INTERNATIONAL
HELLENIC
UNIVERSITY

ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

&

ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

“ Απομακρυσμένη λειτουργία Η/Υ μέσω τηλεχειριστηρίου και κινητού τηλεφώνου”

INTERNATIONAL
HELLENIC
UNIVERSITY

Φοιτητής

Σπάνδος Ιωάννης

ΑΜ : 515135

Επιβλέπων καθηγητής

Άγγελος Γιακουμής

Βαθμίδα : Επίκουρος Καθηγητής

Τίτλος Π.Ε : Απομακρυσμένη λειτουργία Η/Υ μέσω τηλεχειριστηρίου και κινητού τηλεφώνου

Κωδικός Π.Ε: 20212

Ονοματεπώνυμο φοιτητή : Σπάνδος Ιωάννης

Ονοματεπώνυμο εισηγητή : Άγγελος Γιακουμής

Ημερομηνία ανάληψης Π.Ε: (25-10-2020)

Ημερομηνία περάτωσης Π.Ε (24-5-2023)

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Η παρούσα πτυχιακή εργασία με τίτλο «Απομακρυσμένη λειτουργία Η/Υ μέσω τηλεχειριστηρίου και κινητού τηλεφώνου», εκπονήθηκε στα πλαίσια της ολοκλήρωσης των προϋποθέσεων, για την λήψη του πτυχίου Ηλεκτρονικού Μηχανικού Α.Τ.Ε.Ι.Θ. Η εργασία χωρίζεται σε τρία μέρη και απασχολεί διαφορετικές τεχνολογίες για την υλοποίηση τους. Το πρώτο μέρος είναι ο κεντρικός κόμβος επικοινωνίας των συσκευών, μία εφαρμογή σε περιβάλλον Windows προγραμματισμένη σε γλώσσα C#. Το δεύτερο και το τρίτο μέρος είναι οι συσκευές χειρισμού, όπου το ένα είναι μία φυσική κατασκευή βασισμένη στο MCU ESP32 προγραμματισμένο σε γλώσσα C/C++ επικοινωνώντας με Bluetooth και το άλλο, μία εφαρμογή σε περιβάλλον Android προγραμματισμένο σε Xamarin.NET επικοινωνώντας μέσω LAN

Πηγή έμπνευσης αυτής της εργασίας είναι η ραγδαία αύξηση έξυπνων συσκευών στα σπίτια μας. Πάρα πολλές συσκευές όπως κινητά , smart TV κ.α. έχουν αντικαταστήσει σε τον Η/Υ σε έναν μεγάλο βαθμό διότι εξειδικεύονται στον να κάνουν ορισμένες λειτουργίες πολύ πιο γρήγορα και άμεσα. Αυτό έχει σαν αποτέλεσμα την αμέληση των Η/Υ που βρίσκονται στα σπίτια μας για πολύ περισσότερα χρόνια όπου είναι ικανοί για πολύ περισσότερα πράγματα από τις σύγχρονες έξυπνες συσκευές. Όπου είναι ανώτεροι σε κάθε επίπεδο τόσο λογισμικού όσο και hardware.

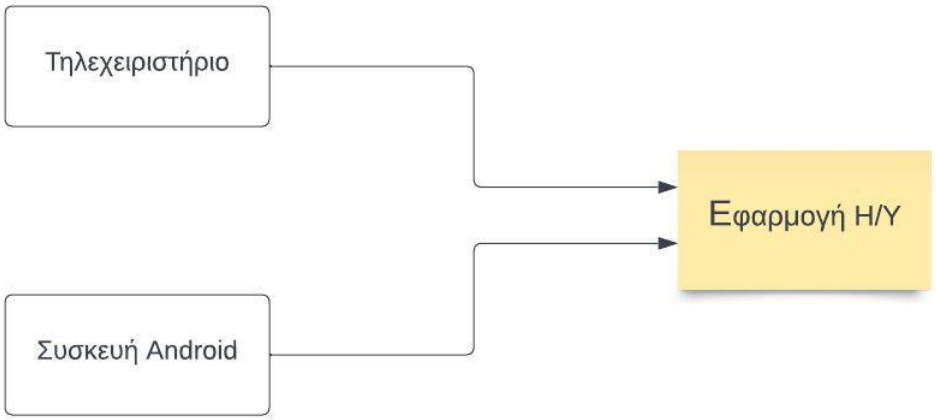


Figure I : Διάγραμμα ροής συστήματος

Περίληψη

Σκοπός της πτυχιακής εργασίας ήταν να γίνει έρευνα, για την απομακρυσμένη λειτουργία ενός Η/Υ με περιβάλλον Microsoft Windows, μέσω ενός παραδοσιακού τρόπου τηλεχειρισμού με φυσική συσκευή και ενός πιο σύγχρονου με τη χρήση μία εφαρμογής έξυπνου τηλεφώνου, με στόχο την απλούστευση ορισμένων λειτουργιών του Η/Υ.

Το μόνο όριο που υπάρχει στις δυνατότητες αυτής της εργασίας είναι το όριο που θέτει ο χρήστης της. Καθώς είναι πλήρως παραμετροποιήσιμη στις προτιμήσεις του χρήση. Με τις κατάλληλες ρυθμίσεις, οποιαδήποτε ενέργεια επιθυμεί ο χρήστης μπορεί να γίνει με το πάτημα ενός πλήκτρου πραγματικού ή/και ψηφιακού. Η επικοινωνία γίνεται σε δύο επίπεδα ταυτόχρονα. Με πρώτο να είναι η επικοινωνία του τηλεχειριστηρίου και του Η/Υ, όπου επιτυγχάνεται με τη τεχνολογία Bluetooth, έως 10 μέτρα καθαρής απόστασης και σε δεύτερο επίπεδο μέσω WiFi τεχνολογίας σε τοπικό δίκτυο από τη Android εφαρμογή προς τον Η/Υ.

Στο Κεφάλαιο 1 γίνεται μία γενική ανασκόπηση στο κομμάτι μερικών από τις πιο συχνές τεχνολογίες ασύρματου χειρισμού που υπάρχουν σήμερα στο εμπόριο, όσο και για το τι είναι ο τηλεχειρισμός και ποιες οι προϋποθέσεις για την ύπαρξη του. Στο Κεφάλαιο 2 αναλύεται η διαφορά των τεχνολογιών Bluetooth και τοπικού δικτύου . Στο 3ο κεφάλαιο αναφέρεται ο τρόπος ανάπτυξης εφαρμογών σε ένα λειτουργικό σύστημα. Το 4ο κεφάλαιο γίνεται μία αναδρομή στους μικροελεγκτές και τη τυπική αρχιτεκτονική τους. Από το 5ο κεφάλαιο ξεκινάει η ανάλυση του τρόπου λειτουργίας της εργασίας ξεκινώντας από τις εφαρμογές που χρησιμοποιούνται στην εργασία και στο κεφάλαιο 6 γίνεται συνέχεια στη λειτουργία του τηλεχειριστηρίου. Στα κεφάλαια 7 και 8 γίνεται ανάλυση του τρόπου κατασκευής της εργασίας και τέλος, στο 9ο κεφάλαιο γίνεται επεξήγηση της τυπικής λειτουργίας όλου του συστήματος.

“ Remote PC operation via remote control and mobile phone”

Spandos Ioannis

Abstract

The purpose of the thesis was to conduct research on the remote operation of a PC with a Microsoft Windows environment, through a traditional remote control method with a physical device and a more modern one using a smart phone application, with the aim of simplifying certain functions of the PC.

The only limit to what this job can do is the limit set by its user**, a**s it is fully customizable to user preferences. With the proper settings, any action the user desires can be done with the push of a real or digital button. Communication takes place on two levels simultaneously. With the first, being the communication of the remote control and the PC, which it is achieved with Bluetooth technology, up to 10 meters of clear distance and on the second level via WiFi technology in a local network from the Android application to the PC.

Chapter 1 provides a general overview of some of the most common wireless control technologies commercially available today, as well as what remote control is and what the conditions for its existence are. Chapter 2 discusses the difference between Bluetooth and LAN technologies. Chapter 3 describes how to develop applications in an operating system. Chapter 4 is a review of microcontrollers and their typical architecture. From the 5th chapter, the analysis of how the work works begins, starting with the applications used in the work, and in chapter 6, the operation of the remote control is continued. In chapters 7 and 8 there is an analysis of how the work is constructed and finally, in chapter 9 there is an explanation of the typical operation of the entire system.

Περιεχόμενα

Κατάλογος εικόνων

Χρήσιμα ακρώνυμα και ορολογίες

LAN	Local Area Network (Περιοχή τοπικού δικτύου)
WLAN	Wireless Local Area Network (Περιοχή Ασύρματου τοπικού δικτύου)
MCU	Micro Controller Unit (Μονάδα Μικροελεγκτή)
Εφαρμογή Κόμβος	Εφαρμογή που τρέχει στο περιβάλλον MicrosoftWindowsκαι διαχειρίζεται τις εισερχόμενες εντολές από τη συσκευή Androidκαι τη φυσική συσκευή
OS	Operating System (Λειτουργικό Σύστημα)
Gestures	Χειρονομίες αφής

Κεφάλαιο 1ο– Τηλεχειρισμός

Με την εξέλιξη της τεχνολογίας να γίνεται όλο και πιο γρήγορη, ο μέσος άνθρωπος χρησιμοποιεί όλο και περισσότερες συσκευές για να εξυπηρετήσει τις ανάγκες του. Μία από τις πρώτες συσκευές που χρησιμοποίησε μαζικά ο πληθυσμός είναι η οικιακή τηλεόραση. Αρχικά για να έχει πρόσβαση κάποιος στις λειτουργίες που προσέφερε η τηλεόραση, έπρεπε να χρησιμοποιήσει τα κουμπιά χειρισμού πάνω στη συσκευή. Αργότερα, σχεδόν τριάντα χρόνια μετά, εφευρέθηκε το πρώτο τηλεχειριστήριο. Από τότε έγινε φανερό σε όλο τον κόσμο ότι εξυπηρετεί καλύτερα ο απομακρυσμένος χειρισμός οποιαδήποτε συσκευής. Έτσι όλο και περισσότερες καθημερινές συσκευές αποκτούν σταδιακά μία μέθοδο απομακρυσμένου χειρισμού.

Τα υπάρχοντα τηλεχειριστήρια υπερύθρων μπορούν να χρησιμοποιηθούν για τον έλεγχο εφαρμογών υπολογιστή. Κάθε εφαρμογή που υποστηρίζει πλήκτρα συντόμευσης μπορεί να ελεγχθεί μέσω τηλεχειριστηρίων υπερύθρων από άλλες οικιακές συσκευές (τηλεόραση, VCR, AC). Χρησιμοποιείται ευρέως με εφαρμογές πολυμέσων για συστήματα οικιακού κινηματογράφου που βασίζονται σε υπολογιστή . Για να λειτουργήσει αυτό, χρειάζεται μια συσκευή που αποκωδικοποιεί τα σήματα δεδομένων του τηλεχειριστηρίου υπερύθρων και μια εφαρμογή υπολογιστή που επικοινωνεί με αυτήν τη συσκευή συνδεδεμένη με υπολογιστή. Μια σύνδεση μπορεί να γίνει μέσω σειριακής θύρας, θύρας USB ή υποδοχής IrDA μητρικής πλακέτας. Τέτοιες συσκευές διατίθενται στο εμπόριο, αλλά μπορούν να κατασκευαστούν στο σπίτι χρησιμοποιώντας μικροελεγκτές χαμηλού κόστους. Το LIRC (Linux IR Remote control) και το WinLIRC (για Windows) είναι πακέτα λογισμικού που αναπτύχθηκαν με σκοπό τον έλεγχο του υπολογιστή χρησιμοποιώντας τηλεχειριστήριο τηλεόρασης και μπορούν επίσης να χρησιμοποιηθούν για τηλεχειριστήριο homebrew με μικρότερη τροποποίηση.

Τα τηλεχειριστήρια χρησιμοποιούνται στη φωτογραφία, ιδίως για τη λήψη λήψεων μεγάλης έκθεσης. Πολλές κάμερες δράσης, όπως οι GoPros, καθώς και οι τυπικές DSLR, συμπεριλαμβανομένης της σειράς Alpha της Sony, ενσωματώνουν συστήματα τηλεχειρισμού που βασίζονται σε Wi-Fi. Αυτά μπορούν συχνά να είναι προσβάσιμα και ακόμη και να ελεγχθούν μέσω κινητών τηλεφώνων και άλλων φορητών συσκευών

Οι κονσόλες βιντεοπαιχνιδιών δεν είχαν χρησιμοποιήσει ασύρματα χειριστήρια μέχρι πρόσφατα, κυρίως λόγω της δυσκολίας που συνεπαγόταν η αναπαραγωγή του παιχνιδιού, ενώ ο πομπός υπερύθρων ήταν στραμμένος στην κονσόλα. Το πρώτο επίσημο ασύρματο χειριστήριο παιχνιδιών που κατασκευάστηκε από έναν κατασκευαστή πρώτου κατασκευαστή ήταν το CX-42 για την Atari 2600. Η σειρά CD-i 400 της Philips ήρθε επίσης με τηλεχειριστήριο, το WaveBird κατασκευάστηκε επίσης για το GameCube. Στην έβδομη γενιά κονσολών παιχνιδιών, τα ασύρματα χειριστήρια έγιναν στάνταρ. Ορισμένα ασύρματα χειριστήρια, όπως αυτά του PlayStation 3 και του Wii, χρησιμοποιούν Bluetooth. Άλλα, όπως το Xbox 360, χρησιμοποιούν ιδιόκτητα ασύρματα πρωτόκολλα.

Είναι σημαντικό να αναφερθεί ότι, για να ενεργοποιηθεί από ασύρματο τηλεχειριστήριο, η ελεγχόμενη συσκευή πρέπει να είναι πάντα εν μέρει ενεργοποιημένη, καταναλώνοντας ισχύ σε αναμονή.

1.1. Αρχή λειτουργίας τηλεχειρισμού

Στην ηλεκτρονική ένα ασύρματο χειριστήριο είναι μία ηλεκτρονική συσκευή που χρησιμοποιείται για τον απομακρυσμένο χειρισμό μιας άλλης συσκευής από μία απόσταση. Συσκευές που μπορεί να έχουν απομακρυσμένο χειρισμό μπορεί να είναι καθημερινές συσκευές όπως η τηλεόραση, το ραδιόφωνο μέχρι και μηχανές βιομηχανικής χρήσης. Τα τηλεχειριστήρια είναι ιδιαίτερα χρήσιμα όταν θέλουμε να χρησιμοποιήσουμε μία συσκευή που είναι πολύ δύσκολο έως απίθανο να πλησιάσουμε. Ένα τηλεχειριστήριο μπορεί να έχει μερική έως και πλήρη δυνατότητα χειρισμού μιας συσκευής και πολλές φορές να παρέχει περισσότερες δυνατότητες χειρισμού της συσκευής από τις δυνατότητες χειρισμού επάνω με στη συσκευή. Αυτό προφανώς εξαρτάται από το κατασκευαστή και σχετίζεται με τον λόγο που προορίζεται η συσκευή. Τα τηλεχειριστήρια λειτουργούν καλύτερα όταν είναι μέσα στη προτεινόμενη εμβέλεια που προτείνει ο κατασκευαστής. Πλέον όμως με το Internet υπάρχει η δυνατότητα μετατροπής οποιασδήποτε συσκευής σε συσκευή απομακρυσμένου ελέγχου, με τη προϋπόθεση κάλυψης δικτύου, όπως θα αναλύσουμε παρακάτω.

1.2. Τρόπος λειτουργίας συστημάτων ασύρματου χειρισμού

Για να είναι επιτυχής ο τηλεχειρισμός μίας συσκευής, υπάρχει μία βασική αρχή: Το τηλεχειριστήριο είναι ο πομπός και η συσκευή ο δέκτης. Από εκεί και πέρα ανάλογα με τον τρόπο λειτουργίας και τις δυνατότητες της συσκευής μπορεί να υπάρχουν και άλλες λειτουργίες όπως η αμφίδρομη επικοινωνία μεταξύ των δύο συσκευών. Ωστόσο οποιαδήποτε παραλλαγή δεν αναιρεί την βασική αρχή του πομπού και του δέκτη. Οι εντολές αποστέλλονται σε άυλη μορφή μέσω σημάτων στον αέρα σε συγκεκριμένη συχνότητα που έχει επιλεχθεί από τον κατασκευαστή. Πριν την κατασκευή ενός τηλεχειριστήριου γίνεται η επιλογή ,ανάλογα με τις ανάγκες της συσκευής, για το πρωτόκολλο επικοινωνίας που πρόκειται να λειτουργεί η συσκευή. Το κάθε πρωτόκολλο έχει την ανάλογη συχνότητα και λογική που λειτουργεί. Τόσο ο πομπός όσο και ο δέκτης θα πρέπει να γνωρίζει το πρωτόκολλο επικοινωνίας για την επιτυχή επικοινωνία των δύο συσκευών. Μερικά από τα πρωτόκολλα ανήκουν συνήθως στις ραδιοσυχνότητες (RF) επικοινωνίας και είναι το LPD433, Bluetooth , LoRa , IR και το Internet. Στο τελευταίο είναι δυνατό να διαχειρίζεται κανείς μία συσκευή απομακρυσμένα σε οποιοδήποτε σημείο του κόσμου, με την προϋπόθεση ότι και οι δύο συσκευές έχουν πρόσβαση στο Internet. Με το Internet μπορούμε εύκολα να μετατρέψουμε μία συσκευή όπως έναν Η/Υ ή ένα κινητό σε συσκευή απομακρυσμένου χειρισμού. Αυτή η συσκευή μπορεί να είναι συνδεδεμένη στο δίκτυο ενσύρματα ή ασύρματα. Σε κάθε περίπτωση η λογική επικοινωνίας παραμένει η ίδια.

1.2.1. Υπέρυθρη ακτινοβολία

Η υπέρυθρη ακτινοβολία ή υπέρυθρες ακτίνες είναι τμήμα του φάσματος της ηλεκτρομαγνητικής ακτινοβολίας. Στο φάσμα τοποθετούνται ως μικρότερη συχνότητα στην προέκταση της κόκκινης ορατής ακτινοβολίας, εξ ου και το όνομα «υπέρυθρες» (υπό το ερυθρόν χρώμα). Υπέρυθρες ακτίνες χρησιμοποιούνταν για μεταφορά δεδομένων από τα κινητά τηλέφωνα πριν την κυκλοφορία του Bluetooth.

Η κύρια τεχνολογία που χρησιμοποιείται στα οικιακά τηλεχειριστήρια είναι το υπέρυθρο φως (IR). Το σήμα μεταξύ ενός τηλεχειριστηρίου και της συσκευής που ελέγχει αποτελείται από παλμούς υπέρυθρου φωτός, το οποίο είναι αόρατο στο ανθρώπινο μάτι, αλλά μπορεί να δει μέσω ψηφιακής

κάμερας, βιντεοκάμερας ή κάμερας τηλεφώνου. Ο πομπός στο ακουστικό του τηλεχειριστηρίου εκπέμπει μια ροή παλμών υπέρυθρου φωτός όταν ο χρήστης πατήσει ένα κουμπί στο ακουστικό. Ένας πομπός είναι συχνά μια δίοδος εκπομπής φωτός (LED) που είναι ενσωματωμένη στο άκρο κατάδειξης του ακουστικού του τηλεχειριστηρίου. Οι παλμοί υπέρυθρου φωτός σχηματίζουν ένα μοτίβο μοναδικό σε αυτό το κουμπί. Ο δέκτης στη συσκευή αναγνωρίζει το μοτίβο και αναγκάζει τη συσκευή να ανταποκριθεί ανάλογα.

Τα περισσότερα τηλεχειριστήρια για ηλεκτρονικές συσκευές χρησιμοποιούν μια δίοδο σχεδόν υπέρυθρης ακτινοβολίας για να εκπέμπουν μια δέσμη φωτός που φτάνει στη συσκευή. Ένα LED μήκους κύματος 940 nm είναι χαρακτηριστικό. Αυτό το υπέρυθρο φως δεν είναι ορατό στο ανθρώπινο μάτι αλλά λαμβάνεται από αισθητήρες στη συσκευή λήψης. Οι βιντεοκάμερες βλέπουν τη δίοδο σαν να παράγει ορατό μωβ φως. Με ένα τηλεχειριστήριο ενός καναλιού (μονής λειτουργίας, ενός κουμπιού), η παρουσία ενός σήματος φορέα μπορεί να χρησιμοποιηθεί για την ενεργοποίηση μιας λειτουργίας. Για τηλεχειριστήρια πολλαπλών καναλιών (κανονικά πολλαπλών λειτουργιών) απαιτούνται πιο περίπλοκες διαδικασίες: η μία συνίσταται στη διαμόρφωση του φορέα με σήματα διαφορετικών συχνοτήτων. Αφού ο δέκτης αποδιαμορφώσει το λαμβανόμενο σήμα, εφαρμόζει τα κατάλληλα φίλτρα συχνότητας για να διαχωρίσει τα αντίστοιχα σήματα. Συχνά μπορεί κανείς να ακούσει τα σήματα να διαμορφώνονται στον φορέα υπερύθρων χειρίζοντας ένα τηλεχειριστήριο σε πολύ κοντινή απόσταση από ένα ραδιόφωνο AM που δεν είναι συντονισμένο σε σταθμό. Σήμερα, τα τηλεχειριστήρια υπερύθρων χρησιμοποιούν σχεδόν πάντα έναν κωδικό διαμορφωμένου πλάτους παλμού, κωδικοποιημένο και αποκωδικοποιημένο από ψηφιακό υπολογιστή: μια εντολή από ένα τηλεχειριστήριο αποτελείται από μια σύντομη σειρά παλμών φέροντος-παρόντων και φορέα-μη-παρόντων διαφορετικών πλάτη.

Δεδομένου ότι τα τηλεχειριστήρια υπερύθρων (IR) χρησιμοποιούν φως, απαιτούν οπτική επαφή για τη λειτουργία της συσκευής προορισμού. Το σήμα μπορεί, ωστόσο, να ανακλάται από καθρέφτες, όπως και κάθε άλλη πηγή φωτός. Εάν απαιτείται λειτουργία όπου δεν είναι δυνατή η οπτική επαφή, για παράδειγμα όταν ελέγχετε εξοπλισμό σε άλλο δωμάτιο ή είναι εγκατεστημένο σε ντουλάπι, πολλές μάρκες υπερύθρων υπερύθρων διατίθενται για αυτό στην αγορά. Τα περισσότερα από αυτά διαθέτουν δέκτη υπερύθρων, που λαμβάνει το σήμα IR και το μεταδίδει μέσω ραδιοκυμάτων στο απομακρυσμένο τμήμα, το οποίο έχει έναν πομπό υπερύθρων που μιμείται τον αρχικό έλεγχο υπερύθρων. Οι δέκτες υπερύθρων τείνουν επίσης να έχουν λίγο πολύ περιορισμένη γωνία λειτουργίας, η οποία εξαρτάται κυρίως από τα οπτικά χαρακτηριστικά του φωτοτρανζίστορ. Ωστόσο, είναι εύκολο να αυξήσετε τη γωνία λειτουργίας χρησιμοποιώντας ένα ματ διαφανές αντικείμενο μπροστά από τον δέκτη.

1.2.2. Zigbee

Η τεχνολογία που ορίζεται από την προδιαγραφή Zigbee προορίζεται να είναι απλούστερη και λιγότερο δαπανηρή από άλλα ασύρματα προσωπικά δίκτυα περιοχής (WPAN), όπως το Bluetooth ή γενικότερη ασύρματη δικτύωση όπως το Wi-Fi. Οι εφαρμογές περιλαμβάνουν ασύρματους διακόπτες φωτός, οικιακές ενεργειακές οθόνες, συστήματα διαχείρισης κυκλοφορίας και άλλο καταναλωτικό και βιομηχανικό εξοπλισμό που απαιτεί ασύρματη μεταφορά δεδομένων μικρής εμβέλειας.

Η χαμηλή κατανάλωση ενέργειας περιορίζει τις αποστάσεις μετάδοσης στα 10–100 μέτρα οπτικής επαφής, ανάλογα με την ισχύ εξόδου και τα περιβαλλοντικά χαρακτηριστικά. Οι συσκευές Zigbee μπορούν να μεταδώσουν δεδομένα σε μεγάλες αποστάσεις περνώντας δεδομένα μέσω ενός δικτύου ενδιάμεσων συσκευών για να φτάσουν σε πιο απομακρυσμένες. Το Zigbee χρησιμοποιείται συνήθως σε εφαρμογές χαμηλού ρυθμού δεδομένων που απαιτούν μεγάλη διάρκεια ζωής της μπαταρίας και ασφαλή δικτύωση.

Το Zigbee είναι ένα πρότυπο ασύρματου πλέγματος χαμηλής κατανάλωσης που στοχεύει σε συσκευές που τροφοδοτούνται από μπαταρία σε εφαρμογές ασύρματου ελέγχου και παρακολούθησης. Το Zigbee παρέχει επικοινωνία χαμηλής καθυστέρησης. Τα τσιπ Zigbee είναι συνήθως ενσωματωμένα με ραδιόφωνα και με μικροελεγκτές.

1.2.3. LPD433

Το LPD433 είναι μία μπάντα συχνοτήτων στο UHF εύρος που συνήθως δε χρειάζεται κάποια ειδική άδεια για τη χρήση του. Χρησιμοποιείται συνήθως σε συσκευές χαμηλής ενέργειας και το εύρος που χρησιμοποιείται είναι από 433,05 MHz έως 434.790 Mhz. Δε χρειάζεται κάποια ιδιαίτερη ζεύξη μεταξύ πομπού, δέκτη και παρέχει αρκετά καλή εμβέλεια. Να σημειωθεί ότι δε χρειάζεται οπτική επαφή με το δέκτη.

1.2.4. Bluetooth

Το Bluetooth είναι μία ασύρματη τεχνολογία μικρής εμβέλειας. Λειτουργεί στο εύρος μεταξύ 2,402GHz έως 2,48GHz. Δε χρειάζεται οπτική επαφή και απαιτείται ζεύξη μεταξύ πομπού και δέκτη. Στις περισσότερες περιπτώσεις η ισχύς μετάδοσης δε ξεπερνάει τα 2,5mW με αποτέλεσμα η μέγιστη εμβέλεια να μη ξεπερνάει τα 10 μέτρα. Πλέον οι περισσότερες συσκευές που δε προορίζονται για ιδιαίτερη απομακρυσμένη χρήση χρησιμοποιούν το Bluetooth όπως κινητά, ακουστικά, άλλες φορητές ή ακόμη και σταθερές συσκευές. Παράλληλα με το Bluetooth υπάρχει και το Bluetooth Low Energy (BLE) που έχει την ίδια λογική λειτουργίας, αλλά είναι πιο προσεγμένο έτσι ώστε να δαπανάει τη λιγότερη δυνατή ενέργεια για την κάθε λειτουργία που πραγματοποιείται.

1.2.5. Internet

Ο χειρισμός μία συσκευής μέσω του Internet δε διαφέρει από την οποιαδήποτε άλλη χρήση του Internet. Πακέτα δεδομένων μεταφέρονται από μία συσκευή σε μία άλλη. Η χρήση που προσφέρει το Internet στον τηλεχειρισμό είναι ότι κάθε συσκευή που έχει πρόσβαση στο Internet μπορεί να χρησιμοποιηθεί σαν ασύρματο χειριστήριο. Το μόνο που χρειάζεται είναι ένα πρόγραμμα διεπαφής με τον χρήστη ώστε να στέλνονται οι επιθυμητές εντολές προς τη απομακρυσμένη συσκευή. Αυτό το πρόγραμμα μπορεί να μην είναι άμεσα ορατό προς το χρήστη, όπως ένα πρόγραμμα σε επίπεδο «οδηγού» (driver). Οποιαδήποτε από τις δύο συσκευές ή ακόμα και οι δύο συσκευές μπορούν να είναι συνδεμένες ασύρματα είτε ενσύρματα στο Internet. Ασύρματη σύνδεση στο Internet γίνεται συνήθως μέσω WiFi ή Κυψελωτό δίκτυο(CellularNetwork), όπου το δεύτερο χρησιμοποιεί κεραίες κινητής τηλεφωνίας και μία κάρτα SIM για την επιτυχή σύνδεση.

1.3. Ειδικές κατηγορίες τηλεχειριστηρίων

Σαν ειδικές κατηγορίες αναφέρονται τα τηλεχειριστήρια τα οποία διαθέτουν επιπλέον λειτουργίες πέραν του τρόπου επικοινωνίας με μία ή περισσότερες συσκευές. Αυτό το χαρακτηριστικό τα κάνει να ξεχωρίζουν από τα κοινά τηλεχειριστήρια και αναβαθμίζουν τη χρησιμότητα τους

1.3.1. Προγραμματιζόμενα τηλεχειριστήρια

Τα προγραμματιζόμενα τηλεχειριστήρια είναι αυτά τα οποία μπορούν να αντιγράψουν διαφορετικούς κωδικούς λειτουργιών ή διαθέτουν «ειδικές» λειτουργίες τις οποίες έχει ορίσει ο κατασκευαστής για μία η περισσότερες συσκευές. Ένα κλασικό παράδειγμα μίας τέτοιας συσκευής είναι το τηλεχειριστήριο υπέρυθρων που χρησιμοποιείτε για τη λειτουργία της τηλεόρασης και του αποκωδικοποιητή της. Συνήθως ένα τέτοιο χειριστήριο περιλαμβάνεται μαζί με τη συσκευασία του αποκωδικοποιητή. Όπου με μία αλληλουχία πλήκτρων η συσκευή μπορεί να «μάθει» τους κωδικούς που χρησιμοποιεί το τηλεχειριστήριο της τηλεόρασης. Ώστε να είναι δυνατή η λειτουργία της από τα «επιπλέον» προγραμματιζόμενα πλήκτρα αυτού του χειριστηρίου

1.3.2. Τηλεχειριστήρια Γενική χρήσης

Τα τηλεχειριστήρια γενικής χρήσης, σε αντίθεση με τα απλά τηλεχειριστήρια (όπου είναι προγραμματισμένα να λειτουργούν με μία συγκεκριμένη συσκευή), συνήθως είναι μίας συγκεκριμένης εταιρίας και μπορεί να λειτουργεί αυτόματα με διαφορετικές συσκευές της ίδιας εταιρίας. Συνήθως οι υποστηριζόμενες συσκευές αναγράφονται στο εγχειρίδιο χρήσης της συσκευής. Ένα ακόμα κλασσικό παράδειγμα τέτοια συσκευής είναι και πάλι ένα τηλεχειριστήριο τηλεόρασης όπου συνηθίζεται να μην αγοράζουμε το το ίδιο χειριστήριο, της ίδια εταιρίας, αφότου χαλάσει. Παρόλα αυτά, αυτά τα τηλεχειριστήρια μπορούν και λειτουργούν με μία μεγάλη γκάμα τηλεοράσεων. Ένα ακόμα παράδειγμα είναι τηλεχειριστήρια , συνήθως τρίτων κατασκευαστών, για παιχνιδοκονσόλες. Όπου η ίδια συσκευή μπορεί να λειτουργήσει (πέραν από τη παιχνιδοκονσόλα) και συνδέοντας το στον Η/Υ.

Κεφάλαιο 2ο – Καθημερινές Ασύρματες Συνδέσεις

2.1. Επικοινωνία Συσκευών σε τοπικό δίκτυο – LAN

Η επικοινωνίας συσκευών σε τοπικό δίκτυο (LAN) ακολουθεί την ίδια λογική επικοινωνίας μέσω Internet που αναφέραμε παραπάνω. Η μόνη διαφορά στη πράξη και όχι στον τρόπο της επικοινωνίας είναι ότι δεν υπάρχει η εκμετάλλευση του Internet. Αυτό σημαίνει ότι ο χειρισμός μίας συσκευής που βρίσκεται εκτός του τοπικού δικτύου , δηλαδή εκτός του δικτύου το οποίο βρίσκεται η συσκευή απομακρυσμένου χειρισμού, είναι αδύνατη. Σε επέκταση του LAN υπάρχει και το "Ασύρματο Τοπικό Δίκτυο» (WLAN) όπου προσφέρει την ασύρματη σύνδεση στο υπάρχων LAN δίκτυο και ακολουθεί την ίδια λογική. Η επικοινωνία σε επίπεδο LAN γίνεται συνήθως είτε γιατί δεν είναι δυνατή ή μπορεί να διακοπεί η σύνδεση στο Internet, είτε γιατί η επικοινωνία των συσκευών δεν προορίζεται να είναι εκτός εμβέλειας του LAN.

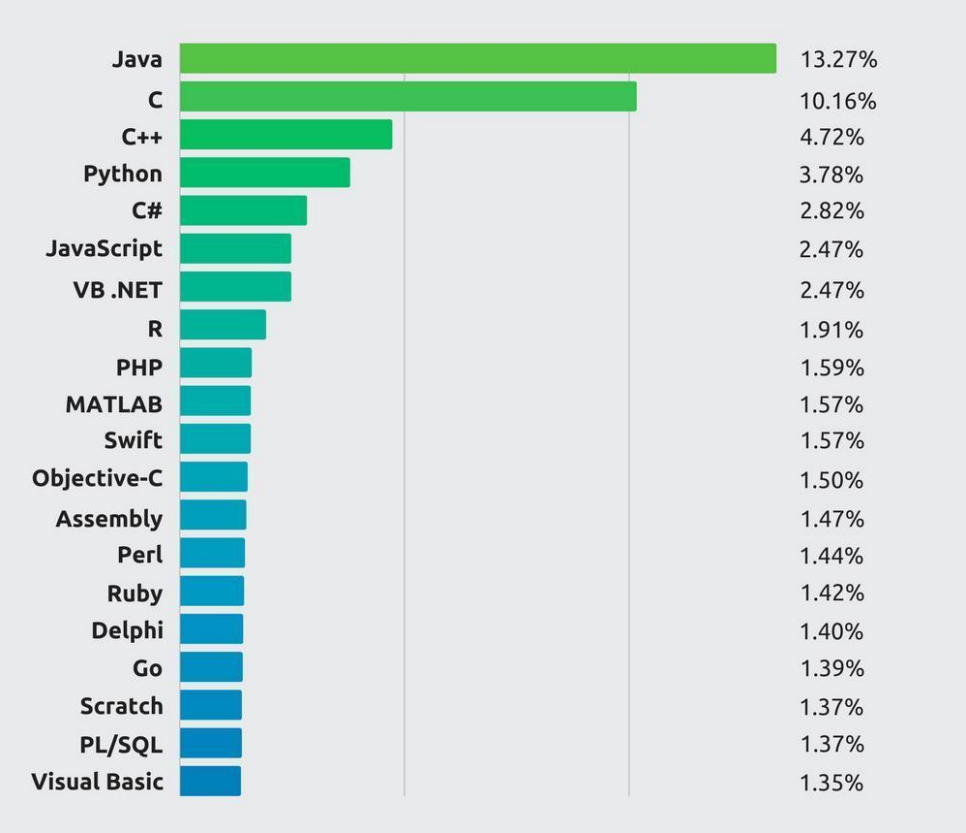
2.2. Bluetooth vs WLAN

Σαν γενική αρχή τα δύο πρωτόκολλα λειτουργούν στο αποτέλεσμα με παρόμοιο τρόπο (όσον αφορά τον χρήστη). Μπορούν να κάνουν τις ίδιες εργασίες. Συχνά παρατηρούνται συσκευές όπως εκτυπωτές, επικοινωνίας μεταξύ έξυπνων κινητών (smartphones) και γενικά συσκευές γενικής ή καθημερινής χρήσης να προσφέρουν τη δυνατότητα επικοινωνίας και με τα δύο πρωτόκολλα. Η κύρια διαφορά τους είναι ότι μία σύνδεση σε επίπεδο τοπικού δικτύου μπορεί να έχει πολλές συσκευές μαζί ταυτόχρονα, χωρίς να επηρεάζει η μία την άλλη. Μία Bluetooth επικοινωνία μπορεί γίνει μόνο με δύο συσκευές. Μία ακόμα κύρια διαφορά είναι ότι για την επιτυχή Bluetooth επικοινωνία χρειάζεται οι δύο συσκευές να είναι εντός εμβέλειας, όπως αναφέρθηκε στο 1ο κεφάλαιο . Είναι δυνατό να πραγματοποιηθεί μία επικοινωνία σε επίπεδο LAN ή WLAN σε οποιοδήποτε μέρος ,ανεξάρτητα της απόστασης, αρκεί να συνδέεται στο ίδιο τοπικό δίκτυο, όπου συνήθως αυτό είναι και στο ίδιο κτίριο ή σύμπλεγμα κτιρίων. Για παράδειγμα, με αυτό τον τρόπο μπορεί η συσκευή προς χρήση να είναι στον 3ο όροφο ενός κτιρίου και η χειριστήρια συσκευή να είναι στο ισόγειο του ίδιου κτιρίου, χωρίς να υπάρχει κανένα πρόβλημα. Αξίζει να σημειωθεί ότι η μέγιστη ταχύτητα μεταφοράς δεδομένων του Bluetooth είναι κατά πολύ χαμηλότερη της μέγιστης ταχύτητας του WLAN.

Κεφάλαιο 3ο

3.1. Ανάπτυξη εφαρμογών για Λειτουργικό Σύστημα

Για την δημιουργία ή αλλιώς ανάπτυξη εφαρμογών για οποιοδήποτε σύστημα αλλά και για ένα λειτουργικό σύστημα χρειάζεται να γίνει χρήση μίας γλώσσας προγραμματισμού. Η επιλογή της γλώσσας προγραμματισμού γίνεται συνήθως ανάλογα για το επιθυμητό αποτέλεσμα σε συνδυασμό με το ανάλογο OS, στο οποίο πρόκειται να γίνει η χρήση του. Οι πιο γνωστές σε χρήση γλώσσες προγραμματισμού για λειτουργικό σύστημα αναγράφονται παρακάτω.



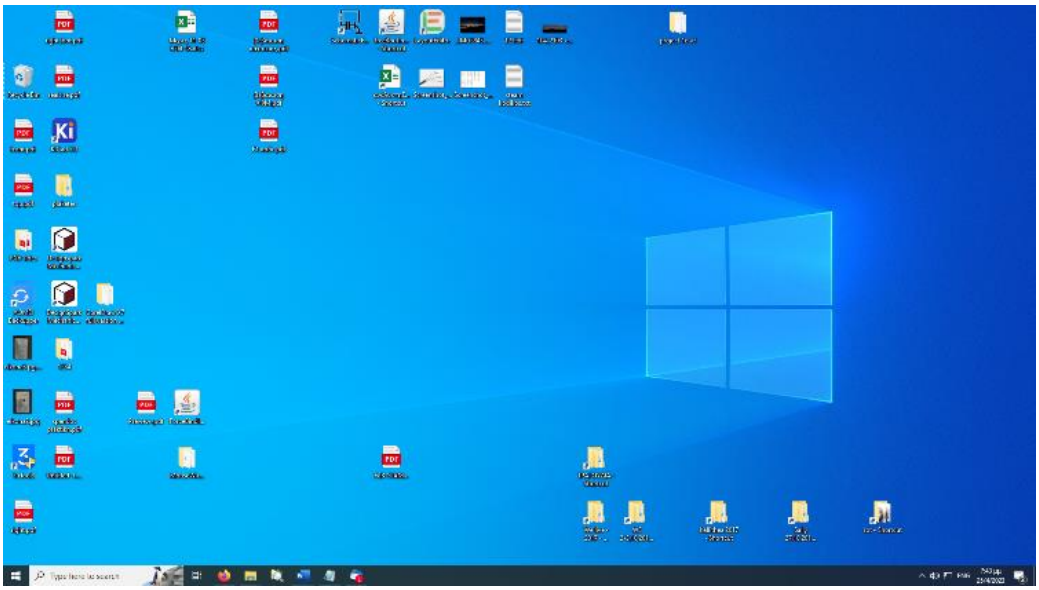
Εικόνα 3.1 : Δημοτικότητα γλωσσών προγραμματισμού το έτος 2017

Το διάγραμμα αυτό είναι ενδεικτικό και δείχνει απλά μερικές από τις πιο γνωστές σε χρήση γλώσσες προγραμματισμού. Μία από τις πιο παλιές γλώσσες είναι η C/C++ η οποία ακόμα και σήμερα βρίσκει εφαρμογή σε πολλές συσκευές. Επιπλέον για την επιλογή της γλώσσας προγραμματισμού πρέπει να ληφθούν και οι εξής παράμετροι: Πρώτον αν η γλώσσα που θα επιλέξουν τηρεί τις δυνατότητες για αυτό που θέλουμε να κάνουμε και δεύτερον αν είναι συμβατή με τη συσκευή ή το λειτουργικό σύστημα που τη προορίζουμε. Για παράδειγμα το λειτουργικό σύστημα Linux ή MacOS δεν παρέχει απευθείας υποστήριξη για την γλώσσα προγραμματισμού 'C#', οπότε και καθίσταται ανεπιθύμητη για χρήση σε ένα τέτοιο λειτουργικό χωρίς την βοήθεια κάποιου τρόπου «μετάφρασης» της πριν την εκτέλεση της εργασίας.

3.1.1. Λειτουργικό Σύστημα (OS)

Ένα λειτουργικό σύστημα είναι ένα λογισμικό συστήματος που διαχειρίζεται το hardware , το software του ίδιου το λογισμικού και παρέχει υπηρεσίες και άλλα για τις εφαρμογές του λογισμικού. Το λειτουργικό σύστημα διαφέρει ανάλογα με τη συσκευή. Μία κονσόλα παιχνιδιού έχει το δικό της λογισμικό, ένα smartphone έχει το δικό του, ένας Η/Υ το δικό του κ.α. Αξίζει να σημειωθεί ότι σε πολλές περιπτώσεις, αν το επιτρέπει ο κατασκευαστής και ο τρόπος κατασκευής της συσκευής μπορεί να γίνει αλλαγή του λειτουργικού συστήματος. Αυτό μπορεί να συμβεί, για παράδειγμα σε έναν υπολογιστή που λειτουργεί με το «κλειστό» λειτουργικό σύστημα των «Windows» σε σύγκριση με το «ανοιχτό» ελεύθερο προς χρήση και τροποποίηση «Linux».

Εικόνα 3.2 : Λειτουργικό σύστημα Linux

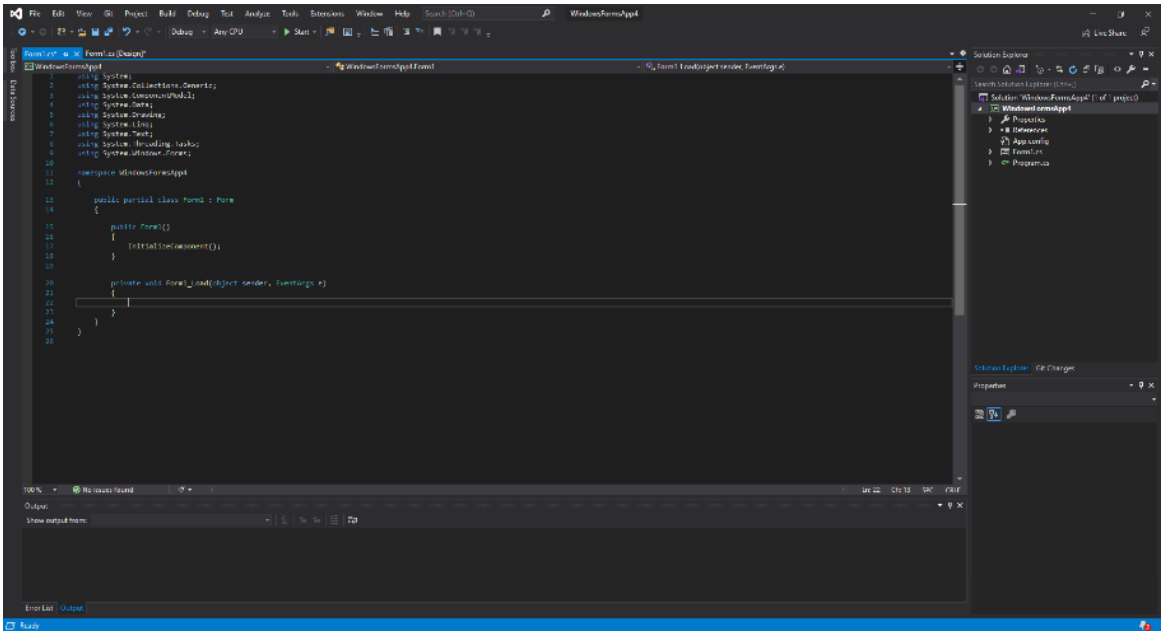


Εικόνα 3.3 : Λειτουργικό σύστημα Microsoft Windows

3.1.2. Ολοκληρωμένο περιβάλλον ανάπτυξης (IDE)

Ένα ολοκληρωμένο περιβάλλον ανάπτυξης είναι ένα ολοκληρωμένο λογισμικό πρόγραμμα που εκτελείται μέσα από ένα λογισμικό σύστημα, το οποίο βοηθάει στην ανάπτυξη προγραμμάτων ή εφαρμογών για ένα λογισμικό σύστημα. Τα βασικά χαρακτηριστικά ενός IDE είναι ότι περιλαμβάνει κάποιον τρόπο επεξεργασίας πηγαίου κώδικα, έναν «μεταγλωττιστή», και αναγνώριση σφαλμάτων .Ανάλογα με το πόσο εξελιγμένο είναι ένα IDE μπορεί να έχει εργαλεία αυτόματης παραγωγής κώδικα, όπου βοηθάνε στην ποιο γρήγορη ή «έξυπνη» γραφή κώδικα, Debugger και όπου με τη περίπτωση δίνει και πολλές άλλες δυνατότητες. Αξίζει να αναφερθεί πως ανάλογα με τον τύπο κώδικα που χρησιμοποιείται, το IDE μπορεί να προσφέρει και ένα γραφικό περιβάλλον, ώστε να είναι εύκολο

να αντιλαμβάνεται ο προγραμματιστής πως εμφανίζεται και πως θα είναι στη πράξη ο κώδικας που γράφει.



Εικόνα 3.4 : Visual Studio IDE

Κεφάλαιο 4ο- Σύστημα Μικροελεγκτή MCU

4.1. Εισαγωγή στους μικροελεγκτές

Οι μικροελεγκτές είναι ένα σύστημα υπολογισμού και εκτέλεσης εργασιών που έχει τη μορφή ολοκληρωμένου κυκλώματος, τα οποία είναι καλυπτόμενα από ένα πλαστικό περίβλημα και ανήκουν στην οικογένεια των επεξεργαστών. Συνήθως τους μπερδεύουμε με τους μικροεπεξεργαστές, οι οποίοι μπορούν να λειτουργούν με ελάχιστα, σε σχέση με τους μικροελεγκτές, εξωτερικά εξαρτήματα. Αυτό είναι δυνατό λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει. Με τη βοήθεια μία κατάλληλης συσκευής προγραμματισμού γίνεται να αναθέσουμε σε ένα MCU διαφορετικές λειτουργίες, αρκετά πολύπλοκες ώστε μία MCU μονάδα να ανταποκριθεί σε πληθώρα εφαρμογών. Ένα μικροελεγκτής διαθέτει τουλάχιστον μία κεντρική μονάδα επεξεργασίας “CPU”, μνήμη προγράμματος “Program Memory”, μνήμη δεδομένων “Data Memory”, αριθμητική λογική μονάδα “ALU”, μονάδες εισόδου/εξόδου “I/O”. Στην εργασία αυτή έγινε χρήση μικροελεγκτή ESP32. Η έκδοση ESP32-WROOM-D32 παρέχει ενσωματωμένη κεραία, με το οποίο δίνεται η δυνατότητα στην άμεση χρήση των WiFi και Bluetooth. Επίσης λόγω της πολύ μικρής έως ελάχιστης κατανάλωσης ρεύματος, θεωρείται ιδανικός για πάρα πολλές IoT εφαρμογές.

4.2. Τυπική αρχιτεκτονική μικροελεγκτή

Οι αρχιτεκτονικές των μικροελεγκτών ανήκουν σε μία εκ των δύο κυρίων αρχιτεκτονικών. Τη VonNeumann και τη Harvard. Ένας AVR μικροελεγκτής ανήκει σε μία τροποποιημένη μορφή της αρχιτεκτονικής Harvard.

Η αρχιτεκτονική VonNeumann αποτελείται από μια κοινή μνήμη, η οποία εμπεριέχει τις εντολές του προγράμματος καθώς και τα αποθηκευμένα δεδομένα. Η εκτέλεση των εντολών γίνεται με σχετικά πιο αργό ρυθμό, επειδή υπάρχει μόνο ένας δίαυλος και για τις δύο μνήμες. Αδυνατεί να προσπελάσει στη μνήμη ταυτόχρονα για τις εντολές και τα δεδομένα. Σχετικά με την εκτέλεση μιας εντολής απαιτούνται περισσότεροι κύκλοι μηχανής (MC). Επίσης, ο επεξεργαστής ανήκει στην οικογένεια των λεγόμενων Υπολογιστών με Διευρυμένο Ρεπερτόριο Εντολών (CISC). Ένας μικροελεγκτής CISC περιλαμβάνει πιο περίπλοκες εντολές, με συνέπεια να εκτελεί πιο πολύπλοκες διαδικασίες. Ωστόσο, όσο μεγαλύτερο είναι το πλήθος του συνόλου των εντολών, τόσο πιο πολύπλοκη είναι η σχεδίαση του, με περισσότερη καθυστέρηση για την εκτέλεση της κάθε εντολής.

Από την άλλη πλευρά, η αρχιτεκτονική Harvard αποτελείται από ξεχωριστές μνήμες για το πρόγραμμα και για τα δεδομένα. Οι εντολές εκτελούνται με γρηγορότερο ρυθμό, λόγω του ότι υπάρχουν δύο μοναδικοί δίαυλοί που επικοινωνούν με την ΚΜΕ, καθιστώντας το αρκετά πιο αποδοτικό. Η εκτέλεση μιας εντολής χρειάζεται λιγότερους κύκλους μηχανής. Ο μικροελεγκτής σε αυτήν την περίπτωση αποτελεί μέρος των Υπολογιστών Μειωμένου Ρεπερτορίου Εντολών (RISC). Η κατηγορία RISC παρέχει μικρό σύνολο εντολών, οι οποίες εκτελούνται ταχύτατα. Έτσι, η σχεδίαση του μικροελεγκτή απλοποιείται σημαντικά, ώστε οι εντολές να εκτελούνται αρκετά πιο γρήγορα σε σχέση με τους CISC.

Είναι σημαντικό να αναφερθεί πως υπάρχει μία παραλλαγή αυτής της αρχιτεκτονικής, η “τροποποιημένη αρχιτεκτονική Harvard. Η διαφορά της είναι ότι επιτρέπει τα περιεχόμενα των εντολών στην μνήμη να είναι προσβάσιμα σαν δεδομένα. Οι περισσότεροι Η/Υ που κατατάσσονται στην αρχιτεκτονική Harvard, στη πραγματικότητα είναι στην “τροποποιημένη αρχιτεκτονική Harvard.

Το AVR ήταν μια από τις πρώτες οικογένειες μικροελεγκτών που χρησιμοποίησαν μνήμη flash στο τσιπ για αποθήκευση προγραμμάτων, σε αντίθεση με την εφάπαξ προγραμματιζόμενη ROM, EPROM ή EEPROM που χρησιμοποιούσαν άλλοι μικροελεγκτές εκείνη την εποχή. Οι μικροελεγκτές AVR βρίσκουν πολλές εφαρμογές ως ενσωματωμένα συστήματα. Είναι ιδιαίτερα κοινές σε ενσωματωμένες εφαρμογές για χομπίστες και εκπαιδευτικές εφαρμογές, οι οποίες έχουν γίνει δημοφιλείς με τη χρήση τους σε πολλές πλακέτες ανάπτυξης υλικού, όπως το Arduino. Το AVR είναι ένα τροποποιημένο μηχάνημα αρχιτεκτονικής του Χάρβαρντ, όπου το πρόγραμμα και τα δεδομένα αποθηκεύονται σε χωριστά συστήματα φυσικής μνήμης που εμφανίζονται σε διαφορετικούς χώρους διευθύνσεων, αλλά έχουν τη δυνατότητα ανάγνωσης στοιχείων δεδομένων από τη μνήμη προγράμματος χρησιμοποιώντας ειδικές οδηγίες.

Οι βασικότερες οικογένειες AVR μικροελεγκτών είναι :

- AVR
- tinyAVR
- megaAVR
- AVR Dx
- XMEGA
- Application-specific AVR

- FPSLIC (AVR with FPGA)
- 32-bit AVRs

Κάθε στοιχείο ή υποσύστημα σε ένα μικροελεγκτή έχει σχεδιαστεί για να εξυπηρετεί τουλάχιστον μία συγκεκριμένη λειτουργία στο σύστημα. Παρακάτω αναφέρονται μερικά από τα τυπικά στοιχεία που περιλαμβάνει ένας μικροελεγκτής στο πυρήνα του, τις περιφερειακές μονάδες , μνήμες, μετατροπείς αναλογικού και ψηφιακού σήματος, “πόρτες” εισόδου και εξόδου και άλλα ειδικά χαρακτηριστικά.

4.2.1. Μνήμη συσκευής

Οι μνήμες Flash, EEPROM και SRAM είναι όλ**ες** ενσωματωμέν**ες** σε ένα ενιαίο τσιπ, καταργώντας την ανάγκη για εξωτερική μνήμη στις περισσότερες εφαρμογές. Ορισμένες συσκευές διαθέτουν μια επιλογή παράλληλου εξωτερικού διαύλου που επιτρέπει την προσθήκη πρόσθετης μνήμης δεδομένων ή συσκευών με χαρτογράφηση μνήμης. Σχεδόν όλες οι συσκευές (εκτός από τα μικρότερα τσιπ TinyAVR) διαθέτουν σειριακές διεπαφές, οι οποίες μπορούν να χρησιμοποιηθούν για τη σύνδεση μεγαλύτερων σειριακών EEPROM ή flash.

4.2.2. Μνήμη τυχαία προσπέλασης RAM

Ο χώρος διευθύνσεων δεδομένων αποτελείται από το αρχείο καταχωρητή, από τους καταχωρητές εισόδου/εξόδου και από την SRAM. Ορισμένα μικρά μοντέλα αντιστοιχίζουν επίσης τη ROM του προγράμματος στο χώρο διευθύνσεων δεδομένων, κάτι όμως που δεν συμβαίνει σε μεγαλύτερα μοντέλα.

4.2.3. Μνήμη προγράμματος

Οι οδηγίες του προγράμματος αποθηκεύονται σε μη-πτητική μνήμη flash. Αν και τα MCU είναι 8-bit, κάθε εντολή παίρνει μία ή δύο λέξεις 16-bit. Το μέγεθος της μνήμης του προγράμματος συνήθως υποδεικνύεται στην ονομασία της ίδιας της συσκευής (π.χ., η γραμμή ATmega64x έχει 64 KB flash, ενώ η γραμμή ATmega32x έχει 32 KB. Δεν υπάρχει πρόβλεψη για μνήμη προγράμματος εκτός τσιπ. Όλος ο κώδικας που εκτελείται από τον πυρήνα AVR πρέπει να βρίσκεται στη μνήμη Flash στο chip. Ωστόσο, αυτός ο περιορισμός δεν ισχύει για τα τσιπ AT94, FPSLIC ,AVR/FPGA.

4.2.4. EEPROM

Σχεδόν όλοι οι μικροελεγκτές AVR έχουν εσωτερική μνήμη EEPROM για ημί-μόνιμη αποθήκευση δεδομένων. Όπως και η μνήμη flash, έτσι και η EEPROM μπορεί να διατηρεί το περιεχόμενο της ακόμα και χωρίς παροχή ρεύματος.

Στις περισσότερες παραλλαγές της αρχιτεκτονικής AVR, η ενσωματωμένη μνήμη EEPROM δεν αντιστοιχίζεται στον διευθυνσιοδοτούμενο χώρο μνήμης του μικροελεγκτή. Είναι δυνατή η πρόσβαση μόνο με τον ίδιο τρόπο με μια εξωτερική περιφερειακή συσκευή, χρησιμοποιώντας

ειδικούς καταχωρητές δείκτη και οδηγίες ανάγνωσης/εγγραφής, γεγονός που καθιστά την πρόσβαση EEPROM πολύ πιο αργή από άλλες εσωτερικές RAM.

Ωστόσο, ορισμένες συσκευές της οικογένειας SecureAVR (AT90SC) χρησιμοποιούν μια ειδική αντιστοίχιση EEPROM στη μνήμη δεδομένων ή του προγράμματος, ανάλογα με τη διαμόρφωση. Η οικογένεια XMEGA επιτρέπει επίσης την αντιστοίχιση του EEPROM στο χώρο διευθύνσεων δεδομένων.

Ο αριθμός των εγγραφών στην EEPROM είναι περιορισμένος. Η Atmel καθορίζει 100.000 κύκλους εγγραφής στα φύλλα δεδομένων της. Μία καλά σχεδιασμένη ρουτίνα εγγραφής EEPROM θα πρέπει να συγκρίνει τα περιεχόμενα μιας διεύθυνσης EEPROM με τα επιθυμητά περιεχόμενα και να εκτελεί μια πραγματική εγγραφή μόνο εάν χρειάζεται αλλαγή του περιεχομένου. Ακόμη πρέπει να λάβουμε υπόψη ότι η διαγραφή και η εγγραφή μπορούν να εκτελεστούν χωριστά σε πολλές περιπτώσεις, byte-προς-byte, κάτι που μπορεί επίσης να βοηθήσει στην παράταση της διάρκειας ζωής όταν τα bit χρειάζεται μόνο να οριστούν σε όλα τα 1 (διαγραφή) ή επιλεκτικά να διαγραφούν σε 0 (εγγραφή).

4.2.5. Μετατροπείς Αναλογικού και ψηφιακού σήματος

Στην ηλεκτρονική, ένας μετατροπέας αναλογικού σε ψηφιακό (ADC, A/D ή A-to-D) είναι ένα σύστημα που μετατρέπει ένα αναλογικό σήμα, όπως έναν ήχο που λαμβάνεται από ένα μικρόφωνο ή το φως που εισέρχεται σε μια ψηφιακή φωτογραφική μηχανή, σε ένα ψηφιακό σήμα. Ένα ADC μπορεί επίσης να παρέχει μια απομονωμένη μέτρηση, όπως μια ηλεκτρονική συσκευή που μετατρέπει μια αναλογική τάση ή ρεύμα εισόδου σε ψηφιακό αριθμό που αντιπροσωπεύει το μέγεθος της τάσης ή του ρεύματος. Συνήθως η ψηφιακή έξοδος είναι ένας συμπληρωματικός δυαδικός αριθμός **δύο** που είναι ανάλογος της εισόδου, αλλά υπάρχουν και άλλες δυνατότητες.

Υπάρχουν πολλές αρχιτεκτονικές ADC. Λόγω της πολυπλοκότητας και της ανάγκης για ακριβή αντιστοιχισμένα εξαρτήματα, όλα εκτός από τα πιο εξειδικευμένα ADC, υλοποιούνται ως ολοκληρωμένα κυκλώματα (IC). Αυτά συνήθως έχουν τη μορφή ολοκληρωμένων κυκλωμάτων μικτού σήματος μετάλλου-οξειδίου-ημιαγωγού (MOS) που ενσωματώνουν αναλογικά και ψηφιακά κυκλώματα. Αντίθετα ένας μετατροπέας ψηφιακού σε αναλογικό (DAC) εκτελεί την αντίστροφη λειτουργία. μετατρέπει ένα ψηφιακό σήμα σε αναλογικό.

Πόρτες εισόδου/εξόδου (I/O Ports)

Κάθε θύρα GPIO σε ένα tiny ή mega AVR οδηγεί έως και οκτώ “ποδαράκια” και ελέγχεται από τους παρακάτω 8-bit καταχωρητές:

- **DDRx: DataDirectionRegister**
  Διαμορφώνει τα “ποδαράκια” είτε ως εισόδους είτε ως εξόδους.
- **PORTx: Outputportregister**
  Ορίζει την τιμή εξόδου στα “ποδαράκια” που έχουν διαμορφωθεί ως έξοδοι. Ενεργοποιεί ή απενεργοποιεί την αντίσταση pull-up στα “ποδαράκια” που έχουν διαμορφωθεί ως είσοδοι.
- **PINx: Inputregister**
  Χρησιμοποιείται για την ανάγνωση ενός σήματος εισόδου. Σε ορισμένες συσκευές, αυτός ο καταχωρητής μπορεί να χρησιμοποιηθεί για εναλλαγή αντιστροφή**ς** κατάστασης ενός από τα

“ποδαράκια”. Γράφοντας ένα bit σε ένα από τα “ποδαράκια” (PINx), **αυτός** εναλλάσσει το αντίστοιχο bit στην ανάλογη πόρτα (PORTx), ανεξάρτητα από τη ρύθμιση του bit από το καταχωρητή DDRx.

Νεότερα ATtiny AVR, έχουν τους καταχωρητές ελέγχου λίγο διαφορετικά ρυθμισμένους. Άλλοι έχουν επιπλέον καταχωρητές για push/pull, αντιστάσεις pull-up κ.α.

4.2.6. Πρόγραμμα εκτέλεσης

Τα AVR της Atmel έχουν σχεδιασμό αγωγού δύο σταδίων, μονού επιπέδου. Αυτό σημαίνει ότι η επόμενη εντολή μηχανής ανακτάται καθώς εκτελείται η τρέχουσα. Οι περισσότερες οδηγίες απαιτούν μόνο έναν ή δύο κύκλους ρολογιού, καθιστώντας τα AVR σχετικά γρήγορα μεταξύ των μικροελεγκτών οκτώ bit. Οι επεξεργαστές AVR σχεδιάστηκαν με γνώμονα την αποτελεσματική εκτέλεση μεταγλωττισμένου κώδικα C και διαθέτουν αρκετούς ενσωματωμένους δείκτες για την εργασία. Αυτοί είναι:

- **Αριθμητική λογική μονάδα ALU**
  Η Αριθμητική και Λογική Μονάδα (ALU) είναι υπεύθυνη για την εκτέλεση αριθμητικών και λογικών πράξεων. Μπορεί να κάνει πράξεις, όπως δυαδική πρόσθεση, αφαίρεση, τις λογικές πράξεις NOT, AND, OR, πολλαπλασιασμούς, διαιρέσεις, συμπλήρωμα ως προς 1 (NOT) και 2, ολίσθηση και περιστροφή. Πάντοτε υπάρχει ένας καταχωρητής εργασίας W (WorkRegister), ο οποίος είναι 8 bit και βρίσκεται μέσα στην Κεντρική Μονάδα Επεξεργασίας. Στον καταχωρητή W δύναται να του φορτωθεί μια δυαδική τιμή και στη συνέχεια να αποθηκευτεί σε μια θέση της μνήμης δεδομένων. Όταν πρέπει να εκτελεστεί μια εντολή με ένα όρισμα μόνο, μπορεί να γίνει είτε από τον καταχωρητή W είτε από οποιοδήποτε άλλο που το αφορά. Έτσι ολοκληρώνεται μια πράξη μεταξύ του καταχωρητή W, πάντα σε σχέση με κάποιον άλλον καταχωρητή. Σύμφωνα με την εντολή ή την πράξη που εκτελείται κάθε φορά από την ALU, ο καταχωρητής κατάστασης (Status Register) δίνει κάποιες πληροφορίες σχετικά με το αποτέλεσμα της τελευταίας ενέργειας, μαζί με κάποιες σημαίες (Flags).

- **Συγκριτής (Comparator)**
  Ο Συγκριτής (Comparator) είναι ένα αναλογικό κύκλωμα, το οποίο συγκρίνει την τάση δύο σημάτων και εμφανίζει στην έξοδό του**ς** μια ψηφιακή στάθμη λογικού "0" ή "1". Πρέπει να τονιστεί πως οι είσοδοι χωρίζονται στην μη αναστρέφουσα (Non-inverted) και την αναστρέφουσα (Inverted). Σε περίπτωση που η αναλογική τάση του ακροδέκτη VIN+ γίνει μεγαλύτερη από την τάση του VIN-, τότε το ψηφιακό επίπεδο της εξόδου θα είναι λογικό "1". Όμως, εάν η αναλογική τάση για το ποδαράκι VIN+ γίνει μικρότερη από την τάση του VIN-, αυτό θα έχει ως αποτέλεσμα το ψηφιακό επίπεδο της εξόδου να αλλάξει σε λογικό "0". Ο Συγκριτής περιλαμβάνει τα παρακάτω χαρακτηριστικά :
    - Ανεξάρτητος Έλεγχος Συγκριτή
    - Προγραμματιζόμενη Επιλογή Εισόδου
    - Η Σύγκριση Εξόδου είναι διαθέσιμη Εξωτερικά/Εσωτερικά
    - Προγραμματιζόμενη Πολικότητα Εξόδου
    - Διακοπή Κατά την Αλλαγή
    - Αφύπνιση από τον Ύπνο
    - Προγραμματιζόμενη Βελτιστοποίηση Ταχύτητας/Ισχύος
    - Απενεργοποίηση PWM
    - Προγραμματιζόμενη και Σταθερή Αναφορά Τάση

- **Interrupts**

  Η Interrupt είναι μια μέθοδος, με την οποία διακόπτεται η ροή της εκτέλεσης του τρέχοντος προγράμματος στον μικροελεγκτή, εξαιτίας κάποιου εσωτερικού ή εξωτερικού συμβάντος. Αν συμβεί διακοπή λόγο κάποιου ακροδέκτη ή χρονιστή, τότε αυτό έχει ως αποτέλεσμα ο μικροελεγκτής να απαντήσει σε αυτή την αίτηση. Τότε πραγματοποιείται αλλαγή διεύθυνσης του τρέχοντος προγράμματος σε μια συγκεκριμένη διεύθυνση της μνήμης, για να εκτελεστεί η υπο-ρουτίνα ειδικής εξυπηρέτησης διακοπής. Μόλις ολοκληρωθεί αυτή η διαδικασία, επαναφέρεται το πρόγραμμα στο σημείο από όπου έγινε η διακοπή.

- **Timers**

  Οι Χρονιστές είναι εξειδικευμένα περιφερειακά κυκλώματα, που αυξάνουν ή μειώνουν την τιμή ενός μετρητή, δημιουργώντας χρονικές καθυστερήσεις με τη συχνότητα του ρολογιού. Η τιμή του μετρητή μεγαλώνει ή μικραίνει κατά μια μονάδα σε κάθε κύκλο μηχανής και είναι αρκετά ευέλικτη με την δυνατότητα προγραμματισμού διαιρέτη (ProgrammablePrescaler). Μια εξωτερική πηγή χρονισμού, χρησιμοποιείται για τον συγχρονισμό με την εσωτερική φάση του ρολογιού. Μερικοί χρονιστές λειτουργούν είτε σαν μετρητές (Counter), έτσι ώστε να αυξάνεται η τιμή τους με κάθε ανοδικό ή καθοδικό παλμό, από έναν ορισμένο εξωτερικό ακροδέκτη.

- **Register**

  Τα AVR έχουν 32 καταχωρητές ενός byte και ταξινομούνται ως συσκευές RISC 8 bit. Στις παραλλαγές tinyAVR και megaAVR της αρχιτεκτονικής AVR, οι καταχωρητές εργασίας αντιστοιχίζονται ως οι πρώτες 32 διευθύνσεις μνήμης (000016–001F16), ακολουθούμενες από 64 καταχωρητές I/O (002016–005F16). Σε συσκευές με πολλά περιφερειακά, αυτοί οι καταχωρητές ακολουθούνται από 160 καταχωρητές "εκτεταμένης εισόδου/εξόδου", προσβάσιμοι μόνο ως I/O με αντιστοίχιση μνήμης (006016–00FF16). Η πραγματική SRAM ξεκινά μετά από αυτές τις ενότητες του καταχωρητή, στη διεύθυνση 006016 ή, σε συσκευές με "εκτεταμένη I/O", στο 010016. Η κατά πολύ μικρότερη από τις παραλλαγές tinyAVR χρησιμοποιεί μειωμένη αρχιτεκτονική με μόνο 16 καταχωρητές, οι οποίοι δεν είναι διευθυνσιοδοτούμενοι ως θέσεις μνήμης. Η μνήμη I/O ξεκινά στη διεύθυνση 000016, ακολουθούμενη από τη SRAM. Επιπλέον, αυτές οι συσκευές έχουν μικρές αποκλίσεις από το τυπικό σετ εντολών AVR. Πιο συγκεκριμένα, οι οδηγίες άμεσης φόρτωσης/αποθήκευσης (LDS/STS) έχουν μειωθεί από 2 λέξεις (32 bit) σε 1 λέξη (16 bit), περιορίζοντας τη συνολική άμεση διευθυνσιοδοτούμενη μνήμη (το άθροισμα και των δύο I/O και SRAM) σε 128 byte.

  Αντίθετα, ο χώρος διευθύνσεων 16-bit της έμμεσης εντολής φόρτωσης (LD) επεκτείνεται για να περιλαμβάνει επίσης μη πτητική μνήμη, όπως Flash και bit διαμόρφωσης. Επομένως, η εντολή LoadProgram Memory (LPM) δεν είναι απαραίτητη και παραλείπεται. Στην παραλλαγή XMEGA, το αρχείο μητρώου εργασίας δεν αντιστοιχίζεται στο χώρο διευθύνσεων δεδομένων. Ως εκ τούτου, δεν είναι δυνατό να αντιμετωπιστεί κανένας από τους καταχωρητές εργασίας του XMEGA σαν να ήταν SRAM. Από την άλλη πλευρά, οι καταχωρητές εισόδου/εξόδου αντιστοιχίζονται στον χώρο διευθύνσεων δεδομένων ξεκινώντας από την αρχή του χώρου διευθύνσεων. Επιπλέον, η ποσότητα του χώρου διευθύνσεων δεδομένων που αφιερώνεται σε καταχωρητές εισόδου/εξόδου έχει αυξηθεί σημαντικά στα 4096 byte (000016–0FFF16). Όπως και με τις προηγούμενες γενιές, ωστόσο, οι οδηγίες γρήγορης χειρισμού I/O μπορούν να φτάσουν μόνο τις πρώτες 64 θέσεις καταχωρίσεων I/O (οι πρώτες 32 θέσεις για οδηγίες bitwise). Μετά τους καταχωρητές εισόδου/εξόδου, η σειρά XMEGA θέτει στην άκρη ένα εύρος 4096 byte του χώρου διευθύνσεων δεδομένων, το οποίο μπορεί να χρησιμοποιηθεί

προαιρετικά για την αντιστοίχιση του εσωτερικού EEPROM στον χώρο διευθύνσεων δεδομένων (100016–1FFF16). Η πραγματική SRAM βρίσκεται μετά από αυτές τις περιοχές, ξεκινώντας από το 200016.

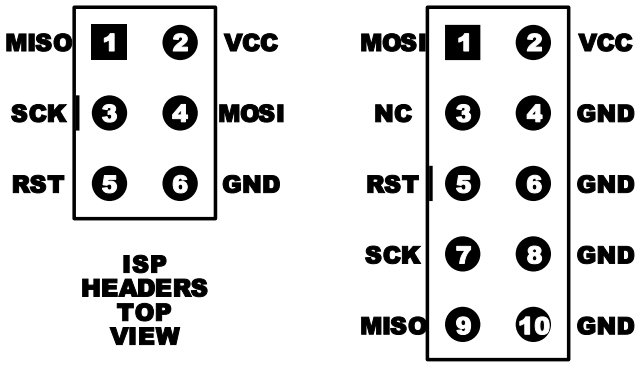
- **Ταλαντωτής**
  Η Μονάδα Ταλαντωτή (OscillatorModule), έχει μια μεγάλη ποικιλία από χαρακτηριστικά για τη ρύθμιση του ρολογιού, μεγιστοποιώντας την απόδοση και ελαχιστοποιώντας την καταναλωμένη ενέργεια. Ως πηγή ρολογιού, ορίζεται ένα εξωτερικό κύκλωμα (ExternalClockSource), δομημένο από ταλαντωτή των 16MHz, με δύο κεραμικούς πυκνωτές 22pF . Στο atmega328p έχει εσωτερικούς ταλαντωτές έως 8Mhz με τη δυνατότητα έως 16Mhz με εξωτερικό. Συνήθως γίνεται χρήση εξωτερικού ταλαντωτή για μεγαλύτερη αξιοπιστία λόγω drift του εσωτερικού συστήματος ταλάντωσης.

4.3. Διεπαφές προγραμματισμού

Υπάρχουν πολλά μέσα για να φορτώσετε τον κώδικα προγράμματος σε ένα τσιπ AVR. Οι μέθοδοι προγραμματισμού των AVR διαφέρουν από οικογένεια AVR σε οικογένεια. Οι περισσότερες από τις μεθόδους που περιγράφονται παρακάτω χρησιμοποιούν τη γραμμή RESET για να εισέλθουν σε λειτουργία προγραμματισμού. Προκειμένου να αποφευχθεί η τυχαία είσοδος του τσιπ σε τέτοια λειτουργία, συνιστάται να συνδέσετε μια αντίσταση έλξης μεταξύ του ακροδέκτη RESET και του θετικού τροφοδοτικού.

4.3.1. SPI (Serial PeripheraInterface)

Το SPI Είναι μία διεπαφή σύγχρονης σειριακής επικοινωνίας που χρησιμοποιείται για μικρής εμβέλειας εφαρμογές, κυρίως στα ενσωματωμένα συστήματα. Η διεπαφή αυτή παρέχει full-duplex λειτουργία χρησιμοποιώντας τη master-slave αρχιτεκτονικής με έναν και μόνο master. Η συσκευή Master (δηλαδή ο ελεγκτής) δημιουργεί το “πλαίσιο” για ανάγνωση και εγγραφή. Πολλαπλές slave συσκευές μπορούν να υποστηριχτούν χρησιμοποιώντας τη λειτουργία CS (Chip-select) της εκάστοτε slave συσκευής. Το CS αναφέρεται πολλές φορές και σαν SS (Slave-select). Το SPI μπορεί να αναφερθεί και σαν δίαυλος τεσσάρων καλωδίων (four-wirebus).



Εικόνα 4.1 : Ακροδέκτες περιφερειακού SPI

Η διεπαφή SPI παρέχει τις παρακάτω συνδέσεις/σήματα:

- SCLK : Serial Clock (αποτο master)
- MOSI : Master-Out-Slave-In (αποτο master)
- MISO : Master-In-Slave-Out (αποτον slave)
- CS ή SS : Chip ή SlaveSelect ,συνήθως λογικό '0' (από τον master για ένδειξη εισερχόμενων δεδομένων προς τον slave)

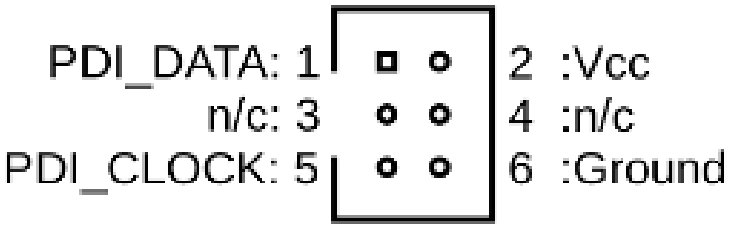
Άλλες πιθανές ονομασίες των παραπάνω είναι :

- SCLK - SCK, SCLK, CLK, SCL
- MOSI - SIMO, MTSR : γιασυσκευές master και slave, συνδέονταιμεταξύτους
  - SDI,DI,DIN,SI: γιασυσκευές slave
  - SDO,DO,DOUT,SO : γιασυσκευές master
- MISO - SOMI, MRST :γιασυσκευές master και slave, συνδέονταιμεταξύτους
  - SDO, DO, DOUT, SO : γιασυσκευές slave

  -SDI, DI, DIN, SI :γιασυσκευές master
- CS- SS, SS, SSEL, NSS, /SS, SS# (slave select)
  - CS, CS (chip select)
  - CSN (chip select/enable)
  - CE (chip enable)

4.3.2. PDI (Program and Debug Interface)

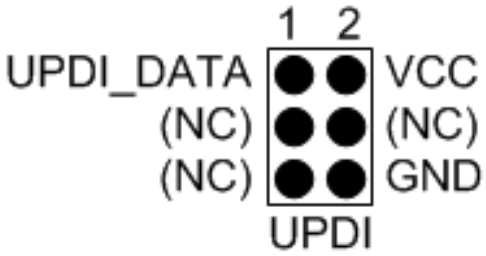
Η διεπαφή προγράμματος και εντοπισμού σφαλμάτων (PDI) είναι για το προγραμματισμό και εντοπισμό σφαλμάτων σε τσιπ συσκευών XMEGA. Το PDI υποστηρίζει προγραμματισμό υψηλής ταχύτητας όλων των χώρων της μη πτητικής μνήμης (NVM), Flash, EEPROM, Fuseskαι Lock-bits. Αυτό γίνεται με την πρόσβαση στον ελεγκτή XMEGA NVM μέσω της διεπαφής PDI και την εκτέλεση εντολών του ελεγκτή της NVM. Το PDI είναι μια διεπαφή 2 ακίδων που χρησιμοποιεί την ακίδα επαναφοράς για είσοδο ρολογιού (PDI_CLK) και μια ειδική ακίδα δεδομένων (PDI_DATA) για είσοδο και έξοδο.



Εικόνα 4.2 : Ακροδέκτες περιφερειακού PDI

4.3.3. UPDI

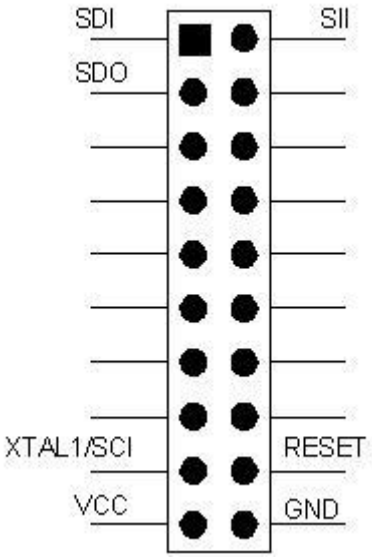
Το UnifiedProgram and DebugInterface (UPDI) είναι μια διεπαφή ενός καλωδίου για εξωτερικό προγραμματισμό και εντοπισμό σφαλμάτων σε τσιπ νεότερων συσκευών ATtiny και ATmega. Το Atmel-ICE και το PICkit 4 είναι ικανά να προγραμματίσουν τσιπ με UPDI. Είναι επίσης δυνατή η χρήση ενός Arduino χάρη στο jtag2updi, ή ενός τυπικού προσαρμογέα USB-UART με την ακίδα TX και RX βραχυκυκλωμένη από μια αντίσταση 1 kΩ και το βοηθητικό πρόγραμμα υπολογιστή pymcuprog που παρέχεται από τη Microchip



Εικόνα 4.3 : Ακροδέκτες περιφερειακού UPDI

4.3.4. High-Voltage Serial Programming

Ο σειριακός προγραμματισμός υψηλής τάσης (HVSP) είναι ουσιαστικά η εφεδρική λειτουργία σε μικρότερα AVR. Ένα πακέτο AVR 8 ακίδων δεν αφήνει πολλούς μοναδικούς συνδυασμούς σημάτων για να τοποθετήσει το AVR σε λειτουργία προγραμματισμού. Ένα σήμα 12 Volt, ωστόσο, είναι κάτι που το AVR πρέπει να βλέπει μόνο κατά τον προγραμματισμό και ποτέ κατά την κανονική λειτουργία. Η λειτουργία υψηλής τάσης μπορεί επίσης να χρησιμοποιηθεί σε ορισμένες συσκευές όπου η ακίδα επαναφοράς έχει απενεργοποιηθεί από ασφάλειες.



Εικόνα 4.4 : Ακροδέκτες περιφερειακού HVSP της Microchip, ξαιρούνται τα μοντέλαATtiny24/44/84.

4.3.5. High-voltage parallel programming

Ο παράλληλος προγραμματισμός υψηλής τάσης (HVPP) θεωρείται η «τελική λύση» και μπορεί να είναι ο μόνος τρόπος για να διορθωθούν οι κακές ρυθμίσεις που έχει γίνει είδη στα Fuses ενός AVR.

4.3.6. Bootloader

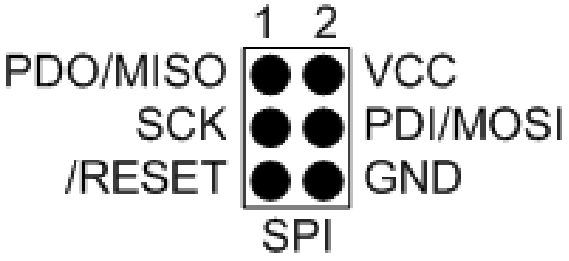
Τα περισσότερα μοντέλα AVR μπορούν να δεσμεύσουν μια περιοχή φορτωτή εκκίνησης, 256 byte έως 4 KB, όπου μπορεί να βρίσκεται ο κώδικας επαναπρογραμματισμού. Κατά την επαναφορά, ο φορτωτής εκκίνησης εκτελείται πρώτα και αποφασίζει, όντας προγραμματισμένο από τον χρήστη εάν θα επαναπρογραμματιστεί ή θα μεταβεί στην κύρια εφαρμογή. Ο κώδικας μπορεί να επαναπρογραμματιστεί μέσω οποιασδήποτε διαθέσιμης διεπαφής ή θα μπορούσε να διαβάσει ένα κρυπτογραφημένο δυαδικό αρχείο μέσω ενός προσαρμογέα Ethernet όπως το PXE.

4.4. Debugging interface

Το Debugging είναι ευρέως διαδεδομένο στο χώρο της πληροφορικής. Στα περισσότερα λογισμικά ανάπτυξης εφαρμογών είναι από μόνο του ενεργοποιημένο χωρίς κάποια ρύθμιση του προγράμματος και δίνει τη δυνατότητα στο χρήστη να σταματήσει την εκτέλεση του προγράμματος σε μία προκαθορισμένη θέση, που έχει θέσει ο χρήστης, να αναγνωρίσει αλλαγές σε μεταβλητές, να προχωρήσει βήμα-βήμα τον κώδικα και πολλά άλλα.

4.4.1. debugWIRE

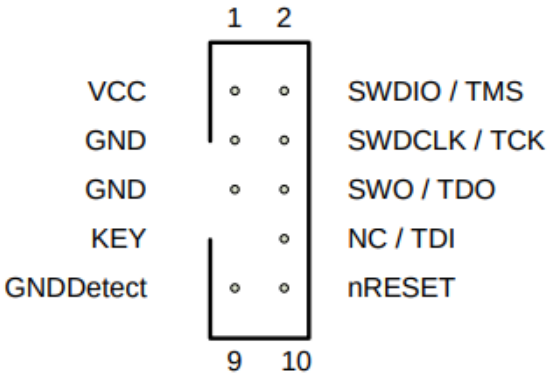
Το debugWIRE είναι η λύση της Atmel για την παροχή δυνατοτήτων εντοπισμού σφαλμάτων on-chip μέσω ενός μόνο ακροδέκτη μικροελεγκτή. Είναι ιδιαίτερα χρήσιμο για εξαρτήματα χαμηλότερου αριθμού ακίδων που δεν μπορούν να παρέχουν τις τέσσερις «εφεδρικές» ακίδες που απαιτούνται για το JTAG. Τα JTAGICE mkII, mkIII και AVR Dragon υποστηρίζουν debugWIRE. Το debugWIRE αναπτύχθηκε μετά την αρχική έκδοση του JTAGICE και τώρα το υποστηρίζουν οι κλώνοι του.



Εικόνα 4.5 : Ακροδέκτες περιφερειακού SPI

4.4.2. JTAG

Η δυνατότητα Joint TestActionGroup (JTAG) παρέχει πρόσβαση στη λειτουργία εντοπισμού σφαλμάτων στο chip, με την ταυτόχρονη εκτέλεση του κώδικα στο σύστημα προορισμού. Το JTAG επιτρέπει την πρόσβαση στην εσωτερική μνήμη και τους καταχωρητές, τον ορισμό σημείων διακοπής στον κώδικα και την εκτέλεση με ένα βήμα για την παρατήρηση της συμπεριφοράς του συστήματος. Το JTAG μπορεί επίσης να χρησιμοποιηθεί για την εκτέλεση μιας δοκιμής σάρωσης ορίων, που ελέγχει τις ηλεκτρικές συνδέσεις μεταξύ AVR και άλλων τσιπ με δυνατότητα σάρωσης ορίων σε ένα σύστημα. Η σάρωση ορίων είναι κατάλληλη περισσότερο για μια γραμμή παραγωγής, παρά για τη χρήση από έναν χομπίστα που είναι καλύτερα να δοκιμάσει την εκτέλεση με ένα πολύμετρο ή ένα παλμογράφο.

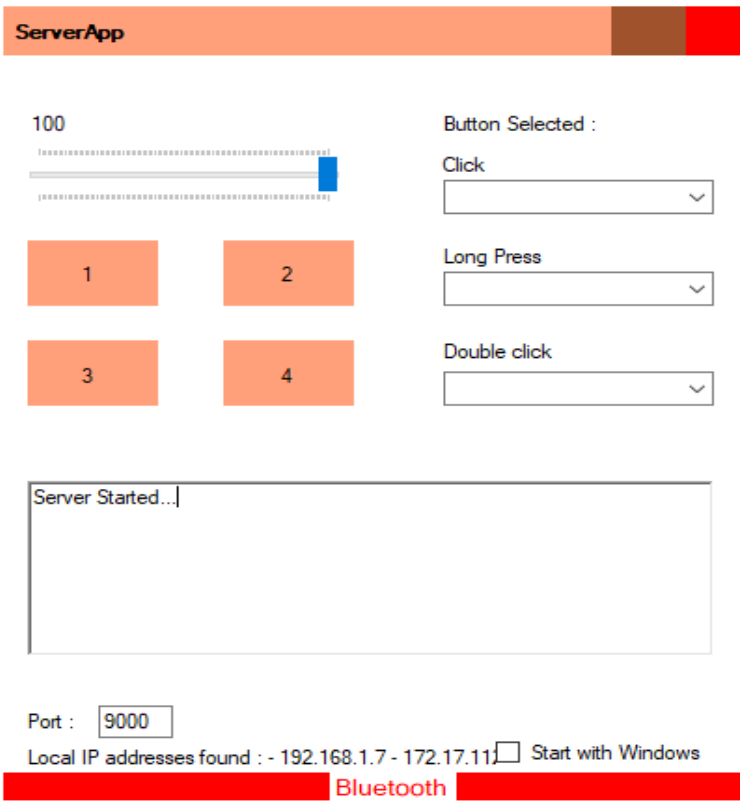


Εικόνα 4.6 : Ακροδέκτες περιφερειακού JTAG

Κεφάλαιο 5ο - Ανάλυση Λειτουργίας Εφαρμογών

5.1. Ανάλυση εφαρμογής διαχείρισης σε περιβάλλον Windows

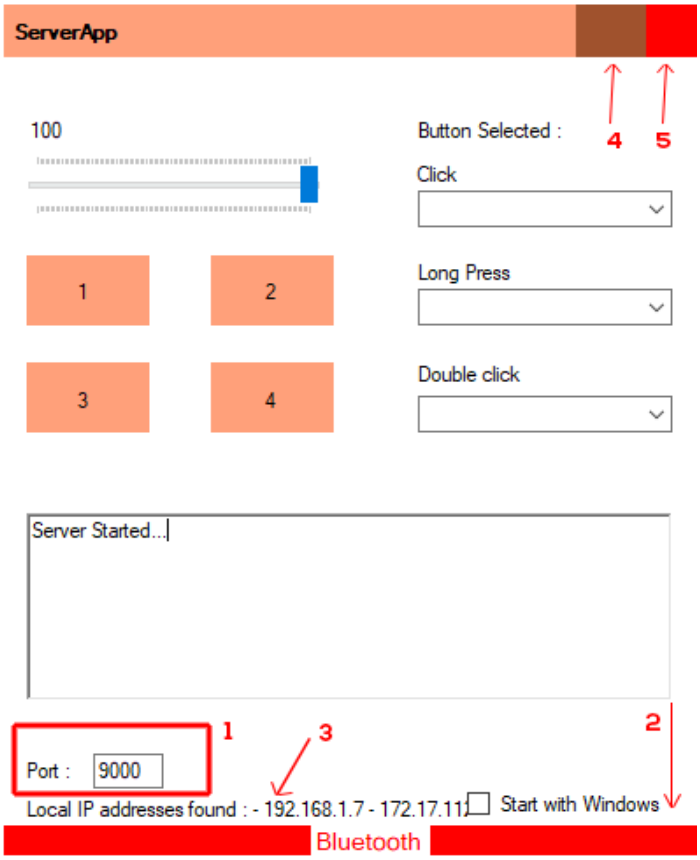
Η εφαρμογή του τίτλου εξυπηρετεί το ρόλο του «Server» μεταξύ των συσκευών χειρισμού, τόσο για τη συσκευή σε περιβάλλον «Android» όσο και στο «φυσικό» τηλεχειριστήριο. Η αρχή λειτουργίας τη εφαρμογής είναι η λήψη της εκάστοτε εντολής από την όποια συσκευή, στην αποδικοποίηση και στην εκτέλεση της εντολής στο περιβάλλον των Windows. Η εφαρμογή χρησιμοποιεί ταυτόχρονα δύο τεχνολογίες ασύρματης επικοινωνίας. Η πρώτη είναι η επικοινωνία μέσω τοπικού δικτύου (LAN) και η δεύτερη η επικοινωνία με χρήση Bluetooth, όπου η πρώτη αφορά την εφαρμογή Android και η δεύτερη το τηλεχειριστήριο. Για την ανάπτυξη και την αποσφαλμάτωση της εφαρμογής έγινε χρήση της γλώσσας προγραμματισμού C# WinForms σε συνδυασμό με το Visual Studio IDE.



Εικόνα 5.1 : Εφαρμογή Κόμβος

5.2. Ρυθμίσεις επικοινωνίας

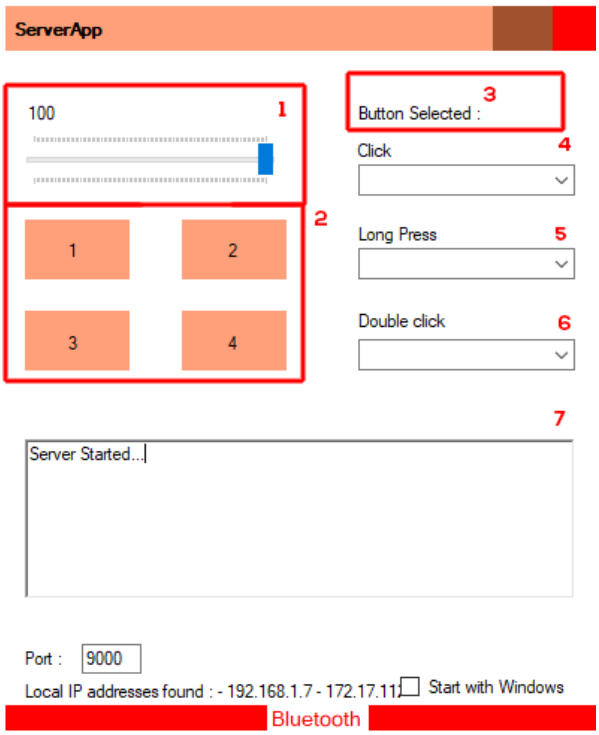
Η εφαρμογή λειτουργεί σχεδόν αυτόματα αμέσως μετά το πρώτο άνοιγμα της. Οι ρυθμίσεις που χρειάζονται είναι ελάχιστες και ποιο πολύ έχουν να κάνουν με το ποιες λειτουργίες επιθυμεί ο χρήστης να εκτελεί απομακρυσμένα. Η εφαρμογή αποτελείται από μία σελίδα που περιέχει ότι χρειάζεται ο χρήστης (Εικόνα 5.2). Στο κάτω μέρος της εφαρμογής έχει το κουτί επιθυμητής πόρτα (1) το οποίο πρέπει να είναι το ίδιο με της επιλεγμένη πόρτα στην Android εφαρμογή (Ενότητα 5.4.1). Ακριβώς από κάτω είναι η τοπική διεύθυνση IPv4 του Η/Υ που επίσης πρέπει να γίνει εισαγωγή της στις ρυθμίσεις της Android εφαρμογής. Στο άκρο κάτω μέρος της συσκευής είναι η μπάρα ένδειξης σύνδεσης με τη συσκευή Bluetooth. Όταν είναι κόκκινη δεν υπάρχει σύνδεση με τη συσκευή και αντίθετα όταν είναι πράσινο, έχει γίνει επιτυχή σύνδεση με τη συσκευή. Στο ίδιο σημείο υπάρχει και η επιλογή «Start with Windows» η οποία όταν είναι επιλεγμένη γίνεται αυτόματη εκτέλεση της εφαρμογής με το άνοιγμα του περιβάλλοντος των Microsoft Windowsκαι τέλος στο πάνω δεξιά μέρος με πράσινο (4) συμβολίζεται η ελαχιστοποίηση της εφαρμογής και με κόκκινο (5) ο τερματισμός της εφαρμογής.



Εικόνα 5.2 : Ρυθμίσεις Εφαρμογής Κόμβος

5.3. Παραμετροποίηση λειτουργιών

Εφόσον έχουν γίνει οι απαραίτητες ρυθμίσεις για την ετοιμασία της εφαρμογής στην Εικόνα 5.2 φαίνονται οι επιλογές για τη προσωποποίηση της εφαρμογής. Η μπάρα ήχου (1) εξυπηρετεί περισσότερο σαν ένδειξη για την επιτυχή αλλαγή της έντασης μέσω των συσκευών χειρισμού. Ακριβώς από κάτω απεικονίζονται τέσσερεις εικόνες που αναπαριστούν τα πλήκτρα των τηλεχειριστηρίων. Κάνοντας «κλικ» σε ένα από αυτά αμέσως έχουμε πρόσβαση στις ρυθμίσεις του συγκεκριμένου πλήκτρου. Η ένδειξη «Button Selected :" (3) εμφανίζει το νούμερο από το επιλεγμένο πλήκτρο **(**στη**ν** παρακάτω εικόνα δεν έχει γίνει κάποια επιλογή πλήκτρου για αυτό δεν αναγράφεται ο αριθμός κάποιου πλήκτρου**)**. Στη συνέχεια στα «drop-down menu" (4,5,6) μπορεί να γίνει εύκολα η επιλογή της επιθυμητής δράσης του πλήκτρου ανάλογα με τρόπο πατήματος του. Με την αλλαγή όποιας επιλογής, η λειτουργία αποθηκεύεται αυτόματα ώστε να είναι γνωστή στο πρόγραμμα κάθε φορά που εκτελείται.



Εικόνα 5.3 : Παραμετροποίηση λειτουργιών εφαρμογής

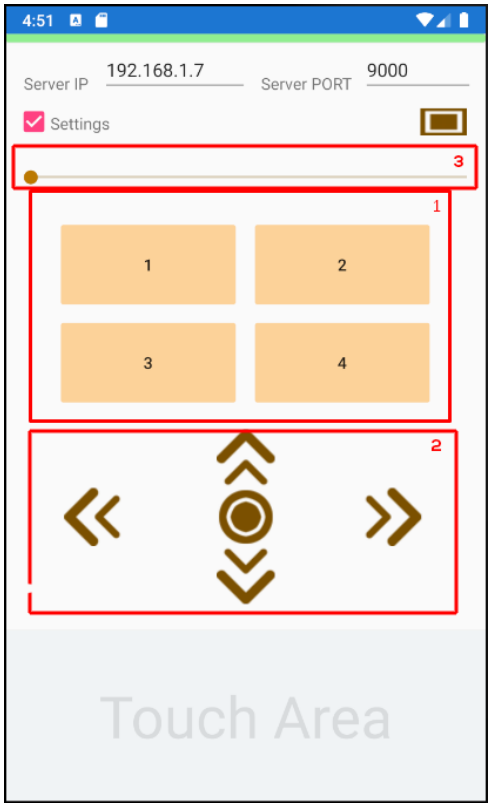
Η εφαρμογή διαθέτει μία αρκετά διευρυμένη μένη λίστα λειτουργιών που μπορούν να επιλεχτούν για απομακρυσμένη εκτέλεση. Ωστόσο, στη περίπτωση που ο χρήστης χρειάζεται να εκτελέσει μία λειτουργία, η οποία δεν υπάρχει μέσα στη λίστα, δίνεται η επιλογή να εισάγει χειροκίνητα την εντολή σε μορφή εντολής **όπου** χρησιμοποιείται στο περιβάλλον του Command Prompt (CMD). Με αυτό τον τρόπο μπορεί να επιτευχθεί η πλήρη απομακρυσμένη χρήση του περιβάλλοντος Windows ανάλογα με τις ανάγκες του χρήστη, όσο ιδιαίτερη και να είναι η απομακρυσμένη λειτουργία που απαιτείται.

5.4. Ανάλυση εφαρμογής τηλεχειρισμού σε περιβάλλον Android

Η εφαρμογή στο περιβάλλον Android λειτουργεί σαν τηλεχειριστήριο. Η εφαρμογή αναλαμβάνει το ρόλο του Client που συνδέεται μέσω LAN στην Server εφαρμογή. Η εφαρμογή αποτελείται από δύο σελίδες. Η πρώτη είναι η αρχική (Εικόνα 5.4), όπου υπάρχουν όλα τα απαραίτητα πλήκτρα για τη λειτουργία της. Τα ρυθμιζόμενα πλήκτρα (1) είναι ίδιας λογικής με αυτά της φυσικής συσκευής και έχουν την ίδια λογική λειτουργίας με τα πλήκτρα ενός φυσικού τηλεχειριστηρίου. Ακριβώς από κάτω είναι κάποια προκαθορισμένα πλήκτρα (2) που εκτελούν λειτουργίες ίδιες με αυτές που πραγματοποιούν τα «βελάκια» στο πληκτρολόγιο και το κεντρικό πλήκτρο πραγματοποιεί τη λειτουργία του «space». Ακριβώς από πάνω φαίνεται μία μπάρα (3), με τη οποία γίνεται ρύθμιση της έντασης του ήχου στον Η/Υ. Το κεντρικό πλήτρο (2) δέχεται τις παρακάτω εντολές:

- Tap ,λειτουργία αριστερού κλικ
- DoubleTap ,λειτουργία διπλού κλικ
- LongPress ,λειτουργία δεξί κλικ
- Slide ,λειτουργία μετακίνησης του κέρσορα

Αξίζει να σημειωθεί ότι οι παραπάνω λειτουργίες , κυρίως του ‘Slide” εξαρτώνται πολύ από τη ταχύτητα του δικτύου για **να είναι εφικτή η** καλύτερη δυνατή εμπειρία του χρήστη.



Εικόνα 5.4 : Πλήκτρα διεπαφής εφαρμογής Android

Όπως αναφέρθηκε παραπάνω η εφαρμογή για την επικοινωνία χρησιμοποιεί το τοπικό δίκτυο (LAN). Για την δημιουργία της εφαρμογής απαιτείται η χρήση των γλωσσών προγραμματισμού “Xamarin” και “C#” σε συνδυασμό με την εφαρμογή Visual Studio IDE, τόσο για την ανάπτυξη όσο και για την αποσφαλμάτωση του κώδικα.

5.4.1. Ρυθμίσεις επικοινωνίας εφαρμογής Android

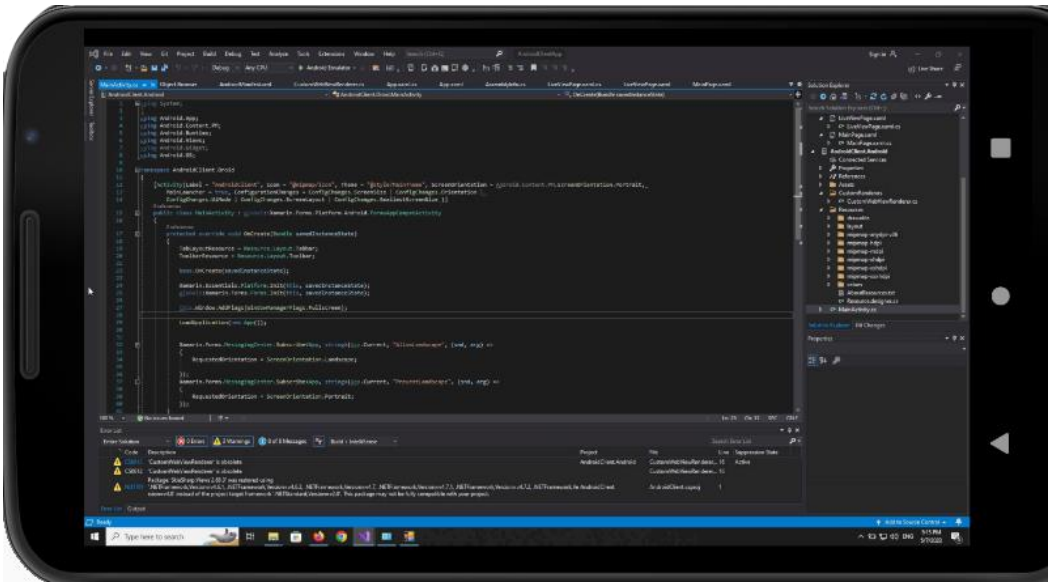
Για να είναι δυνατή η επικοινωνία μεταξύ των δύο εφαρμογών μέσω των δύο εφαρμογών στα δύο λογισμικά, είναι απαραίτητη η μερική ρύθμιση της εφαρμογής. Για να εμφανιστούν οι ρυθμίσεις σύνδεσης της εφαρμογής στον Η/Υ πρέπει το κουτί (1) να είναι επιλεγμένο. Μετά πρέπει να πληκτρολογηθεί η σωστή διεύθυνση IPv4 (2)του Η/Υ, καθώς και η πόρτα επικοινωνίας (3). Έπειτα, εφόσον είναι επιθυμητό, το κουτί επιλογής μπορεί να αλλάξει κατάσταση.

Εικόνα 5.5 : Ρυθμίσεις εφαρμογής Android

Για την αποφυγή ρύθμισης της διεύθυνσης IPv4 ξανά μπορεί να γίνει ρύθμισης του περιβάλλοντος Windows σε χρήση στατικής διεύθυνσης IP (static IP address).Με αυτό τον τρόπο μπορεί να επιτευχθεί η αποφυγή αλλαγής της IP διεύθυνσης των Windows, που θα είχε ως αποτέλεσμα την αποτυχία σύνδεσης της εφαρμογής “Client” στο «Server».

5.4.2. Ζωντανή απεικόνιση οθόνης Server

Με τη δυνατότητα της ζωντανής απεικόνισης της οθόνης μπορεί ο χρήστης να έχει άμεση γνώση για το τι γίνεται στην οθόνη του υπολογιστή άμεσα, ακόμα και χωρίς να υπάρχει οπτική επαφή με την οθόνη. Η προϋπόθεση για να μπορεί να γίνει χρήση αυτής της λειτουργίας είναι παράλληλα με την εφαρμογή του Client να εκτελείται και η Server εφαρμογή στο περιβάλλον του Windows. Με αυτό τον τρόπο με το που γίνει η εκτέλεση της εφαρμογής Client στο Android φαίνεται στο κάτω μέρος της οθόνης η μικρογραφία της οθόνης του Server. Με την επιλογή του εικονιδίου ακριβώς κάτω από την εισαγωγή πόρτας δικτύου, μπορεί ο χρήστης να βλέπει σε πλήρες μέγεθος την οθόνη του Server.

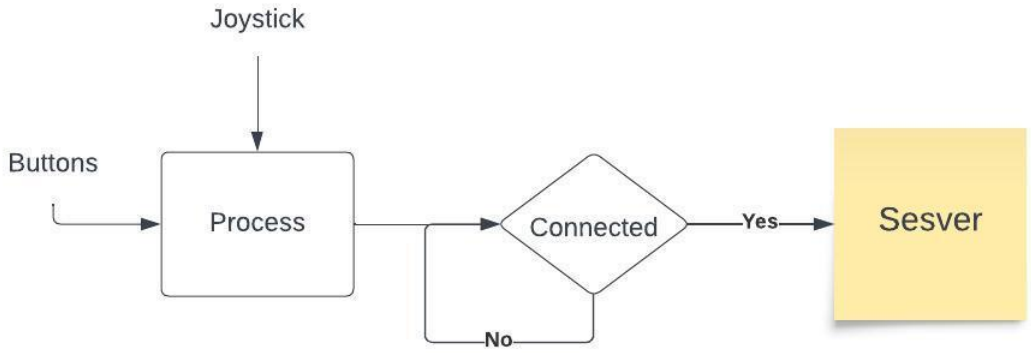


Εικόνα 5.6 : Ζωντανή απεικόνιση Η/Υ από εφαρμογή Android

Η οθόνη αυτή επιτρέπει και την χρήση Gestures , για τη μετακίνηση του κέρσορα καθώς προβάλλεται η ζωντανή προβολή της οθόνης του Η/Υ. Τα Gestures που επιτρέπονται είναι τα : Tap, Double Tap, Long Press, Track touch.

Κεφάλαιο 6ο - Ανάλυση συσκευής τηλεχειριστηρίου

Το φυσικό τηλεχειριστήριο ακολουθεί την ίδια λογική με όλα τα ασύρματα χειριστήρια, ως προς την αρχή λειτουργίας. Με το πάτημα ενός πλήκτρου ενεργοποιείται μία διαδικασία αποστολής της εκάστοτε εντολής που είναι ανατεθειμένη στο κάθε πλήκτρο και αποστέλλεται μέσω του ασύρματού πρωτοκόλλου που έχει επιλεχθεί. Στη παρούσα συσκευή το πρωτόκολλο αυτό είναι το Bluetooth, το οποίο έχει επιλεχθεί τόσο λόγω του ότι δε χρειάζεται να υπάρχει οπτική επαφή με το δέκτη, όσο και λόγω της ευκολίας που παρέχει αυτό το πρωτόκολλο. Επισημαίνεται ότι σαν όψη η συσκευή δεν έχει ιδιαίτερη διαφορά από οποιαδήποτε άλλη αντίστοιχη συσκευή.



Εικόνα 6.1 : Διάγραμμα ροής σύνδεσης φυσικής συσκευής στην Εφαρμογή Κόμβος

Εικόνα 6.2 : Κάτω όψη πλακέτας

Εικόνα 6.3 : Πάνω όψη πλακέτας

6.1. Ηλεκτρονικά εξαρτήματα συσκευής

Τα υλικά που αναφέρονται παρακάτω έχουν επιλεχθεί με τέτοιο σκοπό ώστε να εξυπηρετούν μία συγκεκριμένη λειτουργία της συσκευής ή να βοηθούν στην καλύτερη λειτουργία του συστήματος.Τα χαρακτηριστικά αυτών των εξαρτημάτων, εξετάστηκαν προσεκτικά προτού επιλεχθούν ως τελικά υλικά. Αυτό έγινε γιατί αναλόγως με τη χρήση και την ποιότητα τους, προβλέπονται καθοριστικά για την σωστή λειτουργικότητα της συσκευής. Με άλλα λόγια, παίζει σημαντικό ρόλο η κατάλληλη επιλογή ηλεκτρονικών υλικών με οδηγό τα φύλλα δεδομένων (datasheet).

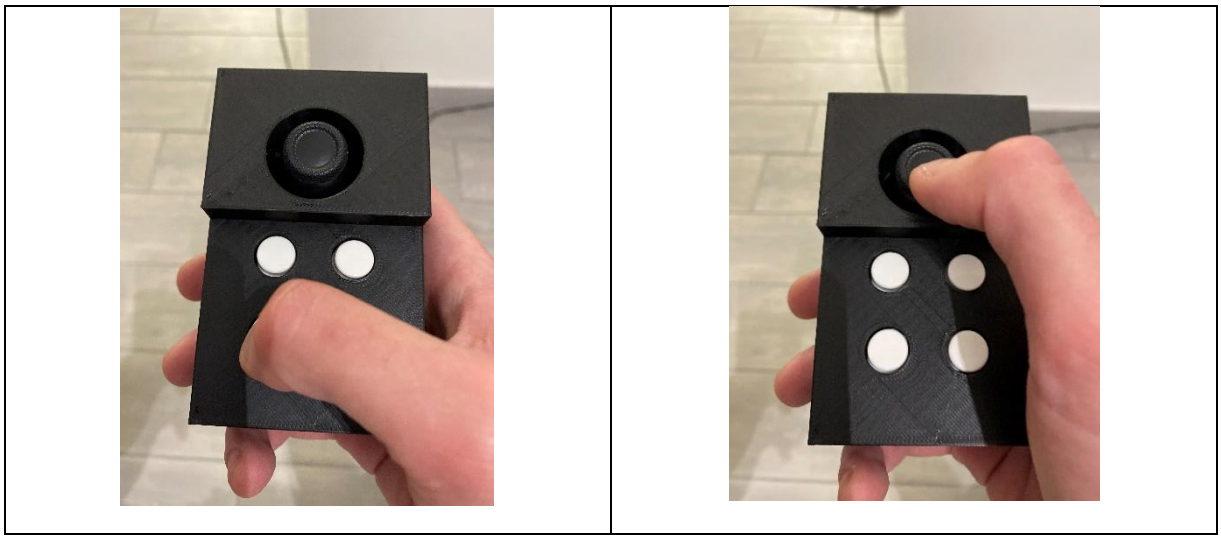
Για τη δημιουργία της συσκευής έγινε χρήση πυκνωτών, αντιστάσεων διαφόρων τιμών, ανάλογα με τη χρήση τους. **Ως** κεντρική μονάδα επεξεργασίας επιλέχθηκε ο μικροελεγκτής ESP32. **Η** διεπαφή χρήστη έγινε **με τη** χρήση μεμονωμένων μπουτόν , όπως και ένα Joystick Module που αναπαριστά το ποντίκι ενός Η/Υ. Τέλος για την επικοινωνία με το περιβάλλον Windows χρησιμοποιήθηκε η κεραία Bluetooth του ESP32. Παρακάτω γίνεται ανάλυση των ποιο σημαντικών στοιχείων που χρησιμοποιήθηκαν.

6.1.1. Μικροελεγκτής ESP32

Ο ESP32 καθίσταται ιδανικός για πολλές εφαρμογές, όπου επιθυμείται χαμηλή κατανάλωση ρεύματος και όσον το δυνατό χαμηλότερο κόστος κατασκευής. Η έκδοση του ESP32-WROOM-32D περιλαμβάνει ενσωματωμένη κεραία για άμεση χρήση WiFi/Bluetooth. Παρέχει πολλαπλά πρωτόκολλα σειριακής επικοινωνίας όπως SPI , UART , USB , I2C κ.α. Ακόμη παρέχει αρκετές λειτουργίες αναστολής λειτουργιών εξοικονόμησης ενέργειας (sleepmodes) καθιστώντας δυνατή την κατανάλωση ρεύματος της τάξης των uA.

6.1.2. Διακόπτες συσκευής

Για τους διακόπτες έγινε χρήση μπουτόν μεγέθους 12mm για πιο εύκολη επαφή στο δάχτυλο του χρήστη και για τη κίνηση του ποντικιού επιλέχθηκε ένα thumb joystick καλής ποιότητας. Όλοι οι διακόπτες τοποθετήθηκαν όσο το δυνατόν καλύτερα για να μπορεί να γίνει χρήση της συσκευής με το ένα χέρι και χωρίς να δυσκολεύεται ο χρήστης στο πάτημα κάποιου διακόπτη.



Εικόνα 6.4 : Λειτουργία πλήκτρων με τον αντίχειρα

6.1.3. Step-downregulator

Για τη σωστή παροχή ρεύματος επιλέχθηκε ο μετατροπέας υποβιβασμού τάσης LD1117-S33 λόγω της χαμηλού drop-out τάσης (το μέγιστο 1V) ρυθμισμένος στα 3,3 Volt, έτσι ώστε να μπορεί να γίνει μετατροπή της τάσης των μπαταριών 4.5V για τη ορθή και χωρίς προβλήματα λειτουργία του κυκλώματος.

6.1.4. Μπαταρία AAA (3xAAA)

Η επιλογή μπαταρίας έγινε με πρώτο παράγοντα τον όσο δυνατό μικρότερο όγκο και έπειτα τη χωρητικότητα. Τρεις μπαταρίες AAA σε συνδεσμολογία σειράς 1.5V/900mAh μπορούν να παρέχουν πολλές μέρες έως και μήνες χρήσης της συσκευής με κανονική χρήση. Η κατανάλωση ρεύματος της συσκευής είναι 45mAh, όπου εγγυάται να δουλεύει συνεχόμενα για σχεδόν είκοσει ώρες χωρίς διακοπή. Σε συνδυασμό με τη συσκευή να εισέρχεται σε λειτουργία ύπνου, δύο λεπτά μετά τη τελευταία χρήση της συσκευής, η διάρκεια χρήσης της συσκευής αυξάνεται κατά πάρα πολύ με κατανάλωση ρεύματος της συσκευής σε κατάσταση ύπνου σχεδόν 10uA.

- Χρόνος λειτουργίας μπαταρίας (hours) = (Χωρητικότητα μπαταρίας (Ah) * Τάση μπαταρίας (V)) / Φορτίο (W)

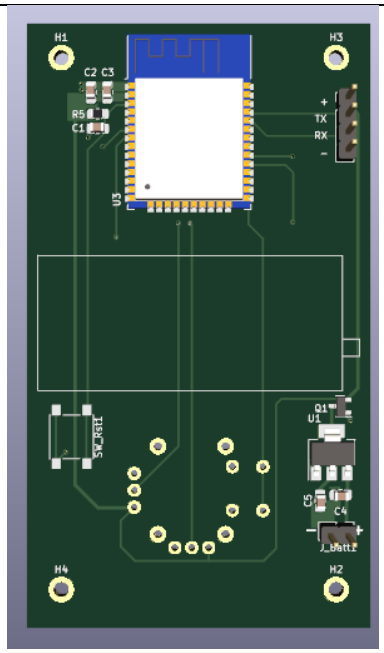
Εικόνα 6.5 : Μπαταριοθήκη συσκευής

Κεφάλαιο 7ο - Προγραμματισμός, συσκευή

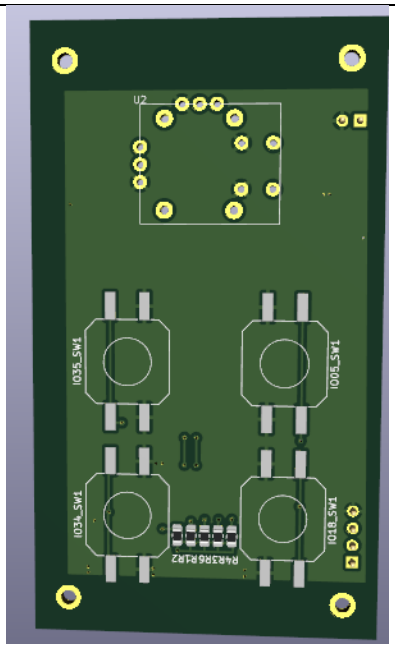
Το πρόγραμμα KiCad είναι μία ολοκληρωμένη σουίτα εφαρμογών σχεδίαση ηλεκτρονικών κυκλωμάτων. Καθίσταται ιδανικό για μία μεγάλη μερίδα πληθυσμού καθώς παρέχει σύγχρονα εργαλεία σχεδίασης και ταυτόχρονα διανέμεται δωρεάν. Χρησιμοποιώντας το KiCad μπορούν να γίνουν όλες οι απαραίτητες εργασίας όπως, ανάπτυξη ηλεκτρονικών κυκλωμάτων, ανάπτυξη τυπωμένων κυκλωμάτων, δημιουργία αποτυπωμάτων (footprints) εξαρτημάτων, δημιουργία και προβολή gerber αρχείων, εξαγωγή .STEP αρχείων και πολλά άλλα. Σε συνδυασμό με τα παραπάνω δύνεται και η δυνατότητα 3D προβολής της πλακέτας και των εξαρτημάτων. Επίσης το πρόγραμμα είναι διαθέσιμο σε όλα τα προηγμένα λογισμικά όπως Windows, Linux, MacOs καθώς και Solaris.

7.1. Εξωτερική όψη και περιγραφή PCB

Τα υλικά που αποτελείται το PCB διανέμονται και στις δύο όψεις. Η πλακέτα αποτελείται από 2 επίπεδα (Layer), όπου στο ένα βρίσκονται τα κύρια κομμάτια του συστήματος όπως ο ESP32 , ο μετατροπέας τάσης και οι μπαταρίες και στο άλλο τα περιφερειακά μπουτόν.



Εικόνα 7.1 : Κάτω όψη πλακέτας (περιβάλλον KiCAD)

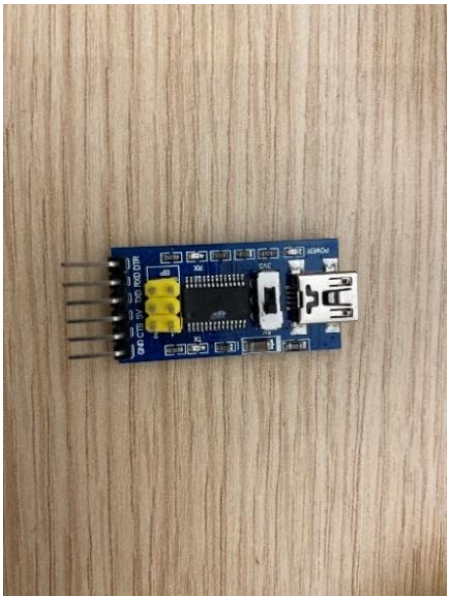


Εικόνα 7.2 : Πάνω όψη πλακέτας (περιβάλλον KiCAD)

7.2. Μέθοδος προγραμματισμού συσκευής

Ο προγραμματισμός έγινε μέσω C/C++ χρησιμοποιώντας το πρόγραμμα Visual Studio Code , χρησιμοποιώντας το Framework ESP-IDF. Ο προγραμματισμός έγινε με χρήση του πρωτοκόλλου UART. Για την επίτευξη αυτού χρησιμοποιήθηκε εξωτερική πλακέτα που περιλαμβάνει το τσιπ FT232RL για τη μετατροπή των δεδομένων από USB σε UART, καθώς τα δεδομένα αποστέλλονται από τον Η/Υ στη προγραμματίστρια πλακέτα μέσω ενός καλωδίου USB.

Εικόνα 7.3 : Ακίδες προγραμματισμού πλακέτας



Εικόνα 7.4 : UARTπλακέτα προγραμματισμού CH340

7.3. Προγραμματισμός εφαρμογών

i. Πως έγινε ο προγραμματισμός

Ο προγραμματισμός και των δύο εφαρμογών έγινε κυρίως με τη γλώσσα C# και συγκεκριμένα για την εφαρμογή του κινητού **σ**τη γλώσσα Xamarin όπου συνδυάζει τη C# με την XML.

Ποια η χρήση τους

ii. Εφαρμογή Windows

Η εφαρμογή Windows αποτελεί στην ουσία τον κεντρικό κόμβο των πληροφοριών που αποστέλλονται από τη συσκευή ή την ψηφιακή συσκευή/εφαρμογή του κινητού. Λαμβάνει τις ανάλογες εντολές και εκτελεί τις επιθυμητές λειτουργίες που έχει προκαθορίσει ο χρήστης μέσω του προγράμματος.

iii. Εφαρμογή Android

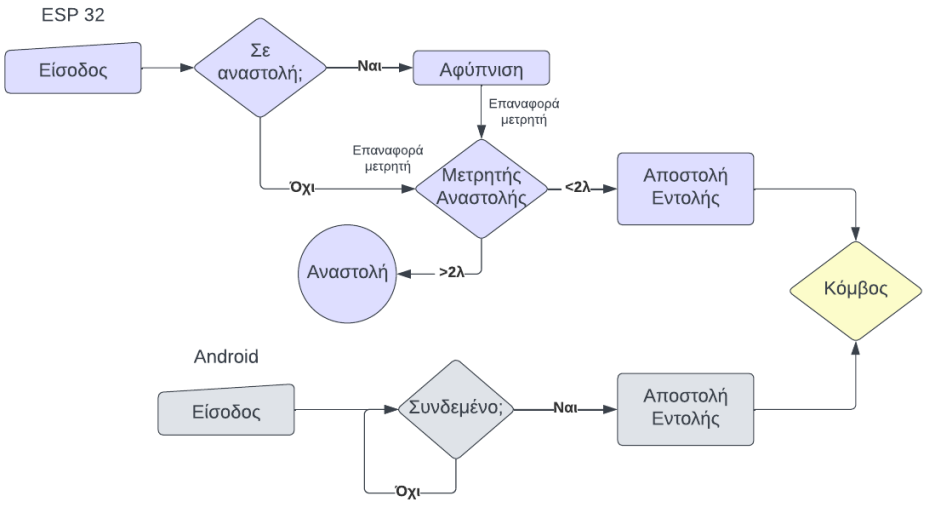
Η εφαρμογή στο περιβάλλον Android είναι επί της ουσίας ένα ψηφιακό τηλεχειριστήριο. Επιπλέον παρέχει και τη δυνατότητα απομακρυσμένης προβολής της οθόνης του περιβάλλοντος Windows όπου εκτελείται η προαναφερθείσα εφαρμογή, έτσι ώστε ο χρήστης να μπορεί να ελέγξει τη κατάσταση του Η/Υ, τη πρόοδο μιας εργασίας κ.α.

iv. Τρόπος επικοινωνίας εφαρμογών

Ο τρόπος επικοινωνίας μεταξύ των δύο εφαρμογών γίνεται με τη χρήση του τοπικού δικτύου (LAN). Με αυτό το τρόπο η επικοινωνία είναι αξιόπιστη σε όποια απόσταση και να βρίσκεται το ψηφιακό-χειριστήριο με μόνη προϋπόθεση να υπάρχει η κάλυψη τοπικού δικτύου.

v. Τρόπος επικοινωνίας εφαρμογής με χειριστήριο

Η διαδικασία επικοινωνίας της συσκευής με την εφαρμογή Windows γίνεται μέσω Bluetooth. Αυτό προαπαιτεί και οι δύο πλευρές της επικοινωνίας να έχουν πρόσβαση σε μία κεραία Bluetooth. Λόγω του περιορισμού της τεχνολογίας Bluetooth η απόσταση μεταξύ των δύο κεραιών δεν μπορεί να είναι πάνω από 10 μέτρα σε περιβάλλον χωρίς παρεμβολές.



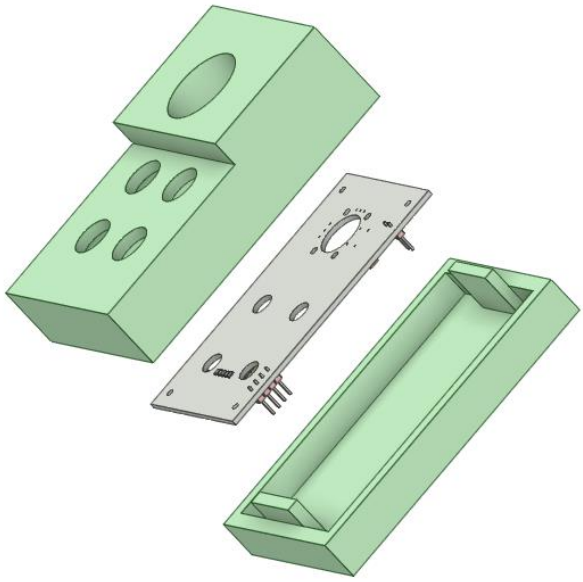
Εικόνα 7.5 : Διάγραμμα ροής συστήματος

Κεφάλαιο 8ο - Δημιουργία κουτιού

8.1. Περιβάλλον CAD

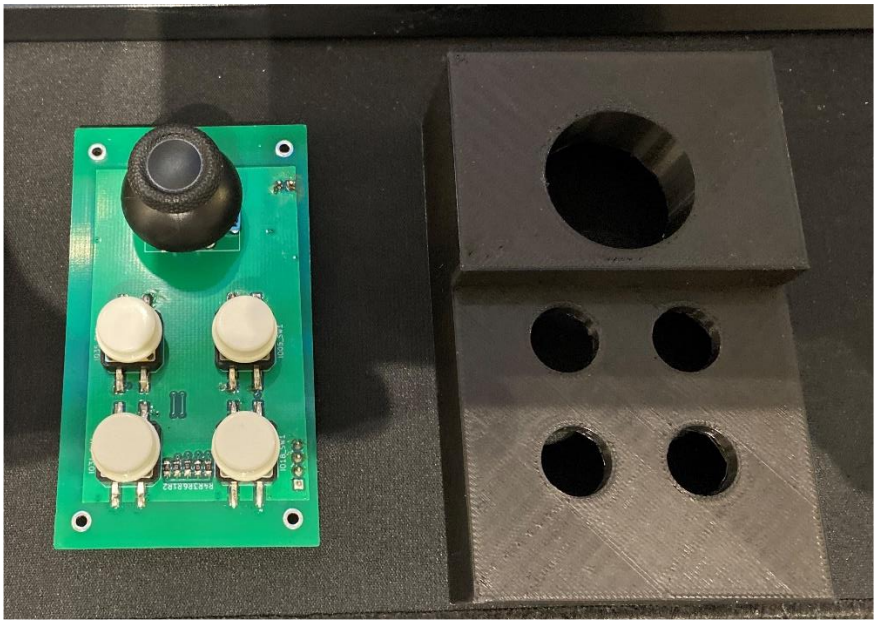
Σχεδίαση με τη βοήθεια υπολογιστή (CAD) είναι η χρήση υπολογιστών (ή σταθμών εργασίας) για να βοηθήσουν στη δημιουργία, τροποποίηση, ανάλυση ή βελτιστοποίηση ενός σχεδίου.[1] Αυτό το λογισμικό χρησιμοποιείται για να αυξήσει την παραγωγικότητα του σχεδιαστή, να βελτιώσει την ποιότητα του σχεδιασμού, να βελτιώσει τις επικοινωνίες μέσω της τεκμηρίωσης και να δημιουργήσει μια βάση δεδομένων για την κατασκευή.[2] Τα σχέδια που γίνονται μέσω λογισμικού CAD είναι χρήσιμα για την προστασία προϊόντων και εφευρέσεων όταν χρησιμοποιούνται σε αιτήσεις διπλωμάτων ευρεσιτεχνίας. Το αρχείο εξόδου CAD έχει συχνά τη μορφή ηλεκτρονικών αρχείων για εκτύπωση, κατεργασία ή άλλες κατασκευαστικές εργασίες.

Το CAD μπορεί να χρησιμοποιηθεί για το σχεδιασμό καμπυλών και σχημάτων σε δισδιάστατο (2D) χώρο. ή καμπύλες, επιφάνειες και στερεά σε τρισδιάστατο (3D) χώρο.



Εικόνα 8.1 : Κομμάτια συναρμολόγησης κουτιού (περιβάλλον DesignSpark Mechanical)

Η κατασκευή του πρωτότυπου κουτιού έγινε έχοντας ως προτεραιότητα την ευκολία χρήσης και την άνετη πρόσβαση στα κουμπιά της συσκευής με τον αντίχειρα του ενός χεριού. Αυτό είχε ως αποτέλεσμα η αισθητική να αποτελεί το τελευταίο παράγοντα σχεδίασης του.



Εικόνα 8.2 : Κομμάτια συναρμολόγησης κουτιού

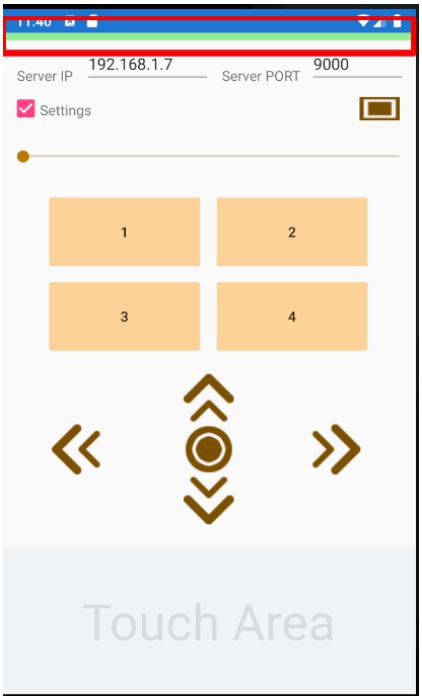
Κεφάλαιο 9ο - Λειτουργίας του συστήματος

Σε αυτό το κεφάλαιο αναφέρεται η τυπική διαδικασία που χρειάζεται να πραγματοποιήσει ο χρήστης, για τη λειτουργία ολόκληρου του συστήματος .

9.1. Ενεργοποίηση εφαρμογής Server

Πριν ξεκινήσει ο χρήστης να χρησιμοποιεί μία ή και τις δύο μεθόδους χειρισμού του Η/Υ πρέπει πρώτα να κάνει ενεργοποίηση «άνοιγμα» της εφαρμογής και τις επιθυμητές ρυθμίσεις στο πρόγραμμα του Server [Ενότητα 5.3] και θα πρέπει να είναι σίγουρος ότι ο Η/Υ διαθέτει Bluetooth ή/και πρόσβαση στο τοπικό δίκτυο.

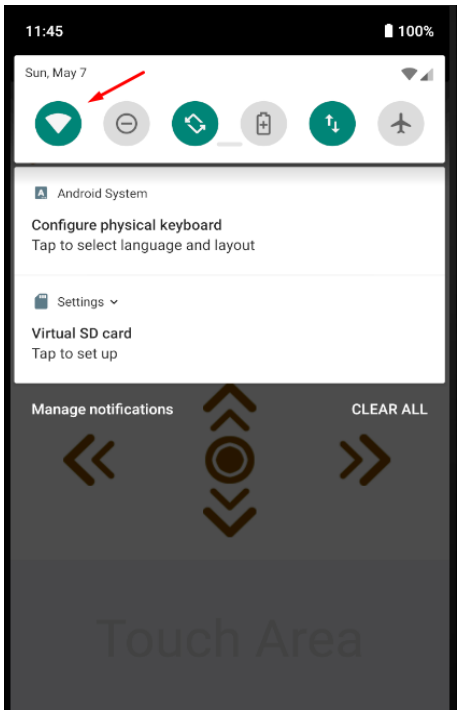
Εικόνα 9.1 : Εφαρμογή Κόμβος σε ετοιμότητα



Εικόνα 9.2 : Εφαρμογή Android συνδεμένη στη Εφαρμογή Κόμβος

9.2. Χειρισμός με εφαρμογή Android

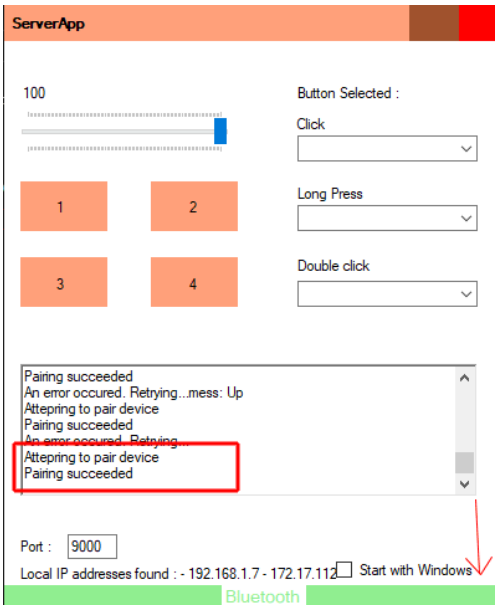
Για να είναι δυνατή η επικοινωνία των δύο εφαρμογών θα πρέπει να γίνει η σωστή εισαγωγή των στοιχείων της Server εφαρμογής [Ενότητα 5.4.1] και να έχει φυσικά πρόσβαση στο τοπικό δίκτυο (Εικόνα 9.3) , όπως αναφέρθηκε και παραπάνω.



Εικόνα 9.3 : Πρόσβαση σε τοπικό δίκτυο στο περιβάλλον Androi

9.3. Φυσικό τηλεχειριστήριο

Όπως αναφέρεται και παραπάνω, οι μόνες ρυθμίσεις που χρειάζονται για τη επικοινωνία των τηλεχειριστηρίων με τον Η/Υ γίνονται αποκλειστικά από τον Η/Υ [Εικόνα 5.3]. Η μόνη προϋπόθεση είναι το τηλεχειριστήριο να διαθέτει φορτισμένες μπαταρίες για την σωστή λειτουργία του. Στη περίπτωση που η συσκευή διαθέτει μπαταρίες και το πρόγραμμα δεν μπορεί να συνδεθεί στη συσκευή, σημαίνει ότι η συσκευή έχει μπει σε κατάσταση εξοικονόμησης και χρειάζεται να πατηθεί οποιοδήποτε πλήκτρο για την ενεργοποίηση του.



Εικόνα 4.1 : Επιτυχής σύνδεση μεταξύ της φυσικής συσκευής

Κεφάλαιο 10ο - Συμπεράσματα και Προτάσεις Βελτίωσης

Ανακεφαλαιώνοντας, με την ολοκλήρωση της παρούσας πτυχιακής εργασίας, έγινε εφικτή η απομακρυσμένη λειτουργία ανάμεσα σε τρεις διαφορετικές συσκευές, με τη χρήση δύο διαφορετικών τεχνολογιών ασύρματης επικοινωνίας. Το εν λόγω πρωτότυπο σύστημα πέραν της χρησιμότητας που είδη παρέχει, μπορεί να αναβαθμιστεί εύκολα και χωρίς κάποια παραπάνω κατασκευή. Όποια πιθανή αναβάθμιση ή αλλαγή αναφερθεί παρακάτω, δεν πραγματοποιήθηκε καθώς δεν κρίθηκε απαραίτητο για την διεκπεραίωση του θέματος της εργασίας, με στόχο την αποφυγή της ανάπτυξης του συστήματος έξω από το πλαίσιο της εργασίας.

Το τηλεχειριστήριο είναι ικανό να δεχθεί πιθανές αναβαθμίσεις τις οποίες μπορούν να το κάνουν πιο γρήγορο και λεπτομερές στις λειτουργίες του, καθώς ο ESP32 είναι ένας αρκετά ικανός μικροελεγκτής με ποικίλες δυνατότητες. Επίσης, όσο αφορά την εφαρμογή Android, θα μπορούσε να βελτιωθεί σε μελλοντικό χρόνο, η απόκριση κίνησης του κέρσορα κατά τη διάρκεια προβολής της ζωντανής εικόνας του Server μέσω Gestures.

Τέλος, είναι απαραίτητο να αναφερθεί ότι παρά τις προτάσεις βελτίωσης, ένα φανερό αδύνατο σημείο του συστήματος είναι το TrackPad της εφαρμογής Android, διότι εξαρτάται άμεσα από τη ταχύτητα του τοπικού δικτύου καθώς και τη δύναμη επεξεργασίας της συσκευής. Αυτά τα δύο μπορούν να δημιουργήσουν την εντύπωση στον χρήστη, ότι αργεί να ανταποκριθεί η συσκευή ή μπορούν να προκαλέσουν μειωμένη ή καθυστερημένη απόκριση του κέρσορα, όταν χρησιμοποιείται το trackpad.

Βιβλιογραφία

ΒιΒλία

1. Ε. Μπολανάκης, Ευριπίδης Γλαβάς, Γεώργιος Α. Ευαγγελάκης, Κωνσταντίνος Θ. Κώτσης, Θόδωρος Λαόπουλος, "ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ"
2. Δημήτριος Β. Νικολός, "Αρχιτεκτονική υπολογιστών"

Datasheet

3. https://www.st.com/resource/en/datasheet/ld1117.pdf
4. https://eu.mouser.com/datasheet/2/891/esp32_wroom_32d_esp32_wroom_32u_datasheet_en-1365844.pdf

Πηγές από το διαδίκτυο

5. https://en.wikipedia.org/wiki/Remote_control
6. https://en.wikipedia.org/wiki/LPD433
7. https://en.wikipedia.org/wiki/Bluetooth
8. https://en.wikipedia.org/wiki/Operating_system
9. https://el.wikipedia.org/wiki/%CE%9F%CE%BB%CE%BF%CE%BA%CE%BB%CE%B7%CF%81%CF%89%CE%BC%CE%AD%CE%BD%CE%BF_%CF%80%CE%B5%CF%81%CE%B9%CE%B2%CE%AC%CE%BB%CE%BB%CE%BF%CE%BD_%CE%B1%CE%BD%CE%AC%CF%80%CF%84%CF%85%CE%BE%CE%B7%CF%82
10. https://en.wikipedia.org/wiki/AVR_microcontrollers
11. https://en.wikipedia.org/wiki/Modified_Harvard_architecture#Split-cache_(or_almost-von-Neumann)_architecture
12. https://en.wikipedia.org/wiki/AVR_microcontrollers
13. https://en.wikipedia.org/wiki/Analog-to-digital_converter
14. https://en.wikipedia.org/wiki/Computer-aided_design
15. https://el.wikipedia.org/wiki/%CE%A5%CF%80%CE%AD%CF%81%CF%85%CE%B8%CF%81%CE%B7_%CE%B1%CE%BA%CF%84%CE%B9%CE%BD%CE%BF%CE%B2%CE%BF%CE%BB%CE%AF%CE%B1
16. https://en.wikipedia.org/wiki/Zigbee
17.

Εικόνες από το διαδίκτυο

18. https://onlinedocs.microchip.com/pr/GUID-04E3421E-81C3-4E6C-BD29-9E1A7BAEBA7E-en-US-1/index.html?GUID-995553E0-AB09-4965-B761-FEE58C5FE5E1

19. https://stackify.com/wp-content/uploads/2017/12/word-image-9.png

Παραρτήματα

Α. Κώδικας ESP32 (C/C++)

```
# include <stdio.h>
# include <Arduino.h>
# include "BleSerial.h"
# include "sdkconfig.h"
# include "nvs_flash.h"
# include <freertos/FreeRTOS.h>
# include <freertos/task.h>
# include <freertos/queue.h>
# include <driver/gpio.h>
# include <esp_timer.h>
# include <esp_sleep.h>
# include <time.h>
# include <string.h>
# include <stdlib.h>
# include <OneButton.h>

#define BTN1 GPIO_NUM_35
#define BTN2 GPIO_NUM_34
#define BTN3 GPIO_NUM_5
#define BTN4 GPIO_NUM_18
#define BTN5 GPIO_NUM_0 // joystick
#define VRx GPIO_NUM_32 // analog ADC1
#define VRy GPIO_NUM_33 // analog ADC1

#define TAG "PROGRAM"
BleSerial ble;
OneButton button1(BTN1,true);
OneButton button2(BTN2,true);
OneButton button3(BTN3,true);
OneButton button4(BTN4,true);
OneButton button5(BTN5,true);

int SecondsCount = 0;
void SendMsg(char letter, char number)
{
	char msg[] = "12345678";
	msg[0] = letter;
	msg[1] = number;
	msg[2] = letter;
	msg[3] = number;
	msg[4] = letter;
	msg[5] = number;
	msg[6] = letter;
	msg[7] = number;
	ble.print(msg);

	SecondsCount = 0;
}
void Btn1SingleClick()
{
	char cmd[] = "a1";
```

```
        SendMsg(cmd[0], cmd[1]);
}
void Btn1DoubleClick()
{
        char cmd[] = "a2";
        SendMsg(cmd[0], cmd[1]);
}
void Btn1LongClick()
{
        char cmd[] = "a3";
        SendMsg(cmd[0], cmd[1]);
}

void Btn2SingleClick()
{
        char cmd[] = "b1";
        SendMsg(cmd[0], cmd[1]);
}
void Btn2DoubleClick()
{
        char cmd[] = "b2";
        SendMsg(cmd[0], cmd[1]);
}
void Btn2LongClick()
{
        char cmd[] = "b3";
        SendMsg(cmd[0], cmd[1]);
}

void Btn3SingleClick()
{
        char cmd[] = "c1";
        SendMsg(cmd[0], cmd[1]);
}
void Btn3DoubleClick()
{
        char cmd[] = "c2";
        SendMsg(cmd[0], cmd[1]);
}
void Btn3LongClick()
{
        char cmd[] = "c3";
        SendMsg(cmd[0], cmd[1]);
}
void Btn4SingleClick()
{
        char cmd[] = "d1";
        SendMsg(cmd[0], cmd[1]);
}
void Btn4DoubleClick()
{
        char cmd[] = "d2";
        SendMsg(cmd[0], cmd[1]);
}
void Btn4LongClick()
{
        char cmd[] = "d3";
        SendMsg(cmd[0], cmd[1]);
}

void Btn5SingleClick()
{
```

```
	char cmd[] = "e1";
	SendMsg(cmd[0], cmd[1]);
}
void Btn5DoubleClick()
{
	char cmd[] = "e2";
	SendMsg(cmd[0], cmd[1]);
}
void Btn5LongClick()
{
	char cmd[] = "e3";
	SendMsg(cmd[0], cmd[1]);
}

void gpio_init()
{
	gpio_pad_select_gpio(BTN1);
	gpio_pad_select_gpio(BTN2);
	gpio_pad_select_gpio(BTN3);
	gpio_pad_select_gpio(BTN4);
	gpio_pad_select_gpio(BTN5);

	gpio_set_direction(BTN1, GPIO_MODE_INPUT);
	gpio_set_direction(BTN2, GPIO_MODE_INPUT);
	gpio_set_direction(BTN3, GPIO_MODE_INPUT);
	gpio_set_direction(BTN4, GPIO_MODE_INPUT);
	gpio_set_direction(BTN5, GPIO_MODE_INPUT);
}

void Joystick_task(void* args)
{
	while (1)
	{
		int VRx_value = analogRead(VRx);
		int VRy_value = analogRead(VRy);
		if (VRx_value < 1500)
		{ // f1 f2 f3 LEFT
			char cmd[] = "ff";
			if (VRx_value < 1500 && VRx_value > 750)
				cmd[1] = '1';
			else if (VRx_value < 750)
				cmd[1] = '2';

			SendMsg(cmd[0], cmd[1]);
		}
		else if (VRx_value > 1800)
		{ // g1 g2 g3 RIGHT
			char cmd[] = "gg";
			if (VRx_value > 1800 && VRx_value < 2850)
				cmd[1] = '1';
			else if (VRx_value > 2850)
				cmd[1] = '2';

			SendMsg(cmd[0], cmd[1]);
		}
		if (VRy_value < 1200)
		{ // h1 h2 h3 DOWN
			char cmd[] = "hh";
			if (VRy_value < 1200 && VRy_value > 750)
				cmd[1] = '1';
			else if (VRy_value < 750)
				cmd[1] = '2';
```

```
                SendMsg(cmd[0], cmd[1]);
            }
            else if (VRy_value > 2000)
            { // i1 i2 i3 UP
                char cmd[] = "ii";
                if (VRy_value > 2000 && VRy_value < 2900)
                {
                    cmd[1] = '1';
                }
                if (VRy_value > 2900)
                {
                    cmd[1] = '2';
                }
                SendMsg(cmd[0], cmd[1]);
            }

            delay(1000);
        }
}
void CountDownTimer_task(void* args)
{
        while (1)
        {
            if (SecondsCount > 300)
            {
                esp_sleep_pd_config(ESP_PD_DOMAIN_RTC_SLOW_MEM,
ESP_PD_OPTION_OFF);
                esp_sleep_pd_config(ESP_PD_DOMAIN_RTC_FAST_MEM,
ESP_PD_OPTION_OFF);
                esp_sleep_pd_config(ESP_PD_DOMAIN_RTC_PERIPH,
ESP_PD_OPTION_AUTO);
                esp_sleep_enable_ext0_wakeup(BTN1, 0); // only one btn wakeup
                esp_deep_sleep_start();
            }
            vTaskDelay(1000 / portTICK_PERIOD_MS);
            SecondsCount++;

        }
}
int x = 0;
int y = 0;
extern "C" void app_main()
{
        nvs_flash_init();
        ble.begin("Dev");

        gpio_init();
        xTaskCreate(&CountDownTimer_task, "CountDownTimer_task task", 2048, NULL,
tskIDLE_PRIORITY, NULL);
        xTaskCreate(&Joystick_task, "Joystick task", 4096, NULL, tskIDLE_PRIORITY,
NULL);

        button1.attachClick(Btn1SingleClick);
        button1.attachDoubleClick(Btn1DoubleClick);
        button1.attachLongPressStop(Btn1LongClick);

        button2.attachClick(Btn2SingleClick);
        button2.attachDoubleClick(Btn2DoubleClick);
        button2.attachLongPressStop(Btn2LongClick);

        button3.attachClick(Btn3SingleClick);
```

```
        button3.attachDoubleClick(Btn3DoubleClick);
        button3.attachLongPressStop(Btn3LongClick);

        button4.attachClick(Btn4SingleClick);
        button4.attachDoubleClick(Btn4DoubleClick);
        button4.attachLongPressStop(Btn4LongClick);

        button5.attachClick(Btn5SingleClick);
        button5.attachDoubleClick(Btn5DoubleClick);
        button5.attachLongPressStop(Btn5LongClick);


        while (1)
        {
                button1.tick();
                button2.tick();
                button3.tick();
                button4.tick();
                button5.tick();

                delay(10);
        }
}
```

Β. Κώδικας Αρχικής Σελίδας Android (Xamarin.NET/C#)

```
using SkiaSharp;
using System;
using System.Threading;
using TouchTracking;
using Xamarin.Essentials;
using Xamarin.Forms;

namespace AndroidThesisApp
{

    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            InitializeComponent();
            NavigationPage.SetHasNavigationBar(this, false); // απενεργοποιεί το nav
bar

        }


        private static Client.ClientSocket ClientSocket = new Client.ClientSocket();

        private void SendMsg(string ip, int port, string msg, ushort code)
        {
            ClientSocket.Connect(ip, port);

            for (int i = 0; i < 5; i++)
            {
                if (ClientSocket.connected() == true)
                {
                    StatusFrame.BackgroundColor = Color.LightGreen;
                    Client.Message packet = new Client.Message(msg, code);
                    ClientSocket.Send(packet.Data);
                    return;
                }
                Thread.Sleep(100);
            }
            ClientSocket.Disconnect();

            StatusFrame.BackgroundColor = Color.Red;
        }


        private void VolumeSlider_ValueChanged(object sender, ValueChangedEventArgs
e)
        {
            int someInt = (int)Math.Round(VolumeSlider.Value);
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text),
someInt.ToString(), 3000);
        }
        #region // Macro buttons

        private void Macro1Btn_Tapped(object sender, MR.Gestures.TapEventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
1010);
```

```
        }
        private void Macro1Btn_DoubleTapped(object sender, MR.Gestures.TapEventArgs
e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
1011);
        }
        private void Macro1Btn_LongPressed(object sender,
MR.Gestures.LongPressEventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
1012);
        }
        private void Macro2Btn_Tapped(object sender, MR.Gestures.TapEventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
1020);
        }
        private void Macro2Btn_DoubleTapped(object sender, MR.Gestures.TapEventArgs
e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
1021);
        }

        private void Macro2Btn_LongPressed(object sender,
MR.Gestures.LongPressEventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
1022);
        }


        private void Macro3Btn_Tapped(object sender, MR.Gestures.TapEventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
1030);
        }
        private void Macro3Btn_DoubleTapped(object sender, MR.Gestures.TapEventArgs
e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
1031);
        }

        private void Macro3Btn_LongPressed(object sender,
MR.Gestures.LongPressEventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
1032);
        }
        private void Macro4Btn_Tapped(object sender, MR.Gestures.TapEventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
1040);
        }
        private void Macro4Btn_DoubleTapped(object sender, MR.Gestures.TapEventArgs
e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
1041);
        }
```

```
        private void Macro4Btn_LongPressed(object sender,
MR.Gestures.LongPressEventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
1042);
        }

        #endregion
        #region // Navigation Buttons
        private void UpBtn_Clicked(object sender, EventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
2000);
        }

        private void LeftBtn_Clicked(object sender, EventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
2001);
        }

        private void CenterBtn_Clicked(object sender, EventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
2002);
        }

        private void RightBtn_Clicked(object sender, EventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
2003);
        }

        private void DownBtn_Clicked(object sender, EventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
2004);
        }
        #endregion

        #region touch area

        private void canvasViewGrid_Tapped(object sender, MR.Gestures.TapEventArgs
e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
5000);
            Cursorlbl.Text = "Tap1";

        }
        private void canvasViewGrid_DoubleTapped(object sender,
MR.Gestures.TapEventArgs e)
        {

            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
5001);
            Cursorlbl.Text = "Double Tap";
        }


        private void canvasViewGrid_LongPressed(object sender,
MR.Gestures.LongPressEventArgs e)
```

```
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
5002);
            Cursorlbl.Text = "Long Press";
        }

        #endregion

        float x = 0;
        float y = 0;
        private void TouchEffect_TouchAction(object sender, TouchActionEventArgs
args)
        {
            TouchTrackingPoint pt = args.Location;
            SKPoint point =
           new SKPoint((float)(canvasView.CanvasSize.Width * pt.X /
canvasView.Width),
                        (float)(canvasView.CanvasSize.Height * pt.Y /
canvasView.Height));


            switch (args.Type)
            {
                case TouchActionType.Pressed:
                    x = point.X; // zero X
                    y = point.Y; // zero Y


                    break;
                case TouchActionType.Moved:

                    float xDif = x - point.X;
                    float yDif = y - point.Y;
                    int round_x = (int)Math.Round(xDif);
                    int round_y = (int)Math.Round(yDif);
                    SendMsg(TargetIP_Entry.Text,
Convert.ToInt32(TargetPORT_Entry.Text), round_x.ToString() + "," +
round_y.ToString(), 4000);

                    break;
                case TouchActionType.Released:

                    break;
            }
        }



        private void FullScreenBtn_Clicked(object sender, EventArgs e)
        {
            var mainDisplayInfo = DeviceDisplay.MainDisplayInfo;
            var width = mainDisplayInfo.Width;
            var height = mainDisplayInfo.Height;
            var density = mainDisplayInfo.Density;

            double heightin = height / density;
            double widthin = width / density;
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text),
widthin.ToString() + "," + heightin.ToString(), 6000); /// WIDTH , HEIGHT


            Navigation.PushAsync(new LiveViewPage());
```

```
        }

        private void ConnectionSettings_checkBox_CheckedChanged(object sender,
CheckedChangedEventArgs e)
        {
            if (ConnectionSettings_checkBox.IsChecked == true)
            {
                TargetSettingsGrid.IsVisible = true;
                ConnectionSettings_label.Margin = new Thickness(0, 7, 0, 0);
            }
            else
            {
                ConnectionSettings_label.Margin = new Thickness(0, 9, 0, 0);
                TargetSettingsGrid.IsVisible = false;
            }
        }

        private void TargetPORT_Entry_TextChanged(object sender,
TextChangedEventArgs e)
        {
            App.Current.Resources["PortResource"] = TargetPORT_Entry.Text;
        }

        private void TargetIP_Entry_TextChanged(object sender, TextChangedEventArgs
e)
        {
            App.Current.Resources["IPResource"] = TargetIP_Entry.Text;
            ClientSocket.Disconnect();
        }
        private void CenterBtn_DoubleTapped(object sender, MR.Gestures.TapEventArgs
e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
5001);
        }

        private void CenterBtn_LongPressed(object sender,
MR.Gestures.LongPressEventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
5002);
        }

        private void CenterBtn_Tapped(object sender, MR.Gestures.TapEventArgs e)
        {
            SendMsg(TargetIP_Entry.Text, Convert.ToInt32(TargetPORT_Entry.Text), "",
5000);
        }
    }
}
```

C. Κώδικας σχεδιασμού αρχικής σελίδας Android (XAML)

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
```

```
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:mr="clr-namespace:MR.Gestures;assembly=MR.Gestures"
             xmlns:skia="clr-
namespace:SkiaSharp.Views.Forms;assembly=SkiaSharp.Views.Forms"
             xmlns:tt="clr-
namespace:TouchTracking.Forms;assembly=TouchTracking.Forms"
             x:Class="AndroidThesisApp.MainPage">
    <StackLayout>
        <!-- TITLE -->
        <Frame BackgroundColor="Red" Padding="1" HeightRequest="10"
x:Name="StatusFrame">
        </Frame>
        <Grid x:Name="TargetSettingsGrid" ColumnDefinitions="4" Margin="0,0,0,0">
            <Label Grid.Column="1" x:Name="TargetIP_label" Text="Server IP"
HorizontalOptions="Start" Padding="5,0,0,0" Margin="0,20,0,0"></Label>
            <Entry Grid.Column="2" x:Name="TargetIP_Entry"
TextChanged="TargetIP_Entry_TextChanged" HorizontalOptions="FillAndExpand" Margin="-
30,0,0,0" HeightRequest="80" FontSize="16" Text="192.168.1.7"></Entry>
            <Label Grid.Column="3" x:Name="TargetPort_label" Text="Server PORT"
HorizontalOptions="Start" Padding="5,0,0,0" Margin="0,20,0,0"></Label>
            <Entry Grid.Column="4" x:Name="TargetPORT_Entry"
TextChanged="TargetPORT_Entry_TextChanged" HorizontalOptions="FillAndExpand"
Margin="-10,0,10,0" HeightRequest="80" FontSize="16" Text="9000" ></Entry>

        </Grid>
        <Grid ColumnDefinitions="2">
            <Label x:Name="ConnectionSettings_label" Grid.Column="1" Text="Settings"
Padding="30,2,0,0" Margin="0,7,0,0"></Label>
            <CheckBox x:Name="ConnectionSettings_checkBox" IsChecked="True"
Margin="0,0,0,0" CheckedChanged="ConnectionSettings_checkBox_CheckedChanged"
Grid.Column="1" ></CheckBox>
            <Button x:Name="FullScreenBtn" Clicked="FullScreenBtn_Clicked" Text=""
Margin="0,0,10,0" ImageSource="fullscreenbtn.png" BackgroundColor="Transparent"
                    Grid.Column="2" HorizontalOptions="End" VerticalOptions="Start"
WidthRequest="50" HeightRequest="70"></Button>
        </Grid>
        <!-- Volume Slider -->

        <Slider HeightRequest="50" BackgroundColor="transparent"
ThumbColor="#BB7900" MaximumTrackColor="#7B5000" Margin="5,0,0,0"
MinimumTrackColor="#FCD299"
                        Maximum="100" Minimum="0" x:Name="VolumeSlider"
ValueChanged="VolumeSlider_ValueChanged" ></Slider>

        <!--Macro Buttons-->
        <Grid>
            <mr:Button x:Name="Macro1Btn" Grid.Row="0" Grid.Column="0"
HorizontalOptions="End" BackgroundColor="#FCD299"
                    Text="1" HeightRequest="150" WidthRequest="150" Margin="0 , 10
,5,0"
                       Tapped="Macro1Btn_Tapped"
                       DoubleTapped="Macro1Btn_DoubleTapped"
                       LongPressed="Macro1Btn_LongPressed"></mr:Button>
            <mr:Button x:Name="Macro2Btn" Grid.Row="0" Grid.Column="1"
HorizontalOptions="Start"  WidthRequest="150" Margin="5 , 10,0,0"
BackgroundColor="#FCD299"
                    Text="2"
                       Tapped="Macro2Btn_Tapped"
                       DoubleTapped="Macro2Btn_DoubleTapped"
                       LongPressed="Macro2Btn_LongPressed"></mr:Button>
```

```
            <mr:Button x:Name="Macro3Btn" Grid.Row="1" Grid.Column="0"
HorizontalOptions="End" WidthRequest="150" Margin="0 , 10 ,5,0"
BackgroundColor="#FCD299"
                    Text="3"
                       Tapped="Macro3Btn_Tapped"
                       DoubleTapped="Macro3Btn_DoubleTapped"
                       LongPressed="Macro3Btn_LongPressed"></mr:Button>
            <mr:Button x:Name="Macro4Btn" Grid.Row="1" Grid.Column="1"
HorizontalOptions="Start" WidthRequest="150" Margin="5 , 10,0,0"
BackgroundColor="#FCD299"
                    Text="4"
                       Tapped="Macro4Btn_Tapped"
                       DoubleTapped="Macro4Btn_DoubleTapped"
                       LongPressed="Macro4Btn_LongPressed"></mr:Button>
        </Grid>

        <Grid Padding="20" Margin="0,10,0,0">
            <Label  Grid.Row="0" Grid.Column="0"></Label>
            <Button x:Name="UpBtn" Grid.Row="0" Grid.Column="1"
ImageSource="Up_arrow50.png" BackgroundColor="Transparent" Margin="0,-20,0,0"
                    Text="" Clicked="UpBtn_Clicked"></Button>
            <Label  Grid.Row="0" Grid.Column="2"></Label>


            <Button x:Name="LeftBtn" Grid.Row="1" Grid.Column="0"
ImageSource="Left_arrow50.png"  BackgroundColor="Transparent" Margin="-10,0,0,0"
                    Text="" Clicked="LeftBtn_Clicked"  ></Button>

            <Button x:Name="RightBtn" Grid.Row="1" Grid.Column="2"
HeightRequest="80" ImageSource="Right_arrow50.png" BackgroundColor="Transparent"
                    Text="" Clicked="RightBtn_Clicked"></Button>

            <Label  Grid.Row="2" Grid.Column="0"></Label>
            <Button x:Name="DownBtn" Grid.Row="2" Grid.Column="1"
ImageSource="Down_arrow50.png" BackgroundColor="Transparent" Margin="0,20,0,0"
                    Text="" Clicked="DownBtn_Clicked"></Button>
            <Label  Grid.Row="2" Grid.Column="2"></Label>
            <mr:Button x:Name="CenterBtn" Grid.Row="1" Grid.Column="1"
ImageSource="Center50.png"  BackgroundColor="Transparent" Padding="0,20,0,20"
                    Text="" Clicked="CenterBtn_Clicked"
DoubleTapped="CenterBtn_DoubleTapped" LongPressed="CenterBtn_LongPressed"
Tapped="CenterBtn_Tapped"></mr:Button>
        </Grid>
        <!-- Touch Area - Trackpad -->
        <mr:Grid x:Name="canvasViewGrid"
        Grid.Row="0"
        BackgroundColor="#F0F3F5" HeightRequest="300"

                AnchorX="0"
                AnchorY="0"
                AutomationProperties.IsInAccessibleTree="False">

            <Label x:Name="Cursorlbl"  Text="Touch Area" TextColor="Black"
FontSize="50" Opacity="0.1"
                  VerticalTextAlignment="Center" HorizontalTextAlignment="Center"
/>
            <skia:SKCanvasView x:Name="canvasView" />
            <Grid.Effects>
                <tt:TouchEffect Capture="True"
                        TouchAction="TouchEffect_TouchAction"/>

            </Grid.Effects>
```

```
        </mr:Grid>
    </StackLayout>

</ContentPage>
```

D. Εκκίνηση εφαρμογής Κόμβος και διαχείριση δεδομένων Bluetooth (C#/.Net)

```
using InTheHand.Net.Bluetooth;
using InTheHand.Net.Sockets;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.IO.Ports;
using System.Linq;
using System.Management;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using WindowsServer.Extras;
using Microsoft.Win32;
using System.Diagnostics;
using InTheHand.Net;
using System.Text.RegularExpressions;
using System.Runtime.InteropServices;
using Windows.Devices.Bluetooth.GenericAttributeProfile;
using Windows.Devices.Enumeration;
using Windows.Storage.Streams;
using Windows.Devices.Bluetooth;
using System.Timers;

namespace WindowsServer
{
public partial class form : Form
    {
        string localIp = Extras.NetworkingAid.GetLocalIPAddress();
        private static Server.ServerSocket server;
        public delegate void delUpdateUIText(string text);
        public static bool StartThread = false;
        Stopwatch timer1 = new Stopwatch();
        private Extras.ImageStreamingServer _Server;
```

```
        public static form _Form1;
        List<string> items = new List<string>();
/*        BluetoothDeviceInfo[] devices;
*/        Guid mUUID = new Guid("00001101-0000-1000-8000-00805F9B34FB");
        int[] action = new int[5]; // up to 4 single tap actions (skip [0])
        int[] actionLongAction = new int[5];
        int[] actionDoubleAction = new int[5];
        int selectedButton;

        string[] CmdText = new string[5];
        string[] CmdText2 = new string[5];
        string[] CmdText3 = new string[5];

        /*        BluetoothDeviceInfo deviceInfo;
        *//*        int count = 0;
        */
        public form()
        {
            InitializeComponent();
        }
        RegistryKey rkApp =
Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run",
true);
        private void form_Load(object sender, EventArgs e)
        {
            loadActions();

            string cleaned = localIp.Replace("\n", " - ").Replace("\r", "");
            Ip_label.Text += cleaned;
            _Form1 = this;
            Volumelbl.Text = Extras.AudioMixerHelper.GetMasterVolume().ToString();
            volume_bar.Value =
Convert.ToInt32(Extras.AudioMixerHelper.GetMasterVolume());

           server = new Server.ServerSocket();


           server.Bind(Convert.ToInt32(Port_text.Text));
           server.Listen(500);
           server.Accept();


            _Server = new ImageStreamingServer();
            _Server.Start(8080);



            Thread BluetoothScanThread = new Thread(new ThreadStart(ScanBT));
            BluetoothScanThread.Start();

            Thread TimerThread = new Thread(new ThreadStart(TimerCreate));
            TimerThread.Start();

            pictureBox2_Click(this.pictureBox2, e);
            if (rkApp.GetValue("WindowsServer") == null)
            {
                Startup_CheckBox.Checked = false;
            }
            else
            {
                Startup_CheckBox.Checked = true;
            }
```

```
            StatusBox.AppendText("Server Started...");
        }
        void TimerCreate()
        {
            Thread.Sleep(1000);
            System.Timers.Timer newTimer = new System.Timers.Timer();
            newTimer.Elapsed += new ElapsedEventHandler(DisplayTimeEvent);
            newTimer.Interval = 5000;
            newTimer.Start();
        }
        public void DisplayTimeEvent(object source, ElapsedEventArgs e)
        {
            if (bluetoothLeDevice != null)
            {
                if (bluetoothLeDevice.ConnectionStatus.ToString() != "Connected")
                {

                        updatePictureBox(Color.Red);

Console.WriteLine(bluetoothLeDevice.ConnectionStatus.ToString());



                        Console.WriteLine("in timer");

                    Thread BluetoothScanThread = new Thread(new
ThreadStart(ScanBT));
                        BluetoothScanThread.Start();

                }
                else
                {
                    updatePictureBox(Color.LightGreen);
                    UpdateStatusBox("Device Connected");
                }
            }
        }
        ///
        /// Functions διενεργιών
        ///
        #region Functions Διαχείριση
        /// <summary>
        /// Ανάκτηση αποθηκευμένων επιλογών
        /// </summary>
        private void loadActions()
        {
            action[1] = Properties.Settings.Default.Button1;
            action[2] = Properties.Settings.Default.Button2;
            action[3] = Properties.Settings.Default.Button3;
            action[4] = Properties.Settings.Default.Button4;


            actionLongAction[1] = Properties.Settings.Default.Button001;
            actionLongAction[2] = Properties.Settings.Default.Button002;
            actionLongAction[3] = Properties.Settings.Default.Button003;
            actionLongAction[4] = Properties.Settings.Default.Button004;

            actionDoubleAction[1] = Properties.Settings.Default.Button01;
            actionDoubleAction[2] = Properties.Settings.Default.Button02;
            actionDoubleAction[3] = Properties.Settings.Default.Button03;
            actionDoubleAction[4] = Properties.Settings.Default.Button04;
```

```
            CmdText[1] = Properties.Settings.Default.CMD1;
            CmdText[2] = Properties.Settings.Default.CMD2;
            CmdText[3] = Properties.Settings.Default.CMD3;
            CmdText[4] = Properties.Settings.Default.CMD4;

            CmdText2[1] = Properties.Settings.Default.CMD01;
            CmdText2[2] = Properties.Settings.Default.CMD02;
            CmdText2[3] = Properties.Settings.Default.CMD03;
            CmdText2[4] = Properties.Settings.Default.CMD04;

            CmdText3[1] = Properties.Settings.Default.CMD001;
            CmdText3[2] = Properties.Settings.Default.CMD002;
            CmdText3[3] = Properties.Settings.Default.CMD003;
            CmdText3[4] = Properties.Settings.Default.CMD004;


        }

        /// <summary>
        /// Δειαχίριση Master ήχου με την αλλαγή της στάθμης
        /// </summary>
        private void volume_bar_ValueChanged(object sender, EventArgs e)
        {
            Extras.AudioMixerHelper.SetMasterVolume(volume_bar.Value);
            Volumelbl.Text = "Volume : " + volume_bar.Value.ToString();
            if (volume_bar.Value == 0)
                Extras.AudioMixerHelper.SetMasterVolumeMute(true);
            else
                Extras.AudioMixerHelper.SetMasterVolumeMute(false);
        }

        ///
        /// Ανανέωση UI
        ///
        private void updatePictureBox(Color Color)
        {
            Func<int> del = delegate ()
            {
                pictureBox1.BackColor = Color;
                label7.ForeColor = Color;
                return 0;
            };
            Invoke(del);
        }
        private void updateUI(string msg)
        {
            Func<int> del = delegate ()
            {
                StatusBox.AppendText(msg + System.Environment.NewLine);
                StatusBox.ScrollToCaret();
                return 0;
            };
            Invoke(del);
        }
        string lastLine = "";
        private void UpdateStatusBox(string str)
        {
            if (str != lastLine)
            {
                WindowsServer.form._Form1.StatusBox.Invoke((MethodInvoker)(() =>
                WindowsServer.form._Form1.StatusBox.AppendText(Environment.NewLine +
str)));
```

```
                WindowsServer.form._Form1.StatusBox.Invoke((MethodInvoker)(() =>
                WindowsServer.form._Form1.StatusBox.SelectionStart =
StatusBox.TextLength));

                WindowsServer.form._Form1.StatusBox.Invoke((MethodInvoker)(() =>
                WindowsServer.form._Form1.StatusBox.ScrollToCaret()));

                lastLine = str;
            }
        }
        #endregion

        ///
        /// Διεργασίες Ανίχνευσης και Σύνδεσης/Επανασύνδεσης συσκευής Bluetooth
        ///
        #region Λειτουργίες Bluetooth
        string a = "";
        private async void ScanBT()
        {
            // Query for extra properties you want returned
            string[] requestedProperties = { "System.Devices.Aep.DeviceAddress",
"System.Devices.Aep.IsConnected" };
            try
            {

                deviceWatcher =
                        DeviceInformation.CreateWatcher(

BluetoothLEDevice.GetDeviceSelectorFromPairingState(false),
                                requestedProperties,
                                DeviceInformationKind.AssociationEndpoint);

                a = deviceWatcher.Status.ToString();
                deviceWatcher.Added += DeviceWatcher_Added;
            deviceWatcher.Updated += DeviceWatcher_Updated;
            deviceWatcher.Removed += DeviceWatcher_Removed;

                deviceWatcher.Start();
            }
            catch(Exception ex)
            {
            }

            while (true)
            {
                if (device == null)
                {
                    Thread.Sleep(200);
                }
                else
                {
                    Console.WriteLine("Press Any to pair .. \n WARNING: Remove
bluetooth device from windows before try to pair");

                    try
                    {

                            bluetoothLeDevice = await
BluetoothLEDevice.FromIdAsync(device.Id);

                            Console.WriteLine("Attempting to pair with device");
```

```
                            UpdateStatusBox("Attepring to pair device");
                            result = await bluetoothLeDevice.GetGattServicesAsync();

                            if (result.Status == GattCommunicationStatus.Success)
                            {
                                UpdateStatusBox("Pairing succeeded");
                                updatePictureBox(Color.LightGreen);
                                var services = result.Services;
                                foreach (var service in services)
                                {

                                    if (service.Uuid.ToString("N").Substring(0, 8)
== "6e400001") // first 8 bits from UUID - UART service
                                    {
                                        Console.WriteLine("Found UART service");

                                        GattCharacteristicsResult
charactiristicResult = await service.GetCharacteristicsAsync();

                                        if (charactiristicResult.Status ==
GattCommunicationStatus.Success)
                                        {
                                            var characteristics =
charactiristicResult.Characteristics;
                                            foreach (var characteristic in
characteristics)
                                            {
                                                Console.WriteLine("----------------
");
                                                Console.WriteLine(characteristic);
                                                GattCharacteristicProperties
properties = characteristic.CharacteristicProperties;

                                                if
(properties.HasFlag(GattCharacteristicProperties.Read))
                                                {
                                                    GattCommunicationStatus status =
await characteristic.WriteClientCharacteristicConfigurationDescriptorAsync(

GattClientCharacteristicConfigurationDescriptorValue.Notify);
                                                    Console.WriteLine("status
passed");
                                                    if (status ==
GattCommunicationStatus.Success)
                                                    {
                                                        characteristic.ValueChanged
+= Characteristic_ValueChanged;
                                                    }
                                                }

                                            }
                                        }
                                        else
                                        {

                                            updatePictureBox(Color.Red);
                                            Console.WriteLine("service connection
failed");
                                            DateTime end =
DateTime.Now.AddSeconds(2);
                                            while (DateTime.Now <= end)
```

```
                                        {

                                        }
                                    }



                                }
                            }
                            else
                            {
                                bluetoothLeDevice.Dispose();
                                updatePictureBox(Color.Red);

                            }

                            break;

                    }

                    catch (Exception ex)
                    {
                        updatePictureBox(Color.Red);
                        Console.WriteLine(ex.Message);
                    }

                }
            }
        }
        GattDeviceServicesResult result;
        DeviceWatcher deviceWatcher;
        static DeviceInformation device = null;
        BluetoothLEDevice bluetoothLeDevice;
        private void timer2_Tick(object sender, EventArgs e)
        {
            if (bluetoothLeDevice != null)
            {
                if (bluetoothLeDevice.ConnectionStatus.ToString() != "Connected")
                {
                    updatePictureBox(Color.Red);

Console.WriteLine(bluetoothLeDevice.ConnectionStatus.ToString());

                    device = null;
                    bluetoothLeDevice.Dispose();
                    Console.WriteLine("in timer");
                    Thread BluetoothScanThread2 = new Thread(new
ThreadStart(ScanBT));
                    BluetoothScanThread2.Start();
                }
                else
                {
                    updatePictureBox(Color.LightGreen);
                }
            }
        }

        string CommandForButton = "";
        private void Characteristic_ValueChanged(GattCharacteristic sender,
GattValueChangedEventArgs args)
        {
```

```
            var reader =
DataReader.FromBuffer(args.CharacteristicValue).ReadString(args.CharacteristicValue.
Length);
            if(reader.Contains("a") == true)
            {

                if (reader.Contains("a1") == true)
                {
                    CommandForButton = "1";
                    MakeAction(1);

                }
                else if(reader.Contains("a2") == true )
                {
                    CommandForButton = "2";
                    MakeDoubleAction(1);

                }
                else if (reader.Contains("a3") == true)
                {
                    CommandForButton = "3";
                    MakeActionLong(1);
                }
            }
            if (reader.Contains("c") == true)
            {

                if (reader.Contains("c1") == true)
                {
                    CommandForButton = "01";
                    MakeAction(2);
                }
                else if (reader.Contains("c2") == true)
                {
                    CommandForButton = "02";
                    MakeDoubleAction(2);
                }
                else if (reader.Contains("c3") == true)
                {
                    CommandForButton = "03";
                    MakeActionLong(2);
                }
            }
            if (reader.Contains("b") == true)
            {

                if (reader.Contains("b1") == true)
                {
                    CommandForButton = "001";
                    MakeAction(3);
                }
                else if (reader.Contains("b2") == true)
                {
                    CommandForButton = "002";
                    MakeDoubleAction(3);
                }
                else if (reader.Contains("b3") == true)
                {
                    CommandForButton = "003";
                    MakeActionLong(3);
                }
```

```
            }
            if (reader.Contains("d") == true)
            {

                if (reader.Contains("d1") == true)
                {
                    CommandForButton = "0001";
                    MakeAction(4);
                }
                else if (reader.Contains("d2") == true)
                {
                    CommandForButton = "0002";
                    MakeDoubleAction(4);
                }
                else if (reader.Contains("d3") == true)
                {
                    CommandForButton = "0003";
                    MakeActionLong(4);
                }
            }
            if (reader.Contains("e") == true)
            {
                if (reader.Contains("e1") == true)
                {
                    LeftMouseClick(Cursor.Position.X, Cursor.Position.Y);
                }
                else if (reader.Contains("e2") == true)
                {
                    LeftMouseClick(Cursor.Position.X, Cursor.Position.Y);
                    LeftMouseClick(Cursor.Position.X, Cursor.Position.Y);
                }
                else if (reader.Contains("e3") == true)
                {
                    RightMouseClick(Cursor.Position.X, Cursor.Position.Y);
                }
            }
            if (reader.Contains("f") == true)
            {
                if (reader.Contains("f1") == true)
                {
                    Cursor.Position = new Point(Cursor.Position.X - 10,
Cursor.Position.Y);

                }
                else if (reader.Contains("f2") == true)
                {
                    Cursor.Position = new Point(Cursor.Position.X - 100,
Cursor.Position.Y);
                }
            }
            if (reader.Contains("g") == true)
            {
                if (reader.Contains("g1") == true)
                {
                    Cursor.Position = new Point(Cursor.Position.X + 10,
Cursor.Position.Y);
                }
                else if (reader.Contains("g2") == true)
                {
                    Cursor.Position = new Point(Cursor.Position.X + 100,
Cursor.Position.Y);
                }
```

```
            }
            if (reader.Contains("h") == true)
            {
                if (reader.Contains("h1") == true)
                {
                    Cursor.Position = new Point(Cursor.Position.X ,
Cursor.Position.Y + 10);
                }
                else if (reader.Contains("h2") == true)
                {
                    Cursor.Position = new Point(Cursor.Position.X, Cursor.Position.Y
+ 100);
                }
            }
            if (reader.Contains("i") == true)
            {
                if (reader.Contains("i1") == true)
                {
                    Cursor.Position = new Point(Cursor.Position.X, Cursor.Position.Y
- 10);
                }
                else if (reader.Contains("i2") == true)
                {
                    Cursor.Position = new Point(Cursor.Position.X, Cursor.Position.Y
- 100);
                }
            }
        }

        private static void DeviceWatcher_Added(DeviceWatcher sender,
DeviceInformation args)
        {
            Console.WriteLine(args.Name);
            if (args.Name == "Dev")
            {
                device = args;
            }
        }
        static bool deviceFound = false;


        #endregion

        ///
        /// Εκτέλεση εισερχόμενης Εντολής
        ///
        #region Διεργασία εισερχόμενης εντολής


        private void MakeAction(int v)
        {
            Func<int> del = delegate ()
            {
                switch (v)
                {
                    case 1:
                        label3.Text = "One Tap Action selected" +
Options_cb.GetItemText(Options_cb.Items[Properties.Settings.Default.Button1]);

CommandsList(Options_cb.GetItemText(Options_cb.Items[Properties.Settings.Default.But
ton1]));
                        break;
```

```
                    case 2:
                        label3.Text = "One Tap Action selected" +
Options_cb.GetItemText(Options_cb.Items[Properties.Settings.Default.Button2]);

CommandsList(Options_cb.GetItemText(Options_cb.Items[Properties.Settings.Default.But
ton2]));
                        break;
                    case 3:
                        label3.Text = "One Tap Action selected" +
Options_cb.GetItemText(Options_cb.Items[Properties.Settings.Default.Button3]);

CommandsList(Options_cb.GetItemText(Options_cb.Items[Properties.Settings.Default.But
ton3]));
                        break;
                    case 4:
                        label3.Text = "One Tap Action selected" +
Options_cb.GetItemText(Options_cb.Items[Properties.Settings.Default.Button4]);

CommandsList(Options_cb.GetItemText(Options_cb.Items[Properties.Settings.Default.But
ton4]));
                        break;

                }

                return 0;
            };
            Invoke(del);
        }
        private void MakeActionLong(int v)
        {
            Func<int> del = delegate ()
            {
                switch (v)
                {
                    case 1:
                        label3.Text = "Long Action selected" +
Options_cb3.GetItemText(Options_cb3.Items[Properties.Settings.Default.Button001]);

CommandsList(Options_cb3.GetItemText(Options_cb3.Items[Properties.Settings.Default.B
utton001]));
                        break;
                    case 2:
                        label3.Text = "Long Action selected" +
Options_cb3.GetItemText(Options_cb3.Items[Properties.Settings.Default.Button002]);

CommandsList(Options_cb3.GetItemText(Options_cb3.Items[Properties.Settings.Default.B
utton002]));
                        break;
                    case 3:
                        label3.Text = "Long Action selected" +
Options_cb3.GetItemText(Options_cb3.Items[Properties.Settings.Default.Button003]);

CommandsList(Options_cb3.GetItemText(Options_cb3.Items[Properties.Settings.Default.B
utton003]));
                        break;
                    case 4:
                        label3.Text = "Long Action selected" +
Options_cb3.GetItemText(Options_cb3.Items[Properties.Settings.Default.Button004]);

CommandsList(Options_cb3.GetItemText(Options_cb3.Items[Properties.Settings.Default.B
utton004]));
                        break;
```

```
                }
                return 0;
            };
            Invoke(del);
        }

        private void MakeDoubleAction(int v)
        {
            Func<int> del = delegate ()
            {
                switch (v)
                {
                    case 1:
                        label3.Text = "Long Action selected" +
Options_cb2.GetItemText(Options_cb2.Items[Properties.Settings.Default.Button01]);

CommandsList(Options_cb2.GetItemText(Options_cb.Items[Properties.Settings.Default.Bu
tton01]));
                        break;
                    case 2:
                        label3.Text = "Long Action selected" +
Options_cb2.GetItemText(Options_cb2.Items[Properties.Settings.Default.Button02]);

CommandsList(Options_cb2.GetItemText(Options_cb.Items[Properties.Settings.Default.Bu
tton02]));
                        break;
                    case 3:
                        label3.Text = "Long Action selected" +
Options_cb2.GetItemText(Options_cb2.Items[Properties.Settings.Default.Button03]);

CommandsList(Options_cb2.GetItemText(Options_cb.Items[Properties.Settings.Default.Bu
tton03]));
                        break;
                    case 4:
                        label3.Text = "Long Action selected" +
Options_cb2.GetItemText(Options_cb2.Items[Properties.Settings.Default.Button04]);

CommandsList(Options_cb2.GetItemText(Options_cb2.Items[Properties.Settings.Default.B
utton04]));
                        break;
                }
                return 0;
            };
            Invoke(del);
        }
        #endregion
        [DllImport("user32.dll")]
        public static extern void keybd_event(byte bVk, byte bScan, uint dwFlags,
uint dwExtraInfo);
        private const int key_Up = 0x26;
        private const int key_Down = 0x28;
        private const int key_Right = 0x27;
        private const int key_Left = 0x25;
        private const int key_Space = 0x20;
        private const int key_Tab = 0x10;
        private const int key_PageUp = 0x21;
        private const int key_PageDown = 0x22;
        private const int key_PrintScreen = 0x2C;
        private const int key_esc = 0x1B;
        private const int key_alt = 0xA4;
        private const int key_space = 0x20;
        void CommandsList(string c)
```

```
        {
            Process cmd = new Process();
            cmd.StartInfo.FileName = "cmd.exe";
            switch (c)
            {
                case "None":
                    break;
                case "Shutdown":
                    cmd.StartInfo.Arguments = @"/C shutdown /t 0 /s";  // <-- /C
Εκτέλεση και μετά κλείσιμο
                    cmd.Start();
                    cmd.WaitForExit();
                    break;
                case "Sleep":
                    cmd.StartInfo.Arguments = @"/C %windir%\System32\rundll32.exe
powrprof.dll,SetSuspendState 0,1,0";  // <-- /C  Εκτέλεση και μετά κλείσιμο
                    cmd.Start();
                    cmd.WaitForExit();
                    break;
                case "Hibernate":
                    cmd.StartInfo.Arguments = @"/C  %windir%\System32\rundll32.exe
powrprof.dll,SetSuspendState Hibernate";  // <-- /C  Εκτέλεση και μετά κλείσιμο
                    cmd.Start();
                    cmd.WaitForExit();
                    break;
                case "Space":
                    keybd_event((byte)key_space, 0, 0x0001 | 0, 0);
                    break;
                case "Alt":
                    keybd_event((byte)key_alt, 0, 0x0001 | 0, 0);
                    break;
                case "Mute":
                    Extras.AudioMixerHelper.ToggleMasterVolumeMute();
                    break;
                case "Volume Up":

Extras.AudioMixerHelper.SetMasterVolume(Extras.AudioMixerHelper.GetMasterVolume() +
10);
                    Extras.AudioMixerHelper.SetMasterVolumeMute(false);
                    break;
                case "Volume Down":

Extras.AudioMixerHelper.SetMasterVolume(Extras.AudioMixerHelper.GetMasterVolume() -
10);
                    Extras.AudioMixerHelper.SetMasterVolumeMute(false);
                    break;
                case "Esc":
                    keybd_event((byte)key_esc, 0, 0x0001 | 0, 0);
                    break;
                case "Tab":
                    keybd_event((byte)key_Tab, 0, 0x0001 | 0, 0);
                    break;
                case "Left":
                    keybd_event((byte)key_Left, 0, 0x0001 | 0, 0);
                    break;
                case "Right":
                    keybd_event((byte)key_Right, 0, 0x0001 | 0, 0);
                    break;
                case "Up":
                    keybd_event((byte)key_Up, 0, 0x0001 | 0, 0);
                    break;
                case "Down":
```

```
                    keybd_event((byte)key_Down, 0, 0x0001 | 0, 0);
                    break;
                case "Left Click":
                    LeftMouseClick(Cursor.Position.X, Cursor.Position.Y);
                    break;
                case "Right Click":
                    RightMouseClick(Cursor.Position.X, Cursor.Position.Y);
                    break;
                case "Page Up":
                    keybd_event((byte)key_PageUp, 0, 0x0001 | 0, 0);
                    break;
                case "Page Down":
                    keybd_event((byte)key_PageDown, 0, 0x0001 | 0, 0);
                    break;
                case "Printscreen":
                    keybd_event((byte)key_PrintScreen, 0, 0x0001 | 0, 0);
                    break;
                case "Command Prompt":
                    switch (CommandForButton)
                    {
                        case "1":
                            cmd.StartInfo.Arguments = @"/C " + CmdText[1];  // <--
/C  Εκτέλεση και μετά κλείσιμο
                            break;
                        case "2":
                            cmd.StartInfo.Arguments = @"/C " + CmdText2[1];
                            break;
                        case "3":
                            cmd.StartInfo.Arguments = @"/C " + CmdText3[1];
                            break;
                        case "01":
                            cmd.StartInfo.Arguments = @"/C " + CmdText[2];
                            break;
                        case "02":
                            cmd.StartInfo.Arguments = @"/C " + CmdText2[2];
                            break;
                        case "03":
                            cmd.StartInfo.Arguments = @"/C " + CmdText3[2];
                            break;
                        case "001":
                            cmd.StartInfo.Arguments = @"/C " + CmdText[3];
                            break;
                        case "002":
                            cmd.StartInfo.Arguments = @"/C " + CmdText2[3];
                            break;
                        case "003":
                            cmd.StartInfo.Arguments = @"/C " + CmdText3[3];
                            break;
                        case "0001":
                            cmd.StartInfo.Arguments = @"/C " + CmdText[4];
                            break;
                        case "0002":
                            cmd.StartInfo.Arguments = @"/C " + CmdText2[4];
                            break;
                        case "0003":
                            cmd.StartInfo.Arguments = @"/C " + CmdText3[4];
                            break;
                        default:
                            UpdateStatusBox("Error while excecuting Command Prompt
argument");
                            break;
                    }
```

```
                    cmd.Start();
                    cmd.WaitForExit();
                    break;
                default:
                    break;
            }


        }
        ///
        /// Λειτουργίες Κέρσορα
        ///
        #region Mouse Events
        [System.Runtime.InteropServices.DllImport("user32.dll")]
        static extern bool SetCursorPos(int x, int y);

        [System.Runtime.InteropServices.DllImport("user32.dll")]
        public static extern void mouse_event(int dwFlags, int dx, int dy, int
cButtons, int dwExtraInfo);
        public const int MOUSEEVENTF_LEFTDOWN = 0x02;
        public const int MOUSEEVENTF_LEFTUP = 0x04;
        private const int MOUSEEVENTF_RIGHTDOWN = 0x08;
        private const int MOUSEEVENTF_RIGHTUP = 0x10;
        public static void LeftMouseClick(int xpos, int ypos)
        {
            SetCursorPos(xpos, ypos);
            mouse_event(MOUSEEVENTF_LEFTDOWN, xpos, ypos, 0, 0);
            mouse_event(MOUSEEVENTF_LEFTUP, xpos, ypos, 0, 0);
        }
        public static void RightMouseClick(int xpos, int ypos)
        {
            SetCursorPos(xpos, ypos);
            mouse_event(MOUSEEVENTF_RIGHTDOWN, xpos, ypos, 0, 0);
            mouse_event(MOUSEEVENTF_RIGHTUP, xpos, ypos, 0, 0);
        }
        #endregion


        /// <summary>
        /// Ενεργοποίση/Απενεργοποίηση Προγράμματος για εκκίνηση μαζί με τα Windows
        /// </summary>
        private void Startup_CheckBox_CheckedChanged(object sender, EventArgs e)
        {
            if (Startup_CheckBox.Checked)
            {
                rkApp.SetValue("WindowsServer", Application.ExecutablePath);
            }
            else
            {
                rkApp.DeleteValue("WindowsServer", false);
            }
        }

        ///
        /// Αποθηκευση συμπεριφοράς μπουτόν
        ///
        #region Remote Button Actions Select
        private void selectAction(object tag)
        {
            selectedButton = Convert.ToInt32(tag);
```

```
            Options_cb.SelectedIndex = action[selectedButton];
            Options_cb2.SelectedIndex = actionDoubleAction[selectedButton];
            Options_cb3.SelectedIndex = actionLongAction[selectedButton];

            CMD_textbox.Text = CmdText[selectedButton];
            CMD2_textbox.Text = CmdText2[selectedButton];
            CMD3_textbox.Text = CmdText3[selectedButton];
        }
        private void pictureBox2_Click(object sender, EventArgs e)
        {
            selectAction(pictureBox2.Tag);
            btnSelected_label.Text = "Button Selected : " + "1";


        }

        private void pictureBox3_Click(object sender, EventArgs e)
        {
            selectAction(pictureBox3.Tag);
            btnSelected_label.Text = "Button Selected : " + "2";
        }

        private void pictureBox4_Click(object sender, EventArgs e)
        {
            selectAction(pictureBox4.Tag);
            btnSelected_label.Text = "Button Selected : " + "3";
        }

        private void pictureBox5_Click(object sender, EventArgs e)
        {
            selectAction(pictureBox5.Tag);
            btnSelected_label.Text = "Button Selected : " + "4";
        }

        #endregion
        ///
        /// Αποθηκευση συμπεριφοράς μπουτόν
        ///

        #region Save Button-Preference and Behavior
        private void Options_cb_SelectedIndexChanged(object sender, EventArgs e)
        {
            action[selectedButton] = Options_cb.SelectedIndex;
            if (Options_cb.SelectedItem.ToString() == "Command Prompt")
                CMD_textbox.Visible = true;
            else
                CMD_textbox.Visible = false;

            switch (selectedButton)
            {
                case 1:
                    Properties.Settings.Default.Button1 = Options_cb.SelectedIndex;
                    break;
                case 2:
                    Properties.Settings.Default.Button2 = Options_cb.SelectedIndex;
                    break;
                case 3:
                    Properties.Settings.Default.Button3 = Options_cb.SelectedIndex;
                    break;
                case 4:
                    Properties.Settings.Default.Button4 = Options_cb.SelectedIndex;
                    break;
```

```
            }
            Properties.Settings.Default.Save();

        }
        private void Options_cb2_SelectedIndexChanged(object sender, EventArgs e)
        {

            actionDoubleAction[selectedButton] = Options_cb2.SelectedIndex;
            if (Options_cb2.SelectedItem.ToString() == "Command Prompt")
                CMD2_textbox.Visible = true;
            else
                CMD2_textbox.Visible = false;

            switch (selectedButton)
            {
                case 1:
                    Properties.Settings.Default.Button01 =
Options_cb2.SelectedIndex;
                    break;
                case 2:
                    Properties.Settings.Default.Button02 =
Options_cb2.SelectedIndex;
                    break;
                case 3:
                    Properties.Settings.Default.Button03 =
Options_cb2.SelectedIndex;
                    break;
                case 4:
                    Properties.Settings.Default.Button04 =
Options_cb2.SelectedIndex;
                    break;
            }
            Properties.Settings.Default.Save();
        }
        private void Options_cb3_SelectedIndexChanged(object sender, EventArgs e)
        {
            actionLongAction[selectedButton] = Options_cb3.SelectedIndex;
            if (Options_cb3.SelectedItem.ToString() == "Command Prompt")
                CMD3_textbox.Visible = true;
            else
                CMD3_textbox.Visible = false;

            switch (selectedButton)
            {
                case 1:
                    Properties.Settings.Default.Button001 =
Options_cb3.SelectedIndex;
                    break;
                case 2:
                    Properties.Settings.Default.Button002 =
Options_cb3.SelectedIndex;
                    break;
                case 3:
                    Properties.Settings.Default.Button003 =
Options_cb3.SelectedIndex;
                    break;
                case 4:
                    Properties.Settings.Default.Button004 =
Options_cb3.SelectedIndex;
                    break;
            }
            Properties.Settings.Default.Save();
```

```
        }

        private void CustomizePanel_MouseClick(object sender, MouseEventArgs e)
        {
            if (e.Clicks > 0)
            {
                Options_cb.Visible = false;
                Options_cb2.Visible = false;
                Options_cb3.Visible = false;
            }
        }

        #endregion

        ///
        /// Custom Buttons
        ///
        #region Flat Buttons
        private void CloseBtn_Click(object sender, EventArgs e)
        {
            Environment.Exit(Environment.ExitCode);
        }
        private void MinimizeBtn_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Minimized;
        }
        #endregion

        /// <summary>
        /// Κωδικας για μετακίνιση φόρμας απο το panel
        /// </summary>
        #region Move Form
        public const int WM_NCLBUTTONDOWN = 0xA1;
        public const int HT_CAPTION = 0x2;

        [System.Runtime.InteropServices.DllImport("user32.dll")]
        public static extern int SendMessage(IntPtr hWnd, int Msg, int wParam, int
lParam);
        [System.Runtime.InteropServices.DllImport("user32.dll")]
        public static extern bool ReleaseCapture();
        private void panel1_MouseDown(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                ReleaseCapture();
                SendMessage(Handle, WM_NCLBUTTONDOWN, HT_CAPTION, 0);
            }
        }
        #endregion


        private void CMD_textbox_TextChanged(object sender, EventArgs e)
        {
            CmdText[selectedButton] = CMD_textbox.Text;
            switch (selectedButton)
            {
                case 1:
                    Properties.Settings.Default.CMD1 = CMD_textbox.Text;
                    break;
                case 2:
                    Properties.Settings.Default.CMD2 = CMD_textbox.Text;
                    break;
```

```
                case 3:
                    Properties.Settings.Default.CMD3 = CMD_textbox.Text;
                    break;
                case 4:
                    Properties.Settings.Default.CMD4 = CMD_textbox.Text;
                    break;

            }
            Properties.Settings.Default.Save();
        }
        private void CMD2_textbox_TextChanged(object sender, EventArgs e)
        {
            CmdText2[selectedButton] = CMD2_textbox.Text;
            switch (selectedButton)
            {
                case 1:
                    Properties.Settings.Default.CMD01 = CMD2_textbox.Text;
                    break;
                case 2:
                    Properties.Settings.Default.CMD02 = CMD2_textbox.Text;
                    break;
                case 3:
                    Properties.Settings.Default.CMD03 = CMD2_textbox.Text;
                    break;
                case 4:
                    Properties.Settings.Default.CMD04 = CMD2_textbox.Text;
                    break;

            }
            Properties.Settings.Default.Save();
        }
        private void CMD3_textbox_TextChanged(object sender, EventArgs e)
        {
            CmdText3[selectedButton] = CMD3_textbox.Text;
            switch (selectedButton)
            {
                case 1:
                    Properties.Settings.Default.CMD001 = CMD3_textbox.Text;
                    break;
                case 2:
                    Properties.Settings.Default.CMD002 = CMD3_textbox.Text;
                    break;
                case 3:
                    Properties.Settings.Default.CMD003 = CMD3_textbox.Text;
                    break;
                case 4:
                    Properties.Settings.Default.CMD004 = CMD3_textbox.Text;
                    break;

            }
            Properties.Settings.Default.Save();
        }

        private void Options_cb_TextChanged(object sender, EventArgs e)
        {
            if (!Options_cb.Items.Contains(Options_cb.Text))
            {
                Options_cb.Text = @"{invalid}";
            }
        }
        private void Options_cb2_TextChanged(object sender, EventArgs e)
        {
```

```
            if (!Options_cb2.Items.Contains(Options_cb2.Text))
            {
                Options_cb2.Text = @"{invalid}";
            }
        }

        private void Options_cb3_TextUpdate(object sender, EventArgs e)
        {
            if (!Options_cb3.Items.Contains(Options_cb3.Text))
            {
                Options_cb3.Text = @"{invalid}";
            }
        }
       private void Port_text_TextChanged(object sender, EventArgs e)
        {
            server.Dis();
            server.Bind(Convert.ToInt32(Port_text.Text));
            server.Listen(500);
            server.Accept();
    }
}
```

Ε. Κώδικα διαχείρισης δεδομένων μέσω LAN

```
using System;
using System.Diagnostics;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Runtime.InteropServices;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;


namespace WindowsServer.Server
{

    public static class PacketHandler
    {
        [DllImport("user32.dll")]
        public static extern void keybd_event(byte bVk, byte bScan, uint dwFlags,
uint dwExtraInfo);

        [System.Runtime.InteropServices.DllImport("user32.dll")]
        static extern bool SetCursorPos(int x, int y);
        [System.Runtime.InteropServices.DllImport("user32.dll")]
        public static extern void mouse_event(int dwFlags, int dx, int dy, int
cButtons, int dwExtraInfo);
        public const int MOUSEEVENTF_LEFTDOWN = 0x02;
        public const int MOUSEEVENTF_LEFTUP = 0x04;
        private const int MOUSEEVENTF_RIGHTDOWN = 0x08;
        private const int MOUSEEVENTF_RIGHTUP = 0x10;
        private const int key_Up = 0x26;
```

```
        private const int key_Down = 0x28;
        private const int key_Right = 0x27;
        private const int key_Left = 0x25;
        private const int key_Space = 0x20;
        private const int key_Tab = 0x10;
        private const int key_PageUp = 0x21;
        private const int key_PageDown = 0x22;
        private const int key_PrintScreen = 0x2C;
        private const int key_esc = 0x1B;
        private const int key_alt = 0xA4;
        private const int key_space = 0x20;

        public static void LeftMouseClick(int xpos, int ypos)
        {
            SetCursorPos(xpos, ypos);
            mouse_event(MOUSEEVENTF_LEFTDOWN, xpos, ypos, 0, 0);
            mouse_event(MOUSEEVENTF_LEFTUP, xpos, ypos, 0, 0);
        }
        public static void RightMouseClick(int xpos, int ypos)
        {
            SetCursorPos(xpos, ypos);
            mouse_event(MOUSEEVENTF_RIGHTDOWN, xpos, ypos, 0, 0);
            mouse_event(MOUSEEVENTF_RIGHTUP, xpos, ypos, 0, 0);
        }

        public static string status;
        public static void Handle(byte[] packet, Socket clientSocket)
        {
            ushort packetLength = BitConverter.ToUInt16(packet, 0);
            ushort packetType = BitConverter.ToUInt16(packet, 2);

            status = "Received packet! Length: " + packetLength + " | Type: " +
packetType;

            /* CODES: BOOK
             *
             *
             * 1010 : M1 TAP
             * 1011      Double tap
             * 1012     long press
             * 1020 : M2
             * 1021     double tap
             * 1022     long press
             * 1030 : M3
             * 1031     double tap
             * 1032     long press
             * 1040 : M4
             * 1041     double tap
             * 1042     long press
             *
             * 2000 : Up
             * 2001 : Left
             * 2002 : Center
             * 2003 : Right
             * 2004 : Down
             *
             * 3000 : Slider
             *
             * 4000 : Touch Event
             *
             * 5000 : Tap
             * 5001 : Double Tap
```

```
             * 5002 : Long Press
             *
             */


            switch (packetType)
            {
                #region Msg types
                case 1010:
                    MakeAction("M1");
                    break;
                case 1011:
                    MakeDoubleAction("M1");
                    break;
                case 1012:
                    LongPressAction("M1");
                    break;
                case 1020:
                    MakeAction("M2");
                    break;
                case 1021:
                    MakeDoubleAction("M2");
                    break;
                case 1022:
                    LongPressAction("M2");
                    break;
                case 1030:
                    MakeAction("M3");
                    break;
                case 1031:
                    MakeDoubleAction("M3");
                    break;
                case 1032:
                    LongPressAction("M3");
                    break;
                case 1040:
                    MakeAction("M4");
                    break;
                case 1041:
                    MakeDoubleAction("M4");
                    break;
                case 1042:
                    LongPressAction("M4");
                    break;
                #endregion
                #region Navigation
                case 2000:
                    keybd_event((byte)0x26, 0, 0x0001 | 0, 0);
                    break;
                case 2001:
                    keybd_event((byte)0x25, 0, 0x0001 | 0, 0);
                    break;
                case 2002:
                    var Center = Task.Run(() =>
                    {
                        LeftMouseClick(Cursor.Position.X, Cursor.Position.Y);

                    });
                    Center.Wait();
                    break;
                case 2003:
                    var RightClick = Task.Run(() =>
```

```
                    {
                        RightMouseClick(Cursor.Position.X, Cursor.Position.Y);
                    });
                    RightClick.Wait();
                    break;
                case 2004:
                    keybd_event((byte)0x28, 0, 0x0001 | 0, 0);
                    break;
                #endregion
                case 3000:
                    Message msg = new Message(packet);
                    Console.WriteLine(msg.Text);
                    SliderValue(msg.Text);
                    break;

                case 4000:
                    Message msg1 = new Message(packet);
                    var pieces = msg1.Text.Split(new[] { ',' }, 2);
                    int X = Convert.ToInt32(pieces[0]);
                    int Y = Convert.ToInt32(pieces[1]);
                    TouchUpdate(X, Y);
                    break;
                case 5000:

                        LeftMouseClick(Cursor.Position.X, Cursor.Position.Y);
                    break;
                case 5001:

                        LeftMouseClick(Cursor.Position.X, Cursor.Position.Y);
                        LeftMouseClick(Cursor.Position.X, Cursor.Position.Y);
                    break;
                case 5002:
                        RightMouseClick(Cursor.Position.X, Cursor.Position.Y);
                    break;
                case 6000:
                    Message dimensions_msg = new Message(packet);
                    break;
                case 6001:
                    Message CursorMoved_msg = new Message(packet);
                    string[] dimentions = CursorMoved_msg.Text.Split(',');
                    int x = Convert.ToInt32(dimentions[0]);
                    int y = Convert.ToInt32(dimentions[1]);


                    break;
            }
        }


        private static void SliderValue(string value)
        {
            Console.WriteLine("In void");
            int x = Convert.ToInt32(value);
            Console.WriteLine("Out void");

            WindowsServer.form._Form1.volume_bar.Invoke((MethodInvoker)(() =>
WindowsServer.form._Form1.volume_bar.Value = x));
        }
        private static void TouchUpdate(int x, int y)
        {
            Cursor.Position = new Point(Cursor.Position.X - x, Cursor.Position.Y -
y);
```

```
        }
        private static void ChangeVolume(int x)
        {
            WindowsServer.form._Form1.volume_bar.Invoke((MethodInvoker)(() =>
WindowsServer.form._Form1.volume_bar.Value += x));

        }

        static string[] LIST_ = {
                        "None",
                        "Shutdown",
                        "Sleep",
                        "Hibernate",
                        "Space",
                        "Alt",
                        "Mute",
                        "Volume Up",
                        "Volume Down",
                        "Esc",
                        "Tab",
                        "Left",
                        "Right",
                        "Up",
                        "Down",
                        "Left Click",
                        "Right Click",
                        "Page Up",
                        "Page Down",
                        "Printscreen",
                        "Command Prompt"
        };

        private static void MakeDoubleAction(string command)
        {
            string str = "";
            switch (command)
            {
                case "M1":
                    str = LIST_[Properties.Settings.Default.Button01];
                    switcher(str);
                    break;
                case "M2":
                    str = LIST_[Properties.Settings.Default.Button02];
                    switcher(str);
                    break;
                case "M3":
                    str = LIST_[Properties.Settings.Default.Button03];
                    switcher(str);
                    break;
                case "M4":
                    str = LIST_[Properties.Settings.Default.Button04];
                    switcher(str);
                    break;
                default:
                    str = "-n";
                    break;
            }
        }
            private static void LongPressAction(string command)
            {
                string str;
```

```
                switch (command)
                {
                    case "M1":
                        str = LIST_[Properties.Settings.Default.Button001];
                        switcher(str);
                        break;
                    case "M2":
                        str = LIST_[Properties.Settings.Default.Button002];
                        switcher(str);
                        break;
                    case "M3":
                        str = LIST_[Properties.Settings.Default.Button003];
                        switcher(str);
                        break;
                    case "M4":
                        str = LIST_[Properties.Settings.Default.Button004];
                        switcher(str);
                        break;
                    default:
                        break;
                }
            }
        private static void MakeAction(string command)
        {
            string str;
            switch (command)
            {
                case "M1":
                    str = LIST_[Properties.Settings.Default.Button1];
                    switcher(str);
                    break;
                case "M2":
                    str = LIST_[Properties.Settings.Default.Button2];
                    switcher(str);
                    break;
                case "M3":
                    str = LIST_[Properties.Settings.Default.Button3];
                    switcher(str);
                    break;
                case "M4":
                    str = LIST_[Properties.Settings.Default.Button4];
                    switcher(str);
                    break;
                default:
                    break;
            }
        }

        static void switcher(string ff)
        {
            Process cmd = new Process();
            cmd.StartInfo.FileName = "cmd.exe";
            switch (ff)
            {
                case "None":
                    break;
                case "Shutdown":
                    cmd.StartInfo.Arguments = @"/C shutdown /t 600 /s";  // <-- /C
Εκτέλεση και μετά κλείσιμο
                    cmd.Start();
                    break;
                case "Sleep":
```

```
                    cmd.StartInfo.Arguments = @"/C %windir%\System32\rundll32.exe
powrprof.dll,SetSuspendState 0,1,0";  // <-- /C  Εκτέλεση και μετά κλείσιμο
cmd.Start();
                    break;
                case "Hibernate":
                    cmd.StartInfo.Arguments = @"/C %windir%\System32\rundll32.exe
powrprof.dll,SetSuspendState Hibernate";  // <-- /C  Εκτέλεση και μετά κλείσιμο
cmd.Start();
                    break;
                case "Space":
                    keybd_event((byte)key_space, 0, 0x0001 | 0, 0);
                    break;
                case "Alt":
                    keybd_event((byte)key_alt, 0, 0x0001 | 0, 0);
                    break;
                case "Mute":
                    Extras.AudioMixerHelper.ToggleMasterVolumeMute();
                    break;
                case "Volume Up":

Extras.AudioMixerHelper.SetMasterVolume(Extras.AudioMixerHelper.GetMasterVolume() +
10);
                    break;
                case "Volume Down":

Extras.AudioMixerHelper.SetMasterVolume(Extras.AudioMixerHelper.GetMasterVolume() -
10);
                    break;
                case "Esc":
                    keybd_event((byte)key_esc, 0, 0x0001 | 0, 0);
                    break;
                case "Tab":
                    keybd_event((byte)key_Tab, 0, 0x0001 | 0, 0);
                    break;
                case "Left":
                    keybd_event((byte)key_Left, 0, 0x0001 | 0, 0);
                    break;
                case "Right":
                    keybd_event((byte)key_Right, 0, 0x0001 | 0, 0);
                    break;
                case "Up":
                    keybd_event((byte)key_Up, 0, 0x0001 | 0, 0);
                    break;
                case "Down":
                    keybd_event((byte)key_Down, 0, 0x0001 | 0, 0);
                    break;
                case "Left Click":
                    LeftMouseClick(Cursor.Position.X, Cursor.Position.Y);
                    break;
                case "Right Click":
                    RightMouseClick(Cursor.Position.X, Cursor.Position.Y);
                    break;
                case "Page Up":
                    keybd_event((byte)key_PageUp, 0, 0x0001 | 0, 0);
                    break;
                case "Page Down":
                    keybd_event((byte)key_PageDown, 0, 0x0001 | 0, 0);
                    break;
                case "Printscreen":
                    keybd_event((byte)key_PrintScreen, 0, 0x0001 | 0, 0);
                    break;
                case "Command Prompt":
```

```
                    cmd.StartInfo.Arguments = @"/C +" ;  // <-- This will execute
the command and wait to close
                    cmd.Start();
                    cmd.WaitForExit();
                    break;
                default:
                    break;
            }
            cmd.WaitForExit();

        }

    }
}
```