ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

ΤΜΗΜΑ ΔΙΟΙΚΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ ΕΚΠΑΙΔΕΥΤΙΚΩΝ ΕΦΑΡΜΟΓΩΝ ΜΕ ΧΡΗΣΗ ΔΙΑΧΥΤΟΥ ΥΠΟΛΟΓΙΣΜΟΥ

ΕΠΙΜΕΛΕΙΑ ΕΡΓΑΣΙΑΣ: ΚΑΛΥΒΑΣ ΘΕΟΔΩΡΟΣ
ΜΗΤΡΟΣ ΧΡΙΣΤΑΚΗΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ : ΠΙΕΡΡΑΚΕΑΣ ΧΡΗΣΤΟΣ

ΠΑΤΡΑ, ΙΟΥΝΙΟΣ 2022

ΕΥΧΑΡΙΣΤΙΕΣ

Με το παρόν θέλουμε να ευχαριστήσουμε τον καθηγητή και επιβλέποντα μας κ. Πιερρακέα Χρήστο, για την πολύτιμη καθοδήγηση του, βοήθεια και άριστη συνεργασία στο έργο που εκπονήσαμε σε κάθε στάδιο αυτού.

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΗΣ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Οι κάτωθι υπογράφων Καλύβας Θεόδωρος του Κωνσταντίνου, με αριθμό μητρώου 12660 και Μήτρος Χριστάκης του Πέτρου με αριθμό μητρώου 12801 προπτυχιακοί φοιτητές του Πανεπιστημίου Πατρών Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας, πρώην τμήματος Διοίκησης Επιχειρήσεων ΤΕΙ Δυτικής Ελλάδας, δηλώνουμε ότι:

Είμαστε οι συγγραφείς και υπεύθυνοι ανάπτυξης αυτής της πτυχιακής εργασίας, οποιαδήποτε βοήθεια για την ανάπτυξη αυτής, αναγράφεται εντός της έρευνας και αναγνωρίζεται πλήρως. Για την ανάπτυξη της εργασίας εκτός από το πηγαίο κώδικα που αναπτύξαμε για την υλοποίηση αυτής, χρησιμοποιήθηκε και κώδικας ανοιχτού λογισμικού ο οποίος αναφέρεται. Επίσης οι πηγές από τις οποίες αντλήσαμε ιδέες, δεδομένα ή λέξεις όπως βρίσκονται στην αρχική τους μορφή ή παραφρασμένες, από βιβλία, περιοδικά, διαδίκτυο, εκδοτικούς οίκους ή ιδέες από φυσικά πρόσωπα, κάνουμε πλήρη αναφορά αυτών των πηγών και των συγγραφέων τους. Να αναφέρουμε πως η ιδέα της συγκεκριμένης πτυχιακής εργασία, έχει συγγραφεί αναπτυχθεί αποκλειστικά από τους αναφερομένους προπτυχιακούς φοιτητές και αποτελεί προϊόν πνευματικής ιδιοκτησίας της σχολής όσο και των αναφερόμενων φοιτητών.

Οι Δηλούντες:

Καλύβας Θεόδωρος.
Μήτρος Χριστάκης.

ΠΕΡΙΛΗΨΗ

Η Παρούσα πτυχιακή εργασία αφορά την ανάπτυξη εκπαιδευτικών εφαρμογών με χρήση διάχυτου υπολογισμού.

Η βασική λειτουργία της εφαρμογής σε συνδυασμό με τον κατάλληλο ηλεκτρονικό εξοπλισμό, μέσω της ανάπτυξης και εφαρμογής της βιομετρικής τεχνολογίας είναι η ταυτοποίηση των φυσικών προσώπων καθώς και η διευκόλυνση αυτών σε πρόσβαση στις υπηρεσίες της τριτοβάθμιας εκπαίδευσης. Στόχος μας είναι να δημιουργηθεί αρχικά μία υπηρεσία καινοτόμου τεχνολογίας για εκπαιδευτικούς σκοπούς που θα καθιστά την προσβασιμότητα στις υπηρεσίες του Τεχνολογικού Ιδρύματος άμεση και ταυτόχρονα έγκυρη ως προς την εξυπηρέτηση τόσο για τους φοιτητές όσο και για τους εργαζομένους του Ιδρύματος. Αναπτύξαμε μια εφαρμογή κατάλληλα δομημένη και με συγκεκριμένο τρόπο ώστε σε πιθανές μελλοντικές αλλαγές και με τη κατάλληλη τροποποίηση να μπορεί να χρησιμοποιηθεί σε διάφορες περιπτώσεις χρήσης που περιγράφουμε εντός της εργασίας. Επιπλέον η δομή της συγκεκριμένης εφαρμογής αποτελείται από: Ανάγκη ➔Ανάλυση Ανάγκης ➔ Δημιουργία της κατάλληλης εφαρμογής ➔ Έλεγχος εφαρμογής ➔ Επιβεβαίωση ορθής λειτουργίας ➔ Χρήση εφαρμογής σε πραγματικές συνθήκες.

Λέξεις κλειδιά: Ubiquitous computing, Ubicomp, Διάχυτος υπολογισμός, Πανταχού παρούσα υπολογιστική, Πανταχού παρών υπολογισμός, Βιομετρική τεχνολογία, Εκπαιδευτική εφαρμογή, Μικρό-Επεξεργαστής, Προγραμματισμός, Βάση δεδομένων, Δεδομένα, Βιομετρικά χαρακτηριστικά, Δακτυλικό αποτύπωμα.

ABSTRACT

This thesis concerns the development of educational applications using ubiquitous computing.

The basic function of the application in combination with the appropriate electronic equipment, through the development and application of biometric technology is the identification of natural persons and facilitating their access to the services of the Institute of Technology. Our goal is to create a service of innovative technology for educational purposes that will make accessibility to the services of the Institute of Technology immediate and at the same time valid in terms of service for both students and employees. We developed an application in a way so future changes and modifications can be used in various use cases as described in the paper. In addition, the structure of this application consists of: Needs➔Needs Analysis➔Analysis, Design and Creation of the appropriate application➔Testing of the application➔Confirmation of correct operation Use of the application in real-life conditions.

Keywords.

Keywords: Ubiquitous computing, Ubicomp, Pervasive computing, Biometric technology, educational application, Microprocessor, Programming, Database, Data.

Περιεχόμενα

Εισαγωγή

Αρχικά αυτή η εργασία θέλει να μας αποδείξει πως με τη χρήση κατάλληλου εξοπλισμού και γλώσσας προγραμματισμού, μπορεί να επιτευχθεί η ταυτοποίηση ατόμου με χρήση των μοναδικών του βιομετρικών χαρακτηριστικών και συγκεκριμένα το δακτυλικό αποτύπωμα. Η χρήση εφαρμογής θα επιλύσει την ανάγκη ταυτοποίησης του ατόμου με ταχύτητα, ακρίβεια και αποτελεσματικότητα.

Το πρώτο κεφάλαιο το οποίο είναι και εισαγωγικό γίνεται αναφορά στο αντικείμενο και το σκοπό της πτυχιακής εργασίας. Επιπλέον γίνεται αναφορά στην ανάγκη ανάπτυξης μιας εφαρμογής με χρήση διάχυτου υπολογισμού η οποία θα διευκολύνει τις υπηρεσίες και τη λειτουργία του Ιδρύματος. Επίσης αναλύουμε τη μεθοδολογία που ακολουθήσαμε καθώς και τους περιορισμούς που προκύπτουν.

Το δεύτερο κεφάλαιο μας παραθέτει την έννοια του διάχυτου υπολογισμού. Το συγκεκριμένο κεφάλαιο είναι αφιερωμένο στο τι είναι ο διάχυτος υπολογισμός και το εξηγεί αναλυτικά και με λεπτομέρειες. Επιπλέον γίνεται αναφορά στον διάχυτο υπολογισμό σήμερα καθώς και στην ιστορία των υπολογιστών.

Το τρίτο κεφάλαιο περιγράφει τη πλακέτα ανοιχτού κώδικα – Arduino. Περιγράφεται η συνδεσιμότητα του με τη χρήση υπολογιστικής πλατφόρμας IDE, που γίνεται μέσω γλώσσας προγραμματισμού. Βάση αυτής της πλατφόρμας πραγματοποιείται η επικοινωνία του Arduino και των επιμέρους τμημάτων που μπορούν να συνδεθούν πάνω στη πλακέτα.

Το τέταρτο κεφάλαιο γίνεται εισαγωγή στην έννοια των προσωπικών δεδομένων. Ακολουθεί μια εκτενής ανάλυση αυτών, καθώς και το ισχύων ευρωπαϊκό νομοθετικό πλαίσιο περί προστασίας προσωπικών δεδομένων – GDPR. Επιπλέον, περιγράφονται τα μέτρα προστασίας για επιθέσεις μέσω κακόβουλων λογισμικών και την αποτροπή διαρροής προσωπικών δεδομένων προς τρίτους.

Το πέμπτο κεφάλαιο αναφέρει τα βιομετρικά χαρακτηριστικά και συγκεκριμένα το δακτυλικό αποτύπωμα. Μας περιγράφει το μοναδικό αυτό χαρακτηριστικό καθώς και τα σενάρια εφαρμογής του στη διάρκεια σπουδών των φοιτητών ως μέσο ταυτοποίησης.

Το έκτο κεφάλαιο αναλύει την ανάπτυξη και υλοποίηση της εφαρμογής e-Sitisi. Περιγράφει τη χρήση ειδικού αισθητήρα για την καταγραφή δακτυλικού αποτυπώματος,

την αποθήκευση των δεδομένων σε βάση δεδομένων SQLite και την διαδικασία ταυτοποίησης.

Το έβδομο κεφάλαιο υπάρχει για την αναφορά των αποτελεσμάτων της ανάπτυξης της εφαρμογής και για μελλοντικές αναβαθμίσεις.

Κεφάλαιο 1ο : Αντικείμενο και σκοπός της πτυχιακής εργασίας

1.1 Σύστημα ταυτοποίησης

Το αντικείμενο της πτυχιακής εργασίας αφορά ένα ολοκληρωμένο σύστημα ταυτοποίησης ατόμου, χρησιμοποιώντας ως μέσο αναγνώρισης το δακτυλικό αποτύπωμα του φοιτητή. Με αυτό το σύστημα ταυτοποίησης θα μπορεί να γίνει η εξακρίβωση στοιχείων του φοιτητή καθώς και οι φοιτητική του ιδιότητα στις υπηρεσίες του ιδρύματος. Η ανάγκη για τη δημιουργία αυτής της εφαρμογής προήλθε έπειτα από το μεγάλο αριθμό φοιτητών και το μεγάλο φόρτο εργασίας που καλείτε το προσωπικό του ιδρύματος να ανταπεξέλθει. Με την υπάρχουσα υποδομή ο φοιτητής καλείτε να αποδεικνύει με τη χρήση της φοιτητικής του ταυτότητας τα στοιχεία του σε όλες υπηρεσίες του ιδρύματος (γραμματεία, εργαστηριακά μαθήματα, σίτιση, βιβλιοθήκη, εξετάσεις, κτλ).

1.2 Μειονεκτήματα

Τα μειονεκτήματα που συναντάμε με τη χρήση της φοιτητικής ταυτότητας είναι:

1. Απώλεια / κλοπή της κάρτας
2. Κίνδυνος απώλειας προσωπικών δεδομένων
3. Χρήση κάρτας από μη εξουσιοδοτημένα άτομα
4. Χωρίς την επίδειξη της κάρτας, δεν μπορεί το άτομο να αποδείξει τη φοιτητική του ιδιότητα.

1.3 Δημιουργία εφαρμογής

Για την επίλυση και αντιμετώπιση των παραπάνω συνθηκών απαιτείται η δημιουργία μιας εφαρμογής η οποία θα αποδεικνύει τη φοιτητική ιδιότητα του ατόμου, χωρίς τη χρήση κάποιου επίσημου εγγράφου παρά μόνο με το δακτυλικό αποτύπωμα.

1.4 Πλεονεκτήματα

Τα πλεονεκτήματα που συναντάμε με τη χρήση της εφαρμογής είναι:

1. Δεν θα απαιτείται η χρήση εγγράφου
2. Δεν θα υπάρχει κίνδυνος κλοπής ή απώλειας του φοιτητικού πάσο
3. Δεν θα υπάρχει ψευδής ταυτοπροσωπία
4. Γρήγορη και εύκολη πρόσβαση σε χώρους και υπηρεσίες του ιδρύματος χωρίς την ανάγκη φυσικής παρουσίας προσωπικού

Σύμφωνα με τα παραπάνω δεδομένα και ανάγκες, δημιουργήσαμε μια εφαρμογή η οποία με τη χρήση της τεχνολογίας του διάχυτου υπολογισμού θα διαβάζει το δακτυλικό αποτύπωμα μέσω ενός αισθητήρα και θα πραγματοποιείται ταυτοποίηση στοιχείων μέσω μίας βάσης δεδομένων. Απαραίτητη προϋπόθεση είναι να γίνει εγγραφή του δακτυλικού αποτυπώματος και των στοιχείων του ατόμου στην εφαρμογή.

Καθώς η εφαρμογή δημιουργήθηκε για ερευνητικούς και εκπαιδευτικούς σκοπούς, υπάρχει περιορισμός στο υλικό που χρησιμοποιήθηκε. Ενδεικτικά υπάρχει χαμηλή επεξεργαστική ισχύ, μικρή βάση δεδομένων του αισθητήρα που χρησιμοποιήσαμε με αποτέλεσμα να υπάρχει όριο 20 εγγραφών.

Κεφάλαιο 2ο : Διάχυτος υπολογισμός

2.1 Έννοια διάχυτου υπολογισμού (Ubiquitous Computing)

Διάχυτος υπολογισμός ή αλλιώς Ubiquitous Computing (Weiser, 1991)

Τι είναι ο διάχυτος υπολογισμός ή όπως θα λέγαμε πανταχού παρόν υπολογισμός ;

Με τον όρο διάχυτος – Ubiquitous , μπορούμε να ορίσουμε την έννοια ή οντότητα , η οποία βρίσκεται παντού τριγύρω μας και μας περικλείει, διαχέεται ανάμεσά μας και αλληλοεπιδρά μαζί μας με ή χωρίς να το καταλαβαίνουμε. Για να γίνει ο όρος πιο κατανοητός φέρνουμε τα παρακάτω παραδείγματα με τα οποία μπορούμε να ορίσουμε την έννοια του διάχυτου υπολογισμού:

1. Η διάχυτη επίδραση ενός ηγέτη από ένα τμήμα μπορεί να γίνει αντιληπτή από όλους τους εργαζομένους και να επηρεάσει θετικά στις επιδόσεις αυτών.

2. Το φως της λάμπας διαχέεται στο χώρο αρμονικά με ένα όμορφο τελικό αποτέλεσμα.

3. Η μυρωδιά των λουλουδιών διαχέεται στο σπίτι και προσφέρει μια ευωδία σε όλα τα άτομα.

4. Το σήμα του ασύρματου δικτύου Wi-Fi διαχέεται στο κτήριο με αποτέλεσμα να έχουν όλοι ασύρματη σύνδεση στο διαδίκτυο δεν βλέπουμε τη συχνότητα του ασύρματο δικτύου με την όρασή μας αλλά υπάρχει στο χώρο διότι όλες οι συσκευές μπορούν να συνδεθούν σε αυτό.

Ο διάχυτος υπολογισμός σαν έννοια πληροφορικής (Academic Discipline) και σαν μελέτη αυτής της έννοιας, αναφέρθηκε για πρώτη φορά από τον Mark Weiser[1].

Mark Weiser[2]

Η Ubiquitous Computing (Kang, January 2007)[3] (ή "ubicomp") είναι μια έννοια της μηχανικής λογισμικού και της επιστήμης των υπολογιστών όπου η πληροφορική είναι παρούσα οποιαδήποτε στιγμή και οπουδήποτε. Σε αντίθεση με τους σταθερούς υπολογιστές, η πανταχού παρούσα πληροφορική μπορεί να εμφανιστεί χρησιμοποιώντας οποιαδήποτε συσκευή, σε οποιαδήποτε τοποθεσία και σε οποιαδήποτε μορφή. Ο χρήστης αλληλοεπιδρά με τον υπολογιστή, ο οποίος μπορεί να υπάρχει σε πολλές διαφορετικές μορφές, όπως φορητούς υπολογιστές, tablet και τερματικά, αλλά και σε καθημερινά αντικείμενα, όπως ψυγείο ή ένα ζευγάρι γυαλιά. Οι

[1] "The Computer for the 21st Century" - Scientific American Special Issue on Communications, Computers, and Networks, September, 1991

The Computer for the 21st Century

Specialized elements of hardware and software, connected by wires, radio waves and infrared, will be so ubiquitous that no one will notice their presence

By Mark Weiser

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

[3] https://en.wikipedia.org/wiki/Ubiquitous_computing as seen on 1/6/2022

τεχνολογίες για την υποστήριξη της πανταχού παρούσης πληροφορικής περιλαμβάνουν το διαδίκτυο, λειτουργικά συστήματα, κώδικα κινητής τηλεφωνίας, αισθητήρες, μικροεπεξεργαστές, νέες διεπαφές I/O, δίκτυα υπολογιστών, πρωτόκολλα κινητής τηλεφωνίας, ανίχνευση και τοποθέτηση και νέα υλικά.

The third wave of Computing

“ Ubiquitous computing names the third wave in computing, just now beginning. First were mainframes, each shared by lots of people. Now we are in the personal computing era, person and machine staring uneasily at each other across the desktop. Next comes ubiquitous computing, or the age of calm technology, when technology recedes into the background of our lives. ”

— Mark Weiser[4][5][6]

Ο Mark Weiser είχε ορίσει ως διάχυτο υπολογισμό ή πανταχού παρούσα πληροφορική το τρίτο κύμα πληροφορικής των σύγχρονων υπολογιστών. Το πρώτο κύμα ήταν οι κεντρικοί υπολογιστές ή Mainframe computers όπου το κάθε μηχάνημα χρησιμοποιούνταν από κοινού από πολλούς ανθρώπους. Το δεύτερο στάδιο είναι αυτό όπου κάθε άτομο έχει το δικό του υπολογιστή με τη δική του επιφάνεια εργασίας - χώρο και για να δουλέψει σε αυτό το μηχάνημα πρέπει να υπάρχει η φυσική παρουσία του ατόμου μπροστά από τον υπολογιστή (βλέπε πληκτρολόγιο-ποντίκι-οθόνη. Και ως διάχυτο υπολογισμό ή τρίτο κύμα σύγχρονης πληροφορικής, αναφέρει την εποχή της ήρεμης τεχνολογίας, δηλαδή ότι η τεχνολογία εισχωρεί στις ζωές μας και στην καθημερινότητα μας χωρίς να γίνεται αντιληπτή από εμάς τους ίδιους.

Η αλληλεπίδραση των ανθρώπων και των συσκευών με χρήση διάχυτου υπολογισμού.

[4] http://www.downloadmela.com/1996-computer-science-challenges-for-the-next-10-years-2%3C%3EqWcatlIVH1jN3IYOWM02Aw%3D%3D

[5] **Description:**
Dr. Mark. Weiser, Chief Technologist for the Xerox Corporation presents "Computer Science Challenges in the next 10 years." November 1, 1996. Topics include Visualization, gaming theory, nanotechnology, parallel systems, connectivity, and human factors.

[6] (Είχε προβλέψει το τρίτο κύμα υπολογιστών) Weiser, Mark (November 1, 1996). "Computer Science Challenges for the Next 10 Years"

2.2 Ιστορία υπολογιστών

Η ιστορία των υπολογιστών μέχρι να περάσουμε στο διάχυτο υπολογισμό περνάει από κάποια στάδια, για αυτά τα στάδια ή κύματα τεχνολογίας οι απόψεις διίστανται από πολλούς επιστήμονες της πληροφορικής και πολλές φορές αποτελούν προσωπική άποψη. Στην παρούσα πτυχιακή εργασία θα προσπαθήσουμε να ξεχωρίσουμε αυτά τα στάδια και να τα οργανώσουμε σε ομάδες μέχρι να φτάσουμε στο τελευταίο στάδιο ή κύμα υπολογιστών. Ο Mark Weise έζησε στο 3ο κύμα σύγχρονων υπολογιστών σε αυτό που αναφέρει τον διάχυτο υπολογισμό σαν μια μορφή υπολογιστών. Όμως λόγω του ότι η τεχνολογία έχει κάνει τεράστια άλματα και προόδους θα μπορούσαμε να πούμε ότι διανύουμε το 5 στάδιο το οποίο αποτελεί το I.O.T ή αλλιώς το Internet Of Things και ο διάχυτος υπολογισμός αποτελεί μέρος αυτού.

Για να καταλάβουμε και να κατανοήσουμε καλύτερα τα στάδια και τις κατηγορίες των υπολογιστών και της τεχνολογίας της πληροφορίας κάνουμε μια ιστορική αναδρομή από την αρχαιότητα μέχρι την σύγχρονη εποχή που θα αναλύσουμε τον Διάχυτο Υπολογισμό.

2.3 Η γενιά «0»

Η γενιά 0 : Η πρώτη μορφή υπολογιστή ή τρόπος υπολογισμού.

2.3.1 Ο αρχαίος Άβακας:

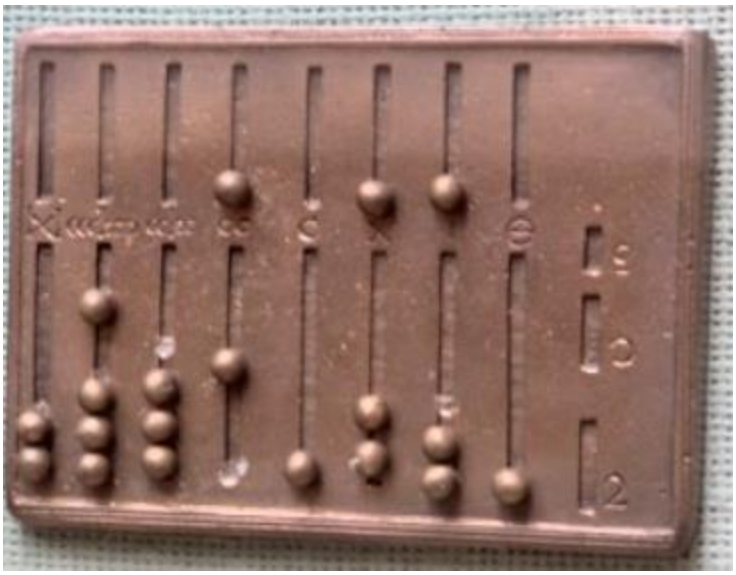
Στην αρχαιότητα περίπου το 2200 π.Χ οι Βαβυλώνιοι ανέπτυξαν ένα όργανο υπολογισμού που τους βοηθούσε να κάνουν υπολογισμούς κυρίως προσθέσων και αφαιρέσεων, αυτό τους βοηθούσε πολύ στους υπολογισμούς τους λόγω του ανεπτυγμένου εμπορίου τους. Τον άβακα τον χρησιμοποιούσαν αρκετά και οι αρχαίοι Έλληνες όπως και οι Ρωμαίοι, επίσης η χρήση του συνεχίστηκε και στην Ευρώπη καθώς και στη διάρκεια των μεσαιωνικών χρόνων. Στη σύγχρονη εποχή το χρησιμοποιούν ακόμη σε αρκετά σχολεία κυρίως στις πρώτες τάξεις του δημοτικού καθώς βοηθάει σημαντικά τις μικρές ηλικίες να κατανοήσουν την αριθμητική και την αφαίρεση – πρόσθεση.

Ανακατασκευή ρωμαϊκού άβακα.[7][8][9]

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

[8] https://upload.wikimedia.org/wikipedia/commons/b/b5/RomanAbacusRecon.jpg
[9] https://el.wikipedia.org/wiki/%CE%86%CE%B2%CE%B1%CE%BA%CE%B1%CF%82

Photographer: Mike Cowlishaw (aus der englischen Wikipedia)
Reconstruction of a Roman Abacus, made by the RGZ Museum in Mainz, 1977 The original is bronze and is held by the Bibliothèque nationale de France, in Paris.



2.3.2 Μηχανισμός των Αντικυθήρων 150 π.Χ. - 100 π.Χ:

Ο μηχανισμός των Αντικυθήρων, γνωστός και ως χρονολόγιο των Αντικυθήρων, υπολογιστής των Αντικυθήρων ή αστρολάβος των Αντικυθήρων, είναι ένα αρχαίο τέχνεργο, το οποίο λειτουργούσε ως αναλογικός, μηχανικός υπολογιστής και όργανο αστρονομικών παρατηρήσεων και παρουσιάζει ομοιότητες με πολύπλοκο ωρολογιακό μηχανισμό. Ο μηχανισμός είναι το αρχαιότερο σωζόμενο μηχάνημα με γρανάζια και μάλιστα ιδιαίτερα πολύπλοκης λειτουργίας. Είναι κατασκευασμένος από μπρούντζο και βρισκόταν εγκιβωτισμένος σε ξύλινο πλαίσιο. Έχει προβληματίσει και συναρπάσει πολλούς ιστορικούς της επιστήμης και της τεχνολογίας αφότου ανακαλύφθηκε. Ήταν ένας αναλογικός υπολογιστής, σχεδιασμένος για να υπολογίζει και να απεικονίζει τις κινήσεις των ουράνιων σωμάτων όπως τις εκλείψεις ηλίου και σελήνης. Πρόσφατες λειτουργικές ανακατασκευές της συσκευής τεκμηριώνουν αυτή την ανάλυση. Από τις πρόσφατες έρευνες καταρρίφθηκε η θεωρία ότι εμπεριέχει ένα διαφορικό γρανάζι, όμως ο μηχανισμός της κίνησης της σελήνης είναι ακόμα εντυπωσιακότερος, καθώς δίνει τη δυνατότητα μεταβλητής γωνιακής ταχύτητας στον άξονα που κινεί τη σελήνη (δεύτερος Νόμος Κέπλερ).

Το κύριο θραύσμα του μηχανισμού. Αθήνα, Εθνικό Αρχαιολογικό Μουσείο.[10][11]

[10] https://en.wikipedia.org/wiki/Antikythera_mechanism
[11] https://commons.wikimedia.org/wiki/File:NAMA_Machine_d%27Anticyth%C3%A8re_1.jpg

2.3.3 Υπολογιστές στην περίοδο της Αναγέννησης:

Η μηχανή του Pascal[12][13][14], 1645: Ο Γάλλος μαθηματικός Μπλεζ Πασκάλ (Blaise Pascal) κατασκεύασε το 1645 την πρώτη αληθινή αριθμομηχανή, η οποία επονομάστηκε Πασκαλίνα (Pascaline). Με τη μηχανή αυτή μπορούσε κάποιος να κάνει (σχετικά) εύκολα μαθηματικούς υπολογισμούς. Η μηχανή του Pascal είχε τροχαλίες, τις οποίες, όταν περιέστρεφε ο χρήστης εμφάνιζαν τα αποτελέσματα. Η μηχανή είχε μικρές διαστάσεις και μπορούσε εύκολα να χωρέσει σε ένα μικρό τραπέζι. Ο αρχικός «υπολογιστής» είχε πέντε γρανάζια (με αποτέλεσμα να μπορεί να κάνει υπολογισμούς με σχετικά μικρούς αριθμούς), αλλά κατασκευάστηκε και σε παραλλαγές με έξι και οκτώ γρανάζια. Η μηχανή εκτελούσε δύο πράξεις, πρόσθεση και αφαίρεση. Στο επάνω μέρος υπήρχε μια σειρά από οδοντωτούς τροχούς (γρανάζια), που το καθένα περιείχε τους αριθμούς από 0 έως 9. Ο πρώτος τροχός συμβόλιζε τις μονάδες, ο δεύτερος τις δεκάδες, ο τρίτος τις εκατοντάδες, κ.ο.κ.

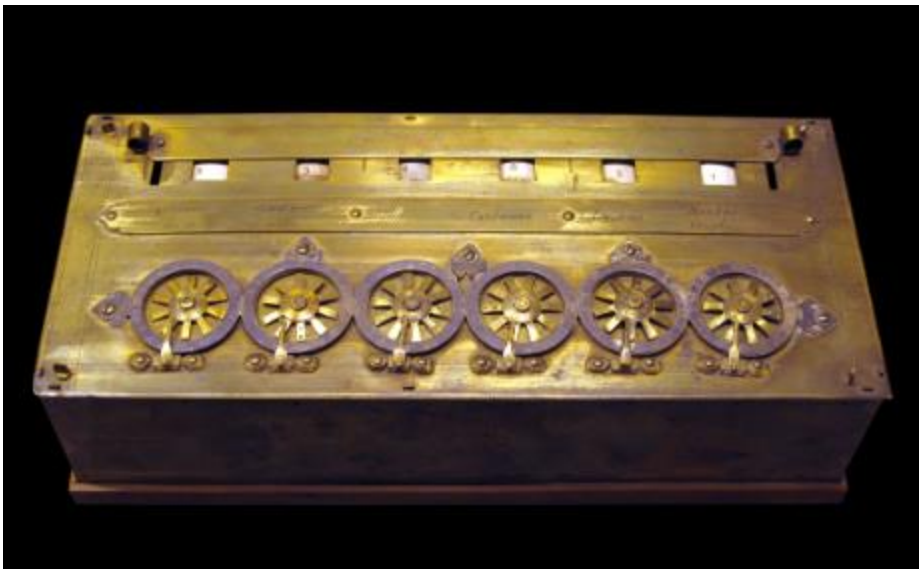
---

[12] https://en.wikipedia.org/wiki/Pascal%27s_calculator#/media/File:Pascaline-CnAM_823-1-IMG_1506-black.jpg

[13] https://el.wikipedia.org/wiki/%CE%99%CF%83%CF%84%CE%BF%CF%81%CE%AF%CE%B1_%CF%84%CF%89%CE%BD_%CF%85%CF%80%CE%BF%CE%BB%CE%BF%CE%B3%CE%B9%CF%83%CF%84%CF%8E%CE%BD#%CE%97_%CE%BC%CE%B7%CF%87%CE%B1%CE%BD%CE%AE_%CF%84%CE%BF%CF%85_Pascal%2C_1645

[14] https://en.wikipedia.org/wiki/Pascal's_calculator

By Rama, CC BY-SA 3.0 fr, https://commons.wikimedia.org/w/index.php?curid=53246694



Στην σύγχρονη εποχή στην οποία ανήκει και η κατηγορία υπολογιστών του Διάχυτου υπολογισμού, οι κατηγορίες σύγχρονων υπολογιστών χωρίζονται σε 4 γενιές με πολλές αναφορές να γίνονται για μια 5η γενιά υπολογιστών αυτήν των κβαντικών υπολογιστών και της τεχνητής νοημοσύνης. Όπως αναφέρεται και προηγουμένος ο διάχυτος υπολογισμός ανήκει στη 4η και 5η σύγχρονη γενιά και ταυτόχρονα έρχεται και δένει με τη τεχνητή νοημοσύνη στο περιβάλλον του I.O.T (το οποίο βασίζεται στο διάχυτο υπολογισμό), προσφέροντας στον άνθρωπο υπηρεσίες που βελτιώνουν στο έπακρο την ποιότητα ζωής του σε όλους τους τομείς.

2.4 1η γενιά Υπολογιστών (1943 – 1956)

Γενιά Υπολογιστών (1943 – 1956)

Κατά την διάρκεια του Β' Παγκοσμίου Πολέμου ο μαθηματικός Άλαν Τιούρινγκ σχεδίασε μια μηχανή όπου μπορούσε να λύσει οποιοδήποτε πρόβλημα με την μορφή αλγορίθμου. Η μηχανή Τιούρινγκ[15] ήταν η αρχική ιδέα για την κατασκευή του ENIAC όμως δεν μπορούσε τότε να κατασκευαστεί. Παράλληλα κατασκευάζει τον Colossus Mark I το 1943,

[15] https://en.wikipedia.org/wiki/History_of_computing_hardware

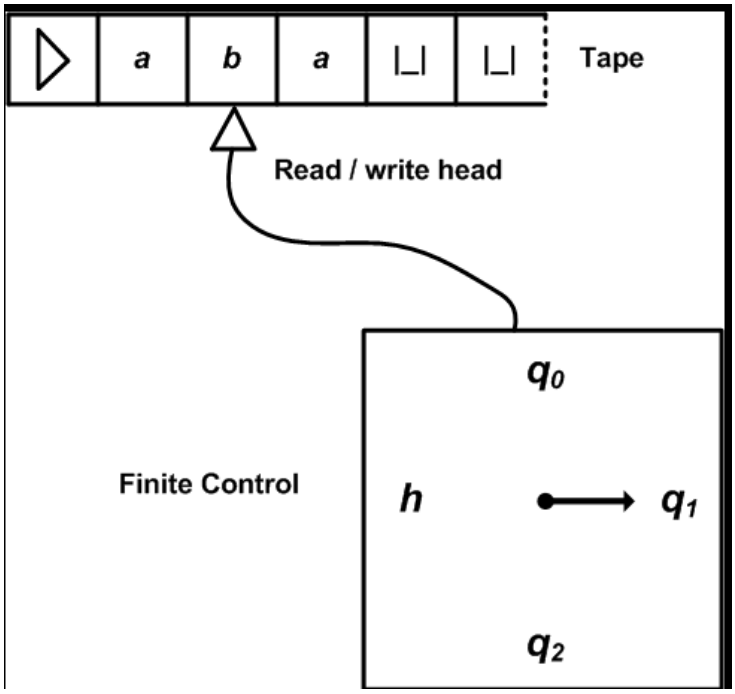
τον πρώτο προγραμματιζόμενο ηλεκτρονικό υπολογιστή, για την αποκρυπτογράφηση της γερμανικής μηχανής Enigma στον Β' Παγκόσμιο πόλεμο. Αργότερα το 1946, μετά το τέλος του Β' Παγκοσμίου Πολέμου, οι Ηνωμένες Πολιτείες χρειάζονταν μια συσκευή η οποία να βοηθά τους στρατιωτικούς στους υπολογισμούς για να βρίσκουν τα όπλα τους το στόχο με μεγαλύτερη ακρίβεια. Ο ENIAC ικανοποιώντας τα κριτήρια του Τιούρινγκ υπήρξε ο πρώτος επαναπρογραμματιζόμενος ηλεκτρονικός υπολογιστής γενικού σκοπού. Για πρώτη φορά δημιουργήθηκε ένα τεράστιο μηχάνημα που αντί για μηχανικά μέρη χρησιμοποιούσε ηλεκτρονικές λυχνίες, κατασκευασμένες από τον Λι Ντε Φόρεστ (Lee De Forest).

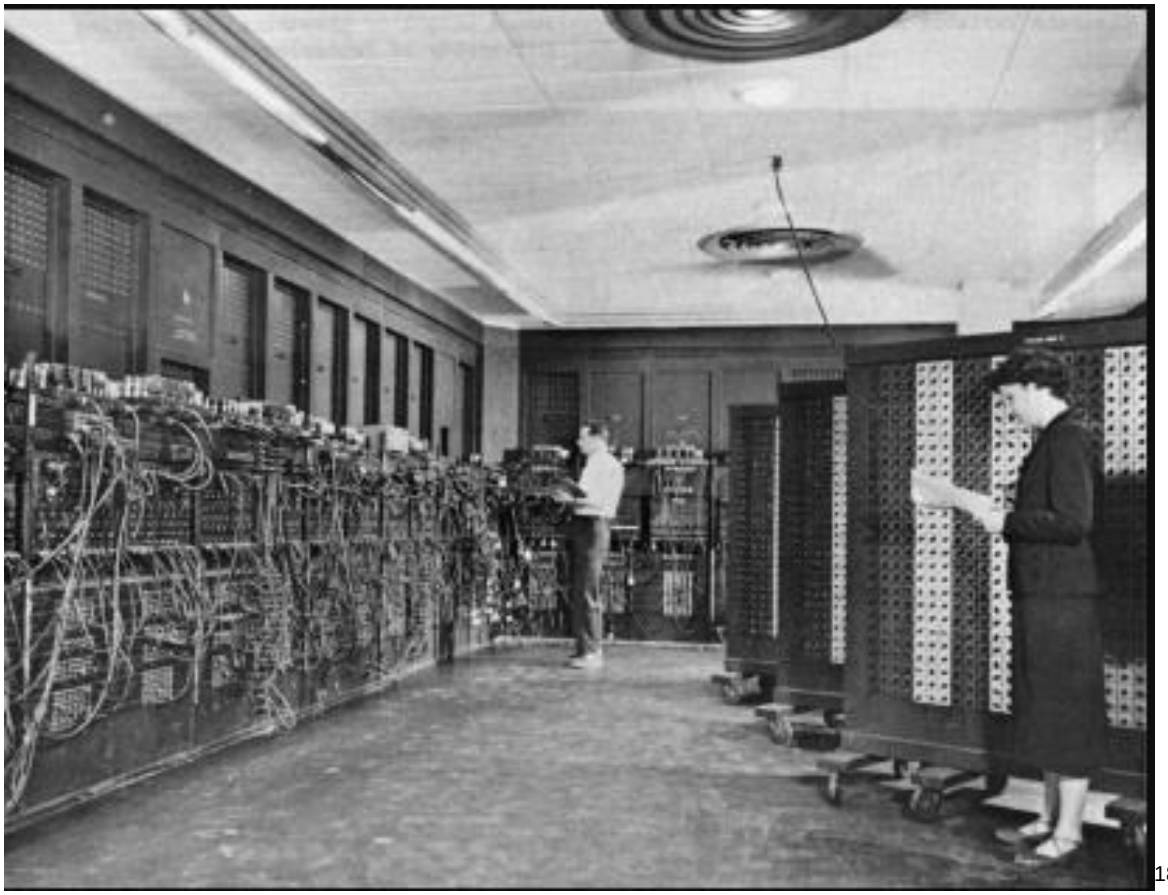
Ο πρώτος επαναπρογραμματιζόμενος ηλεκτρονικός υπολογιστής, λειτουργούσε με λυχνίες και ονομάστηκε ENIAC. Ο ENIAC[16] ήταν τεράστιος σε μέγεθος (καταλάμβανε έναν ολόκληρο όροφο), και έπρεπε να τον ελέγχουν συνεχώς ειδικοί επιστήμονες. Συχνά επίσης, καίγονταν οι λυχνίες του και έπρεπε να τις αντικαθιστούν. Ακόμα και ο πιο ταπεινός σημερινός υπολογιστής είναι χιλιάδες φορές καλύτερος από τον ENIAC ως προς τις δυνατότητες.

Βασική γραφική αναπαράσταση μιας μηχανής Turing [17]

[16] https://upload.wikimedia.org/wikipedia/commons/4/4e/Eniac.jpg

[17] https://el.wikipedia.org/wiki/%CE%9C%CE%B7%CF%87%CE%B1%CE%BD%CE%AE_%CE%A4%CE%BF%CF%8D%CF%81%CE%B9%CE%BD%CE%B3%CE%BA#/media/%CE%91%CF%81%CF%87%CE%B5%CE%AF%CE%BF:Turing_machine.png
Από Kevd1337 - Έργο αυτού που το ανεβάζει, Κοινό Κτήμα, https://commons.wikimedia.org/w/index.php?curid=3916419


a
b
a
|_|
|_|
Tape
Read / write head
Finite Control
q_0
h
q_1
q_2

 1819

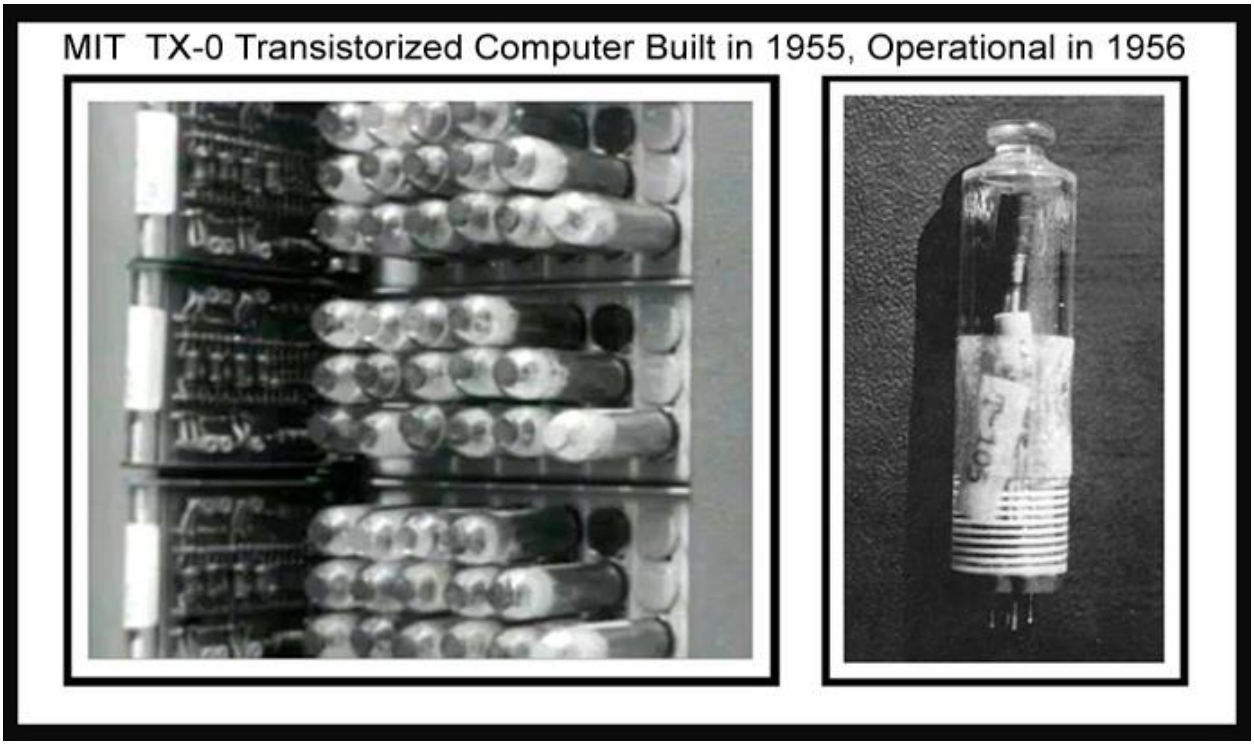
[18] ENIAC (Electronic Numerical Integrator And Computer) in Philadelphia, Pennsylvania. Glen Beck (background) and Betty Snyder (foreground) program the ENIAC in building 328 at the Ballistic Research Laboratory (BRL). This image is a work of a U.S. Army soldier or employee, taken or made as part of that person's official duties. As a work of the U.S. federal government, the image is in the public domain.

[19] https://commons.wikimedia.org/wiki/File:Eniac.jpg
Από Άγνωστος - U.S. Army Photo, Κοινό Κτήμα, https://commons.wikimedia.org/w/index.php?curid=55124

2.5 2η γενιά Υπολογιστών (1956 - 1963)

Γενιά των Υπολογιστών (1956 - 1963)

Την περίοδο αυτή οι λυχνίες αντικαθίστανται από τρανζίστορς. Οι ηλεκτρονικές αυτές κατασκευές (κρυσταλλοτρίοδοι, όπως τις ονομάζουν οι ηλεκτρονικοί), επιτρέπουν τη δημιουργία μικρότερων και ταχύτερων υπολογιστών. Το 1956 στο Τεχνολογικό Ινστιτούτο Μασαχουσέτης (M.I.T.) κατασκευάστηκε ο πρώτος Ηλεκτρονικός Υπολογιστής που λειτουργούσε με τρανζίστορς, ο TX-0[2021].



The MIT TX-0 Transistorized computer used 3600 Philco surface-barrier transistors in its circuitry. These transistors were encapsulated in plug-in vacuum tubes, which made it easier for testing, evaluation, and replacement.

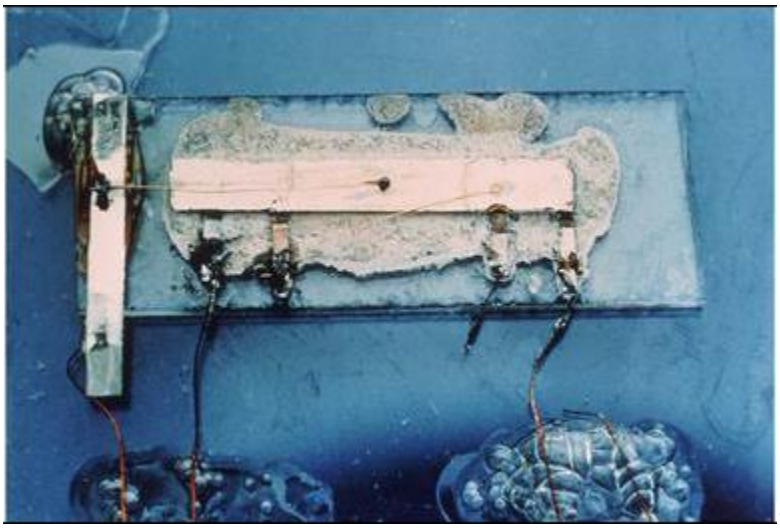
[20] https://en.wikipedia.org/wiki/TX-0#/media/File:MIT_TX-0_computer_Philco_surface-barrier_transistors.JPG By Historianbuff - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=25915303 CC BY-SA 3.0

[21] https://en.wikipedia.org/wiki/TX-0 as seen on 1/6/2022

2.6 3η γενιά Υπολογιστών (1964- 1971)

Γενιά των υπολογιστών(1964- 1971)

Το 1958 ο Τζακ Κίλμπυ (Jack Kilby[22]), της εταιρείας Texas Instruments κατάφερε να δημιουργήσει κάτι που θα άλλαζε τον κόσμο των ηλεκτρονικών για πάντα, κατασκεύασε το πρώτο Ολοκληρωμένο Κύκλωμα συνδυάζοντας τρανζίστορς, πυκνωτές, αντιστάτες και άλλα ηλεκτρονικά εξαρτήματα όλα τοποθετημένα στο ίδιο κομμάτι από πυρίτιο. Το δημιούργημα του Κίλμπυ επέτρεψε στους επιστήμονες να κατασκευάσουν υπολογιστές τόσο μικρούς ώστε να μπορούμε ακόμη και να τους μεταφέρουμε. Χρησιμοποιείται, επίσης, σε μια πληθώρα άλλων εφαρμογών, όπως τηλεπικοινωνίες, πολυμέσα, ακόμη και παιχνίδια.

 [23]

The first working integrated circuit was created by Jack Kilby in 1958. It contains a single transistor and supporting components on a slice of germanium and measures 1/16 by 7/16 inches (1.6 x 11.1 mm).

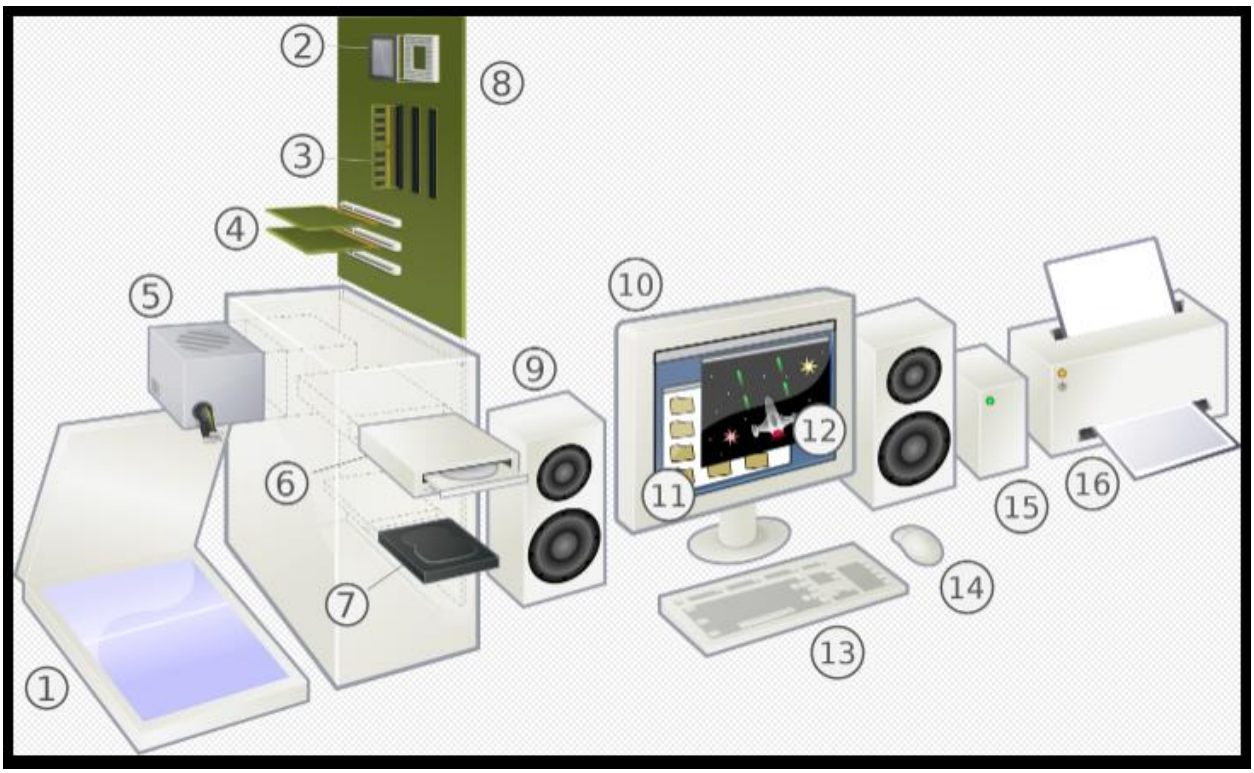
[22] https://en.wikipedia.org/wiki/Jack_Kilby

[23] By http://www.ti.com/corp/graphics/press/image/on_line/co1034.jpg, Fair use, https://en.wikipedia.org/w/index.php?curid=19287598 https://en.wikipedia.org/wiki/Jack_Kilby#/media/File:Kilby_solid_circuit.jpg

2.7 4η γενιά Υπολογιστών (1971 – σήμερα)

Γενιά των υπολογιστών (1971 - σήμερα)

Οι υπολογιστές που έχουμε σήμερα ανήκουν στην 4η Γενιά. Ο κάθε ένας από αυτούς είναι εφοδιασμένος με Επεξεργαστή (CPU), έχει τη δική του Μνήμη, μονάδα αποθήκευσης πληροφοριών, οθόνη, και κάποιο είδος μέσου για να δίνουμε πληροφορίες στον υπολογιστή (πληκτρολόγιο, πενάκι, ποντίκι κλπ).



24

[24] https://en.wikipedia.org/wiki/Personal_computer#/media/File:Personal_computer,_exploded_6.svg
By User:HereToHelp - Self-published work by User:HereToHelp and File:Personal computer, exploded 5.svg, CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=4023664
https://en.wikipedia.org/wiki/History_of_computing_hardware

1. Scanner
2. CPU (microprocessor)
3. Memory (RAM)
4. Expansion cards (graphics cards, etc.)
5. Power supply
6. Optical disc drive
7. Storage (hard disk or SSD)
8. Motherboard
9. Speakers
10. Monitor
11. System software
12. Application software
13. Keyboard
14. Mouse
15. External hard disk
16. Printer

Στη παρούσα πτυχιακή εργασία θα μπορούσαμε να τοποθετήσουμε τον διάχυτο υπολογισμό στο 4ο και 5ο επίπεδο υπολογισμού ή 4ης γενιάς υπολογιστών τα οποία είναι εξοπλισμένα με τον δικό τους επεξεργαστή.

1. 1960 – 1970

Πρώτο επίπεδο ή στάδιο υπολογιστών , είναι οι Mainframe ή κεντρικοί κοινόχρηστοι υπολογιστές οι οποίοι συνήθως ανήκαν σε έναν οργανισμό ή εταιρία και παρείχαν την δυνατότητα χρήσης από πολλά άτομα (Συσχέτιση ή relationship 1 υπολογιστής προς πολλά άτομα), ήταν τεράστιοι σε όγκο ,αρκετές φορές κάλυπταν ολόκληρα δωμάτια ή και ορόφους σε εταιρίες και υπολογιστική τους ισχύς ήταν αρκετή για να κάνουν μικρές σε όγκο επεξεργασία δεδομένων. Η επεξεργαστική ισχύς αυτών των μηχανημάτων δεν μπορεί σε καμία περίπτωση να συγκριθεί και με τον απλούστερο προσωπικό υπολογιστή νέας τεχνολογίας και επεξεργαστών. Επίσης σε καμιά περίπτωση δεν πρέπει να γίνεται σύγκριση των τότε ογκώδη υπολογιστών με τους τωρινούς Supercomputers, των οποίων των μέγεθος και η επεξεργαστική ισχύ αφορά συνήθως επιστημονικούς σκοπούς και όχι απλές διεργασίες.

2. 1980 – μέσα 1990

Δεύτερο επίπεδο ή στάδιο υπολογιστών, σταθεροί υπολογιστές ή Desktop Computer και τα Laptop (Portable Personal Computer) σε αυτό το στάδιο αναφερόμαστε στον προσωπικό υπολογιστή ή Personal Computer εν συντομία P.C. Σε αυτή τη περίπτωση ένας υπολογιστής ανήκει και χρησιμοποιείται συνήθως από έναν χρήστη (συσχέτιση ή relationship 1 υπολογιστής προς 1 άτομο), αυτός ο υπολογιστής βρίσκεται σε μια θέση εργασίας ή γραφείο και αρκετές φορές υπάρχει και μια σύνδεση προς το διαδίκτυο με τη χρήση ενσύρματης σύνδεσης (παρουσία καλωδίου).

3. Αρχές 2000

Τρίτο επίπεδο ή στάδιο υπολογιστών, πλέον οι χρήστες χρησιμοποιούν παραπάνω από έναν υπολογιστή, αρχίζουν να εμφανίζονται στις αγορές τα περιβόητα SmartPhone ή έξυπνα κινητά τα οποία συνδυάζουν υπολογιστές, τηλεφωνία και άλλες τεχνολογίες πχ κάμερα, αν και οι πρώτες αναφορές για συσκευές που συνδύαζαν τηλεφωνία και πληροφορική έχουν αναφερθεί από το 1973 (Το 1973, ο Θεόδωρος Γεώργιος Παρασκευάκος κατοχύρωσε με δίπλωμα ευρεσιτεχνίας τις έννοιες του συνδυασμού νοημοσύνης, της επεξεργασίας δεδομένων και των οθονών οπτικής απεικόνισης με τα τηλέφωνα, περιγράφοντας έτσι τις κοινές πλέον δραστηριότητες των τραπεζικών συναλλαγών και την πληρωμή λογαριασμών κοινής ωφελείας μέσω τηλεφώνου και διατέθηκαν προς πώληση το 1994 βλέπε IBM Simon[25] Personal Communicator (εν συντομία IBM Simon), η πρώτη αναφορά για το έξυπνο κινητό είχε γίνει από την Ericson και ήταν το κινητό ERICSON GS 88 PENELOPE[26]. Σε αυτό το στάδιο οι χρήστες αρχίζουν να έχουν παραπάνω από έναν προσωπικούς υπολογιστές οι οποίοι μπορούν και συνδέονται στο διαδίκτυο και ασύρματα, πχ Laptops νεότερης τεχνολογίας με ασύρματη σύνδεση στο διαδίκτυο μέσω Wi-Fi ή δεδομένων κινητής τηλεφωνίας και τα έξυπνα κινητά στην αρχική τους μορφή τα οποία εκτός από τηλεφωνία συνδυάζουν και άλλες τεχνολογίες όπως ένας υπολογιστής. Σε αυτό το σημείο αρχίζει και κάνει την εμφάνιση του ο διάχυτος υπολογισμός για να καταλήξει στη μορφή που τον γνωρίζουμε σήμερα.

[25] https://en.wikipedia.org/wiki/IBM_Simon#/media/File:IBM_Simon_Personal_Communicator.png as seen on 1/6/2022

[26] https://ericssoners.wordpress.com/2016/06/13/gs88/ as seen on 1/6/2022

27

28

[27] https://en.wikipedia.org/wiki/IBM_Simon#/media/File:IBM_Simon_Personal_Communicator.png
By Bcos47 - File:IBM_SImon_in_charging_station.png, Public Domain,
https://commons.wikimedia.org/w/index.php?curid=20083707
[28] https://ericssoners.wordpress.com/2016/06/13/gs88/

2.8 Η γενιά του διάχυτου υπολογισμού

Γενιά υπολογιστών 2000 έως σήμερα

4ο επίπεδο ή στάδιο υπολογιστών. Ubiquitous computing / Διάχυτος ή κινητός υπολογισμός, ο τρόπος με τον οποίο ο περισσότερος κόσμος στη σημερινή εποχή αντιλαμβάνεται την χρήση των υπολογιστικών συστημάτων και της έννοιας του υπολογισμού συνήθως παραπέμπει στην εικόνα του ενσύρματου ή ασύρματου υπολογιστή είτε αυτό είναι Laptop είτε είναι Desktop, που διαθέτει πληκτρολόγιο, ποντίκι, οθόνη, πρόσβαση σε εκτυπωτή με κάποια καλωδιακή σύνδεση και μπορεί να συνδεθεί στο διαδίκτυο ασύρματα ή ενσύρματα και μπορούν να μοιραστούν να επεξεργαστούν και να παρουσιάσουν πληροφορίες σε άλλους τελικούς χρήστες μέσω διαφόρων συσκευών εξόδου πχ οθόνη ή εκτύπωση της πληροφορίας ή μέσω οπτικοακουστικών μέσων. Επίσης μπορούμε να αναφέρουμε τον διάχυτο υπολογισμό σαν ένα παράδειγμα για τη σχέση των ανθρώπων με τους υπολογιστές.

Με τη χρήση διάχυτου υπολογισμού οι συσκευές οι οποίες μας περικλείουν εναρμονίζονται με τη καθημερινότητα του ατόμου. Προσαρμοζόμαστε με τους υπολογιστές γινόμαστε ένα με αυτούς, συνήθως ξεχνάμε ότι εργαζόμαστε με αυτούς καθώς η πληροφορία διαχέεται μεταξύ χρήστη και συσκευής χωρίς να το καταλαβαίνουμε. Με το διάχυτο υπολογισμό η συσκευή με τη χρήση διάφορων τεχνολογιών μπορεί να αποκρίνεται και να αλληλοεπιδρά με το χρήστη.

Η πληροφορία διαχέεται στον ανθρώπινο κόσμο χωρίς όμως να γίνεται αντιληπτή και τα αποτελέσματα από τις αλλαγές δεν φαίνονται.

Π.χ μια smartphone συσκευή καθώς ο χρήστης πλησιάζει το ακουστικό στο αυτί ο ήχος μπορεί αυτόματα και προσαρμόζεται ανάλογα με τον εξωτερικό θόρυβο, επίσης όταν μια συσκευή την κρατάμε όρθια και τη γυρίζουμε στο πλάι αυτόματα η οθόνη προσαρμόζεται. Επιπλέον καθώς οδηγούμε και το όχημα στρίβει, αυτόματα η συσκευή προσαρμόζει τον χάρτη προς τη κατεύθυνση που κοιτάει το όχημα.

Ο πανταχού παρών ή διάχυτος υπολογισμός εμφανίζεται με ένα εντελώς διαφορετικό τρόπο. Συνεργάζεται και αξιοποιεί δυνατότητες ετερογενών συσκευών, οι οποίες όσο αφορά τουλάχιστον την μορφή και το σχήμα τους δεν έχουν καμία σχέση με τα υπολογιστικά συστήματα όπως περιεγράφηκαν παραπάνω. Μπορεί να βρίσκεται σε μικρά κυκλώματα (επεξεργαστές) μέσα σε έπιπλα, αυτοκίνητα, κινητά τηλέφωνα, PDA's, τηλεοράσεις, ψυγεία, πλυντήρια κλπ. Όλες αυτές οι συσκευές "τρέχουν" διαφορετικά λειτουργικά συστήματα, μπορούν να συνδέονται με άλλες συσκευές, διαθέτουν interfaces για εισαγωγή δεδομένων και φυσικά δυνατότητες παρουσίασης και αλληλεπίδρασης με τους τελικούς χρήστες. Η τέταρτη εποχή, αυτή του διάχυτου υπολογισμού, χαρακτηρίζεται από την εμφάνιση με εκρηκτικούς ρυθμούς, μικρών δικτυωμένων φορητών προϊόντων πληροφορικής με τη μορφή έξυπνων τηλεφώνων, προσωπικών ψηφιακών βοηθών (PDA), αλλά και ενσωματωμένων υπολογιστικών μονάδων σε πολλές συσκευές (πχ πλυντήρια ρούχων) με αποτέλεσμα την διαμόρφωση ενός νέου κόσμου, στον οποίο κάθε άτομο κατέχει και χρησιμοποιεί πολλούς ταυτόχρονα υπολογιστές. (JOHN, 2010)

Κεφάλαιο 3ο: Arduino

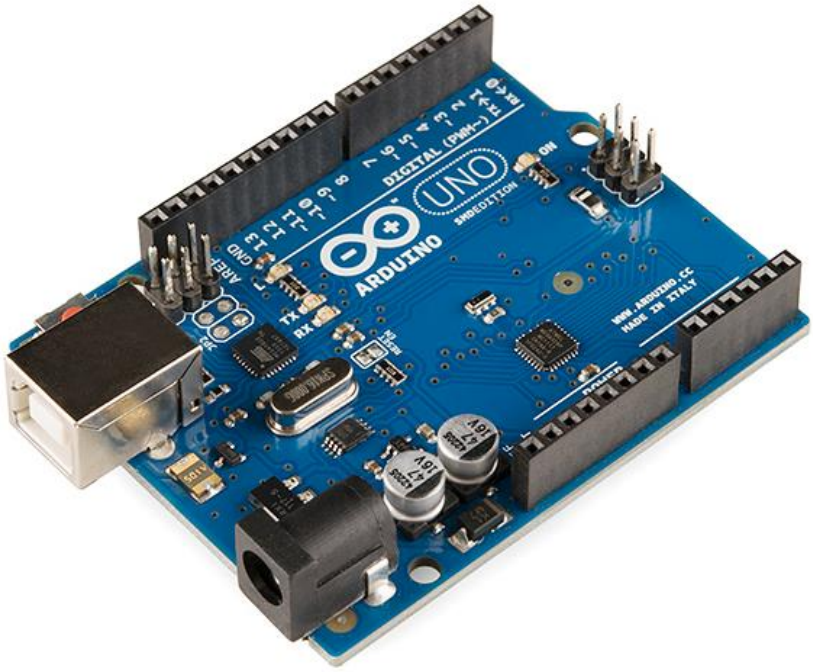
3.1 Η έννοια του Arduino

Τι είναι το Arduino

Το Arduino (Shiloh, 3rd Issue May 2015) είναι μια ανοιχτού κώδικα πλακέτα η οποία έχει ενσωματωμένο μικροελεγκτή με εισόδους/εξόδους και μπορεί να προγραμματιστεί με τη γλώσσα Wiring.

Η γλώσσα προγραμματισμού Wiring είναι στην ουσία η γλώσσα C++ που περιέχει ένα σύνολο από βιβλιοθήκες υλοποιημένες επίσης στην ίδια γλώσσα. Κάθε ολοκληρωμένο σύστημα Arduino αποτελείται από δύο τμήματα, το υλικό και το λογισμικό. Αυτό σημαίνει ότι από άποψη υλικού θα πρέπει να έχουμε τη κατάλληλη πλακέτα Arduino, τους αισθητήρες και τα καλώδια συνδεσμολογίας.

Στη συνέχεια θα πρέπει να χρησιμοποιήσουμε το λογισμικό Arduino IDE ώστε να το προγραμματίσουμε για το σκοπό που επιθυμούμε, δηλαδή να δημιουργήσουμε το πηγαίο κώδικα.



29

[29] https://en.wikipedia.org/wiki/Arduino_Uno#/media/File:Arduino_Uno_-_R3.jpg
By SparkFun Electronics from Boulder, USA - Arduino Uno - R3, CC BY 2.0,
https://commons.wikimedia.org/w/index.php?curid=26785892

3.2 Ιστορία

Ιστορία

Όλα ξεκίνησαν το 2005 από μία ομάδα μαθητών, η οποία ξεκίνησε ένα project με στόχο τη δημιουργία μιας πλατφόρμας με χαμηλό κόστος και προσιτή σε όλους. Έτσι οι ιδρυτές Massimo Banzi και David Cueartielles ξεκίνησαν αυτό το project, δημιουργώντας διάφορες πλακέτες στη Ιβρέα, Τορίνο της Ιταλίας όπου εκεί βρισκόταν και η εταιρία υπολογιστών Olivetti.

3.3 Πλακέτες και χρήσεις

Πλακέτες και χρήσεις του Arduino

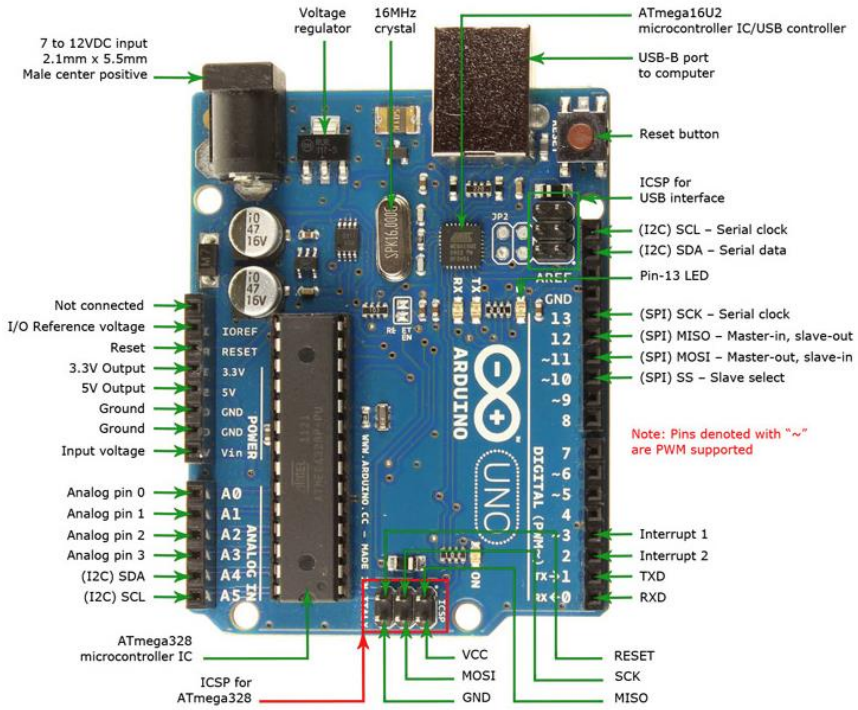
Η πρώτη πλακέτα Arduino που κατασκευάστηκε είχε την ονομασία UNO. Επειδή σήμερα έχουν αυξηθεί οι ανάγκες, έχουν δημιουργηθεί αρκετές παραλλαγές της αρχικής πλακέτας όπως οι: Mega, Nano, Esplora ή Arduino extreme κτλ. Η κάθε μία πλακέτα έχει τα δικά της χαρακτηριστικά και προορίζεται για συγκεκριμένη λειτουργία σε κάθε project. Επιπλέον το κάθε project περιέχει και αισθητήρες, όπως είναι για παράδειγμα οι αισθητήρες θερμοκρασίας, ήχου, φωτεινότητας, υγρασίας ή δακτυλικού αποτυπώματος. Επιπρόσθετα ως έξοδο μπορούμε να έχουμε άναμμα λυχνίας, εμφάνιση κειμένου σε οθόνη led κτλ.

3.4 Λειτουργία

Η λειτουργία του Arduino.[30][31]

Μια πλακέτα Arduino συνδέεται με διάφορα εξαρτήματα ή αισθητήρες και μπορούμε να πούμε ότι η πλακέτα έχει το ρόλο της επεξεργασίας των δεδομένων που δέχεται.

Το Arduino διαθέτει σειριακό interface και ο μικροελεγκτής υποστηρίζει αυτή την επικοινωνία, την οποία το Arduino προωθεί μέσα από έναν ελεγκτή που μας επιτρέπει την σύνδεση με υπολογιστή με τη χρήση θύρας USB. Η σύνδεση αυτή μας επιτρέπει την αμφίδρομη επικοινωνία μεταξύ υπολογιστή και Arduino για τη μεταφόρτωση του πηγαίου κώδικα στη πλακέτα, αλλά και για την επικοινωνία την ώρα που εκτελείται.



32

[30] https://edurobotics.gr/what-is-arduino/

[31] https://4dimkal-robot.weebly.com/tiota-epsilon943nualphaiota-tauomicron-arduino.html

[32] https://i.pinimg.com/originals/63/3c/45/633c45308db32e636c11bb4b852c4f00.png

Στη παραπάνω εικόνα βλέπουμε 14 θηλυκά pin με αρίθμηση 0 έως 13 που λειτουργούν ως ψηφιακές είσοδοι/έξοδοι.

Τα pin 0 και 1 λειτουργούν ως RX και TX δηλαδή για τη μεταφορά δεδομένων από την σειριακή στη θύρα USB.

Τα pin 2 και 3 λειτουργούν ως εξωτερικά interrupt, δηλαδή μπορούν να λειτουργήσουνε ως ψηφιακές είσοδοι για εφαρμογές που απαιτούν συγχρονισμό μεγάλης ακρίβειας.

Τα pin 3,5,6,9,10,11 μπορούν να λειτουργήσουν ως ψευδοαναλογικές είσοδοι.

Στη κάτω πλευρά της πλακέτας υπάρχει η ένδειξη ANALOG IN με έξι pin από το 0 έως το 5. Το κάθε ένα λειτουργεί ως αναλογική είσοδος χρησιμοποιώντας το ADC (Analog to Digital Converter).

Η τροφοδοσία πραγματοποιείται είτε μέσω θύρας USB είτε από εξωτερική τροφοδοσία. Στη πλακέτα θα βρούμε έξι pin με τη σήμανση POWER.

Το πρώτο pin με την ένδειξη RESET λειτουργεί για την επανεκκίνηση του Arduino.

Το δεύτερο pin με την ένδειξη 3.3V τροφοδοτεί τους αισθητήρες με 3.3V.

Το τρίτο pin με την ένδειξη 5V τροφοδοτεί τους αισθητήρες με 5V.

Το τέταρτο και πέμπτο pin με την ένδειξη GND είναι η γείωση.

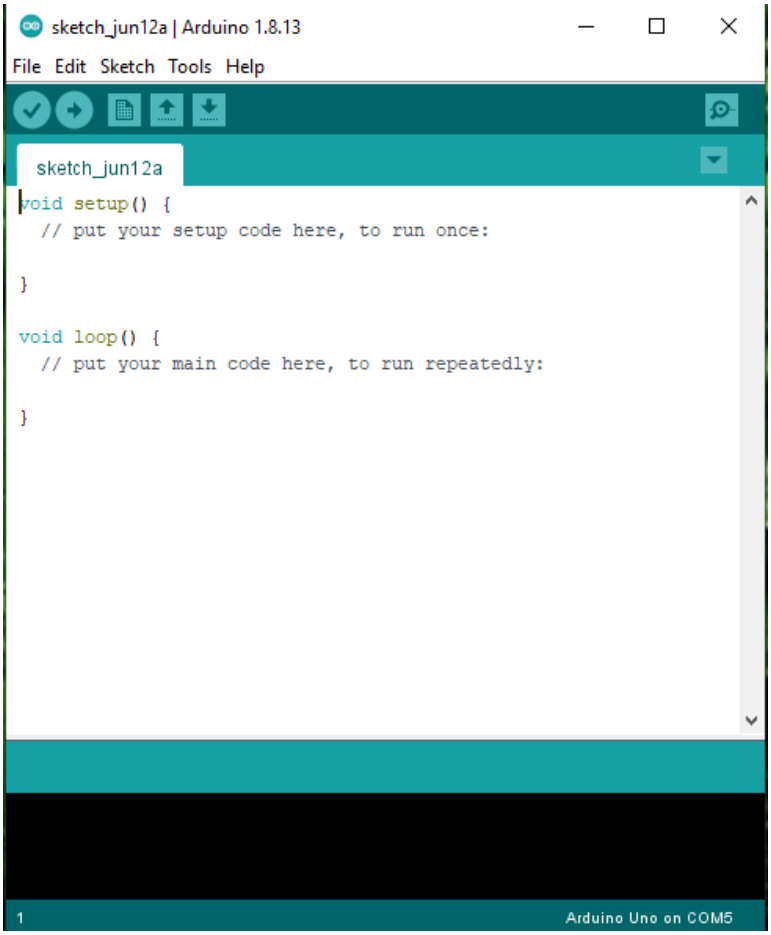
Το έκτο pin με την ένδειξη Vin μπορεί να τροφοδοτήσει τους αισθητήρες με πλήρη τάση εξωτερικής τροφοδοσίας.

3.5 Προγραμματισμός της πλακέτας

Προγραμματισμός Arduino.[33][34]

Το Arduino IDE είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης κώδικα γραμμένο σε JAVA, που έχει σχεδιαστεί να εισαγάγει νέους και άτομα που δεν είναι εξοικειωμένοι με την ανάπτυξη λογισμικού.

Συγκεκριμένα προσφέρει ένα εύχρηστο περιβάλλον για τη δημιουργία προγραμμάτων με έτοιμες βιβλιοθήκες, compiler για τη μεταγλώττιση, serial monitor για τη παρακολούθηση των επικοινωνιών και την επιλογή για μεταφόρτωσης του κώδικα (sketch) στο Arduino.



ARDUINO IDE

[33] https://el.wikipedia.org/wiki/Arduino

[34] https://4dimkal-robot.weebly.com/tiota-epsilon943nualphaiota-tauomicron-arduino.html

Κεφάλαιο 4ο : Προσωπικά Δεδομένα & GDPR

4.1 Η έννοια των Προσωπικών Δεδομένων

Προσωπικά δεδομένα ή αλλιώς δεδομένα προσωπικού χαρακτήρα

Προσωπικά δεδομένα είναι κάθε πληροφορία που αναφέρεται και περιγράφει ένα άτομο όπως:

1. Στοιχεία αναγνώρισης (ονοματεπώνυμο, ηλικία, κατοικία, επάγγελμα)
2. Οικογενειακή κατάσταση κλπ.)
3. Φυσικά χαρακτηριστικά
4. Εκπαίδευση
5. Εργασία (προϋπηρεσία, εργασιακή συμπεριφορά κλπ)
6. Οικονομική κατάσταση (έσοδα, περιουσιακά στοιχεία, οικονομική συμπεριφορά)
7. Ενδιαφέροντα
8. Δραστηριότητες
9. Συνήθειες

Το άτομο (φυσικό πρόσωπο) στο οποίο αναφέρονται τα δεδομένα ονομάζεται υποκείμενο των δεδομένων. Τα προσωπικά δεδομένα χωρίζονται σε δυο κατηγορίες.

Τα δεδομένα προσωπικού χαρακτήρα και τα ευαίσθητα προσωπικά δεδομένα.

Ευαίσθητα χαρακτηρίζονται τα προσωπικά δεδομένα ενός ατόμου που αναφέρονται στη φυλετική ή εθνική του προέλευση, στα πολιτικά του φρονήματα, στις θρησκευτικές ή φιλοσοφικές του πεποιθήσεις, στη συμμετοχή του σε συνδικαλιστική οργάνωση, στην υγεία του, στην κοινωνική του πρόνοια, στην ερωτική του ζωή, τις ποινικές διώξεις και καταδίκες του, καθώς και στη συμμετοχή του σε συναφείς με τα ανωτέρω ενώσεις προσώπων.

Τα ευαίσθητα δεδομένα προστατεύονται από τον Νόμο με αυστηρότερες ρυθμίσεις από ότι τα απλά προσωπικά δεδομένα. (AFFAIRS, January 6, 2012)

Στην παρούσα πτυχιακή εργασία θα ασχοληθούμε μόνο με τα προσωπικά δεδομένα και συγκεκριμένα με τα βιομετρικά δεδομένα (δακτυλικό αποτύπωμα) επεξεργασία προσωπικών δεδομένων και τις μεθόδους αυτών.

Με βιομετρικές μέθοδοι να νοούνται οι τεχνικές πιστοποίησης της ταυτότητας των ατόμων μέσω ανάλυσης σταθερών χαρακτηριστικών τους.

4.2 Τύποι βιομετρικών μεθόδων

Είδη βιομετρικών μεθόδων

Οι βιομετρικές μέθοδοι ταξινομούνται σε δύο κατηγορίες:

Α. Στις τεχνικές που στηρίζονται στην ανάλυση φυσικών ή γενετικών χαρακτηριστικών (όπως δακτυλικών αποτυπωμάτων, γεωμετρίας της παλάμης, ανάλυσης της κόρης του ματιού, των χαρακτηριστικών του προσώπου, του DNA).

Β. Στις τεχνικές που στηρίζονται στην ανάλυση συμπεριφοράς (όπως υπογραφής, φωνής, τρόπου πληκτρολόγησης).

4.3 Επεξεργασία

Επεξεργασία προσωπικών δεδομένων. (Stevens, April 10, 2012)

Επεξεργασία προσωπικών δεδομένων νοούνται οι εργασίες που πραγματοποιούνται σε δεδομένα προσωπικού χαρακτήρα, όπως: συλλογή, καταχώριση, οργάνωση, διατήρηση ή αποθήκευση, τροποποίηση, εξαγωγή, χρήση, διαβίβαση, διάδοση, συσχέτιση ή συνδυασμός, διασύνδεση, δέσμευση, διαγραφή, καταστροφή.

Κάθε φυσικό ή νομικό πρόσωπο του δημόσιου ή ιδιωτικού τομέα που τηρεί και επεξεργάζεται προσωπικά δεδομένα ονομάζεται υπεύθυνος επεξεργασίας.

Κάθε φυσικό ή νομικό πρόσωπο του δημόσιου ή ιδιωτικού τομέα που επεξεργάζεται δεδομένα για λογαριασμό κάποιου υπεύθυνου επεξεργασίας ονομάζεται εκτελών την επεξεργασία.

4.4 Νομοθεσία

Νομοθεσία προσωπικών δεδομένων.[35]

Σχετικά με τη νομοθεσία για τα προσωπικά δεδομένα στην Ελλάδα ισχύει ο Νόμος 2472/1997 για την προστασία των προσωπικών δεδομένων, ο οποίος εποπτεύεται από την Αρχή Προστασίας Δεδομένων Προσωπικού Χαρακτήρα. Επίσης, ισχύει ο Νόμος 3471/2006 για την προστασία των προσωπικών δεδομένων στις ηλεκτρονικές επικοινωνίας Ν. 2472/1997 αφορά κάθε φυσικό πρόσωπο που βρίσκεται εν ζωή.

Τα νομικά πρόσωπα δεν έχουν προσωπικά δεδομένα.

Η επεξεργασία προσωπικών δεδομένων επιτρέπεται μόνο όταν το άτομο έχει δώσει τη συγκατάθεσή του. Κατ εξαίρεση επιτρέπεται η επεξεργασία και χωρίς συγκατάθεση όταν συντρέχουν οι προϋποθέσεις που ορίζει ο Νόμος 2472/1997 στο άρθρο 5.

Στο νόμο 2472/1997 (άρθρο 5) ισχύει ότι

1. Επεξεργασία δεδομένων προσωπικού χαρακτήρα επιτρέπεται μόνον όταν το υποκείμενο των δεδομένων έχει δώσει τη συγκατάθεσή του.

2. Κατ' εξαίρεση επιτρέπεται η επεξεργασία και χωρίς τη συγκατάθεση, όταν:

α) Η επεξεργασία είναι αναγκαία για την εκτέλεση σύμβασης, στην οποία συμβαλλόμενο μέρος είναι υποκείμενο δεδομένων ή για τη λήψη μέτρων κατόπιν αιτήσεως του υποκειμένου κατά το προφυματικό στάδιο.

β) Η επεξεργασία είναι αναγκαία για την εκπλήρωση υποχρεώσεως του υπεύθυνου επεξεργασίας, η οποία επιβάλλεται από το νόμο.

γ) Η επεξεργασία είναι αναγκαία για τη διαφύλαξη ζωτικού συμφέροντος του υποκειμένου, εάν αυτό τελεί σε φυσική ή νομική αδυναμία να δώσει τη συγκατάθεσή του.

δ) Η επεξεργασία είναι αναγκαία για την εκτέλεση έργου δημόσιου συμφέροντος ή έργου που εμπίπτει στην άσκηση δημόσιας εξουσίας και εκτελείται από δημόσια αρχή ή έχει ανατεθεί από αυτή είτε στον υπεύθυνο επεξεργασίας είτε σε τρίτο, στον οποίο γνωστοποιούνται τα δεδομένα.

ε) Η επεξεργασία είναι απολύτως αναγκαία για την ικανοποίηση του έννομου συμφέροντος που επιδιώκει ο υπεύθυνος επεξεργασίας ή ο τρίτος ή οι τρίτοι στους οποίους ανακοινώνονται τα δεδομένα και υπό τον όρο ότι τούτο υπερέχει προφανώς των δικαιωμάτων και συμφερόντων των προσώπων στα οποία αναφέρονται τα δεδομένα και δεν θίγονται οι θεμελιώδεις ελευθερίες αυτών.

3. Η Ελληνική Αρχή μπορεί να εκδίδει ειδικούς κανόνες επεξεργασίας για τις πλέον συνήθεις κατηγορίες επεξεργασιών και αρχείων, οι οποίες προφανώς δεν θίγουν τα δικαιώματα και τις ελευθερίες των προσώπων στα οποία αναφέρονται τα δεδομένα.

[35] https://www.dpa.gr/el/enimerwtiko/nomothesia/proswpikon_dedomenon as seen on 1/6/2022

Οι κατηγορίες αυτές προσδιορίζονται με κανονισμούς που καταρτίζει η Αρχή και κυρώνονται με προεδρικά διατάγματα, τα οποία εκδίδονται με πρόταση του Υπουργού Δικαιοσύνης.

Κάθε υπεύθυνος επεξεργασίας οφείλει να γνωστοποιεί στην Αρχή την επεξεργασία προσωπικών δεδομένων που πραγματοποιεί, εκτός αν εμπίπτει σε μία από τις περιπτώσεις του αρ. 7Α του Ν. 2472/1997. Η Αρχή καταχωρεί τη γνωστοποίηση σε ειδικό μητρώο. Δυστυχώς δεν μπορούμε να γνωρίζουμε αν κάποιοι έχουν τα προσωπικά μας δεδομένα και τα χρησιμοποιούν παράνομα . Η καλύτερη μέθοδος για αυτό το σκοπό είναι η πρόληψη. Ο καθένας μας πρέπει να είναι ευαισθητοποιημένος, να μη δίνει τα στοιχεία του ανεξέλεγκτα, να περιορίζεται στα απολύτως απαραίτητα και, αν αντιμετωπίσει πρόβλημα, να ακολουθήσει τη διαδικασία που προβλέπει ο Ν. 2472/97.

Επίσης δυστυχώς η Ελληνική Αρχή[36] έχει στην κατοχή της αρχεία με προσωπικά μας δεδομένα, στην Αρχή γνωστοποιείται μόνο η επεξεργασία αρχείων με προσωπικά δεδομένα. Η Αρχή δεν γνωρίζει το περιεχόμενο των αρχείων αλλά μόνο τα κύρια χαρακτηριστικά τους καθώς και τον υπεύθυνο επεξεργασίας των αρχείων.

[36] https://www.dpa.gr/el

4.5 Γενικός Κανονισμός για την Προστασία Δεδομένων (General Data Protection Regulation – GDPR)

Σε ποιους απευθύνεται.[37] (GRIVAS, 2018)

Αφορά όλες τις ιδιωτικές και δημόσιες επιχειρήσεις, καθώς και τις κρατικές αρχές που με οποιοδήποτε τρόπο διαχειρίζονται προσωπικά δεδομένα πελατών, εργαζομένων, συνεργατών ή άλλων φυσικών προσώπων. Ως εκ τούτου, ο GDPR αφορά πρακτικά όλες τις επιχειρήσεις, εντός και εκτός Ευρωπαϊκής Ένωσης, εφόσον τα δεδομένα αφορούν Ευρωπαίους πολίτες.

Σε περίπτωση μη τήρησης του κανονισμού GDPR επιβάλλονται πρόστιμα μέχρι 20,000,000€ και όσον αφορά τις επιχειρήσεις μπορεί να φτάσει έως και το 4% του συνολικού παγκόσμιου ετήσιου κύκλου εργασιών του προηγούμενου οικονομικού έτους, ανάλογα με το ποιο είναι υψηλότερο.

Περιορισμένη πρόσβαση εφαρμογών και ηλικιακός περιορισμός

Πολλές εφαρμογές αναγκάζονται να αλλάξουν τους όρους επειδή δεν πληρούν τις προδιαγραφές του GDPR και από την στιγμή που απευθύνονται σε Ευρωπαίους πολίτες δεν θα τους επιτρέπεται η πρόσβαση σε αχρείαστα δεδομένα όπως επαφές ή αρχείο τηλεφωνικών κλήσεων. Επιπλέον, από την 25η Μαΐου απαγορεύεται η συγκέντρωση προσωπικών δεδομένων σε άτομα κάτω των 16 ετών.

Επεξεργασία δεδομένων

Είναι κάθε εργασία που αφορά την επεξεργασία δεδομένων όπως συλλογή, καταχώρηση, οργάνωση, διαγραφή, καταστροφή κτλπ.

Υπεύθυνος Επεξεργασίας (Data Controller)

Ο Υπεύθυνος Επεξεργασίας (Data Controller) είναι φυσικό πρόσωπο που καθορίζει το σκοπό, τις προϋποθέσεις και τον τρόπο επεξεργασίας προσωπικών δεδομένων.

Εκτελών την Επεξεργασία (Data Processor)

[37] https://grillmagazine.gr/2018/08/11/gdpr-%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CE%BA%CE%B1%CE%B9-%CF%83%CE%B5-%CF%80%CE%BF%CE%B9%CE%BF%CF%85%CF%82-%CE%B1%CF%80%CE%B5%CF%85%CE%B8%CF%8D%CE%BD%CE%B5%CF%84%CE%B1%CE%B9/ as seen om 1/6/2022

Είναι το φυσικό πρόσωπο που εκτελεί την επεξεργασία των προσωπικών δεδομένων για λογαριασμό του Υπευθύνου Επεξεργασίας.

Παραβίαση δεδομένου προσωπικού χαρακτήρα

Η παραβίαση της ασφάλειας είναι οι ενέργειες που οδήγησαν τα προσωπικά δεδομένα σε απώλεια, καταστροφή, άνευ άδειας μεταφοράς τους ή υποβλήθηκαν με επεξεργασία με οποιοδήποτε τρόπο. Εάν προκύψει κάποια παραβίαση ασφαλείας και έχουν διαρρεύσει προσωπικά δεδομένα τότε η εταιρεία ή ο οργανισμός πρέπει να ειδοποιήσει την εποπτική αρχή χωρίς αδικαιολόγητη καθυστέρηση και το αργότερο εντός 72 ωρών αφού αντιληφθεί την παραβίαση. Εάν η εταιρεία ή ο οργανισμός είναι ο εκτελών την επεξεργασία, πρέπει να ενημερώνει τον υπεύθυνο επεξεργασίας δεδομένων για κάθε παραβίαση δεδομένων.

Παράδειγμα διαρροής προσωπικών δεδομένων από κυβερνοεπίθεση

Χαρακτηριστικό παράδειγμα η κυβερνοεπίθεση στην εταιρία Uber που έγινε το 2016 και μετά από 1 χρόνο παραδέχθηκε ότι είχαν διαρρεύσει προσωπικά δεδομένα 57 εκατομμύριων πολιτών. Μέχρι να γίνει γνωστή η υπόθεση η Uber είχε πληρώσει τους χάκερς 100,000$ ώστε να «διαγράψουν» την βάση δεδομένων που απέκτησαν πρόσβαση και να μην κάνουν γνωστή την εν λόγο επίθεση. Η εταιρία υποστηρίζει ότι αριθμοί πιστωτικών καρτών και αριθμοί κοινωνικής ασφάλισης δεν έπεσαν ποτέ στα χέρια των χάκερ. Η τιμωρία της εταιρίας ήταν να πληρώσει υπέρογκα ποσά σε κάθε χώρα για τους πολίτες των οποίων τα στοιχεία είχαν εκτεθεί.[38]

[38] https://grillmagazine.gr/2018/08/11/gdpr-%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-%CE%BA%CE%B1%CE%B9-%CF%83%CE%B5-%CF%80%CE%BF%CE%B9%CE%BF%CF%85%CF%82-%CE%B1%CF%80%CE%B5%CF%85%CE%B8%CF%8D%CE%BD%CE%B5%CF%84%CE%B1%CE%B9/

4.6 Λογισμικό προστασίας – Antivirus

Λογισμικό προστασίας από ιούς – Antivirus[39][40]

Το antivirus (IEEE, 2005)είναι ένα πρόγραμμα προστασίας του υπολογιστή που χρησιμοποιείται για την πρόληψη, τον εντοπισμό και την αφαίρεση κακόβουλου λογισμικού. Είναι μία από τις ελάχιστες εφαρμογές –αν όχι η μόνη– που είναι απαραίτητο να ξεκινάει ταυτόχρονα με το λειτουργικό σύστημα του υπολογιστή και να είναι ενεργό για όλη την διάρκεια που είναι ενεργός ο υπολογιστής. Κάθε λογισμικό προστασίας έχει μία βάση δεδομένων γεμάτη με χαρακτηριστικά του κώδικα εκατομμυρίων ιών η οποία ονομάζονται «υπογραφές ιών».

Έτσι κάθε αρχείο που σαρώνει το πρόγραμμα διαβάζει τον κώδικά του και ψάχνει στην βάση δεδομένων αν περιέχει ίχνη από τις υπογραφές ιών που είναι καταχωρημένες.

Ένα σύγχρονο antivirus ανανεώνει συνεχώς την Βάση Δεδομένων του με νέες υπογραφές ιών που κυκλοφορούν καθημερινά καθώς και έχει μηχανισμούς άμυνας που λειτουργούν σε πραγματικό χρόνο, ώστε να αντιμετωπίσουν ιούς που ακόμη δεν έχουν μολύνει υπολογιστές και δεν είναι γνωστοί στον κόσμο (0-day απειλές), συνεπώς μπορεί να είμαστε από τα πρώτα θύματα που θα επιτεθεί αυτός ο νέος ιός. Η σύνδεση με το διαδίκτυο είναι απαραίτητη ώστε να παίρνουμε όλες τις ενημερώσεις για να έχουμε την καλύτερη δυνατή προστασία με τις τελευταίες πληροφορίες που είναι διαθέσιμες από το λογισμικό προστασίας που έχουμε εγκαταστημένο καθώς και όλες τις απαραίτητες ενημερώσεις ασφαλείας που σχετίζονται με λειτουργικό σύστημα και σε οποιοδήποτε άλλο πρόγραμμα υπάρχει στον υπολογιστή.

39

https://el.wikipedia.org/wiki/%CE%91%CE%BD%CF%84%CE%B9%CE%B9%CE%B9%CE%BA%CF%8C_%CF%80%CF%81%CF%8C%CE%B3%CF%81%CE%B1%CE%BC%CE%BC%CE%B1 as seen on 1/6/2022

40 https://www.pcsteps.gr/1391-%CF%80%CF%8E%CF%82-%CE%BB%CE%B5%CE%B9%CF%84%CE%BF%CF%85%CF%81%CE%B3%CE%B5%CE%AF-%CF%84%CE%BF-antivirus/

as seen on 1/6/2022

4.7 Κατηγορίες ιών

Κακόβουλο Λογισμικό και Τύποι Ιών[41][42]

Ένας ιός υπολογιστών (virus) είναι ένα κακόβουλο λογισμικό πρόγραμμα υπολογιστή το οποίο μπορεί να επέμβει και να τροποποιήσει διάφορα κομμάτια του λειτουργικού συστήματος χωρίς την άδεια ή να γνωρίζει ο χρήστης. Ο σκοπός αυτών των επιθέσεων εξαρτάται, για παράδειγμα μπορεί να έχει ως σκοπό αφού μολύνει τον υπολογιστή να καταστρέψει διάφορα προγράμματα ή να διαγράψει αρχεία που είναι απαραίτητα στον χρήστη. Μπορεί να θέλει να παρακολουθεί τις κινήσεις του χρήστη και να εξαπλωθεί σε όσο το δυνατόν περισσότερους υπολογιστές, να συλλέξει πληροφορίες, προσωπικά δεδομένα και να τα χρησιμοποιήσει ενάντια στον δημιουργό με δόλο (οικονομικό όφελος) ή για άλλες παράνομες δραστηριότητες.

- Ιοί περιοχής εκκίνησης – Αυτοί οι ιοί μολύνουν δισκέτες και σκληρούς δίσκους. Ο ιός φορτώνεται πριν από το λειτουργικό σύστημα. Ήταν οι πρώτοι ιοί που εμφανίστηκαν.

- Ιοί αρχείων – είναι εκτελέσιμα αρχεία τα οποία έχουν την δυνατότητα να εξαπλώνονται εύκολα σε χρήσιμα προγράμματα ενός ξένου υπολογιστή με αποτέλεσμα να βλάψει τα αρχεία του χρήστη. Η μετάδοσή τους είναι εύκολη και μπορεί να γίνει από μία φορητή μνήμη USB ή από έναν εξωτερικό σκληρό δίσκο. Ένα στοιχείο διαφοράς του με τους άλλους τύπους ιών είναι ότι μπορεί να μεταδοθεί οπουδήποτε έχει τη δυνατότητα. Οι επιπτώσεις που μπορεί να έχει ένας χρήστης από αυτό τον ιό, είναι η διαγραφή δεδομένων του ή ακόμη και η κατάρρευση ολόκληρου του συστήματος του.

- Ιοί σκουλίκια (Worms) – Είναι κακόβουλο λογισμικό το οποίο μπορεί να μεταδοθεί είτε με την χρήση κάποιας δικτυακής υποδομής όπως είναι τα τοπικά δίκτυα ή μέσω μηνύματος e-mail. Μόλις μολύνει το σύστημα στέλνει προσωπικά δεδομένα και κωδικούς στον επιτιθέμενο ώστε να έχει πλήρη πρόσβαση στο δίκτυο. Επιπλέον επιβαρύνουν το δίκτυο φορτώνοντάς το με άχρηστη δραστηριότητα.

- Δούρειος Ίππος (Trojan horse) – Είναι κακόβουλο λογισμικό που χρησιμοποιεί το στοιχείο της παραπλάνησης. Συγκεκριμένα παριστάνει ότι είναι χρήσιμο για το υπολογιστή αλλά στην πραγματικότητα μέσα από αυτό οι επιτιθέμενοι καταφέρνουν να κλέψουν σημαντικά αρχεία ή να αποκτήσουν τον πλήρη έλεγχο του συστήματος.

[41] https://ioi4.webnode.gr/%CF%83%CF%87%CE%B5%CF%84%CE%B9%CE%BA%CE%AC-%CE%BC%CE%B5-%CF%84%CE%BF%CF%85%CF%82-%CE%B9%CE%BF%CF%8D%CF%82-(about-viruses)/

[42] https://techguru.gr/blog/h-mikri-egkyklopedia-ton-ion-ypologiston/

- Rootkit – Είναι λογισμικό το οποίο μπορεί να ανήκει σε οποιαδήποτε κατηγορία και έχει την ιδιαιτερότητα να αποκρύπτει κάποια κακόβουλα λογισμικά ώστε να μην γίνονται ορατά από το πρόγραμμα προστασίας. Επίσης κάποιες φορές προστατεύουν τους χάκερ διότι διαγράφει πληροφορίες του εισβολέα.

- Ransomware – Είναι ιδιαίτερα γνωστοί το τελευταίο καιρό και εξαπλώνονται όλο και περισσότερο. Μόλις μολυνθεί ο υπολογιστής αυτόματα κρυπτογραφούνται όλα τα αρχεία του συστήματος, συνεπώς και οποιοδήποτε αρχείο του χρήστη ζητώντας από το θύμα να πληρώσει λύτρα ώστε να γίνει η αποκρυπτογράφηση. Η πληρωμή συνήθως ζητείται να γίνει σε bitcoin δηλαδή ένα ψηφιακό - εικονικό νόμισμα που βοηθάει ώστε να μην εντοπιστεί ο παραλήπτης των λύτρων.

Κεφάλαιο 5ο: Βιομετρία και περιπτώσεις χρήσεις

5.1 Η έννοια των βιομετρικών χαρακτηριστικών

Βιομετρικά χαρακτηριστικά.[43]

Τα βιομετρικά χαρακτηριστικά είναι φυσικά, μοναδικά και μετρήσιμα σε κάθε άνθρωπο που χρησιμοποιούνται για την αναγνώριση ταυτότητας ενός ατόμου. Μερικά χαρακτηριστικά μπορεί να είναι εκτός από το δακτυλικό αποτύπωμα, η απόσταση μεταξύ ματιών, μύτης, στόματος και αυτιών.

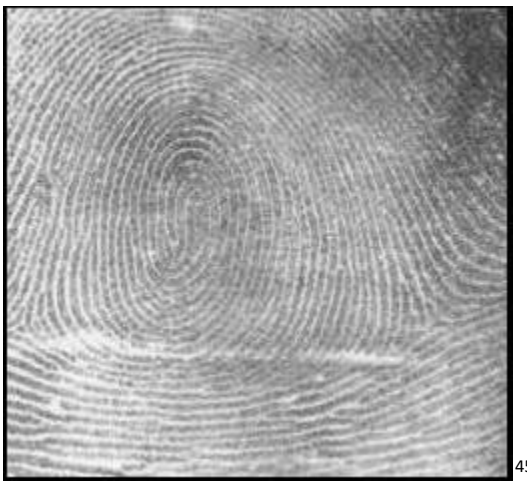
Τα πλέον φημισμένα παραδείγματα βιομετρίας είναι η αναγνώριση δακτυλικών αποτυπωμάτων, η αναγνώριση προσώπου και η σάρωση της ίριδας.

5.2 Δακτυλικό αποτύπωμα

[43] http://www.passport.gov.gr/npc-periexomeno/npc-periexomeno/viometrika-xaraktiristika-diavatiriou.html

Τι είναι δακτυλικό αποτύπωμα.[44]

Αποτύπωμα είναι η εικόνα που σχηματίζεται από την αναπαράσταση των θηλοειδών γραμμών των δακτύλων όταν έρθουν σε επαφή σε μία λεία επιφάνεια. Αποτυπώματα μπορούν να αφήσουν επίσης οι παλάμες και τα πέλματα του ανθρώπου. Οι θηλοειδής αυτές γραμμές πρωτοεμφανίζονται στον άνθρωπο όταν από τον τέταρτο μήνα της κύησης και δεν αλλάζουν καθόλη την διάρκεια της ζωής του ακόμη και μετά τον θάνατο μέχρι την αποσύνθεση.


4546

5.3 Ιδιότητες δακτυλικού αποτυπώματος

[44] http://www.forensiclabs.gr/index.php?option=com_content&task=view&id=37&Itemid=55
[45] https://en.wikipedia.org/wiki/Fingerprint#/media/File:Fingerprint_Whorl.jpg
[46] By NIST database - https://web.archive.org/web/20050403155444/http://www.nist.gov/srd/fing_img.htm, Public Domain, https://commons.wikimedia.org/w/index.php?curid=11296016

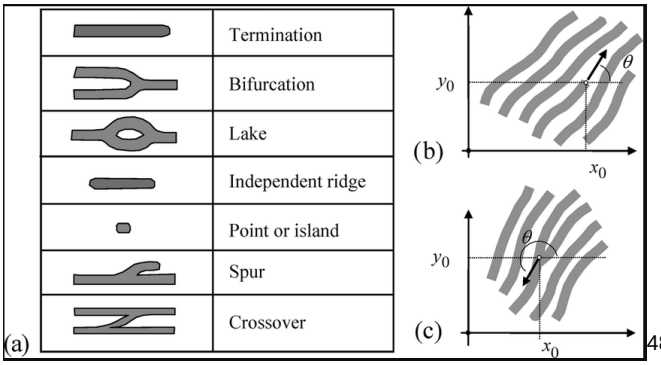
Ιδιότητες[47]

Οι ιδιότητες την δακτυλικών αποτυπωμάτων είναι ότι παραμένουν αμετάβλητα και αναλλοίωτα υπό φυσιολογικές συνθήκες και δεν υπάρχει περίπτωση ταύτισης ενός ατόμου με ένα άλλο, αλλά και ούτε ταύτιση ενός δακτύλου με ένα άλλο δάκτυλο από το ίδιο άτομο.

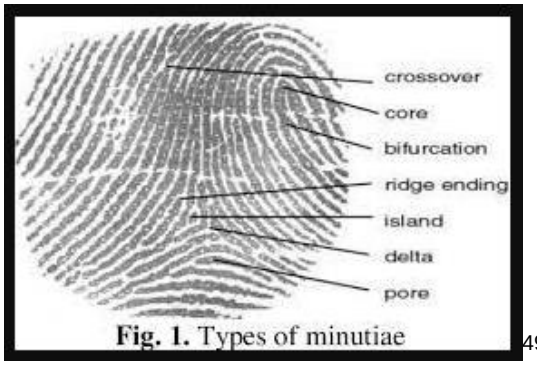
5.4 Μικρολεπτομέρειες

Μικρολεπτομέρειες (minutiae)

Κατά την διαδικασία ανίχνευσης δακτυλικού αποτυπώματος, οι μικρολεπτομέρειες (minutiae) είναι κάποιο συγκεκριμένα σημεία στην εικόνα του δακτύλου. Υπάρχουν δύο βασικοί τύποι που είναι γνωστοί ως «Τερματισμοί Κορυφής» και «Διακλαδώσεις». Ο αριθμός αυτών των σημείων διαφέρει σε κάθε δάκτυλο αλλά και από άτομο σε άτομο.



[48]



[49]

5.5 Περιπτώσεις χρήσεις

[47] http://www.forensiclabs.gr/index.php?option=com_content&task=view&id=37&Itemid=55
[48] https://www.researchgate.net/profile/S-Pankanti/publication/3455239/figure/fig4/AS:669009137651715@1536515616365/Fingerprint-minutiae-a-The-common-fingerprint-minutiae-types-b-ridge-ending-x-y.ppm
[49] https://www.semanticscholar.org/paper/Implementation-of-Minutiae-Based-Fingerprint-System-Chaudhari-Patnaik/f83df46928d38b3f62af5c0d585e3e6dffec45e9

Με την επιτυχή είσοδο του φοιτητή στο Εκπαιδευτικό Ίδρυμα παράλληλα με την εγγραφή του στο τμήμα, θα μπορούσε να δώσει το δακτυλικό του αποτύπωμα για καταγραφή, ώστε ο σπουδαστής να μπορεί μελλοντικά να διευκολυνθεί σε διάφορες υπηρεσίες που θα του χρειαστούν. Η καταγραφή μπορεί να είναι αρχικά προαιρετική εξηγώντας τα πλεονεκτήματα που προσφέρει και στη συνέχεια να γίνει υποχρεωτική. Με το συγκεκριμένο τρόπο όλα θα λειτουργούν ευκολότερα, ταχύτερα και με ακρίβεια.

Περίπτωση χρήσης 1: ΣΙΤΙΣΗ ΣΤΟ ΕΣΤΙΑΤΟΡΙΟ ΤΗΣ ΣΧΟΛΗΣ

Κάθε φοιτητής που έχει δικαίωμα γεύματος στο εστιατόριο της σχολής θα μπορεί να λαμβάνει το γεύμα του, τοποθετώντας μόνο το δακτυλικό του αποτύπωμα στον αισθητήρα ταυτοποίησης που μπορεί να είναι τοποθετημένος σε σημείο που να διευκολύνει τον φοιτητή (πχ. Πριν πάρει τον δίσκο).

Το σύστημα που ισχύει έως και σήμερα για να φάει ένας φοιτητής είναι το εξής:
Ο φοιτητής αφού πάρει το φαγητό του κατευθύνεται στον υπάλληλο του εστιατορίου αναφέρει τον αριθμό μητρώου του, δείχνει το πάσο και εν συνεχεία ο υπάλληλος τον πληκτρολογεί στον Η/Υ ώστε να εμφανίσει τα στοιχεία του και να δει αν έχει δικαιούται να φάει ή αν έχει ξαναπεράσει.

Με την εφαρμογή του συστήματος που υλοποιήσαμε και παρουσιάζουμε δεν θα υπάρχει όλη αυτή η χρονοβόρα διαδικασία. Ο φοιτητής θα μπορεί να κάνει την ταυτοποίηση δίχως να λέει τον αριθμό μητρώου του, δίχως να απαιτείται η επίδειξη του πάσο του, παρά μόνο τοποθετώντας το δακτυλικό του αποτύπωμα στον αισθητήρα ταυτοποίησης. Κατά την διάρκεια της ταυτοποίησης η αλληλεπίδραση με τον χρήστη θα μπορούσε να είναι γίνεται με 2 λαμπάκια τύπου LED (πράσινο, κόκκινο) ώστε να μπορεί να καταλάβει ο φοιτητής αν έγινε με επιτυχία η ταυτοποίηση ή όχι. Θα μπορούσε σε μελλοντική χρήση εάν υπάρξουν 3 συνεχόμενες αποτυχίες ταυτοποίησης ένα ηχείο θα βγάζει έναν χαρακτηριστικό ήχο που θα σημαίνει ότι ο προϊστάμενος θα χρειαστεί να παρέμβει ώστε να αποφασίσει για την συνέχεια.

Περίπτωση χρήσης 2: ΠΑΡΟΥΣΙΕΣ ΣΕ ΕΡΓΑΣΤΗΡΙΑΚΑ ΜΑΘΗΜΑΤΑ

Ο κάθε φοιτητής που καλείται να παρακολουθήσει τα εργαστηριακά του μαθήματα με την έναρξη του μαθήματος θα πρέπει να πάρει και την παρουσία ότι παρακολούθησε το μάθημα. Το σύστημα που ισχύει έως και σήμερα είναι είτε ο διδάσκων (στην αρχή ή στο τέλος του μαθήματος) να διαβάζει ένα-ένα τα ονόματα που έχει σε λίστα στον Η/Υ του και να παίρνει παρουσίες, είτε να υπάρχει μία απλή κόλλα Α4 και να γυρίζει απ' άκρη σε άκρη, συμπληρώνοντας ο καθένας τα στοιχεία του (ονοματεπώνυμο και υπογραφή). Εδώ θα μπορούσε ο οποιοσδήποτε να βάλει ότι βρίσκεται στη θέση άλλου συμφοιτητή του παρόλο που αυτός μπορεί να λείπει.

Σε αντίθεση με την χρήση δακτυλικού αποτυπώματος που μπορεί η συσκευή να βρίσκεται στην είσοδο της αίθουσας κάθε φοιτητής που θα έρχεται θα τοποθετεί το δάκτυλό του στον αισθητήρα ταυτοποίησης και θα δηλώνεται αυτόματα στο σύστημα η παρουσία του. Με αυτόν τον τρόπο εκτός ότι είναι γρήγορος και αποτελεσματικός δεν θα μπορεί να δηλωθεί παρουσία για συνάδελφο που δεν βρίσκεται εκεί.

Περίπτωση χρήσης 3: ΕΞΥΠΗΡΕΤΗΣΗ ΓΡΑΜΜΑΤΕΙΑΣ

Ένα από τα πιο σύνηθες έγγραφα που ζητάει ένας φοιτητής είναι μία απλή βεβαίωση σπουδών ή μία αναλυτική βαθμολογία. Για την έκδοση της βεβαίωσης σπουδών απαιτείται η επίδειξη της φοιτητικής ταυτότητας και του αριθμού μητρώου στη γραμματεία. Στις ώρες αιχμής της λειτουργίας της γραμματείας οι φοιτητές που περιμένουν στην ουρά για να εξυπηρετηθούν, η τρέχουσα κατάσταση ταυτοποίησης μπορεί να αποτελεί μια χρονοβόρα διαδικασία τόσο για τον σπουδαστή όσο και για τη γραμματεία.

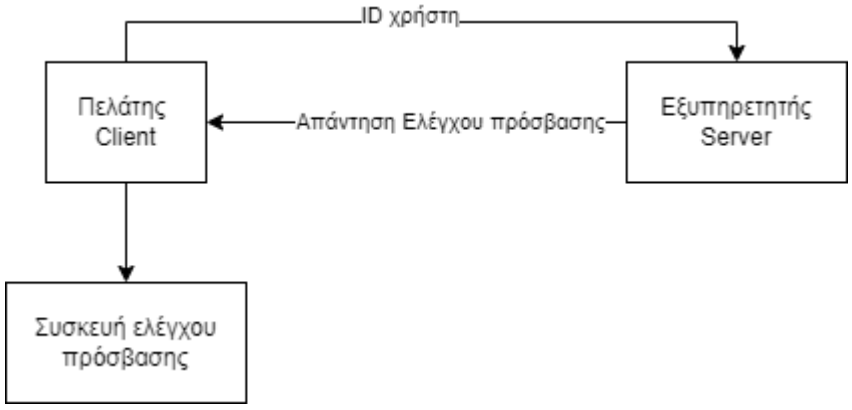
Με την χρήση δακτυλικού αποτυπώματος δεν θα χρειαστεί η επίδειξη της φοιτητικής ταυτότητας αλλά ούτε και ο αριθμός μητρώου για την έκδοση εγγράφου. Αυτό θα έχει ως αποτέλεσμα την καλύτερη, ταχύτερη και με ακρίβεια εξυπηρέτηση.

Επιπλέον θα μπορούσε να δημιουργηθεί ειδικός χώρος με Η/Υ, εκτυπωτή και αισθητήρα δακτυλικού αποτυπώματος έτσι ώστε ο σπουδαστής να μπορεί εξυπηρετηθεί από μόνος του στις περιπτώσεις έκδοσης κάποιου εγγράφου, ανεξάρτητα από τις ώρες λειτουργίας της γραμματείας.

Περίπτωση χρήσης 4: ΕΞΕΤΑΣΤΙΚΗ

Σε κάθε εξεταστική υπάρχουν πολλά παρουσιολόγια για το κάθε τμήμα ώστε να υπογράφει ο εξεταζόμενος στην αντίστοιχη λίστα του τμήματός του ότι πήγε να δώσει το εξεταζόμενο μάθημα. Επειδή πολλές φορές υπάρχουν στην ίδια εξεταστική αίθουσα φοιτητές από διαφορετικά τμήματα που δίνουν το συγκεκριμένο μάθημα, με την υπάρχουσα διαδικασία ταυτοποίησης, υπάρχει μεγάλη χρονοτριβή ως προς την ολοκλήρωση της διαδικασίας ταυτοποίησης από όλους τους σπουδαστές.

Με την χρήση όμως του δακτυλικού αποτυπώματος ο κάθε σπουδαστής που θα εισέρχεται στην αίθουσα εξετάσεων, ταυτόχρονα με την είσοδο του στον χώρο θα πραγματοποιείται και η ταυτοποίηση. Η όλη διαδικασία ταυτοποίησης του κάθε ατόμου θα ολοκληρώνεται σε ελάχιστο χρόνο και με ακρίβεια.



Κεφάλαιο 6ο: Ανάπτυξη υλικολογισμικού ταυτοποίησης χρήστη και ελέγχου της πρόσβασης του φοιτητή σε υπηρεσίες του εκπαιδευτικού ιδρύματος.

6.1 Το υλικό που θα χρησιμοποιηθεί

Σε κάθε έργο που πρόκειται να εκπονηθεί πολλές μεταβλητές και σταθερές πρέπει να υπολογιστούν ώστε να επιλεχθεί ο εξοπλισμός και το υλικό που θα χρησιμοποιηθεί για την ανάπτυξη του έργου.
Τέτοιες μεταβλητές είναι το κόστος, η διαθεσιμότητα, η ευκολία της χρήσης και της αξιοποίησης του υλικού μέσω της βιβλιογραφίας, κώδικα και παραδειγμάτων που υπάρχει γι' αυτό και συνάμα ο χρόνος που θα χρειαστεί για να αναπτυχθεί ένα πρωτότυπο. Καθοριστικός παράγοντας πάντα μπορεί να είναι το υλικό που μπορεί να υπάρχει ήδη στην κατοχή του οργανισμού / ερευνητή για την άμεση αξιοποίηση του πράγμα που είναι απόρροια των προηγούμενων μεταβλητών.
Στα πλαίσια αυτής της πτυχιακής εργασίας αποφασίστηκε η ανάπτυξη υλικολογισμικού για την περίπτωση χρήσης 1 «ΣΙΤΙΣΗ ΣΤΟ ΕΣΤΙΑΤΟΡΙΟ ΤΗΣ ΣΧΟΛΗΣ».
Το προϊόν που θα αναπτυχθεί θα είναι μια συσκευή όπου θα είναι εγκατεστημένη στην σίτιση της σχολής. Το προϊόν θα έχει διασύνδεση με το διαδίκτυο και θα επικοινωνεί με απομακρυσμένο κεντρικό ηλεκτρονικό υπολογιστή (server) ώστε να παρέχει σε αυτόν τα στοιχεία αναγνώρισης του χρήστη / φοιτητή και έπειτα να λαμβάνει απάντηση με την ενέργεια που χρήζει πράξης. Εν συνεχεία θα

Τα βήματα για την κατασκευή του προϊόντος είναι:

1. Επιλογή μικροελεγκτή.
2. Επιλογή περιφερειακών συσκευών για την αλληλεπίδραση με τον χρήστη.
3. Επιλογή μεθόδου αναγνώρισης.
4. Διεπαφή των αισθητήρων.
5. Προγραμματισμός του ελεγκτή.
6. Κατασκευή προγράμματος στον εξυπηρετητή. (server)

6.2 Επιλογή επεξεργαστικής μονάδας

Όπως αναφέρθηκε και πιο πριν, η επιλογή ενός κατάλληλου ελεγκτή είναι πολύπλευρη.

Για την εφαρμογή μας χρειαζόμαστε έναν ελεγκτή που θα μπορεί να επικοινωνεί μέσω διαδικτυακών πρωτοκόλλων καθώς και να έχει τις διεπαφές για να ελέγχει τις περιφερειακές συσκευές που θέλουμε να χειριστούμε.

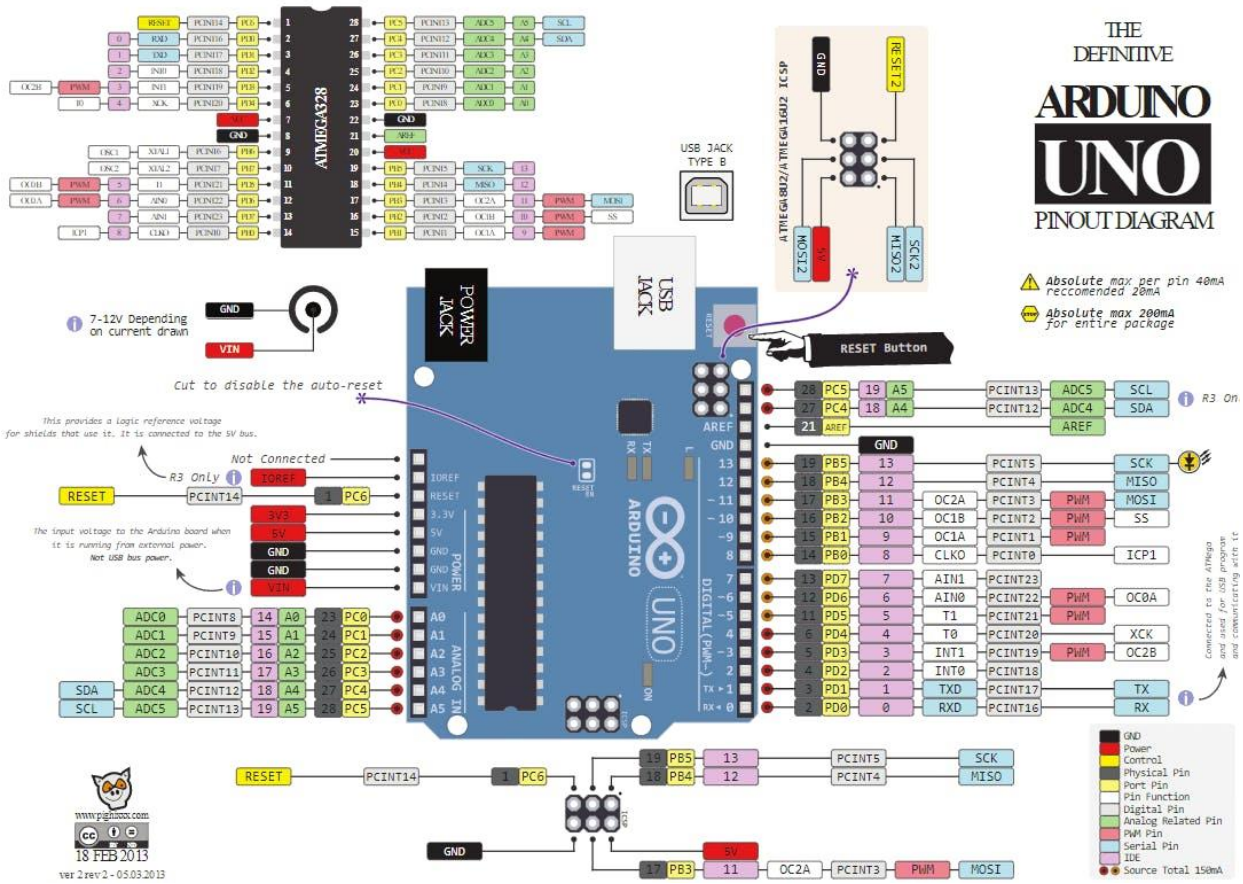
Πιθανοί ελεγκτές που θα μπορούσαν να χρησιμοποιηθούν είναι:

- Μικροελεγκτές (Microcontrollers) τύπου Arduino βασισμένοι σε AVR επεξεργαστές ή άλλους (ST32 ή ESP).
  Δεν διαθέτουν λειτουργικό σύστημα και δεν υποστηρίζουν παράλληλο προγραμματισμό. Παρέχουν χαμηλού επιπέδου προγραμματισμό μέσω της γλώσσας Arduino που είναι μια παραλλαγή της γλώσσας C++ καθώς έχουν αφαιρεθεί μερικές συναρτήσεις / μέθοδοι και έχουν προστεθεί άλλες.
  Η πλειοψηφία των μικροελεγκτών πλην κάποιων καινούργιων προσθηκών, χρειάζονται «κάρτα επέκτασης» ώστε να μπορέσει να επιτευχθεί η επικοινωνία με τα διαδικτυακά πρωτόκολλα.

- Υπολογιστές μονής πλακέτας (Single Board Computers) τύπου Raspberry Pi 3/4, Beaglebone Black ή NVIDIA Jetson.
  Διαθέτουν λειτουργικό σύστημα, συνήθως Linux και υποστηρίζουν παράλληλο προγραμματισμό πραγματικού χρόνου.
  Έχουν ενσωματωμένη διεπαφή για τα πρωτόκολλα Ethernet ή Wi-Fi.
  Είναι πιο κοστοβόρα από έναν απλό μικροελεγκτή αλλά παρέχουν πιο εύκολο προγραμματισμό και ανάπτυξη σε αυτά.

Ο ελεγκτής που τελικά επιλέχθηκε είναι ο μικροελεγκτής Arduino Uno.
Μπορεί να μην είναι ο πιο δυνατός επεξεργαστικά ή εύκολος στον προγραμματισμό του αλλά υπήρχε στην ομάδα και αποφασίστηκε να χρησιμοποιηθεί αυτός.

Επεξεργαστής: ATmega328P @16MHz
SRAM: 2KB
EEPROM: 1KB
Είσοδοι / Έξοδοι: 20



50

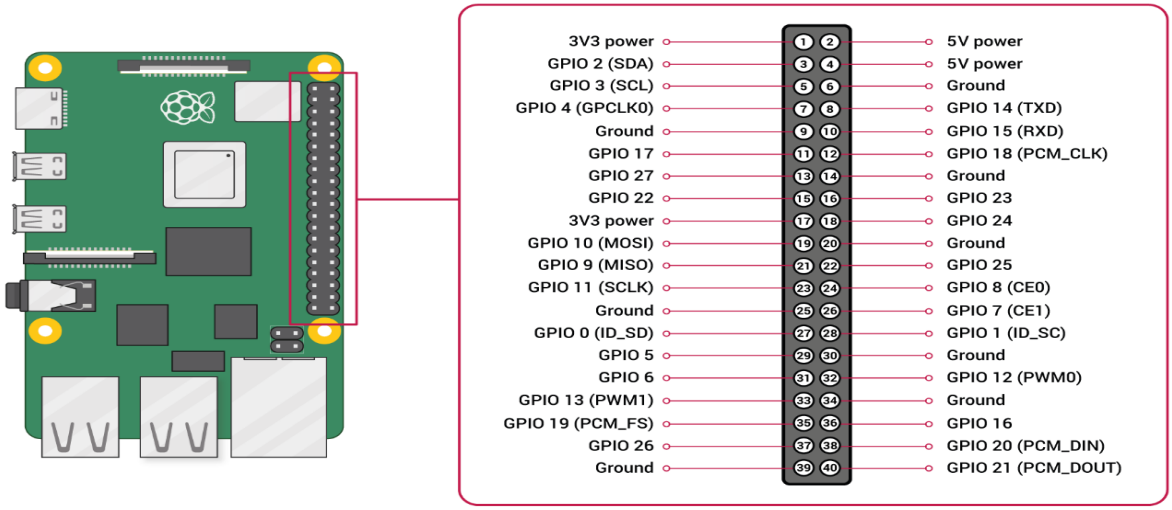
Arduino Uno: €22.00

Ethernet shield: €13.90 / 28.60€[51]

[50] https://images.prismic.io/circuito/8e3a980f0f964cc539b4cbbba2654bb660db6f52_arduino-uno-pinout-diagram.png?auto=compress,format

[51] https://store-usa.arduino.cc/products/arduino-uno-rev3

Raspberry Pi 4 Model B[52]
Επεξεργαστής: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
RAM: 1GB, 2GB, 4GB ή 8GB LPDDR4-3200 SDRAM (ανάλογα το μοντέλο)
Συνδεσιμότητα: 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE, Gigabit Ethernet, 2 θύρες USB 3.0 και 2 θύρες USB 2.0.
Είσοδοι/Έξοδοι: 40



Κόστος: 120€

Beagle Bone Black

[52] https://www.raspberrypi.com/documentation/computers/images/GPIO-Pinout-Diagram-2.png

Επεξεργαστής: AM335x 1GHz ARM® Cortex-A8
RAM: 512MB DDR3 RAM
ROM: 4GB 8-bit eMMC on-board flash storage
2x PRU 32-bit microcontrollers
Συνδεσιμότητα: Θύρες USB 2.0, Ethernet
Είσοδοι / Έξοδοι: 2x 46 pin headers

Beaglebone Black Pinout Diagram

P9

Function	Physical Pins		Function
DGND	1	2	DGND
VDD 3.3 V	3	4	VDD 3.3 V
VDD 5V	5	6	VDD 5V
SYS 5V	7	8	SYS 5V
PWR_BUT	9	10	SYS_RESET
UART4_RXD	11	12	GPIO_60
UART4_TXD	13	14	EHRPWM1A
GPIO_48	15	16	EHRPWM1B
SPIO_CSO	17	18	SPIO_D1
I2C2_SCL	19	20	I2C_SDA
SPIO_DO	21	22	SPIO_SLCK
GPIO_49	23	24	UART1_TXD
GPIO_117	25	26	UART1_RXD
GPIO_115	27	28	SP11_CSO
SP11_DO	29	30	GPIO_112
SP11_SCLK	31	32	VDD_ADC
AIN4	33	34	GND_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	ECAPWMO
DGND	43	44	DGND
DGND	45	46	DGND

LEGEND
Power, Ground, Reset
Digital Pins
PWM Output
1.8 Volt Analog Inputs
Shared I2C Bus
Reconfigurable Digital

P8

Function	Physical Pins		Function
DGND	1	2	DGND
MMC1_DAT6	3	4	MMC1_DAT7
MMC1_DAT2	5	6	MMC1_DAT3
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
EHRPWM2B	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
EHRPWM2A	19	20	MMC1_CMD
MMC1_CLK	21	22	MMC1_DAT5
MMC1_DAT4	23	24	MMC1_DAT1
MMC1_DATO	25	26	GPIO_61
LCD_VSYNC	27	28	LCD_PCLK
LCD_HSYNC	29	30	LCD_AC_BIAS
LCD_DATA14	31	32	LCD_DATA15
LCD_DATA13	33	34	LCD_DATA11
LCD_DATA12	35	36	LCD_DATA10
LCD_DATA8	37	38	LCD_DATA9
LCD_DATA6	39	40	LCD_DATA7
LCD_DATA4	41	42	LCD_DATA5
LCD_DATA2	43	44	LCD_DATA3
LCD_DATA0	45	46	LCD_DATA1

[53]

Κόστος: 70€
Έχει μικρότερη κοινότητα συγκριτικά με τις άλλες διαθέσιμες επιλογές αλλά υπάρχει υποστήριξη και παραδείγματα που θα βοηθήσουν την ανάπτυξη εφαρμογών.

[53] https://toptechboy.com/wp-content/uploads/2015/06/beaglebone-black-pinout.jpg

Πολλές επιλογές υπάρχουν στο εμπόριο κι από όλες τις παραπάνω επιλογές η πιο σώφρων σε σύγκριση τιμής και απόδοσης θα ήταν το Beagle Bone Black καθώς στοιχίζει κοντά 20€ παραπάνω από το Arduino Uno συνδυαστικά με το Ethernet shield αλλά έχει πολλές παραπάνω δυνατότητες και επεξεργαστική ισχύ και θα έκανε την ανάπτυξη πιο εύκολη.

Στα πλαίσια της πτυχιακής εργασίας και της «επιβεβαίωση της ιδέας / proof of concept» ο ελεγκτής που επιλέχθηκε είναι το Arduino Uno μαζί με την πλακέτα επέκτασης Ethernet καθώς ήταν ήδη στην κατοχή της ομάδας.

6.3 Ταυτοποίηση φοιτητή

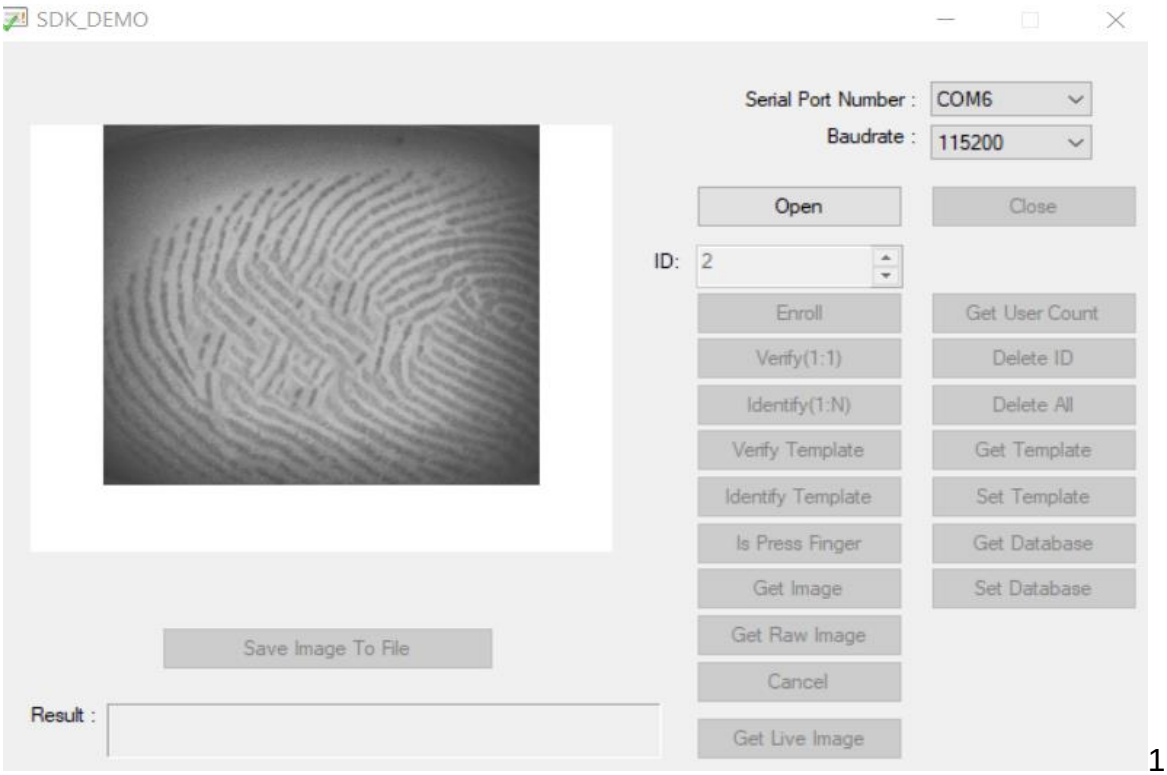
Ο σκοπός της ανάπτυξης του υλικολογισμικού είναι ο έλεγχος πρόσβασης των φοιτητών. Για να ελεγχθεί η ταυτότητα του χρήστη υπάρχουν πολλές μέθοδοι. Πιθανές λύσεις θα μπορούσαν να είναι ένα απλό πληκτρολόγιο και να εισάγεται ένας μοναδικός κωδικός ή η χρήση RFID καρτών. Το θέμα είναι ότι δεν θα μπορεί να πραγματοποιείται ταυτοπροσωπία του χρήστη καθώς οποιασδήποτε θα μπορούσε να πάρει την κάρτα ή να μάθει τον κωδικό και να χρησιμοποιεί τα «προνόμια» του εκάστοτε φοιτητή.
Οι καλύτερες λύσεις για ταυτοπροσωπία των φοιτητών είναι η χρήση των βιομετρικών τους αποτυπωμάτων με καλύτερη λύση απ' όλες να είναι η χρήση των δαχτυλικών αποτυπωμάτων καθώς συστήματα αναγνώρισης προσώπου είναι πιο ασταθή σε μεταβολές της εμφάνισης κτλ.

Για την λήψη του δαχτυλικού αποτυπώματος θα χρησιμοποιηθεί ο αισθητήρας δαχτυλικών αποτυπωμάτων GT-511C1R.
Πρόκειται για έναν αισθητήρα που αποτελείται από ένα 32bit επεξεργαστή και μια ασπρόμαυρη κάμερα. Τοποθετώντας το δάχτυλο ο χρήστης, εκτελεί τους ανάλογους αλγόριθμους για να ταχτοποιήσει το δαχτυλικό αποτύπωμα του χρήστη με τα υπόλοιπα που υπάρχουν στην βάση δεδομένων του.
Ο συγκεκριμένος αισθητήρας μπορεί να αποθηκεύσει 20 δακτυλικά αποτυπώματα τα οποία αποθηκεύονται τοπικά στον αισθητήρα και μπορεί να αναγνωρίσει το εκάστοτε σε οποιαδήποτέ από τις 360ο που μπορεί να εφαρμοστεί το δάχτυλο πάνω στον αισθητήρα.
Ο αριθμός των αποθηκευμένων δακτύλων είναι πολύ μικρός για μια εφαρμογή που θα δέχεται εκατοντάδες δάκτυλα αλλά τον καθιστά κατάλληλο για την «επιβεβαίωση της ιδέας» ειδικά με το μικρό του κόστος της τάξεως των 20€.
Ο αισθητήρας επικοινωνεί μέσω του πρωτοκόλλου UART και έτοιμες βιβλιοθήκες για το Arduino, (c++) σε python, και ένα SDK για την δοκιμή των λειτουργιών του καθιστούν την χρήση και την ενσωμάτωση του αισθητήρα πολύ εύκολη.

54

55

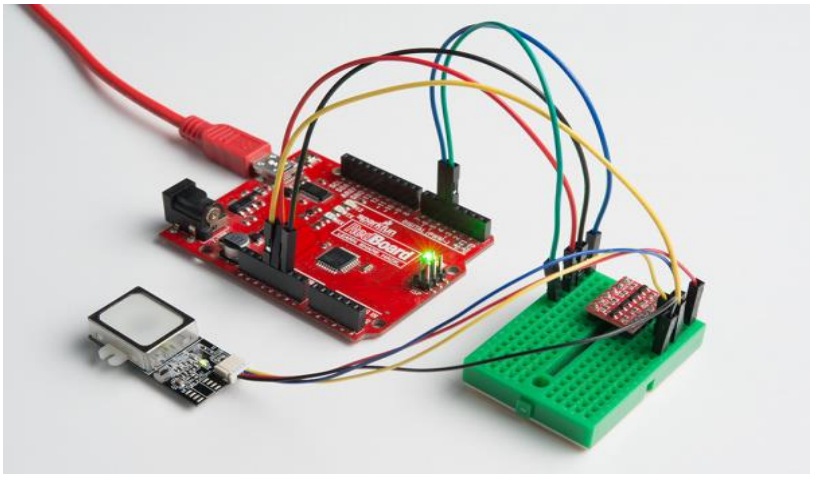
[54] https://cdn.sparkfun.com//assets/parts/9/9/7/2/13007-01.jpg
[55] https://cdn.sparkfun.com/assets/learn_tutorials/6/5/4/GT-511C1ROpenedTopView.jpg



1.1

Στην εικόνα 1.1 φαίνεται το SDK του αισθητήρα την ώρα που τραβάει μια εικόνα του δακτυλικού αποτυπώματος για να κάνει την αναγνώριση.

Ο αισθητήρας έχει τάση λειτουργίας τα 3V με αποτέλεσμα να μην είναι άμεσα συμβατός με τον μικροελεγκτή που θα χρησιμοποιηθεί ο οποίος δέχεται και στέλνει ηλεκτρικά σήματα της τάξεως των 5V.
Γι' αυτόν τον λόγο θα χρησιμοποιηθεί ένας μετατροπέας[56] επιπέδου λογικής ώστε να επιτευχθεί η μετατροπή του.



[56] https://cdn.sparkfun.com/assets/learn_tutorials/6/5/4/FPSQwiicRedBoardLLC.jpg

6.4 Ανάδραση με τον χρήστη

Η εφαρμογή χρειάζεται να δίνει μηνύματα στον χρήστη για την κατάσταση της αναγνώρισης κτλ.
Τα μηνύματα θα μπορούσαν να είναι ηχητικά μέσω ενός ηχείου, οπτικά με την χρήση led διαφόρων χρωμάτων ή θα μπορούσαν να εμφανίζονται σε μια οθόνη ώστε να είναι πιο κατανοητά σε άπειρους χρήστες.
Αποφασίστηκε να χρησιμοποιηθεί μια οθόνη δεκαέξι (16) χαρακτήρων και δύο (2) σειρών για την εμφάνιση μηνυμάτων.

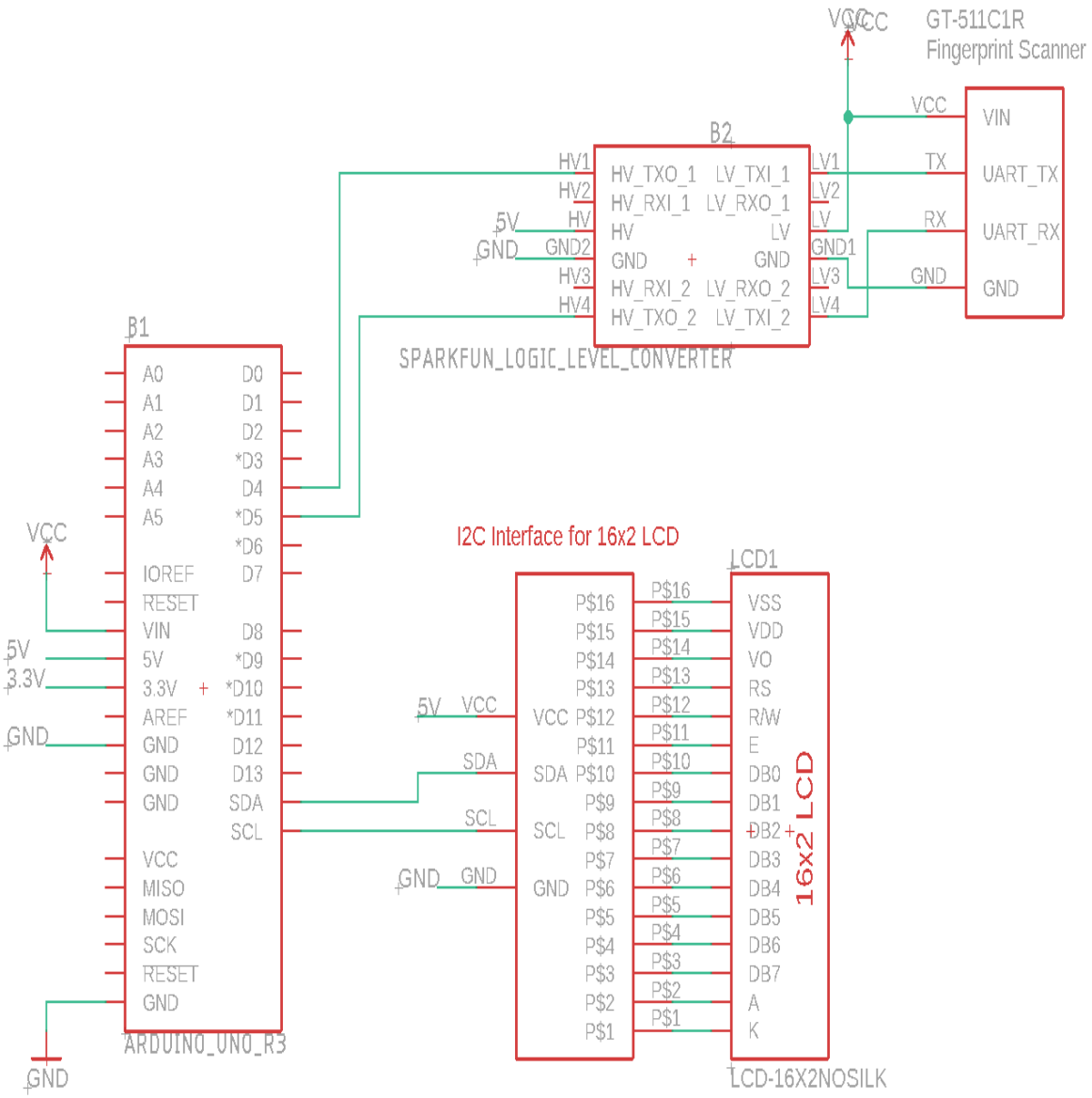
57

Η οθόνη αν και μικρή χρειάζεται έντεκα καλώδια για την διασύνδεση της στον μικροελεγκτή, γεγονός που δημιουργεί πρόβλημα καθώς δεν φτάνουν οι είσοδοι / έξοδοι για την διασύνδεση της. Λύση στο πρόβλημα θα δώσει μια διεπαφή πρωτοκόλλου I2C η οποία θα επιτρέψει τον έλεγχο της οθόνης μόνο με τέσσερα καλώδια. Τροφοδοσία Vcc, Gnd και τα καλώδια ελέγχου SDA SCL.

[57] https://grobotronics.com/images/detailed/110/18039-img_2076__1__grobo.jpg

6.5 Κατασκευή ηλεκτρικού κυκλώματος

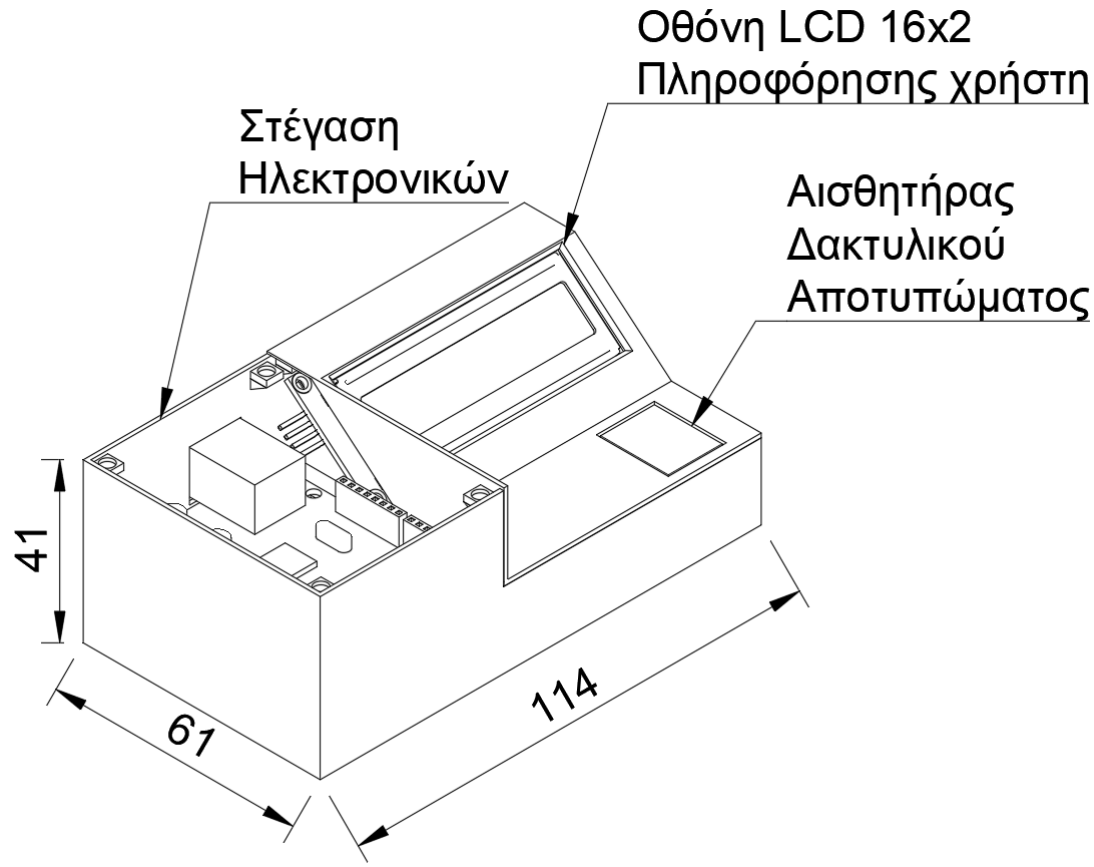
Με βάση όλα τα παραπάνω χρησιμοποιήθηκε το πρόγραμμα Autodesk Eagle για να κατασκευαστεί το σχηματικό διάγραμμα της εργασίας.

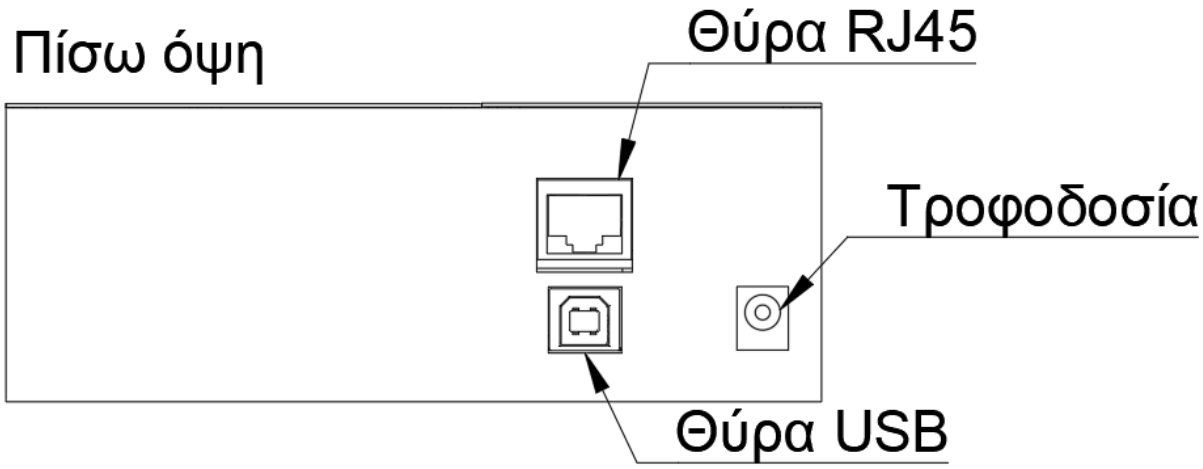


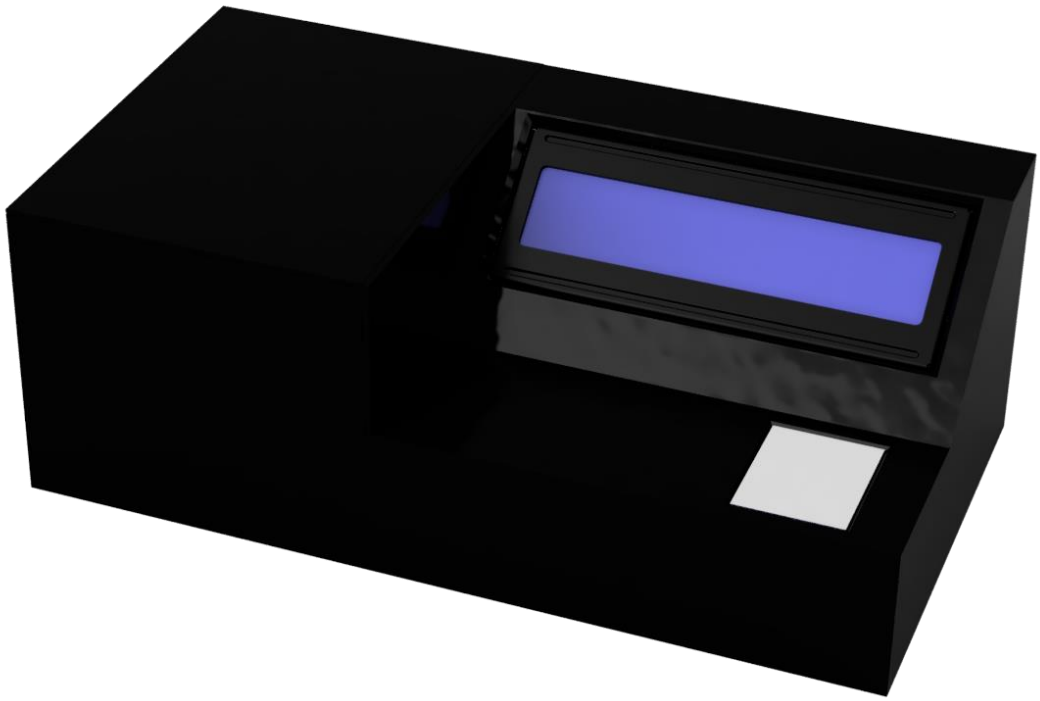
6.6 Κατασκευή «θήκης» για τα ηλεκτρονικά

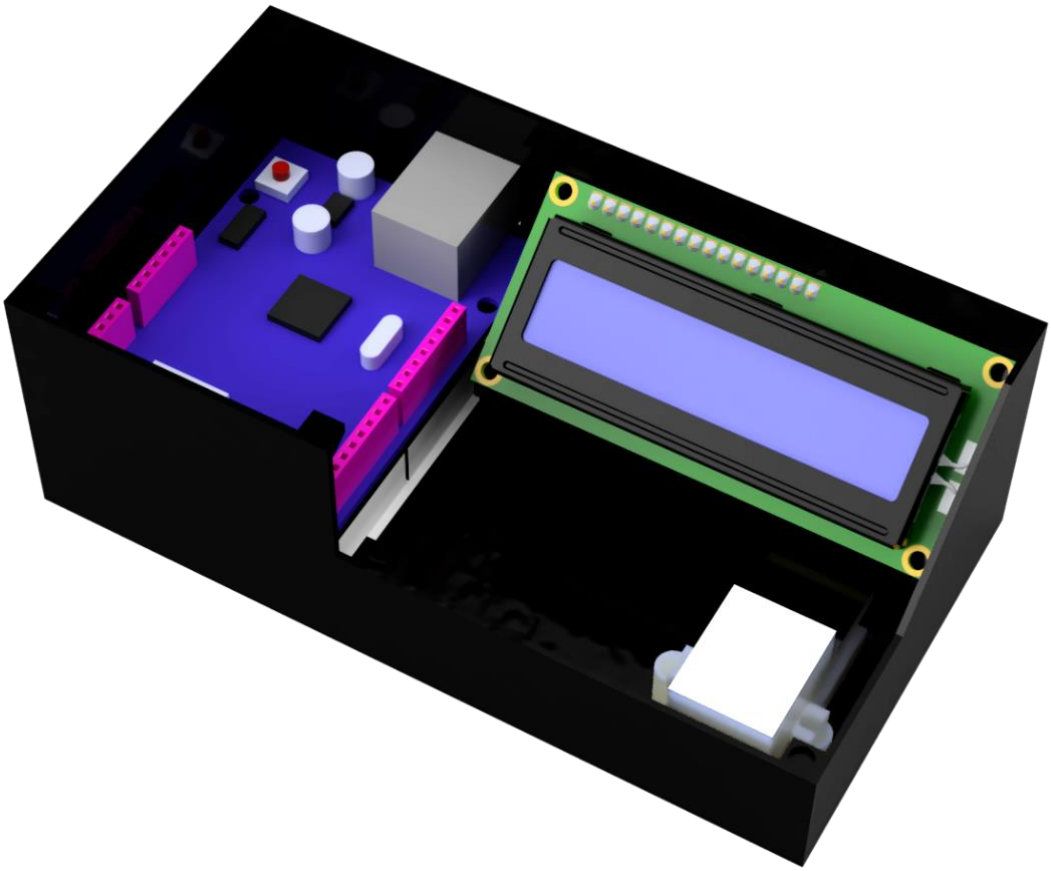
Τα ηλεκτρονικά επιμέρους μέρη το σώφρον θα ήταν να στεγάζονται και να μην είναι εκτεθειμένα στους χρήστες.
Σχεδιάστηκε λοιπόν μια θήκη στο τρισδιάστατο σχεδιαστικό πρόγραμμα Autodesk Fusion360 για να εξυπηρετήσει την παραπάνω ζήτηση και εν συνεχεία «ήρθε στην ζωή» μέσω τρισδιάστατης εκτύπωσης των επιμέρους κομματιών σε πλαστικό τύπου PET-G.
Αξίζει να σημειωθεί πως η μορφή της θήκης είναι τέτοια ώστε να εξυπηρετήσει την ανάγκη της εργασίας μας σε ερευνητικό επίπεδο. Αν μιλούσαμε για «τελικό προϊόν» προφανώς και θα έμοιαζε εντελώς διαφορετικό καθώς η οθόνη και ο αισθητήρας δαχτυλικών αποτυπωμάτων θα ήταν προσβάσιμα στον χρήστη / φοιτητή και τα υπόλοιπα ηλεκτρονικά (μικροελεγκτής κτλ) θα ήταν κρυμμένα σε άλλη θήκη ώστε να τοποθετηθούν σε μη εμφανές σημείο.

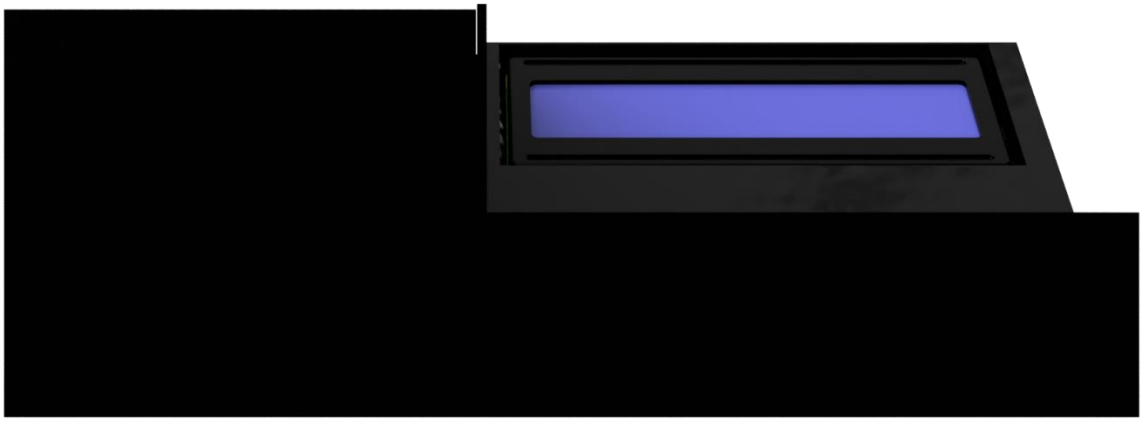
Παρακάτω φαίνονται οι ψηφιακές αποδόσεις (renderings) της θήκης καθώς και σχεδιαγράμματα με τις διαστάσεις και την επεξήγηση των επιμέρους μερών.




Πίσω όψη
Θύρα RJ45
Τροφοδοσία
Θύρα USB

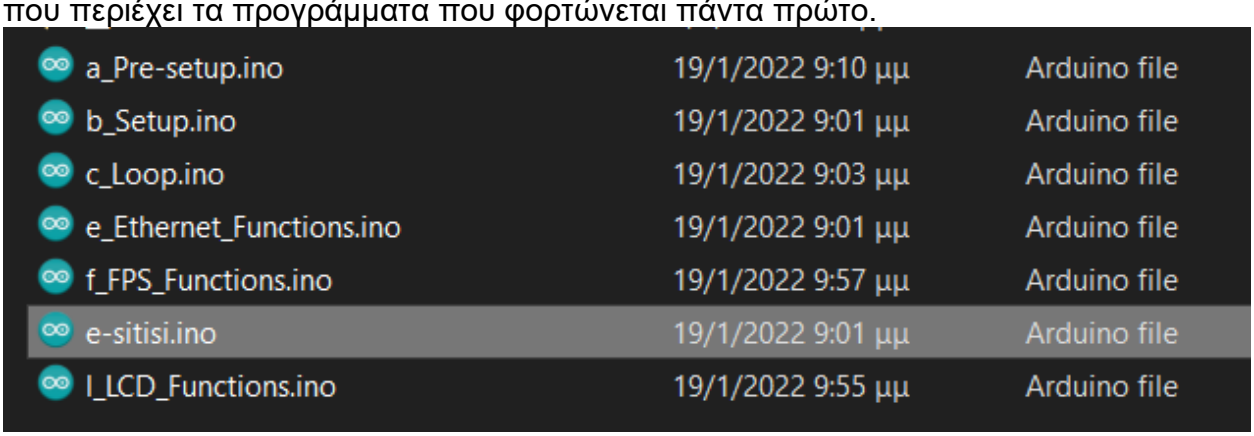






6.7 Το πρόγραμμα του μικροελεγκτή

Όπως αναφέρθηκε πιο νωρίς, ο κώδικας του Arduino γράφεται στην γλώσσα Arduino, δηλαδή σε C++ με μικρές προσθαφαιρέσεις μεθόδων και συναρτήσεων.
Το πρόγραμμα του μικροελεγκτή έχει κατάληξη .ino και μπορεί να μοιραστεί σε πολλά αρχεία αν αρχίσει ο κώδικας να γίνεται πολύ μεγάλος.
Τα αρχεία έχουν όλα την κατάληξη .ino, πρέπει να βρίσκονται στον ίδιο φάκελο και φορτώνονται με αλφαβητική σειρά, εκτός εκείνου που έχει το ίδιο όνομα με τον φάκελο που περιέχει τα προγράμματα που φορτώνεται πάντα πρώτο.



Τα αρχεία φορτώνονται στο Arduino IDE και υπάρχει μια συνέχεια μεταξύ τους, δηλαδή το IDE τα ενώνει όλα σαν να είναι ένα ενιαίο αρχείο.
Όταν κάποιος ανοίξει ένα καινούργιο πρόγραμμα θα αντικρίσει την εξής σελίδα:

Αμέσως γίνονται αντιληπτές δυο συναρτήσεις τύπου “void” δηλαδή που δεν επιστρέφουν τίποτα. Η μία συνάρτηση είναι η void setup() που κατά την ενεργοποίηση του μικροελεγτή τρέχει μια φορά μόνο και την void loop() που μετά την περάτωση της setup() τρέχει για πάντα σαν ένας ατέρμωνας βρόχος.
Με τα παραπάνω συμπεραίνουμε ότι οι δυο αυτές συναρτήσεις είναι οι πιο σηματνικές για ένα πρόγραμμα αλλά θα μπορούσαμε να πούμε ότι στην πραγματικότητα ένα πρόγραμμα για το Arduino μπορεί να χωριστεί σε πέντε (5) στάδια.

Το πρώτο στάδιο είναι η εισαγωγή τυχών εξωτερικών βιβλιοθηκών.
Στην περίπτωση της εργασίας αυτής έχουμε:

```
#include "FPS_GT511C3.h" // Βιβλιοθήκη για χρήση του αισθητήρα δακτυλικού
αποτυπώματος.

#include <Ethernet.h> //Βιβλιοθήκη για τον την κάρτα δικτύου ethernet
#include <SPI.h> //Βιβλιοθήκη για το πρωτόκολλο επικοινωνίας SPI
#include "SoftareSerial.h" //Βιβλιοθήκη για το πρωτόκολλο επικοινωνίας SPI
μέσω οποιασδήποτε πόρτας.
```

```
#include <Wire.h> //Βιβλιοθήκη για το πρωτόκολλο επικοινωνίας I2C.
#include <LiquidCrystal_I2C.h> //Βιβλιοθήκη για χρήση της LCD μέσω I2C
```

Το δεύτερο στάδιο θα μπορούσαμε να πούμε ότι είναι η αρχικοποίηση και ο ορισμός μεταβλητών που θα χρησιμοποιηθούν στο πρόγραμμα.

```
// Ορισμός της διεύθυνσης MAC που θα έχει η κάρτα δικτύου
byte mac[] = {  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

// Ορισμός της διεύθυνσης IP που θα έχει η κάρτα δικτύου
IPAddress ip_local(192, 168, 1, 250);
unsigned int localPort = 5000;      // Η πόρτα που θα «ακούει» η κάρτα δικτύου


// Ορισμός της διεύθυνσης IP που θα έχει ο κεντρικός υπολογιστής
byte ip_remote[] = {192,168,1,3};
unsigned int remotePort = 1100; // Η πόρτα που θα «μιλάει» η κάρτα δικτύου στον
κεντρικό υπολογιστή


// buffers για να στέλνει και να λαμβάνει δεδομένα
char packetBuffer[UDP_TX_PACKET_MAX_SIZE];  // buffer για τα εισερχόμενα πακέτα,

char ReplyBuffer[] = "acknowledged";        // Η απάντηση που θα στέλνει πίσω

// Δημιουργία ενός αντικειμένου τύπου EthernetUDP
// ώστε να λαμβάνει και να στέλνει το arduino πακέτα τύπου UDP
EthernetUDP Udp;

// Ορισμός του πρωτοκόλλου SPI μέσω κώδικα και όχι των πορτών που εξυπηρετούν
κανονικά το πρωτόκολλο.
// Το FPS (TX) συνδέεται στην πόρτα 4 του arduino (Software RX)
//Το FPS (RX) συνδέεται στην πόρτα 5 του arduino (Software TX)
// Δημιουργία ενός αντικειμένου τύπου FPS_GT511C3 και ορισμός σε ποιες πόρτες θα
επικοινωνεί
FPS_GT511C3 fps(4, 5); // SS_RX = πόρτα 4, SS_TX = πόρτα 5

/* Μεταβλητή που θα αποθηκεύει τον αριθμό του αποθηκευμένου δακτυλικού αποτυπώματος
* που ταιριάζει με αυτό του χρήστη. Αν είναι από 21 και πάνω τότε δεν έγινε η
*αναγνώριση.*/
int fps_id = 21;

// Μεταβλητές για τον έλεγχο της θέσης των γραμμάτων στην οθόνη LCD
int Li          = 16;
int Lii         = 0;
int Ri          = -1;
int Rii         = -1;


const int lcd_x = 2;
const int lcd_y = 16;

/* Δημιουργία ενός αντικειμένου τύπου LiquidCrystal_I2C για τον έλεγχο της οθόνης.
* το 0x27 είναι η διεύθυνση της οθόνης για το πρωτόκολλο I2C και οι μεταβλητές
* lcd_y, lcd_x για τον ορισμό των χαρακτήρων που υποστηρίζει. */
LiquidCrystal_I2C lcd(0x27, lcd_y, lcd_x);
```

Το τρίτο στάδιο είναι η συνάρτηση setup() που θα τρέξει μια φορά κατά την ενεργοποίηση του μικροελεγκτή ή μετά από μια επανεκκίνηση / reset.

Το τέταρτο στάδιο είναι η συνάρτηση loop() που θα τρέχει για πάντα μέχρι να αποσυνδεθεί από το ρεύμα ή να πραγματοποιηθεί reset.

```
void setup()
{
  Ethernet.begin(mac, ip_local); // Ξεκίνα το πρωτόκολλο Ethernet.

  Serial.begin(9600); // Ξεκίνα το πρωτόκολλο SPI του Arduino.

  lcd.begin(); // Ξεκίνα την επικοινωνία με την οθόνη.

  // Έλεγξε αν είναι συνδεδεμένη η κάρτα δικτύου και αν δεν είναι μην προχωρήσεις.
  if (Ethernet.hardwareStatus() == EthernetNoHardware) {
    Serial.println("Ethernet shield was not found.  Sorry, can't run without hardwa
re. :(");
    LCD_print("Ethernet shield",0);
    delay(300);
    LCD_print("was not found",1);
    delay(500);
    while (true) {
      delay(1); // Μην κάνεις τίποτα μέχρι να συνδεθεί η κάρτα δικτύου…
    }
  }
  // Έλεγξε αν το καλώδιο του Ethernet είναι συνδεδεμένο
  if (Ethernet.linkStatus() == LinkOFF) {
    Serial.println("Ethernet cable is not connected.");
    LCD_print("Ethernet cable",0);
    delay(300);
    LCD_print("is not connected",1);
    delay(500);
  }

  // Ξεκίνα το πρωτόκολλο UDP
  Udp.begin(localPort);

  delay(10);

  // Τύπωσε στην οθόνη ότι το Ethernet αρχικοποιήθηκε επιτυχώς
  LCD_print("Ethernet",0);
  delay(300);
  LCD_print("Innitialized",1);
  delay(500);

  lcd.clear(); // Σβήσε το περιεχόμενο της οθόνης

  fps.Open(); // Ξεκίνα την επικοινωνία με τον αισθητήρα δακτυλικού αποτυπώματος
  fps.SetLED(true); // Άνοιξε το LED στον αισθητήρα ώστε να βλέπει τα δάκτυλα
}
```

```
void loop() {

  fps_id = 21; // ορισμός του δακτύλου ως μη αναγνωρισμένο.

// ορισμός του μεγέθους του πακέτου με γνώμονα το μέγεθος του UDP πακέτου που
έφτασε
  int packetSize = Udp.parsePacket()
  // αν είναι >0 σημαίνει ότι ένα πακέτο έφθασε
  if (packetSize) {
    Serial.print(F("Received packet of size "));
    Serial.println(packetSize);
    Serial.print(F("From "));
    IPAddress remote = Udp.remoteIP(); // αποθήκευσε την IP διεύθυνση του αποστολέα
    // Τύπωσε την διεύθυνση IP (είναι τύπου char όχι τύπου String)
    for (int i = 0; i < 4; i++) {
      Serial.print(remote[i], DEC);
      if (i < 3) {
        Serial.print(".");
      }
    }
    // Τύπωσε την πόρτα που χρησιμοποιήθηκε για την αποστολή του πακέτου
    Serial.print(", port ");
    Serial.println(Udp.remotePort());

    // ανάγνωσε το περιεχόμενο του πακέτου και τύπωσε το
    Udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);
    Serial.print(F("Contents of message: "));
    Serial.println(packetBuffer);

    String response = String(packetBuffer);
    // κάλεσε την συνάρτηση sitisi_manager για να ελεγθεί τι απόφαση πρέπει να
    // παρθεί με βάση την απάντηση του κεντρικού υπολογιστή
    String dec = sitisi_manager(response);
    Serial.println(dec); // τύπωσε την απάντηση στην κονσόλα
    // τύπωσε την απάντηση στην οθόνη όσο το δάκτυλο είναι πάνω στον αισθητήρα
    LCD_print(dec, 0);
    LCD_print("", 1);
    delay(500);
    while (fps.IsPressFinger() == true) delay(100);
  }

  delay(10);

  // Αν υπάρχει δάκτυλο πάνω στον αισθητήρα τότε
  if (fps.IsPressFinger()) {
    fps_id = fps_identify(); // προσπάθησε να το αναγνωρίσεις
    if (fps_id < 21) { // και αν υπάρχει ταύτιση
      sendID(fps_id); // στείλε στον κεντρικό υπολογιστή τον α/αριθμό του δακτύλου
      delay(500);
    }
  }
  // Αλλιώς εμφάνισε στην οθόνη να βάλει ο χρήστης το δάκτυλο του
  else {
    LCD_print("Please", 0);
    LCD_print("input finger", 1);
  }
}
```

Το πέμπτο και τελευταίο στάδιο είναι η σύνταξη βοηθητικών συναρτήσεων, αν υπάρχουν πάντα.

```
// Συνάρτηση για την αποστολή UDP μηνύματος στον απομακρυσμένο υπολογιστή /server
void sendID(int message) {
  char id[1];
  itoa(message, id, 10); //μετατρέπει την μεταβλητή τύπου int σε τύπου char και την
  // εκχωρεί στην μεταβλητή id
  Udp.beginPacket(ip_remote, remotePort);
  Udp.write(id);
  Udp.endPacket();
  Serial.print(("Sent to ip: "));
  Serial.println(message);
}

// Συναρτήσεις για τον έλεγχο του αισθητήρα GT-511C3
// Πολλές δημιουργήθηκαν από τον Josh Hawley στις 23 Ιουλίου 2013

// Αναβόσβησε το LED στον αισθητήρα
void fps_blink(bool stat)
{
  fps.SetLED(stat); // turn on or off the LED inside the fps
}

// Αναγνώρισε το δακτυλικό αποτύπωμα
int fps_identify()
{
  fps.CaptureFinger(false);
  int id = fps.Identify1_N();

  if (id < 20)
  { // Αν ο α/α του δακτύλου είναι <20 τότε αναγνωρίστηκε ο χρήστης.
    // εμφάνισε τον αριθμό του χρήστη στην οθόνη και επέστρεψε τον αριθμό
    LCD_print("Verified ID:", 0);
    LCD_print(String(id), 1);
    return id; //
  }
  else
  { // Αλλιώς αν δεν βρέθηκε το δάκτυλο εμφάνισε το κατάλληλο μήνυμα
    LCD_print("Finger not found", 0);
    LCD_print("Would you like to register?", 1);
    return 21;
  }
}
```

```
// Συνάρτηση για την εκχώρηση νέου δακτύλου
void fps_enroll()
{
  // find open enroll id
  int enrollid = 0;
  bool usedid = true;
  while (usedid == true)
  {
    usedid = fps.CheckEnrolled(enrollid);
    if (usedid==true) enrollid++;
  }
  fps.EnrollStart(enrollid);

  // enroll
  Serial.print(F("Press finger to Enroll #"));
  Serial.println(enrollid);
  LCD_print("Press finger to Enroll #", 0);
  LCD_print(String(enrollid), 1);
  while(fps.IsPressFinger() == false) delay(100);
  bool bret = fps.CaptureFinger(true);
  int iret = 0;
  if (bret != false)
  {
    Serial.println(F("Remove finger"));
    fps.Enroll1();
    LCD_print("Remove finger", 0);
    LCD_print("", 1);
    while(fps.IsPressFinger() == true) delay(100);
    Serial.println(F("Press same finger again"));
    LCD_print("Press same finger again", 0);
    LCD_print("", 1);
    while(fps.IsPressFinger() == false) delay(100);
    bret = fps.CaptureFinger(true);
    if (bret != false)
    {
      Serial.println(F("Remove finger"));
      LCD_print("Remove finger", 0);
      LCD_print("", 1);
      fps.Enroll2();
      while(fps.IsPressFinger() == true) delay(100);
      Serial.println(F("Press same finger yet again"));
      LCD_print("Press same finger", 0);
      LCD_print("yet again", 1);
      while(fps.IsPressFinger() == false) delay(100);
      bret = fps.CaptureFinger(true);
      if (bret != false)
      {
        Serial.println(F("Remove finger"));

        LCD_print("Remove finger", 0);
        LCD_print("", 1);

        iret = fps.Enroll3();
        if (iret == 0)
        {
          Serial.println(F("Enrolling Successful"));
          LCD_print("Enrolling", 0);
          LCD_print("Successful", 1)
```

```
// Συνέχεια της συνάρτησης
        else
        {
          Serial.print(F("Enrolling Failed with error code:"));
          Serial.println(iret);

          LCD_print("Enrolling Failed. error:", 0);
          LCD_print(String(iret), 1);
        }
      }
      else{
        Serial.println(F("Failed to capture third finger"));
        LCD_print("Failed to capture", 0);
        LCD_print("third finger", 1);
      }

    }
    else{
      Serial.println(F("Failed to capture second finger"));
      LCD_print("Failed to capture", 0);
      LCD_print("second finger", 1);
    }
  }
  else{
    Serial.println(F("Failed to capture first finger"));
    LCD_print("Failed to capture", 0);
    LCD_print("first finger", 1);
  }

}

// Συνάρτηση για την απόφαση πρόσβασης του φοιτητή με βάση την απάντηση του server
String sitisi_manager(String response)
{
  /*  Αν η απάντηση είναι 0 ο χρήστης έχει φάει
   *  Αν η απάντηση είναι 1 ο χρήστης επιτρέπεται να φάει
   *  Αν η απάντηση είναι 2 ο χρήστης δεν επιτρέπεται να φάει
  */

  if (response == "0"){
    return "Exeis faei";
  }
  else if (response == "1"){
    return "Kali oreksi!";
  }
  else if (response == "2"){
    return "Den vrethike";
  }
  else{
    return "N/A";
  }
}
```

Από το πρόγραμμα γίνεται αντιληπτό ότι η επικοινωνία των δυο υπολογιστικών συστημάτων γίνεται μέσω του πρωτοκόλλου UDP. Ο λόγος που δεν χρησιμοποιείται κάποιο άλλο πρωτόκολλο όπως το MQTT είναι καθαρά ο περιορισμένος χώρος του Arduino Uno στην RAM.

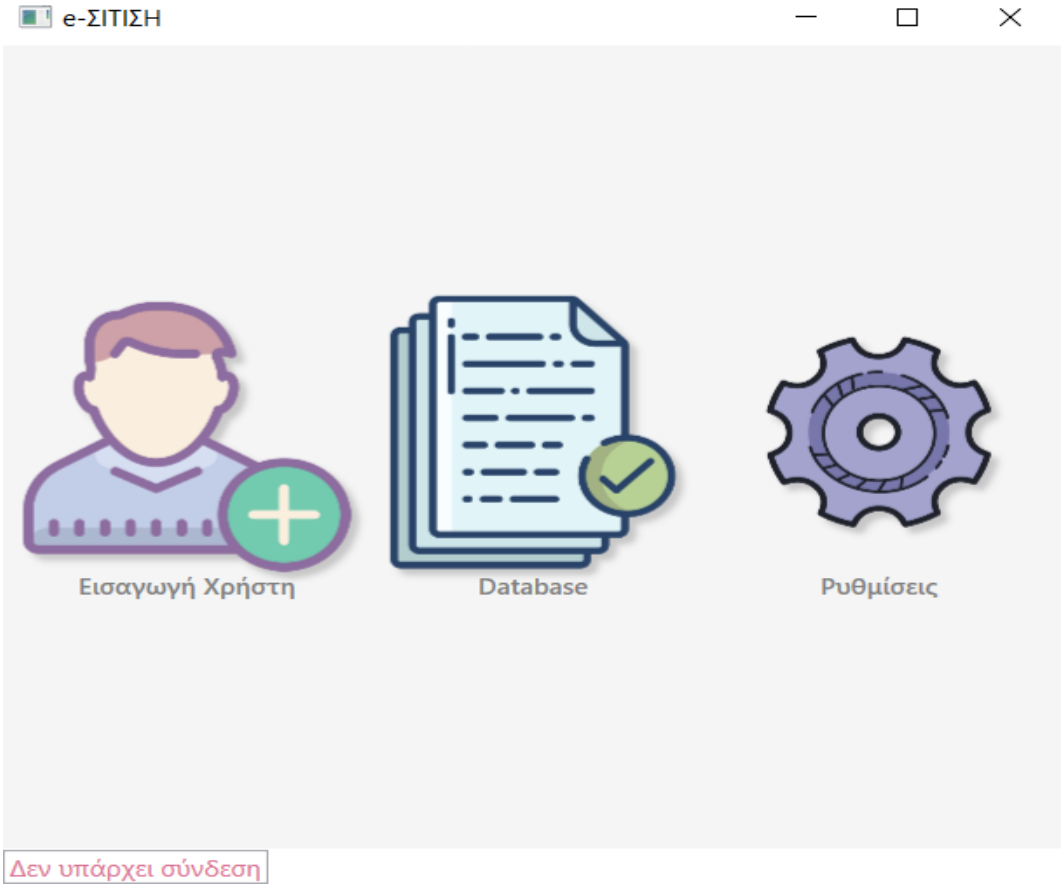
Κεφάλαιο 7ο: Το πρόγραμμα διαχείρισης των αποφάσεων e-ΣΙΤΙΣΗ

7.1 Η λογική του προγράμματος

Στο προηγούμενο υπό κεφάλαιο παρατέθηκε και αναλύθηκε ο κώδικας που εκτελείτε στο Arduino. Όπως είδαμε, στο περιβάλλον της σίτισης, αν αναγνωριστεί ο χρήστης / φοιτητής, το Arduino στην συνέχεια στέλνει ένα μήνυμα με το πρωτόκολλο UDP στην διεύθυνση IP και στην κατάλληλη πόρτα που περιμένει το μήνυμα ο απομακρυσμένος υπολογιστής / Server.
Εν συνεχεία, ο server θα ελέγξει αν υπάρχει ο χρήστης στην βάση δεδομένων και αν όντως υπάρχει ο χρήστης στην συνέχεια πραγματοποιεί άλλη μια πληθώρα από ελέγχους (αν είναι ενεργός, αν έχει ξανά λάβει το γεύμα του την δεδομένη συνεδρία της σίτισης, κτλ) και στέλνει πίσω στον μικροελεγκτή την απόφαση αν μπορεί να φάει ο φοιτητής ή όχι.
Το πρόγραμμα e-ΣΙΤΙΣΗ αναπτύχθηκε στην γλώσσα προγραμματισμού Visual C# με τον σκελετό (framework) WPF .NET.
Για την αποθήκευση και προσπέλαση πληροφοριών χρησιμοποιήθηκε η βάση δεδομένων SQLite η οποία έχει παρόμοιες εντολές «ερωτήματα SQL / SQL queries» με τις βιβλιοθήκες βάσεων δεδομένων όπως η MySQL με την διαφορά ότι λειτουργεί καθαρά τοπικά σαν μέρος της εφαρμογής και δεν χρειάζεται ένα πρόγραμμα server για την εκτέλεση της και την επικοινωνία της με την εκάστοτε εφαρμογή.

7.2 Το γραφικό περιβάλλον της εφαρμογής

Η πρώτη σελίδα του προγράμματος αποτελείται από τρια (3) κουμπιά.
Το κουμπί «Εισαγωγή Χρήστη» μεταφέρει τον υπεύθυνο εγγραφών σε μια σελίδα για να εγγράψει νέο χρήστη.
Το κουμπί «Database» εμφανίζει την βάση δεδομένων που υπάρχει στο σύστημα (δεν έχει αναπτυχθεί ακόμα).
Το κουμπί «Ρυθμίσεις» εμφανίζει την σελίδα για την ρύθμιση της επικοινωνίας του server με το σύστημα που αναπτύχθηκε και υπάρχει εγκατεστημένο στην σίτηση του πανεπιστημίου.
Στην κάτω πλευρά του παραθύρου υπάρχει μια γραμμή πληροφοριών που εμφανίζει την κατάσταση της σύνδεσης καθώς και μηνύματα για την επικοινωνία των υπολογιστικών συστημάτων.

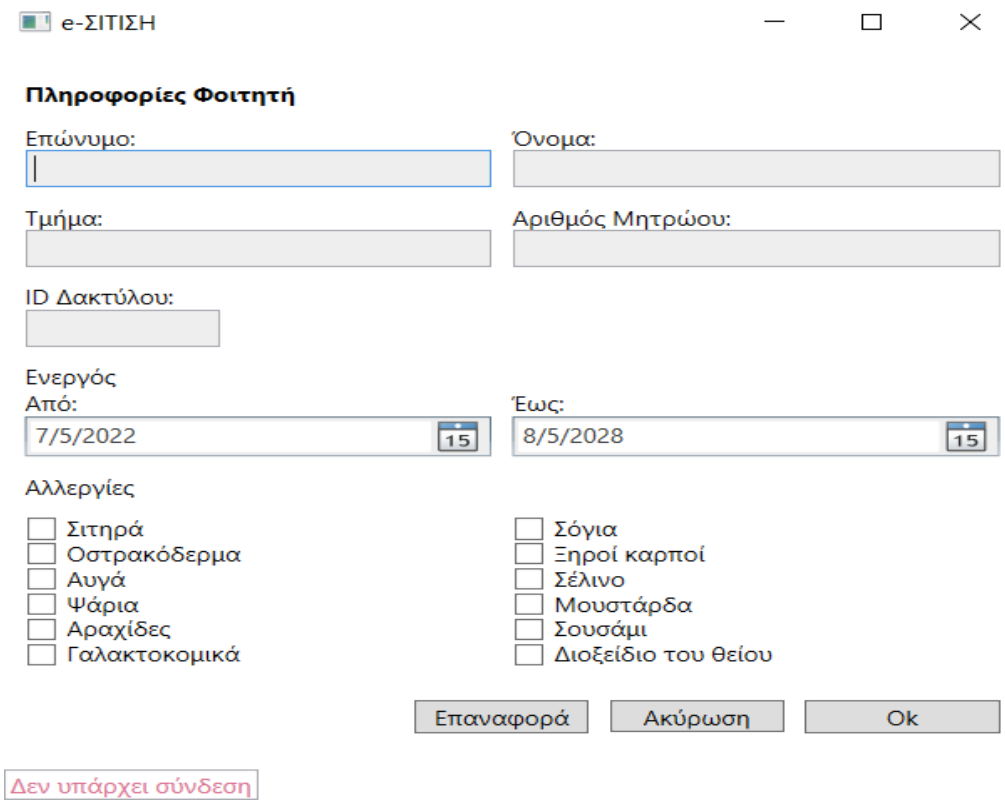


Παρακάτω φαίνεται η σελίδα εισαγωγής χρήστη και όλα τα ζητούμενα που πρέπει να συμπληρωθούν. Το «ID Δακτύλου» εμφανίζεται στην οθόνη του συστήματος της σίτισης όταν περαστεί στο σύστημα το δακτυλικό αποτύπωμα και είναι ένας αύξοντας αριθμός. Δυστυχώς η βάση δεδομένων είναι τοπική στον αισθητήρα και δεν μπορούμε να την αξιοποιήσουμε για να προσθέσουμε ή να αφαιρέσουμε χρήστες άμεσα / online με διαφορετικούς αισθητήρες με την τεχνολογία που χρησιμοποιούμε. Αν είχε γίνει χρήση ενός SBC όπως το raspberry pi, λόγω των καλύτερων και μεγαλύτερων πόρων θα

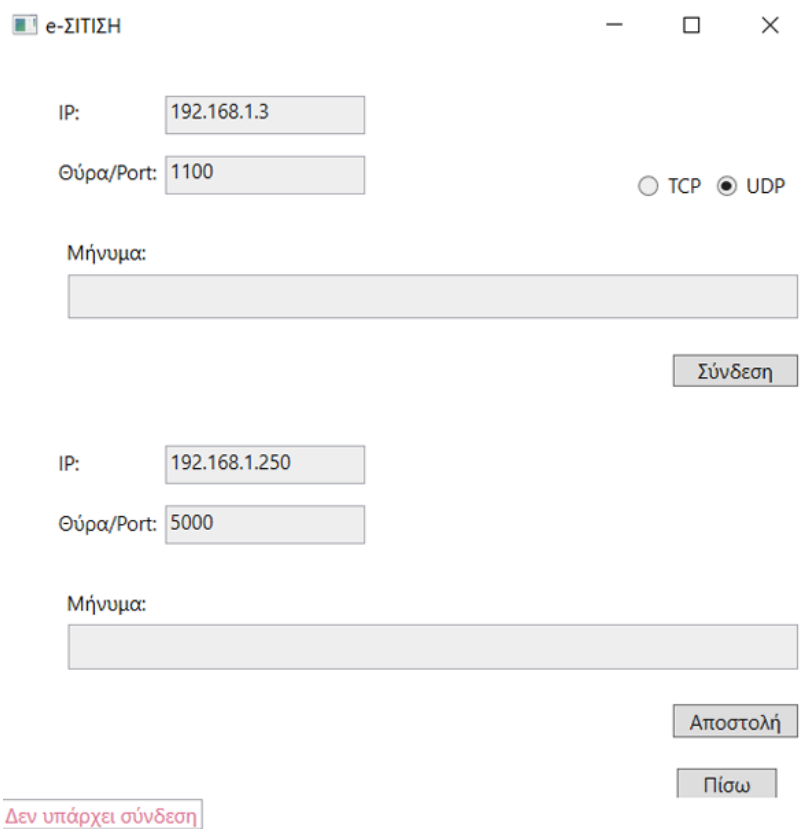
μπορούσε να επιτευχθεί έστω ένας συγχρονισμός όλων των αισθητήρων online στο τέλος κάθε ημέρας.

Τα διαστήματα ημερομηνιών εμφανίζονται αυτόματα με βάση την ημέρα που πάει να εισαχθεί ο χρήστης για να κάνει πιο γρήγορη την διαδικασία η ενεργοποίηση του έως μια ημερομηνία πάλι εμφανίζεται αυτόματα με βάση τον κανόνα Ν+2, όπου Ν τα χρόνια φοίτησης του εκπαιδευτικού προγράμματος.

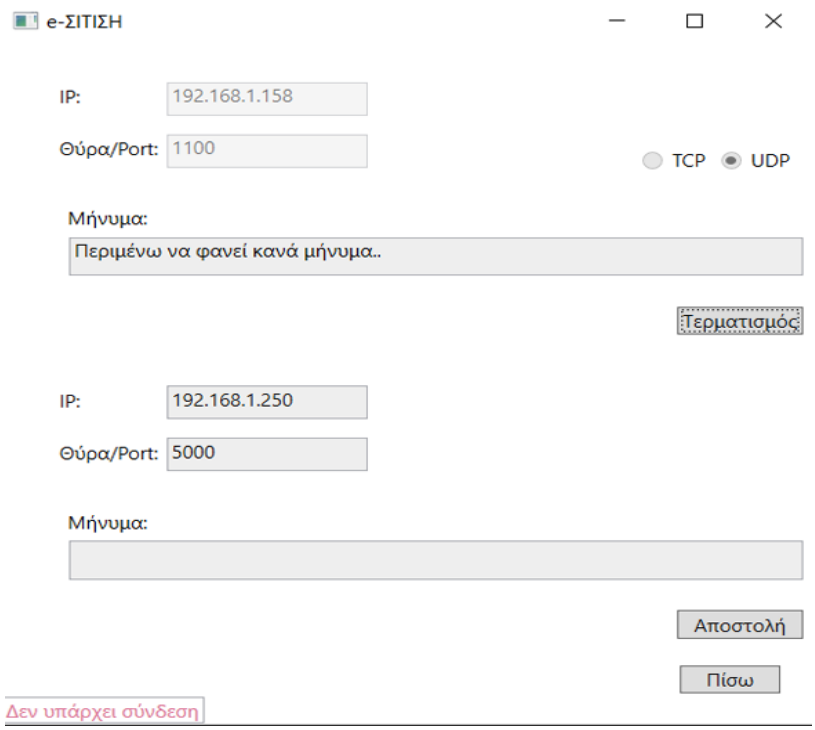
Υπάρχει πρόβλεψη για τυχόν διατροφικές αλλεργίες και ενημέρωση του φοιτητή με βάση το φαγητό της ημέρας αλλά δεν έχει ενσωματωθεί στην εργασία καθώς έπρεπε να αναπτυχθεί και σύστημα για τον κατάλογο φαγητών καθώς και τα υλικά που τα απαρτίζουν την εκάστοτε ημέρα.



Το παράθυρο «Ρυθμίσεις» εμφανίζει τις σχετικές ρυθμίσεις για την επίτευξη της επικοινωνίας των δυο υπολογιστικών συστημάτων.
Το επάνω κομμάτι είναι οι ρυθμίσεις που αφορούν τον Server, δηλαδή τον Υ/Η που εκτελείτε το πρόγραμμα και εισάγεται η διεύθυνση IP του ίδιου του υπολογιστή καθώς και η πόρτα που θα περιμένει τα μηνύματα. Το κουμπί σύνδεση πρέπει να πατηθεί για να ανοίξει την σύνδεση και να περιμένει μηνύματα από άλλες συσκευές. Στο κάτω κομμάτι του παραθύρου πρέπει να εισαχθεί η διεύθυνση IP του Arduino καθώς και η πόρτα που περιμένει να λάβει το μήνυμα. Το κουτί μήνυμα και το κουμπί αποστολή είναι καθαρά για λόγους απασφαλμάτωσης / debug ώστε να μπορεί να σταλεί αυθαίρετο μήνυμα στην συσκευή για να δοκιμαστεί η διασύνδεση τους.

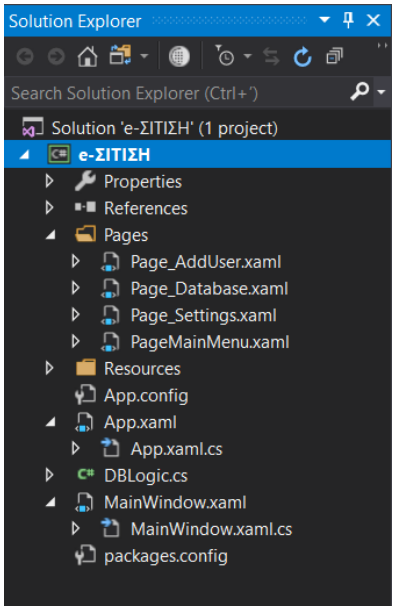


Πατώντας το κουμπί «Σύνδεση» περιμένει για μηνύματα από τα άλλα υπολογιστηκά συστήματα τα οποία θα εμφανίσει και στον χώρο κάτω από την ταμπέλα «Μήνυμα» αλλά και στην κάτω γραμμή πληροφοριών.


e-ΣΙΤΙΣΗ
IP:
192.168.1.158
Θύρα/Port:
1100
TCP
UDP
Μήνυμα:
Περιμένω να φανεί κανά μήνυμα..
Τερματισμός
IP:
192.168.1.250
Θύρα/Port:
5000
Μήνυμα:
Αποστολή
Πίσω
Δεν υπάρχει σύνδεση

7.3 Ο κώδικας του προγράμματος

Το πρόγραμμα αποτελείται από διάφορα αρχεία που φαίνονται στην παρακάτω εικόνα. Αρχικά υπάρχει το κεντρικό παράθυρο «MainWindow» που φιλοξενεί τις σελίδες «Pages».



Στον φάκελο Resources υπάρχουν οι εικόνες που χρησιμοποιήθηκαν για την εφαρμογή και στους φακέλους «Properties» και στα αρχεία «App.» περιέχονται ρυθμίσεις για την εφαρμογή. Στον φάκελο «References» εμπεριέχονται όλες οι βιβλιοθήκες που περιέχει το πρόγραμμα. Τέλος το αρχείο DBLogic.cs είναι μια κλάση που περιέχει όλη την λογική για την προσπέλαση και εγγραφή στην βάση δεδομένων.
Κάθε «παράθυρο» ή σελίδα είναι γραμμένα σε μια μορφή xml αρχείου που ονομάζεται xaml (ζαμλ) και έχει κατάληξη .xaml.

Το αρχείο αυτό περιέχει λοιπόν τον κώδικα για το τι να εμφανίσει ο υπολογιστής τρέχοντας την εφαρμογή. Το μοντέλο που διαχειρίζεται την αλληλεπίδραση του χρήστη με το πρόγραμμα έχει το ίδιο όνομα με το αρχείο xaml με κατάληξη .xaml.cs που ουσιαστικά είναι μια κλάση.

Κάθε φορά που εμφανίζεται ένα παράθυρο ή μια σελίδα στην πραγματικότητα εκτελείται ο κώδικας αυτής της κλάσης με πρώτη εκτέλεση του κατασκευαστή της κλάσης του παραθύρου και δημιουργεί το αντικείμενο. Σημεία στην κλάση που εκτελούνται με βάση την ανάδραση του χρήστη, για παράδειγμα το πάτημα ενός κουμπιού, εκτελούνται μόνο με την αντίστοιχη ενέργεια.

Παρακάτω είναι ο κώδικας όλου του προγράμματος και μια σύντομη περιγραφή του.
Τα αρχεία σχεδιασμού τύπου .xaml δεν θα σχολιαστούν καθώς αναφέρονται καθαρά στον σχεδιασμό της εφαρμογής.

MainWindow.xaml

```
<Window x:Class="e_ΣΙΤΙΣΗ.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:e_ΣΙΤΙΣΗ"
        mc:Ignorable="d"
        Title="e-ΣΙΤΙΣΗ" Height="490" Width="500" WindowState="Minimized">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="77*"/>
            <ColumnDefinition Width="423*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition Height="20" />
        </Grid.RowDefinitions>

        <Border Grid.Row="0" Padding="0" Background="Black" Grid.ColumnSpan="2">
            <Frame x:Name="frame_pages" Content="" VerticalAlignment="Stretch" Horizont
alAlignment="Stretch" Background="WhiteSmoke" NavigationUIVisibility="Hidden"/>
        </Border>

        <TextBox Grid.Row="1" x:Name="Info" HorizontalAlignment="Left" TextWrapping="Wr
ap" Text="Δεν υπάρχει σύνδεση" VerticalAlignment="Center" Foreground="PaleVioletRed" Gr
id.ColumnSpan="2" />

    </Grid>
</Window>
```

MainWindow.xaml.cs

Ο παρακάτω κώδικας είναι ο κώδικας διαχείρισης του «κύριου παραθύρου» της εφαρμογής.
Κάθε φορά που εκτελείται η εφαρμογή τρέχει πρώτα το παρακάτω κομμάτι κώδικα.

Συγκεκριμένα, δημιουργεί ένα νέο παράθυρο, δημιουργεί ένα νέο αντικείμενο της βάσεως δεδομένων που θα χρησιμοποιηθεί (Αυτό γίνεται για να τρέξει ο κατασκευαστής του αντικειμένου. Ο κώδικας του θα πραγματευθεί παρακάτω) και τέλος ορίζεται η σελίδα «PageMainWindow / Σελίδα κύριου παραθύρου» ως η σελίδα που θα εμφανίσει το παράθυρο. Αξίζει να σημειωθεί ότι η εφαρμογή αντί αν έχει ξεχωριστά αντιαισθητικά παράθυρα που θα ανοιγοκλείνουν, στεγάζει σελίδες σε ένα «Frame» και στην ουσία μένει σταθερά ανοιγμένο το παράθυρο και αλλάζουν απλώς οι σελίδες.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace e_ΣΙΤΙΣΗ
{
    public partial class MainWindow : Window
    {
        public static MainWindow main_window;
        public MainWindow()
        {
            InitializeComponent();
            main_window = this;
            DBLogic DB = new DBLogic("e_sitisi.sqlite");
            Page PageMainMenu = new Pages.PageMainMenu();
            this.frame_pages.NavigationService.Navigate(PageMainMenu);
        }
    }
}
```

PageMainMenu.xaml

```
<Page x:Class="e_ΣΤΙΣΗ.Pages.PageMainMenu"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:e_ΣΤΙΣΗ.Pages"
      mc:Ignorable="d"
      d:DesignHeight="450" d:DesignWidth="800"
      Title="PageMainMenu">

    <Border Padding="5"  Background="WhiteSmoke">
        <Grid Margin="0,0,0,0" HorizontalAlignment="Stretch" VerticalAlignment="Center"
 >
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="*"/>
            </Grid.ColumnDefinitions>
            <StackPanel Orientation="Vertical" HorizontalAlignment="Center" VerticalAli
gnment="Center">
                <Button x:Name="btn_AddUser" Grid.Column="0" Width="150" Height="150" B
orderThickness="0" Click="btn_AddUser_Click">
                    <Button.Background>
                        <ImageBrush ImageSource="/e-
ΣΤΙΣΗ;component/Resources/add_user_2.png" Stretch="Uniform"/>
                    </Button.Background>
                    <Button.Effect>
                        <DropShadowEffect Color="#FFE5E5E5"/>
                    </Button.Effect>
                </Button>
                <TextBlock Text="Εισαγωγή Χρήστη" FontWeight="DemiBold" HorizontalAlign
ment="Center" Foreground="Gray"/>
            </StackPanel>
            <StackPanel Grid.Column="1" HorizontalAlignment="Center" VerticalAlignment=
"Center">
                <Button x:Name="btn_Database"  Width="150" Height="150" BorderThickness
="0" Click="Btn_Database_Click">
                    <Button.Background>
                        <ImageBrush ImageSource="/e-
ΣΤΙΣΗ;component/Resources/document-alert.png" Stretch="Uniform"/>
                    </Button.Background>
                    <Button.Effect>
                        <DropShadowEffect Color="#FFE5E5E5"/>
                    </Button.Effect>
                </Button>
</Page>
```

```
                <TextBlock Text="Database" HorizontalAlignment="Center" Foreground="Gra
y" FontWeight="DemiBold"/>
            </StackPanel>
            <StackPanel  Grid.Column="2" HorizontalAlignment="Center" VerticalAlignment
="Center">
                <Button x:Name="btn_Settings" Width="150" Height="150" IsCancel="False"
 BorderThickness="0" Click="Btn_Settings_Click">
                    <Button.Background>
                        <ImageBrush ImageSource="/e-
ΣΙΤΙΣΗ;component/Resources/settings2.png" Stretch="Uniform"/>
                    </Button.Background>
                    <Button.Effect>
                        <DropShadowEffect Color="#FFE5E5E5"/>
                    </Button.Effect>
                </Button>
                <TextBlock Text="Ρυθμίσεις" Foreground="Gray" FontWeight="DemiBold" Hor
izontalAlignment="Center"/>
            </StackPanel>
        </Grid>
    </Border>
```

PageMainMenu.xaml.cs

Ο παρακάτω κώδικας είναι ο κώδικας διαχείρισης της «κύριας σελίδας» της εφαρμογής.

```
<Page x:Class="e_ΣΙΤΙΣΗ.Pages.Page_AddUser"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:e_ΣΙΤΙΣΗ.Pages"
      mc:Ignorable="d"
      d:DesignHeight="460" d:DesignWidth="500" Background="White"
      Title="Page_AddUser">

    <Border Padding="10">
        <StackPanel>

            <TextBlock Text="Πληροφορίες Φοιτητή" FontWeight="Bold" Margin="0 10" />

            <!-- Όνομα & Επώνυμο -->
            <Grid Margin="0 0 0 10">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>

                <!-- Επώνυμο -->
                <StackPanel Grid.Column="0" Margin="0 0 10 0">
                    <TextBlock Text="Επώνυμο:" />
                    <TextBox x:Name="textbox_Surname" Background="#eee" Padding="2" />
                </StackPanel>

                <!-- Όνομα -->
                <StackPanel Grid.Column="1">
                    <TextBlock Text="Όνομα:" />
                    <TextBox x:Name="textbox_Name" Background="#eee" Padding="2" />
                </StackPanel>

            </Grid>

            <!-- ΑΜ & Τμήμα -->
            <Grid Margin="0 0 0 10">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>

                <!-- Τμήμα -->
                <StackPanel Grid.Column="0" Margin="0 0 10 0">
                    <TextBlock Text="Τμήμα:" />
                    <TextBox x:Name="textbox_Department" Background="#eee" Padding="2"
/>
                </StackPanel>
            </Grid>
```

Ο κώδικας της αποτελείτε από τον κατασκευαστή της σελίδας και τρεις συναρτήσεις. Μια για κάθε κουμπί του μενού. Με το πάτημα του ανάλογου κουμπιού απλά εμφανίζει την εκάστοτε σελίδα στο κεντρικό παράθυρο.

PageAddUser.xaml

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace e_ΣΙΤΙΣΗ.Pages
{
    public partial class PageMainMenu : Page
    {
        public PageMainMenu()
        {
            InitializeComponent();
        }
        private void Btn_Settings_Click(object sender, RoutedEventArgs e)
        {
            Page Page_Settings = new Pages.Page_Settings();
            MainWindow.main_window.frame_pages.NavigationService.Navigate(Page_Settings
);
        }
        private void btn_AddUser_Click(object sender, RoutedEventArgs e)
        {
            Page Page_AddUser = new Pages.Page_AddUser();
            MainWindow.main_window.frame_pages.NavigationService.Navigate(Page_AddUser)
;
        }
        private void Btn_Database_Click(object sender, RoutedEventArgs e)
        {
            Page Page_Database = new Pages.Page_Database();
            MainWindow.main_window.frame_pages.NavigationService.Navigate(Page_Database
);
        }
    }
}
```

```
                <!-- AM -->
                <StackPanel Grid.Column="1">
                    <TextBlock Text="Αριθμός Μητρώου:" />
                    <TextBox x:Name="textbox_AM" Background="#eee" Padding="2" />
                </StackPanel>
            </Grid>

            <!-- ID -->
            <Grid Margin="0 0 0 10">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="2*" />
                    <ColumnDefinition Width="2*" />
                </Grid.ColumnDefinitions>
                <!-- ID -->
                <StackPanel Orientation="Vertical" Grid.Column="0" Margin="0 0 0 0">
                    <TextBlock Text="ID Δακτύλου:" />
                    <TextBox  x:Name="textbox_ID" Background="#eee" Padding="2" />
                </StackPanel>
            </Grid>


            <!-- Work Centres -->
            <TextBlock Text="Ενεργός" Margin="0 0 0 0" />
            <!-- Ενεργός; Από - Εως; -->
            <Grid Margin="0 0 0 10">
                <Grid.ColumnDefinitions>

                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>

                <!-- Τμήμα -->
                <StackPanel Grid.Column="0" Margin="0 0 10 0">
                    <TextBlock Text="Από:" />
                    <DatePicker x:Name="DatePicker_From" Background="#eee" Padding="2"

/>
                </StackPanel>

                <!-- AM -->
                <StackPanel Grid.Column="1">
                    <TextBlock Text="Έως:" />
                    <DatePicker x:Name="DatePicker_To" Background="#eee" Padding="2" />
                </StackPanel>

            <TextBlock Text="Αλλεργίες" Margin="0 0 0 10" />

            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                </Grid.ColumnDefinitions>
```

```
                <!-- Column 1 -->
                <StackPanel Grid.Column="0" Margin="0 0 10 0">
                    <CheckBox x:Name="Checkbox_sitira" Content="Σιτηρά" />
                    <CheckBox x:Name="Checkbox_ostraka" Content="Οστρακόδερμα" />
                    <CheckBox x:Name="Checkbox_eggs" Content="Αυγά" />
                    <CheckBox x:Name="Checkbox_fish" Content="Ψάρια"  />
                    <CheckBox x:Name="Checkbox_peanuts" Content="Αραχίδες" />
                    <CheckBox x:Name="Checkbox_milk" Content="Γαλακτοκομικά" />
                </StackPanel>

                <!-- Column 2 -->
                <StackPanel Grid.Column="1">
                    <CheckBox x:Name="Checkbox_soy" Content="Σόγια " />
                    <CheckBox x:Name="Checkbox_nuts" Content="Ξηροί καρποί" />
                    <CheckBox x:Name="Checkbox_selino" Content="Σέλινο" />
                    <CheckBox x:Name="Checkbox_mustard" Content="Μουστάρδα" />
                    <CheckBox x:Name="Checkbox_sesami" Content="Σουσάμι" />
                    <CheckBox x:Name="Checkbox_thio" Content="Διοξείδιο του θείου" />
                </StackPanel>

            </Grid>

            <!--Κουμπιά-->
            <Grid Margin="0 20 0 0">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="*" />

                </Grid.ColumnDefinitions>

                <Button x:Name="OkButton" Click="OkButton_Click" Margin="0 0 0 0" Grid.
Column="4" Content="Ok" />
                <Button x:Name="ResetButton" Click="ResetButton_Click" Margin="0 0 10 0
" Grid.Column="2" Content="Επαναφορά" />
                <Button x:Name="CancelButton" Click="CancelButton_Click" Margin="0 0 10
 0" Grid.Column="3" Content="Ακύρωση" Height="19" VerticalAlignment="Top" />

            </Grid>

        </StackPanel>
    </Border>
</Page>
```

PageAddUser.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace e_ΣΙΤΙΣΗ.Pages
{
    /// <summary>
    /// Interaction logic for Page_AddUser.xaml
    /// </summary>
    public partial class Page_AddUser : Page
    {
        public Page_AddUser()
        {
            InitializeComponent();
            DatePicker_From.SelectedDate = DateTime.Today.AddDays(-1);
            DatePicker_To.SelectedDate = DateTime.Today.AddYears(6);
        }
        private void ResetButton_Click(object sender, RoutedEventArgs e)
        {
            textbox_ID.Text = String.Empty;
            textbox_Surname.Text = String.Empty;
            textbox_Name.Text = String.Empty;
            textbox_AM.Text = String.Empty;
            textbox_Department.Text = String.Empty;
            textbox_AM.Text = String.Empty;
            DatePicker_From.SelectedDate = DateTime.Today.AddDays(-1);
            DatePicker_To.SelectedDate = DateTime.Today.AddYears(6);
            Checkbox_sitira.IsChecked = false;
            Checkbox_ostraka.IsChecked = false;
            Checkbox_eggs.IsChecked = false;
            Checkbox_fish.IsChecked = false;
            Checkbox_peanuts.IsChecked = false;
            Checkbox_milk.IsChecked = false;
            Checkbox_soy.IsChecked = false;
            Checkbox_nuts.IsChecked = false;
            Checkbox_selino.IsChecked = false;
            Checkbox_mustard.IsChecked = false;
            Checkbox_sesami.IsChecked = false;
            Checkbox_thio.IsChecked = false;
        }
```

```
        private void CancelButton_Click(object sender, RoutedEventArgs e)
        {
            Page PageMainMenu = new Pages.PageMainMenu();
            MainWindow.main_window.frame_pages.NavigationService.Navigate(PageMainMenu)
;
        }

        private void OkButton_Click(object sender, RoutedEventArgs e)
        {
            string DBname = "e-sitisi.sqlite";
            string FPS_ID = textbox_ID.Text;
            string Surname = textbox_Surname.Text;
            string Name = textbox_Name.Text;
            string AM = textbox_AM.Text;
            string Department = textbox_Department.Text;
            string Valid_From = DatePicker_From.SelectedDate.Value.Date.ToString("yyyy.
MM.dd");
            string Valid_To = DatePicker_To.SelectedDate.Value.Date.ToString("yyyy.MM.d
d");

            string Allergies = "";
            if (Checkbox_sitira.IsChecked == true)
            {
                Allergies = Allergies + "Σιτηρά,";
            }
            if (Checkbox_ostraka.IsChecked == true)
            {
                Allergies = Allergies + "Οστρακόδερμα,";
            }
            if (Checkbox_eggs.IsChecked == true)
            {
                Allergies = Allergies + "Αυγά,";
            }
            if (Checkbox_fish.IsChecked == true)
            {
                Allergies = Allergies + "Ψάρια,";
            }
            if (Checkbox_peanuts.IsChecked == true)
            {
                Allergies = Allergies + "Αραχίδες,";
            }
            if (Checkbox_milk.IsChecked == true)
            {
                Allergies = Allergies + "Γαλακτοκομικά,";
            }
            if (Checkbox_soy.IsChecked == true)
            {
                Allergies = Allergies + "Σόγια,";
            }
            if (Checkbox_nuts.IsChecked == true)
            {
                Allergies = Allergies + "Ξηροί καρποί,";
            }
```

```
            if (Checkbox_selino.IsChecked == true)
            {
                Allergies = Allergies + "Σέλινο,";
            }
            if (Checkbox_mustard.IsChecked == true)
            {
                Allergies = Allergies + "Μουστάρδα,";
            }
            if (Checkbox_sesami.IsChecked == true)
            {
                Allergies = Allergies + "Σουσάμι,";
            }
            if (Checkbox_thio.IsChecked == true)
            {
                Allergies = Allergies + "Διοξείδιο του θείου";
            }

            DBLogic DB = new DBLogic("e_sitisi.sqlite");

            DB.addtoTable_StudentInfo(FPS_ID, Surname, Name, AM, Department, Valid_From
, Valid_To, Allergies);

            Page PageMainMenu = new Pages.PageMainMenu();
            MainWindow.main_window.frame_pages.NavigationService.Navigate(PageMainMenu)
;
        }

    }
}
```

Ο παραπάνω κώδικας διαχειρίζεται την φόρμα της σελίδας «Page_AddUser»., Με το πάτημα του κουμπιού “Ακύρωση” απλά επιστρέφει στο κεντρικό μενού. Με το πάτημα του κουμπιού “Επαναφορά” απαλείφει όλες τις πληροφορίες που μπορεί να έβαλε στην φόρμα ο χειριστής. Με το πάτημα του κουμπιού “Ok” αποθηκεύει όλες τις πληροφορίες στην βάση δεδομένων "e_sitisi.sqlite" και στην συνέχεια επιστρέφει στο κεντρικό μενού.

Page_Settings.xaml

```
<Page x:Class="e_ΣΙΤΙΣΗ.Pages.Page_Settings"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:e_ΣΙΤΙΣΗ.Pages"
      mc:Ignorable="d"
      d:DesignHeight="460" d:DesignWidth="500" Background="White"
      Title="Page_Settings">

    <Grid>
        <StackPanel>
            <Grid Height="210" Width="500">
                <TextBox x:Name="Txb_Listen_IP" HorizontalAlignment="Left" Height="23"
Margin="99,27,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" Text="192.16
8.1.3" Background="#eee"/>
                <TextBlock HorizontalAlignment="Left" Margin="35,29,0,0" TextWrapping="
Wrap" Text="IP:" VerticalAlignment="Top"/>
                <TextBlock HorizontalAlignment="Left" Margin="35,65,0,0" TextWrapping="
Wrap" Text="Θύρα/Port:" VerticalAlignment="Top" />
                <TextBlock HorizontalAlignment="Left" Margin="40,113,0,0" TextWrapping=
"Wrap" Text="Μήνυμα:" VerticalAlignment="Top"/>
                <TextBox x:Name="Txb_Listen_Port" HorizontalAlignment="Left" Height="23
" Margin="99,63,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" Text="1100
" Background="#eee" />
                <TextBox x:Name="Txb_Listen_Message" HorizontalAlignment="Left" Height=
"27" Margin="41,134,0,0" VerticalAlignment="Top" Width="438" AcceptsReturn="True" Accep
tsTab="True" TextWrapping="Wrap" Background="#eee" IsReadOnly="True" />
                <Button x:Name="BtnConnect" Content="Σύνδεση" HorizontalAlignment="Left
" Margin="404,182,0,0" VerticalAlignment="Top" Width="75" Click="BtnConnect_Click"/>
                <RadioButton x:Name="RadioUdp" Content="UDP" HorizontalAlignment="Left"
 Margin="429,74,0,0" VerticalAlignment="Top" Background="#eee" GroupName="TransportProt
ocol" IsChecked="True"/>
                <RadioButton x:Name="RadioTcp" Content="TCP" HorizontalAlignment="Left"
 Margin="382,74,0,0" VerticalAlignment="Top" Background="#eee" GroupName="TransportProt
ocol" />
            </Grid>
```

```
            <Grid Height="210" Width="500">
                <TextBox x:Name="Txb_Send_IP" HorizontalAlignment="Left" Height="23" Ma
rgin="99,27,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" Text="192.168.
1.250" Background="#eee"/>
                <TextBlock HorizontalAlignment="Left" Margin="35,29,0,0" TextWrapping="
Wrap" Text="IP:" VerticalAlignment="Top"/>
                <TextBlock HorizontalAlignment="Left" Margin="35,65,0,0" TextWrapping="
Wrap" Text="Θύρα/Port:" VerticalAlignment="Top" />
                <TextBlock HorizontalAlignment="Left" Margin="40,113,0,0" TextWrapping=
"Wrap" Text="Μήνυμα:" VerticalAlignment="Top"/>
                <TextBox x:Name="Txb_Send_Port" Text="5000" HorizontalAlignment="Left"
Height="23" Margin="99,63,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"
Background="#eee" />
                <TextBox x:Name="Txb_Send_Message" HorizontalAlignment="Left" Height="2
7" Margin="41,134,0,0" VerticalAlignment="Top" Width="438" AcceptsReturn="True" Accepts
Tab="True" TextWrapping="Wrap" Background="#eee" />
                <Button x:Name="BtnSend" Content="Αποστολή" HorizontalAlignment="Left"
Margin="404,182,0,0" VerticalAlignment="Top" Width="75" Click="BtnSend_Click"/>
            </Grid>
            <Grid HorizontalAlignment="Right" Margin="0,10,20,0" Width="60">
                <Button x:Name="BtnMainWindow" Content="Πίσω" Click="BtnMainWindow_Clic
k">

                </Button>
            </Grid>
        </StackPanel>
    </Grid>
</Page>
```

Page_Settings.xaml.cs

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Windows;
using System.Windows.Input;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace e_ΣΙΤΙΣΗ.Pages
{
    /// <summary>
    /// Interaction logic for Page_Settings.xaml
    /// </summary>
    ///
    enum State
    {
        Listening,
        ReadyToListen
    }

    enum TransportProtocol
    {
        TCP,
        UDP
    }

    public partial class Page_Settings : Page
    {

        private State state;
        private Socket listeningSocket;
        private Socket sendingSocket;
        private TransportProtocol transportProtocol;

        public Page_Settings()
        {
            InitializeComponent();
            state = State.ReadyToListen;
            transportProtocol = TransportProtocol.UDP;

        }
```

```
private void BtnConnect_Click(object sender, RoutedEventArgs e)
{
    // Με το πάτημα του κουμπιού «Σύνδεση» περιμένει εισερχόμενα μηνύματα
    ListenAction();
}

private void BtnSend_Click(object sender, RoutedEventArgs e)
{
    // Αποστολή μηνύματος στον μικροελεγκτή για αποσφαλμάτωση
    SendAction(Txb_Send_Message.Text);
}

private void SendAction(string text_to_send)
{
    // Κώδικας για την αποστολή μηνύματος στον μικροελεγκτή

    MainWindow.main_window.Info.Text = "Started Sending";

    Boolean done = false;
    Boolean exception_thrown = false;

    string ip = Txb_Send_IP.Text;
    int port = int.Parse(Txb_Send_Port.Text);

    Socket sending_socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
    ProtocolType.Udp);

    IPEndPoint sending_end_point = new IPEndPoint(IPAddress.Parse(ip), port);

    if (text_to_send.Length == 0)
    {
        done = true;
        Txb_Send_Message.Text = "";
    }
    else
    {
        byte[] send_buffer = Encoding.ASCII.GetBytes(text_to_send);

        MainWindow.main_window.Info.Text = $"sending to address: {ip} port: {port}";

        try
        {
            sending_socket.SendTo(send_buffer, sending_end_point);
            Txb_Send_Message.Text = "";
        }
```

```
                catch (Exception send_exception)
        {
            exception_thrown = true;
            MainWindow.main_window.Info.Text = " Exception {0}" + send_exception.Messag
e;
        }
        if (exception_thrown == false)
        {
            MainWindow.main_window.Info.Text = $"Message has been sent to address: {ip}
 port: {port}";
        }
        else
        {
            exception_thrown = false;
            MainWindow.main_window.Info.Text = "The exception indicates the message was
 not sent.";
        }
    }
}
private void Listen()
{

    bool done = false;
    string listenIP = Txb_Listen_IP.Text;
    int listenPort = int.Parse(Txb_Listen_Port.Text);

    UdpClient listener = new UdpClient(listenPort);
    IPEndPoint receiving_endpoint = new IPEndPoint(IPAddress.Parse(listenIP), listenPor
t);

    string received_data;
    byte[] receive_byte_array;
    try
    {
        while (!done)
        {
            Txb_Listen_Message.Text = "Waiting for broadcast";
            Txb_Listen_Message.ScrollToEnd();

            receive_byte_array = listener.Receive(ref receiving_endpoint);
            Txb_Listen_Message.Text += $"Received a broadcast from {receiving_endpoint.
ToString()}";
            Txb_Listen_Message.ScrollToEnd();

            received_data = Encoding.ASCII.GetString(receive_byte_array, 0, receive_byt
e_array.Length);
            Txb_Listen_Message.Text += $"data follows \n{received_data}\n\n";
            Txb_Listen_Message.ScrollToEnd();
        }
    }
```

```
            catch (Exception e)
    {
        Txb_Listen_Message.Text += (e.ToString());
        Txb_Listen_Message.ScrollToEnd();
    }
    listener.Close();
}

private void ListenAction()
{

    try
    {


        string ip = Txb_Listen_IP.Text;
        int port = int.Parse(Txb_Listen_Port.Text);
        IPEndPoint ep = new IPEndPoint(IPAddress.Parse(ip), port);

        switch (State)
        {
            case State.Listening:
                listeningSocket?.Close();
                State = State.ReadyToListen;
                break;
            case State.ReadyToListen:
                Txb_Listen_Message.Text = "Περιμένω να φανεί κανά μήνυμα..";
                Txb_Listen_Message.ScrollToEnd();

                new Thread(() => ListenSocket(updateTextboxMessage, ep, transportProtoc
ol, out listeningSocket)) { IsBackground = true }.Start();
                State = State.Listening;
                break;
        }

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}
```

```
private static void ListenSocket(Action<string> callback, IPEndPoint endPoint, Transpor
tProtocol protocol, out Socket listener)
{
    byte[] buffer = new byte[1024];
    using (listener = new Socket(endPoint.AddressFamily, protocol == TransportProtocol.
TCP ? SocketType.Stream : SocketType.Dgram, ProtocolType.IP))
    {
        try
        {
            listener.Bind(endPoint);
            if (protocol == TransportProtocol.TCP)
                listener.Listen(200);

            while (true)
            {
                Socket handler = protocol == TransportProtocol.TCP ? listener.Accept()
: listener;
                string message = ConsumeSocketBuffer(handler, buffer, protocol);
                callback(message);
            }
        }
        catch (SocketException e)
        {
            if (e.SocketErrorCode != SocketError.Interrupted)
                MessageBox.Show($"Erro: {e.Message}");
        }
    }
}
static string ConsumeSocketBuffer(Socket socket, byte[] buffer, TransportProtocol proto
col)
{
    string message = "";
    int receivedBytes;
    while ((receivedBytes = socket.Receive(buffer)) > 0)
    {
        message += Encoding.UTF8.GetString(buffer, 0, receivedBytes);
        if (protocol == TransportProtocol.UDP)
            break;
    }
    return message;
}
```

```
private State State
{
    get => state;
    set
    {
        state = value;
        switch (state)
        {
            case State.ReadyToListen:
                BtnConnect.Content = "Σύνδεση";
                break;
            case State.Listening:
                BtnConnect.Content = "Τερματισμός";
                break;
        }
        bool isListenning = state != State.Listening;
        Txb_Listen_Port.IsEnabled = isListenning;
        Txb_Listen_IP.IsEnabled = isListenning;
        RadioTcp.IsEnabled = isListenning;
        RadioUdp.IsEnabled = isListenning;
    }
}
       private void updateTextboxMessage(string message)
        {
// Αυτή η συνάρτηση εκτελείτε κάθε φορά που λαμβάνεται μήνυμα από τον μικροελεγκτή.
            Dispatcher.Invoke(() =>
            {
// Δημιουργείτε σύνδεση με την βάση δεδομένων μέσω ενός καινούργιου αντικειμένου.
                DBLogic DB = new DBLogic("e_sitisi.sqlite");

                int sitisi_manager = DB.LogStudent(message);
// Η απόφαση για την πρόσβαση του φοιτητή στην υπηρεσία εξάγεται από την συνάρτηση
// DB.LogStudent(message) και στην συνέχεια αποστέλλεται στον μικροελεγκτή η απάντηση.
                SendAction(sitisi_manager.ToString());

                Txb_Listen_Message.Text += Environment.NewLine + message;
                Txb_Listen_Message.ScrollToEnd();

                MainWindow.main_window.Info.Text = Environment.NewLine + message;
                MainWindow.main_window.Info.ScrollToEnd();
            });
        }
        private void RdbUdp_Checked(object sender, RoutedEventArgs e)
        {
            transportProtocol = TransportProtocol.UDP;
        }
        private void RdbTcp_Checked(object sender, RoutedEventArgs e)
        {
            transportProtocol = TransportProtocol.TCP;
        }
        private void BtnMainWindow_Click(object sender, RoutedEventArgs e)
        {
            Page PageMainMenu = new Pages.PageMainMenu();
            MainWindow.main_window.frame_pages.NavigationService.Navigate(PageMainMenu)
;
        }
    }
}
```

Όλος ο παραπάνω κώδικας διαχειρίζεται την σύνδεση του προγράμματος με τα υπόλοιπα υπολογιστικά συστήματα. (τον μικροελεγκτή). Όπως αναφέρθηκε και πιο πριν, ο μικροελεγκτής επικοινωνεί μέσω του απλού πρωτοκόλλου UDP με το πρόγραμμα του απομακρυσμένου υπολογιστή / server καθώς αυτό το πρωτόκολλο επικοινωνίας υποστηρίζεται άμεσα από την επέκταση ethernet του μικροελεγκτή και λόγω πόρων δεν μπορούσε να αναπτυχθεί κώδικας για άλλο πρωτόκολλο επικοινωνίας. Το βασικότερο κομμάτι του κώδικα είναι η συνάρτηση *void updateTextboxMessage(string message)* που δέχεται σαν μήνυμα από τον μικροελεγκτή το ID του φοιτητή, δηλαδή τον αύξοντα αριθμό του δακτυλικού αποτυπώματος που έχει ο χρήστης περασμένο στην βάση δεδομένων την επεξεργασία του και στην συνέχεια την αποστολή της κατάλληλης απάντησης πίσω στον μικροελεγκτή.

DBLogic.cs

Η κλάση DBLogic.cs εμπεριέχει όλη την λογική για την δομή, δημιουργία, διαχείριση της βάσεως δεδομένων καθώς και την λογική για τον έλεγχο πρόσβασης των φοιτητών στις υπηρεσίες του πανεπιστημίου. (Προς το παρόν υποστηρίζεται μόνο η σίτιση)
Εν αρχή ορίζονται οι μεταβλητές που θα χρησιμοποιηθούν από την κλάση.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Data.SQLite;

namespace e_ΣΙΤΙΣΗ
{
    public class DBLogic
    {
        public String dbName { get; set; }
        public String dbPath { get; set; }

        public String DatabaseSource { get; set; }

        public struct Student
        {
            public bool Exists { get; set; }
            public bool IsValid { get; set; }
            public bool IsLogged { get; set; }
            public DateTime ValidTo { get; set; }
            public uint ID { get; set; }
            public string Surname { get; set; }
            public string Name { get; set; }
            public uint AM { get; set; }
            public string Department { get; set; }
            public bool HasEatenBreakfast { get; set; }
            public bool HasEatenLaunch { get; set; }
            public bool HasEatenDinner { get; set; }
        }
        public Student student = new Student();
```

Συγκεκριμένα αφορούν το όνομα της βάσης δεδομένων, την τοπική διεύθυνση που είναι αποθηκευμένη καθώς και το μοντέλο με τις πληροφορίες του φοιτητή.

```
public DBLogic (String db_Name)
{
    dbName = db_Name;
    string path = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
    dbPath = System.IO.Path.Combine(path, dbName);

    SetUp(dbName);
}
```

Ο κατασκευαστής της κλάσης, η συνάρτηση δηλαδή που τρέχει όταν δημιουργείται το αντικείμενο “DBLogic”. Ο κατασκευαστής δέχεται ως είσοδο το όνομα της βάσης δεδομένων, έχει προεπιλεγμένο χώρο αποθήκευσης της βάσης τον φάκελο «Τα έγγραφα» μου και εν συνεχεία καλεί την μέθοδο SetUp(dbName).

```
public void SetUp(string dbName)
        {
            //Αν η βάση δεδομένων υπάρχει
            if (File.Exists(dbPath))
            {
             //Αν υπάρχει συνδέεται τυπικά για να ελέγξει την σύνδεση
                CheckDB(dbPath);
            }
           //Αν δεν υπάρχει δημιουργεί την βάση δεδομένων
            else
            {
                CreateDB(dbPath);
            }
        }
```

```
public void CheckDB(string DBPath)
{
    SQLiteConnection log_dbConnection;
    DatabaseSource = String.Format("Data Source={0}; Version=3;", DBPath);
    log_dbConnection = new SQLiteConnection(DatabaseSource);
    log_dbConnection.Open();
    log_dbConnection.Close();
}
```

```
// Η συνάρτηση αυτή εισάγει στον πίνακα CurrentDay την ώρα, ποιος χρήστης και σε ποιο
γεύμα της σίτισης έχει φάει
public void addtoTable_CurrentDay(string FPS_ID, int meal)
        {
            string date = DateTime.UtcNow.ToShortDateString();
            string time = DateTime.Now.ToShortTimeString();

            if (meal == 1)
            {
                student.HasEatenBreakfast = true;
            }
            else if (meal == 2)
            {
                student.HasEatenLaunch = true;
            }
            else if (meal == 3)
            {
                student.HasEatenDinner = true;
            }

            int Breakfast = Convert.ToInt32((student.HasEatenBreakfast));
            int Launch = Convert.ToInt32((student.HasEatenLaunch));
            int Dinner = Convert.ToInt32((student.HasEatenDinner));
            string sqlLogDate = $"REPLACE INTO CurrentDay (Date, FPS_ID, Breakfast, Din
ner, Launch) VALUES ('{date}','{FPS_ID}','{Breakfast}','{Launch}','{Dinner}')";
            using (SQLiteConnection log_dbConnection = new SQLiteConnection(DatabaseSou
rce))
            {
                log_dbConnection.Open();
                using (SQLiteCommand cmd = new SQLiteCommand(sqlLogDate, log_dbConnecti
// Η συνάρτηση αυτή εισάγει στον πίνακα StudentInfo πληροφορίες του χρήστη
public void addtoTable_StudentInfo(string FPS_ID, string Surname, string Name, string A
M, string Department, string Valid_From, string Valid_To, string Allergies)
{

    string sqlStudentInfo = $"INSERT INTO StudentInfo (FPS_ID, Surname, Name, AM, Depar
tment, Valid_From, Valid_To, Allergies) VALUES ('{FPS_ID}', '{Surname}', '{Name}', '{AM
}', '{Department}', '{Valid_From}', '{Valid_To}', '{Allergies}')";
    using (SQLiteConnection addStudent_dbConnection = new SQLiteConnection(DatabaseSour
ce))
    {
        addStudent_dbConnection.Open();
        using (SQLiteCommand cmd = new SQLiteCommand(sqlStudentInfo, addStudent_dbConne
ction))
        {
            cmd.ExecuteNonQuery();
        }
    }

}
```

```
// Η συνάρτηση CheckToday ελέγχει αν έχει φάει ο συγκεκριμένος φοιτητής στο
συγκεκριμένο γεύμα
public void CheckToday(string FPS_ID)
{
    if (student.Exists && student.IsValid)
    {

        string date = DateTime.UtcNow.ToShortDateString();
        string sql_CheckToday = $"SELECT BREAKFAST, LAUNCH, DINNER, DATE FROM CurrentDa
y WHERE FPS_ID = {FPS_ID}";

        using (SQLiteConnection checkLog_dbConnection = new SQLiteConnection(DatabaseSo
urce))
        {
            checkLog_dbConnection.Open();
            using (SQLiteCommand cmd = new SQLiteCommand(sql_CheckToday, checkLog_dbCon
nection))
            {
                using (SQLiteDataReader reader = cmd.ExecuteReader())
                {
                    bool exists = reader.HasRows;
                    if (exists)
                    {
                        reader.Read();
                        if (reader[3].ToString() == date)
                        {
                            student.IsLogged = true;
                            student.HasEatenBreakfast = Convert.ToBoolean(Int32.Parse(r
eader[0].ToString()));
                            student.HasEatenLaunch = Convert.ToBoolean(Int32.Parse(read
er[1].ToString()));
                            student.HasEatenDinner = Convert.ToBoolean(Int32.Parse(read
er[2].ToString()));
                        }
                    }
                }
            }
        }

    }
}
```

```
// Η συνάρτηση DetermineMeal βρίσκει αν είναι ανοικτή η σίτιση και ποιο γεύμα της
ημέρας είναι ενεργό εκείνη την στιγμή

public int DetermineMeal()
{
    TimeSpan Breakfast_start = new TimeSpan(8, 0, 0); //Αρχή του πρωινού
    TimeSpan Breakfast_end = new TimeSpan(11, 59, 0); //Λήξη του πρωινού
    TimeSpan Launch_start = new TimeSpan(13, 30, 0); //Αρχή του μεσημεριανού
    TimeSpan Launch_end = new TimeSpan(16, 0, 0); //Λήξη του μεσημεριανού
    TimeSpan Dinner_start = new TimeSpan(17, 30, 0); //Αρχή του βραδινού
    TimeSpan Dinner_end = new TimeSpan(20, 59, 0); //Λήξη του βραδινού

    TimeSpan now = DateTime.Now.TimeOfDay;

    //Αν η ώρα είναι μέσα στο διάστημα του πρωινού επέστρεψε «1»
    if ((now > Breakfast_start) && (now < Breakfast_end))

        return 1;
    }
    //Αν η ώρα είναι μέσα στο διάστημα του μεσημεριανού επέστρεψε «2»
    else if ((now > Launch_start) && (now < Launch_end))
    {
        return 2;
    }
    //Αν η ώρα είναι μέσα στο διάστημα του μεσημεριανού επέστρεψε «3»
    else if ((now > Dinner_start) && (now < Dinner_end))
    {
        return 3;
    }
    // Αλλιώς δεν υπάρχει ενεργό γεύμα και επέστρεψε «0»
    else
    {
        return 0;
    }

}
```

```
//Η συνάρτηση αυτή επιτρέπει την πρόσβαση στον φοιτητή όταν όλες οι απαιτήσεις
πληρούνται και τον εγγράφει στην βάση ότι έφαγε. Η πληροφορία που επιστρέφεται από την
μέθοδο στέλνεται στον μικροελεγκτή ώστε να εμφανίσει το ανάλογο μήνυμα.
public int LogStudent(string FPS_ID)
{
    CheckStudent(FPS_ID);
    CheckToday(FPS_ID);

    int current_meal = DetermineMeal();

    if (current_meal != 0) // Το γεύμα είναι 0 όταν δεν είναι ανοιχτή η σίτιση
    {
        if (student.IsValid) // Αν ο φοιτητής είναι ενεργός στο πρόγραμμα σίτισης
        {
            if ((student.HasEatenBreakfast && current_meal == 1)  // Αν το τρέχον γεύμα
 της σίτης είναι το πρωινό & ο φοιτητής έχει φάει ήδη
                || (student.HasEatenLaunch && current_meal == 2)  // Αν το τρέχον γεύμα
 της σίτης είναι το μεσημεριανό & ο φοιτητής έχει φάει ήδη
                || (student.HasEatenDinner && current_meal == 3)) // Αν το τρέχον γεύμα
 της σίτης είναι το βραδινό & ο φοιτητής έχει φάει ήδη
            {

                return 0; // Δεν επιτρέπεται να ξαναφάει
            }
            else
            {
                addtoTable_CurrentDay(FPS_ID, current_meal); // Δημιούργησε την εγγραφή
 στον πίνακα
                return 1; // Επέτρεψε την λήψη φαγητού
            }
        }
        else
        {
            return 2;
        }
    }
    else
    {
        return 3;
    }
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. "The Computer for the 21st Century" - Scientific American Special Issue on Communications, Computers, and Networks, September, 1991 The Computer for the 21st Century e-book

2. Derek De Solla Price. Gears from the Greeks: The Antikythera Mechanism - A Calendar Computer from ca. 80 B.C. Science History Publications, New York, 1975 e-book

3. Ubiquitous Computing: Smart Devices, Environments and Interactions Stefan Poslad © 2009 John Wiley & Sons, Ltd. ISBN: 978-0-470-03560-3 A catalogue record for this book is available from the British Library. ISBN 978 0 470 03560 3 (H/B) Set in 9/11pt Times by Integra Software Services Pvt. Ltd. Pondicherry, India Printed and bound in Great Britain by Antony Rowe, Chippenham, UK e-book

4. Everywhere: The Dawning Age of Ubiquitous Computing 1st Edition Publisher: New Riders Publishing; 1st edition (March 10, 2006) Language: EnglishPaperback: 288 pages ISBN-10: 0321384016 ISBN-13: 978-0321384010 Author Adam Greenfield

5. IEEE Annals of the History of Computing, Edited by Gerardo Con Diaz, Former name(s) Annals of the History of Computing, Publisher IEEE, ISO 4 IEEE Ann. Hist. Comput. e-book as seen on 2017

6. Ubiquitous Computing Fundamentals, Edited by John Krumm, Publisher CRC Press, Taylor & Francis Group, A CHAPMAN & HALL BOOK

7. Εισαγωγή στις Βάσεις Δεδομένων, Συγγραφέας: ΔΡ. Βασίλειος Τ. Ταμπακάς Τρίτη έκδοση: Πάτρα, Ιανουαριος 2021, έκδοση Βασίλειος Τ. Ταμπακάς

8. Εισαγωγή στις Σύγχρονες Τεχνολογίες Επικοινωνιών, Συγγραφέας Ανδρέας Σ. Πομπόρτσης, Εκδόσεις Τζίολα 2009

9. Foundations of Computer Science 2nd Edition, edited by Behrouz Forouzan & Firouz Musharraf, Copyright 2008 Thomson Learning, Published by Cengage Learning Company

10. The c Programming Language, 2nd edition, edited by Brian W. Kernighan & Dennis M. Ritchie, Published by Prentice Hall Software Series, Brian Kernighan, Advisor 1988

11. Computer Networking a Top-Down Approach, edited by Jim Kurose & Keith Ross, Published by Pearson Education, INC 2008

12. The Arduino Projects Book, edited by Scott Fitzgerald & Michael Shiloh & Tom Igor, 3rd Issue 2015, Published by Arduino SRL

13. AFFAIRS, D. O. (January 6, 2012). MANAGEMENT OF DATA BREACHES. Washington, DC 20420: DEPARTMENT OF VETERANS AFFAIRS.

14. AMERICAN, S. (1991). Communications, Computers and Networks. SCIENTIFIC AMERICAN INC.

15. GRIVAS, A. (2018). GDPR: Τι Είναι Και Σε Ποιους Απευθύνεται. grillmagazine.gr.

16. IEEE. (2005, January-March). IEEE Annals of the History of Computing. IEEE Computer Society.

17. JOHN, K. (2010). UBIQUITOUS COMPUTING FUNDAMENTALS. CRS PRESS.

18. Kang, B.-H. (January 2007). Ubiquitous Computing Environment Threats and Defensive Measures. International Journal of Multimedia and Ubiquitous Engineering, 47–60.

19. Shiloh, S. F. (3rd Issue May 2015). Arduino Projects Book. Torino Italy: Arduino Srl.

20. Stevens, G. (April 10, 2012). Data Security Breach Notification Laws. Legislative Attorney.

21. Weiser, M. (1991). Communications, Computers and Networks (Vol. Special Issue). America: Scientific American Inc.