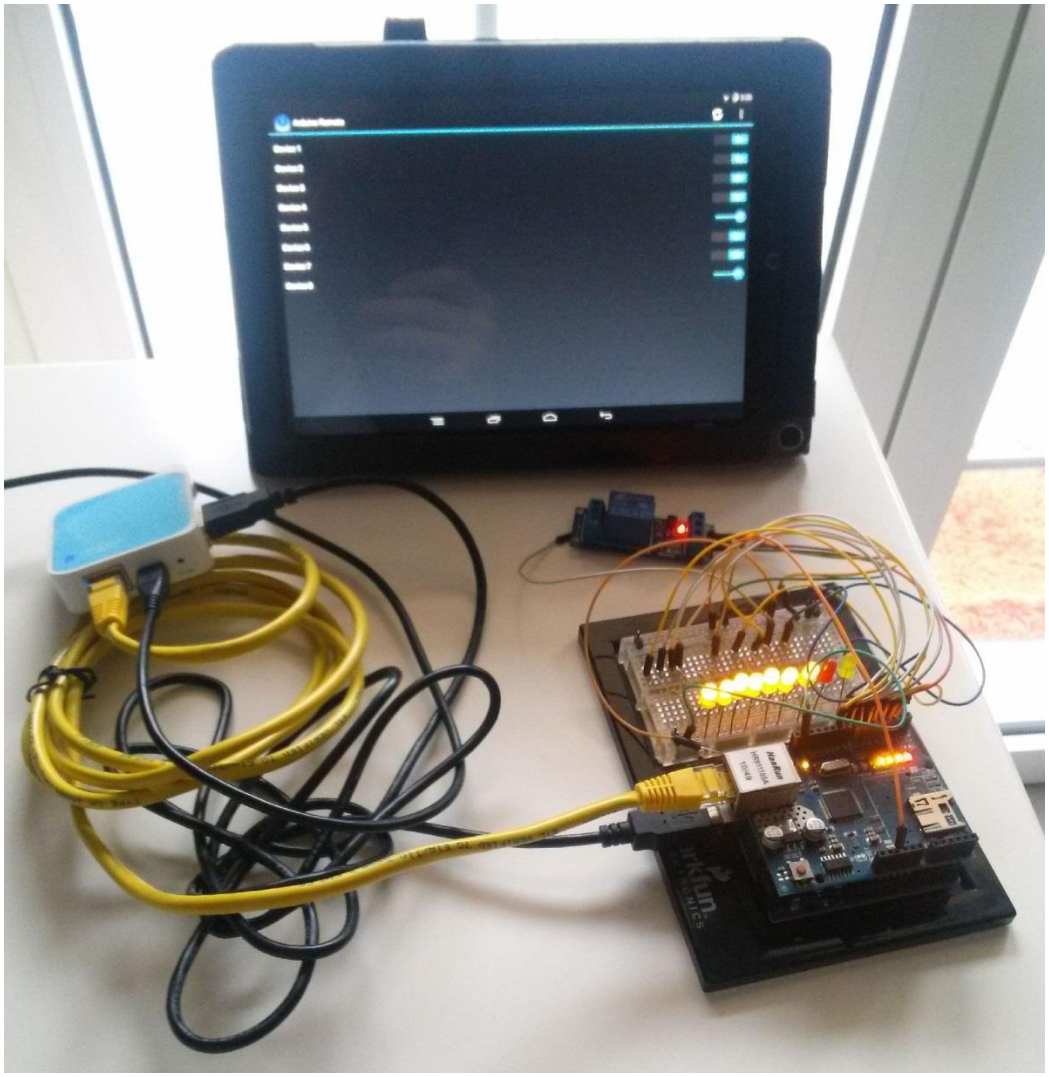
Θα μπορούσε ακόμα να γίνει ενσωμάτωση στον κώδικα για χρήση σερβοκινητήρων με ότι αυτό συνεπάγεται για την χρήση τους από την εφαρμογή Android, όπως για παράδειγμα πλήρης έλεγχος του ύψους μιας γκαραζόπορτας ή των περσίδων ενός παραθύρου.

Επιπλέον θα μπορούσε να γίνει ενσωμάτωση στην εφαρμογή διαφόρων συνθηκών ελέγχου των εξόδων του Server. Κατ' αρχάς, θα μπορούσε να γίνει ενσωμάτωση ενός **Schedule Interface**, βάση του οποίου οι έξοδοι του Server που θα επίλεγε ο χρήστης, θα άνοιγαν και θα έκλειναν συγκεκριμένες ώρες και ημέρες της εβδομάδας.

Επιπρόσθετα θα μπορούσε να ενσωματωθεί μια λειτουργία βάσει της οποίας ο χρήστης θα ρυθμίζει την κατάσταση κάποιων εξόδων του Server, ανάλογα με την **τοποθεσία** στην οποία βρίσκεται η Android συσκευή του (και κατ' επέκταση ο ίδιος). Για παράδειγμα όταν φτάνει κοντά στο σπίτι του, η εφαρμογή θα στέλνει σήμα αυτόματα να ανοίξει ο θερμοσίφωνας ή το φως της σκάλας. Αντίστοιχες λειτουργίες θα μπορούσαν να πραγματοποιηθούν και για την ανίχνευση της MAC Address του **Wi-Fi Access Point** του σπιτιού του εκάστοτε χρήστη από το κινητό του.

Οι φωνητικές εντολές και η ενσωμάτωση τους μέσα στο **Google Now** στα σύγχρονα smartphones μέσω **Accessibility Service**, θα έδινε στην εφαρμογή και στους χρήστες έξτρα λειτουργικότητα και ευκολία, λύνοντας τα χέρια σε πολλές περιπτώσεις.

Όπως βλέπουμε υπάρχουν πάρα πολλά πράγματα που μπορούν να γίνουν ακόμα. Ευελπιστώ να μπορέσω να ενσωματώσω εν καιρώ κάποιες από αυτές τις λειτουργίες στον **Arduino Server**, αλλά και στην **Android Εφαρμογή**.



Εικόνα 103: Το Τελικό Σύστημα και η Android Εφαρμογή

Βιβλιογραφία

[1] «http://www.gantt.com/,» [Ηλεκτρονικό].

[2] «No-IP,» [Ηλεκτρονικό]. Available: http://www.noip.com/.

[3] «Arduino.cc,» [Ηλεκτρονικό]. Available: http://arduino.cc/en/Main/arduinoBoardUno.

[4] «arduino.cc,» [Ηλεκτρονικό]. Available: http://arduino.cc/en/Main/ArduinoEthernetShield.

[5] «Open WRT,» [Ηλεκτρονικό]. Available: http://wiki.openwrt.org/toh/tp-link/tl-wr703n.

[6] «WRT54GWiki,» [Ηλεκτρονικό]. Available: http://en.wikipedia.org/wiki/Linksys_WRT54G_series#WRT54G.

[7] «TG585Wiki,» [Ηλεκτρονικό]. Available: http://wiki.openwrt.org/toh/thomson/tg585.

[8] «Static Relay Wiki,» [Ηλεκτρονικό]. Available: http://en.wikipedia.org/wiki/Static_relay.

[9] S. Kasap, Principles of Electronic Materials and Devices, Second Edition, New York: Mc Graw-Hill, 2002.

[10] «lg.com,» [Ηλεκτρονικό]. Available: http://www.lg.com/hk_en/mobile-phones/lg-E975.

[11] «arduino.cc,» [Ηλεκτρονικό]. Available: http://arduino.cc/en/Main/Software.

[12] «Techspot,» 30 June 2014. [Ηλεκτρονικό]. Available: http://www.techspot.com/news/57228-google-shows-off-new-version-of-android-announces-1-billion-active-monthly-users.html.

[13] «Android Developers,» [Ηλεκτρονικό]. Available: http://developer.android.com/guide/developing/tools/index.html.

[14] [Ηλεκτρονικό]. Available: http://en.wikipedia.org/wiki/Eclipse_(software).

[15] [Ηλεκτρονικό]. Available: http://developer.android.com/tools/help/adt.html.

[16] [Ηλεκτρονικό]. Available: http://www.webopedia.com/TERM/D/dynamic_DNS.html.

[17] «http://en.wikipedia.org/wiki/MAC_address,» [Ηλεκτρονικό].

[18] «DOD Standard Internet Protocol,» January 1980. [Ηλεκτρονικό]. Available: http://tools.ietf.org/html/rfc760.

[19] M. Muuss, 8 September 2010 2010. [Ηλεκτρονικό]. Available: http://www.webcitation.org/5saCKBpgH.

[20] [Ηλεκτρονικό]. Available: http://arduino.cc/en/Reference/SPI.

[21] [Ηλεκτρονικό]. Available: http://arduino.cc/en/pmwiki.php?n=Reference/Ethernet.

[22] [Ηλεκτρονικό]. Available: http://arduino.cc/en/Reference/loop.

[23] [Ηλεκτρονικό]. Available: http://arduino.cc/en/Reference/StringIndexOf.

[24] [Ηλεκτρονικό]. Available: http://arduino.cc/en/Reference/StringConstructor.

[25] [Ηλεκτρονικό]. Available: http://arduino.cc/en/Reference/ClientPrint.

[26] [Ηλεκτρονικό]. Available: http://arduino.cc/en/Reference/analogWrite.

[27] [Ηλεκτρονικό]. Available: http://en.wikipedia.org/wiki/Pulse-width_modulation.

[28] [Ηλεκτρονικό]. Available: http://en.wikipedia.org/wiki/Wireless_Internet_service_provider.

[29] «IEEE 802.11i-2004: Amendment 6: Medium Access Control (MAC) Security Enhancements,» *IEEE Standards,* αρ. IEEE Standards, 2004.

[30] D. C. Plummer, «RFC 826, An Ethernet Address Resolution Protocol -- or -- Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware,» Network Working Group, November 1982. [Ηλεκτρονικό]. Available: http://tools.ietf.org/html/rfc826.

[31] «Definition of Port Forwarding,» *PC Magazine,* 2008.

[32] «Android Developers,» [Ηλεκτρονικό]. Available: http://developer.android.com/sdk/installing.

[33] «Android Developers - Design - Actionbar,» [Ηλεκτρονικό]. Available: http://developer.android.com/design/patterns/actionbar.html.

[34] «Android Developers - Design - Buttons,» [Ηλεκτρονικό]. Available: http://developer.android.com/design/building-blocks/buttons.html.

[35] «Android Developers - Design - Progress & Activity,» [Ηλεκτρονικό]. Available: http://developer.android.com/design/building-blocks/progress.html.

[36] «Android Developers - Guide - Layout - Linear,» [Ηλεκτρονικό]. Available: http://developer.android.com/guide/topics/ui/layout/linear.html.

[37] «Android Developers - Design - Text Fields,» [Ηλεκτρονικό]. Available: http://developer.android.com/design/building-blocks/text-fields.html.

[38] «Android Developers - Design - Scrolling,» [Ηλεκτρονικό]. Available: http://developer.android.com/design/building-blocks/scrolling.html.

[39] «Android Developers - Design - Switches,» [Ηλεκτρονικό]. Available: http://developer.android.com/design/building-blocks/switches.html.

[40] «Android Developers - Design - Seek Bars,» [Ηλεκτρονικό]. Available:

http://developer.android.com/design/building-blocks/seek-bars.html.

[41] «Android Developers - Design - Navigation,» [Ηλεκτρονικό]. Available: http://developer.android.com/design/patterns/navigation.html.

[42] «Android Developers - Reference - Hint,» [Ηλεκτρονικό]. Available: http://developer.android.com/reference/android/R.attr.html#hint.

[43] «Android Developers - Design - CAB,» [Ηλεκτρονικό]. Available: http://developer.android.com/design/patterns/actionbar.html#contextual.

[44] «Android Developers - Guide - Layout - Relative,» [Ηλεκτρονικό]. Available: http://developer.android.com/guide/topics/ui/layout/relative.html.

[45] «Android Developers - Guide - Webapps - Webview,» [Ηλεκτρονικό]. Available: http://developer.android.com/guide/webapps/webview.html.

[46] «Google Developers - Web Badge,» [Ηλεκτρονικό]. Available: https://developers.google.com/+/web/badge/.

[47] «Android Developers - Design - Dialogs,» [Ηλεκτρονικό]. Available: http://developer.android.com/design/building-blocks/dialogs.html.

[48] «Parsing,» [Ηλεκτρονικό]. Available: http://en.wikipedia.org/wiki/Parsing.

[49] «Android Developers - Guide - Manifest,» [Ηλεκτρονικό]. Available: http://developer.android.com/guide/topics/manifest/manifest-intro.html.

[50] «Android Developers - Publishing - App Signing,» [Ηλεκτρονικό]. Available: http://developer.android.com/tools/publishing/app-signing.html.