ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

« Διάταξη εμφάνισης μηνυμάτων με LCD και έλεγχο φωτεινότητας μέσω αισθητήρων φωτός»

Του φοιτητή :
Αθανάσιου Κουγιουμτζόγλου
Αρ. Μητρώου: 516059

Επιβλέπων
Αριστοτέλης Καζακόπουλος
Καθηγητής

Ημερομηνία: 14/1/2021

Τίτλος Π.Ε.: Διάταξη εμφάνισης μηνυμάτων με LCD και έλεγχο φωτεινότητας μέσω αισθητήρων φωτός

Κωδικός Δ.Ε. 19150

Ονοματεπώνυμο φοιτητή: Αθανάσιος Κουγιουμτζόγλου

Ονοματεπώνυμο εισηγητή Αριστοτέλης Καζακόπουλος

Ημερομηνία ανάληψης Π.Ε. 22-11-2019

Ημερομηνία περάτωσης Π.Ε. 14/1/2021

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

Στους γονείς μου Αρίστο και Δέσποινα ,στην αδερφή μου Φιλίτσα
και σε όσους με εμπνέουν να γίνω καλύτερος.

Πρόλογος

Η συγκεκριμένη Π.Ε επιλέχθηκε με σκοπό την θεμελίωση και απόκτηση γνώσεων πάνω σε συστήματα μικροελεγκτών, καθώς επίσης και η εντριβή πάνω στον προγραμματισμό τους για την επίτευξη διάφορων λειτουργιών. Με το πέρας της Π.Ε έγιναν κατανοητές οι βασικές αρχές των μικροελεγκτων, το προγραμματιστικό τους περιβάλλον ,τα προγράμματα σχεδίασης και προσομοίωσης ,τα ηλεκτρονικά χαρακτηριστικά των εξαρτημάτων καθώς επίσης και τα όργανα που χρησιμοποιήθηκαν για τις διάφορες μετρήσεις. Σκοπός της εργασία αυτής είναι η σχεδίαση ,η κατασκευή και ο προγραμματισμός του συστήματος που θα εμφανίζει σε μια οθόνη LCD μηνύματα απο τον χρήστη και ο έλεγχος φωτεινότητας της οθόνης μέσω αισθητήρων φωτός. Τέλος όλα τα παραπάνω έχουν ως αποτέλεσμα την ανάπτυξη κριτικής σκέψης στον τομέα της ηλεκτρονικής και την απόκτηση γνώσεων και εμπειριών επάνω σε ενσωματωμένα συστήματα.

Περίληψη

Στην παρακάτω εργασία παρουσιάζονται τα είδη και η δομή των ενσωματωμένων συστημάτων ανάλογα με τις απαιτήσεις του συστήματος και την απόδοση του μικροεπεξεργαστή. Η χρήση του μικροεπεξεργαστή αποτελεί τον πυρήνα του συστήματος καθώς είναι υπεύθυνος για τον έλεγχο και την διαχείριση των διάφορων λειτουργιών και με βάση τα χαρακτηριστικά τους, μπορούμε να έχουμε την καλύτερη απόδοση του κυκλώματος.Οι μικροεπεξεργαστές αποτελούν την επανάσταση της τεχνολογίας και παρουσιάζουν, απο τα πρώτα μόλις χρόνια της δημιουργίας τους, ραγδαία εξέλιξη που αποσκοπεί στην βελτιώση των λειτουργιών τους. Μία απο τις σημαντικότερες λειτουργίες είναι η ταχύτητα επεξεργασίας και η διαχείρηση των δεδομένων, που αποθηκεύονται στην μνήμη του, καθώς οι περισσότερες, πολύπλοκες, εφαρμογές απαιτούν διαθέσιμο χώρο μνήμης ώστε να γίνεται σωστή διαχείριση πόρων. Στην συγκεκριμένη εργασία σχεδιάστηκε και κατασκευάστηκε σύστημα εμφάνισης μηνυμάτων σε οθόνη LCD με αυτόματη ένταση φωτεινότητας και πλήρη έλεγχο του συστήματος μέσα απο ιστοσελίδα. Μέσα στην ιστοσελίδα εμφανίζονται πληροφορίες σχετικά με την διαχείρηση της μνήμης του μικροελεγκτή, το μήνυμα που στέλνει ασύρματα στην LCD και την κατάσταση επαναπρογραμματισμού του μικροελεγκτή στην λειτουργία Over The Air Update. Η παραπάνω κατασκευή μπορεί να χρησιμοποιηθεί σε πολλούς τομείς που απαιτούν την χρήση οθόνης και τον ασύρματο έλεγχο του μικροελεγκτή. Όλα τα παραπάνω θα αναλυθούν με λεπτομέρειες στα επόμενα κεφάλαια.

«LCD message display device and brightness control via light sensors»

«Athanasios Kougioumtzoglou»

Abstract

This thesis analyses the types and structure of embedded systems depending on the system requirements and the performance of the microprocessor. The use of microprocessors is vital for the system as it is responsible for controlling and managing the various operations and based on their characteristics, we can achieve the best circuit performance. Microprocessors revolutionised technology and showed, from the first years of their creation, a rapid development aimed at improving their use. One of the most important operations is the processing speed and management of the data stored in its memory, as the more complex applications require memory space in order to properly manage resources. For the experimental part of this project a system for displaying messages on an LCD screen with automatic brightness and complete control of the system through a web page was designed and built. The website displays information about the microcontroller's memory management, the message it sends wirelessly to the LCD, and the microcontroller reprogramming mode in Over-The-Air-Update mode. This circuit board can be used in situations where the use of a monitor and the wireless control of the microcontroller is required. All the above will be analyzed in detail in the following chapters.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επόπτη μου, τον κύριο Αριστοτέλη Καζακόπουλο για την πολύτιμη βοήθεια, συμβουλές και καθοδήγησή του όλους αυτούς τους μήνες. Θα ήθελα επίσης να ευχαριστήσω τους φίλους, την οικογένειά μου και όλους τους άλλους που με υποστήριξαν καθ 'όλη τη συγγραφή αυτής της διατριβής.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Κεφάλαιο 2

Κεφάλαιο 3

Κατάλογος Εικόνων

Κεφάλαιο 1

Κεφάλαιο 2

Κεφάλαιο 3

Συντομογραφίες

- Π.Ε Πτυχιακή Εργασία
- LCD Liquid Crystal Display (Οθόνη υργού κρυστάλλου)
- ΕΣ Ενσωματωμένο Σύστημα
- CPU Central Processing Unit (Κεντρική Μονάδα Επεξεργασίας)
- ALU Arithmetic and Logic Unit (Αριθμητική και Λογική Μονάδα)
- A/D Analog to Digital (Αναλογικό σε Ψηφιακό)
- D/A Digital to Analog (Ψηφιακό σε Αναλογικό)
- DSP Digital Signal Processor (Επεξεργαστής Ψηφιακού σήματος)
- RISC Reduced Instruction Set Computers (Υπολογιστής Περιορισμένου Συνόλου Εντολών)
- IDE Integrated Development Environment (Ολοκληρωμένο Περιβάλλον Ανάπτυξης)
- LAN Local Area Network (Τοπικό Δίκτυο)
- WAN Wide Area Network (Δίκτυο Ευρείας Περιοχής)
- GB Giga Byte (Μονάδα Μέτρησης Μνημών)
- MME (Μέσα Μαζικής Ενημέρωσης)
- PIC (Peripheral Interface MicroController)
- DIP (Dual In Line)
- QFP (Quad Flat Package)
- QFN (Quad Flat No leads)
- MSSP (Master Synchronous Serial Port)
- SPI (Serial Peripheral Interface)
- I2C (Interface to Communicate)
- PWM (Pulse Width Modulation)
- ISP (In System Programming)
- SFR Special Function Register (Καταχωρητής Ειδικής Λειτουργίας)
- GPR General Purpose Register (Καταχωρητής Γενικού Σκοπού)
- EEPROM (Electrically Erasable Programmable Read Only Memory)
- AVR (ALF Vegard Risc)
- MCU (Micro Controller Unit)
- SRAM (Static Random Access Memory)
- ADC (Analog to Digital Converter)
- DAC (Digital to Analog Converter)
- I2C (Inter-Integrated Circuit)
- UART (Universal Asynchronous Receiver/Transmitter)
- I2S (Inter-IC Sound)
- CAN (Controller Area Network)’
- SoC (System on Chip)
- AP (Access Point)
- MIMO (Multiple Inputs Multiple Outputs)
- IRAM (Instruction Random Access Memory)
- DRAM (Dynamic Random Access Memory)
- BT (Burst Type)

- SPIRAM ή PSRAM (Serial Peripheral Interface RAM ή Pseudo Static RAM αντίστοιχα)
- IDE (Integrated Development Environment)
- GPIO (General Purpose Input / Output)
- LCD (Liquid Crystal Display)
- HTML (Hyper Text Markup Language)
- LED (Light Emitting Diode)
- RS (Register Select)
- RW (Read/Write)
- E (Enable)
- LDR (Light Dependent Resistor)
- POT (Potentiometer)
- nF (nano Farad)
- uF (micro Farad)
- mF (mili Farad)
- DC (Direct Current)
- AC (Alternative Current)
- PCB (Printed Circuit Board)
- SPIFFS (SPI Flash File System)
- DNS (Domain Name System)

Κεφάλαιο 1: Εισαγωγή στα ενσωματωμένα συστήματα

1.1 Εισαγωγή στα ενσωματωμένα συστήματα

Ένας πρώτος ορισμός για τα Ενσωματωμένα συστήματα (ΕΣ) είναι ότι σε αυτά ανήκουν οποιαδήποτε συσκευή η οποία περιλαμβάνει έναν προγραμματιζόμενο επεξεργαστή με ειδική λειτουργία μέσα σε μεγαλύτερο μηχανικό ή ηλεκτρικό σύστημα. Αναλυτικότερα μπορούμε να πούμε ότι ένα ΕΣ μπορεί να είναι προγραμματιζόμενο ή μη-προγραμματιζόμενο, ανάλογα με την εφαρμογή για την οποία σχεδιάζεται, όπου στο τέλος όλα τα τμήματα του υπολογιστικού συστήματος συνεργάζονται για να φέρουν εις πέρας διάφορες λειτουργίες. Ωστόσο αυτό που κάνει τα ΕΣ να ξεχωρίζουν απο τους ηλεκτρονικούς υπολογιστές (PC), τα κινητά τηλέφωνα και οποιαδήποτε άλλη ηλεκτρονική συσκευή είναι ότι τα συστήματα αυτά χρησιμοποιούνται μόνο για κάποιο ειδικό σκοπό,δηλαδή μπορούν να κάνουν μόνο μία συγκεκριμένη λειτουργία στο συνολικό υπολογιστικό σύστημα. Εκτός απο τα μηχανικά μέρη μπορούμε να τα ξεχωρίσουμε απο το λογισμικό, καθώς τα ΕΣ χρησημοποιούν λογισμικό ικανό να δημιουργήσει κώδικα για μια πλατφόρμα διαφορετική απο αυτήν στην οποία εκτελείται ο μετταγλωττιστής. Για παράδειγμα ένας μετταγλωττιστής που εκτελείται σε υπολογιστή με Windows 10 αλλά δημιουργεί κώδικα που εκτελείται σε κινητό Android είναι ένας διασταυρούμενος μετταγλωττιστής (cross-compiler). Επιπρόσθετα τα ΕΣ αποτελούνται απο υλικό υψηλών επιδόσεων, δηλαδή να είναι αξιόπιστα και ποιοτικά, σε συνδιασμό με εξειδικευμένο λογισμικό ένας σχεδιασμός που έχει πολύ μικρό χρόνο σχεδιασμόυ καθώς έχουν κατασκευαστεί για μία μόνο λειτουργία.

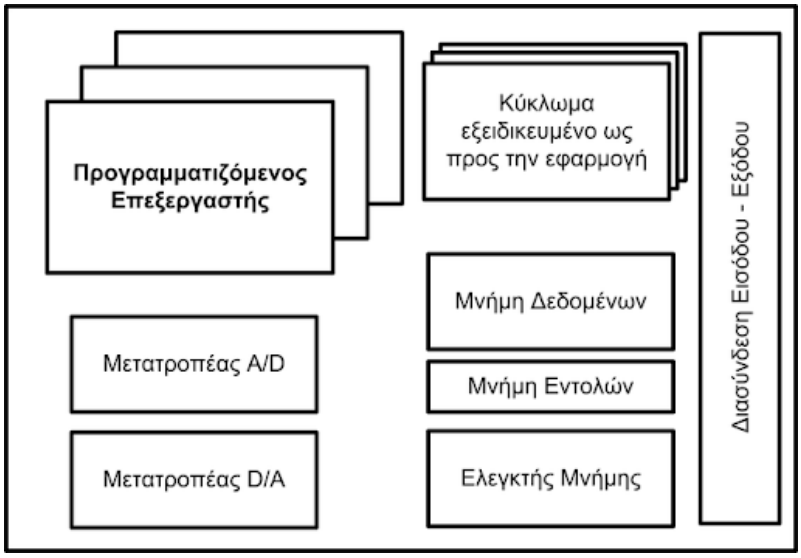
1.2 Τα ενσωματωμένα συστήματα ως μέρος της καθημερινότητας

Τα τελευταία χρόνια, τα ενσωματωμένα συστήματα, εμφανίζονται στην αγορά με ραγδαίους ρυθμούς και τα συναντάμε καθημερινά σε αρκετές απλές καθώς και σύνθετες συσκευές χωρίς οι περισσότεροι να το γνωρίζουν. Γι' αυτό το λόγο, αρκετοί ερευνητές χαρακτηρίζουν τα ΕΣ ως επανάσταση των ηλεκτρονικών συσκευών. Συνήθως τα ΕΣ, σε αντίθεση με τους υπολογιστές γενικού σκοπού, χρησιμοποιούνται για συγκεκριμένες εφαρμογές και έχουν περιορισμένη χρήση ανάλογα με τις απαιτήσεις του συστήματος που θα αναφερθούμε σε αυτό στην ενότητα 1.4. Για να κατανοήσουμε καλύτερα τα συστήματα αυτά θα αναφερθούμε σε μερικά παραδείγματα τέτοιων συστημάτων.

- **Τηλεοράσεις και ΜΜΕ.** Οι τηλεοράσεις περιέχουν επεξεργαστές με σκοπό τον έλεγχο της εικόνας, την ρύθμιση καναλιών και την ενεργοποίηση-απενεργοποίηση κυκλωμάτων της τηλεόρασης, δηλαδή για παράδειγμα η λειτουργία της αυτόματης απενεργοποίησης της τηλεόρασης γνωστή και ως Timer είναι ένα κύκλωμα που ελέγχεται απο τον επεξεργαστή της τηλεόρασης.
- **Κινητά τηλέφωνα.** Τα κινητά τηλέφωνα θεωρούνται σήμερα πολύπλοκες ηλεκτρονικές συσκευές και έχουν ενσωματωμένους επεξεργαστές για να διεκπεραιώνουν τις διαδικασίες αποκωδικοποίηση /κωδικοποίησης της φωνής, αναπαραγωγή βίντεο και μουσικής και γενικά να ελέγχουν τις διάφορες λειτουργίες για τις οποίες προορίζονται.
- **Οικιακές συσκευές.** Οι εκσυγχρονισμενες αυτές συσκεύες όπως φούρνοι, πλυντήρια ,ψυγεία κ.α διαθέτουν ενσωματωμένους επεξεργαστές για να ελέγχουν καλύτερα οι χρήστες τις λειτουργίες αυτων των συσκευών βελτιώνοντας με αυτό τον τρόπο την καθημερινότητά του.
- **Οχήματα.** Με την πάροδο των χρόνων οχήματα έχουν γνωρίσει μεγάλη ανάπτυξη στον τομέα των ΕΣ καθώς πλέον ένα όχημα είναι εξ ολοκλήρου ηλεκτροκίνητο και ελέγχονται όλες οι λειτουργίες του απο επεξεργαστές.

1.3 Βασική δομή ενσωματωμένων συστημάτων

Όπως αναφέραμε στην ενότητα 1.1 τα ΕΣ διαφέρουν απο τις άλλες ηλεκτρονικές συσκευές ως προς το υλικό (Hardware) καθώς έχουν μια πιο απλή δομή για τις λειτουργίες που εκτελούν.
Ένα ΕΣ αποτελείται απο πολλά τμήματα ,στο Σχήμα 1.1 παρουσιάζονται τα στοιχεία που μαζί συνθέτουν ένα τυπικό ΕΣ.



Σχήμα 1.1 Βασική δομή ΕΣ

Παρακάτω θα αναλύσουμε αναλυτικά καθένα απο τα μέρη του:

- **Διασύνδεση Εισόδου-Εξόδου**: Οι μονάδες αυτές αποτελούν το τμήμα του ΕΣ στο οποίο επιτρέπουν την μετατροπή όλων των μορφών πληροφορίας σε δυαδική και το αντίστροφο. Συγκεκριμένα στις μονάδες Εισόδου εισάγονται προς επεξεργασία δεδομένα σε οποιαδήποτε μορφή, ενώ στις μονάδες εξόδου εξάγονται τα δεδομένα έπειτα απο την επεξεργασία τους.

- **Προγραμματιζόμενος Επεξεργαστής ή Κεντρική Μονάδα Επεξεργασίας (CPU):** Η κεντρική μονάδα επεξεργασίας (CPU) η οποία είτε είναι ενσωματωμένη στο μικροελεγκτή ή αποτελεί ξεχωριτό ολοκληρωμένο κύκλωμα ,είναι η καρδιά του συστήματος και υλοποιείται με ένα ολοκληρωμένο κύκλωμα. Μια CPU αποτελέιται απο :
  a) **Εσωτερικούς καταχωρητές**: οι οποίοι είναι υπεύθυνοι για την αποθήκευση δεδομένων και ενδιάμεσων αποτελεσμάτων. Το μέγεθος των εσωτερικών καταχωρητών μας δίνει το εύρος του λεγόμενου εσωτερικού διαδρόμου δεδομένων (internal data bus) και πρόκειται για κυκλώματα που διακινούν και αποθηκεύουν πληροφορίες στην CPU,κάτι που γίνεται με μεγάλη ταχύτητα. Ένας απο τους βασικότερους καταχωρητές είναι ο συσσωρευτής ,που χαρακτηρίζεται και ως καταχωρητής εργασίας και είναι υπεύθυνος για την αποθήκευση αριθμητικών και λογικών πράξεων.
  b) **Την αριθμητική και λογική μονάδα (ALU)**: η οποία εκτελεί με ταχύτητα κάθε αριθμητική και λογική πράξη και στέλνει το αποτέλεσμα στον συσσωρευτή για αποθήκευση.

c) **Την μονάδα ελέγχου ή ελεγκτής μνήμης:** πρόκειται για την μονάδα που παράγει όλα τα σήματα χρονισμού και ελέγχου,τα οποία συντονίζουν τις διάφορες λειτουργίες όλων των κυκλωμάτων του συστήματος. Η μονάδα ελέγχου παράγει σήματα που καθοδηγούν τη συνεργασία των διάφορων μονάδων προκειμένου να εκτελεστούν οι εντολές του προγράμματος.Αυτή αποφασίζει για την προτεραιότητα στην εξυπηρέτηση των διαφόρων μονάδων και κατευθύνει τη διακίνηση των δεδομένων στους διαδρόμους του ΕΣ.

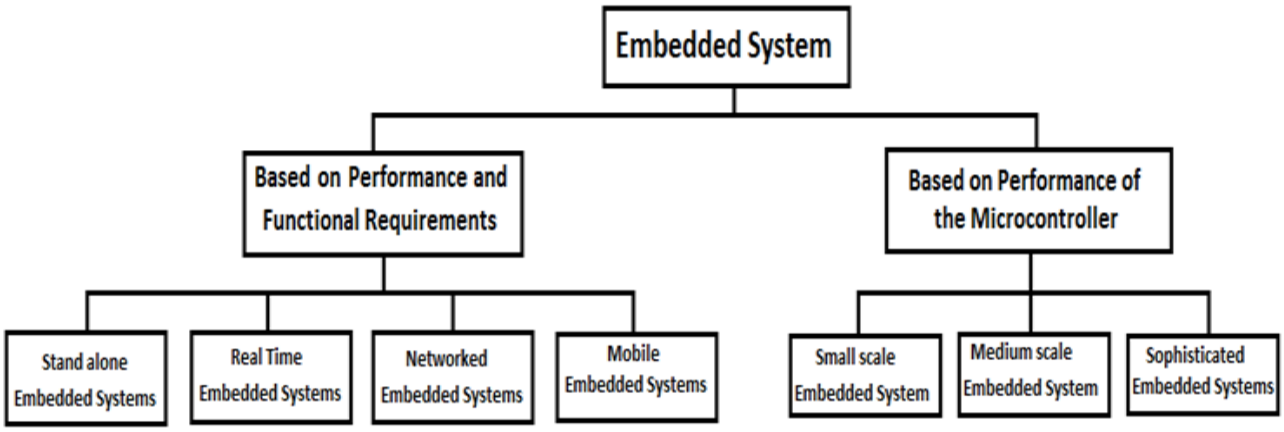
d) **Την Μνήμη:** που μπορεί να αποθηκεύσει ένα σύνολο απο δυαδικά ψηφία(bits) τα οποία θα έχουν λογικές τιμές ‘0’ και ‘1’. Τα ΕΣ αντί για μνήμη ROM έχουν μνήμες EEPROM και πιο συχνά τύπου FLASH που επιτρέπουν τον επαναπρογραμματισμό της CPU.

- **Μνήμη Εντολών:** Περιέχει το λογισμικό του συστήματος.

- **Μνήμη Δεδομένων:** Αποθηκεύονται ή διαβάζονται δεδομένα ,απο κυκλώματα εξειδικευμένα ως προς την εφαρμογή που εκτελείται.

- **Μετατροπέας A/D και D/A:** Είναι μετατροπείς αναλογικού σήματος σε ψηφιακό(A/D) και ψηφιακού σήματος σε αναλογικό(D/A) αντίστοιχα. Οι μονάδες αυτές διαμέσου του λογισμικού επιτρέπουν στον χρήστη να μετατρέψει ένα αναλογικό σήμα σε ψηφιακό ή και το αντίστροφο,για παράδειγμα εάν έχουμε μία τάση των 5V και 0V και θέλουμε να μεταφράσουμε την τάση αυτή στο δυαδικό για να την καταλάβει ο μικροεπεξεργαστής τότε θα αντιστοιχίσουμε την τάση των 5V σε bit ‘1’ και την τάση των 0V σε bit ‘0’. Με αυτό τον τρόπο ο μικροεπεξεργαστής καταλαβαίνει ότι όταν συναντήσει στον κώδικα την τιμή ‘1’ θα δώσει σε κάποια έξοδό του high δηλαδή 5V ενώ όταν συνταντήσει την τιμή ‘0’ θα δώσει στην έξοδό του low δηλαδή 0V. Η παραπάνω διαδικασία ισχύει και για μετατροπή απο ψηφιακό σήμα σε αναλογικό.

1.4 Κατηγοριοποίηση ενσωματωμένων συστημάτων



Σχήμα 1.2 Κατηγορίες ΕΣ

Μπορούμε να ταξινομίσουμε τα ενσωματωμένα συστήματα σε 2 μεγάλες κατηγορίες με βάση την απόδοσή τους. Αρχικά με βάση την απόδοση των μικροεπεξεργαστών τα ΕΣ χωρίζονται σε 3 κατηγορίες:

- **Μικρής κλίμακας ΕΣ:** (small scale systems) Σε αυτή την κατηγορία τα συστήματα σχεδιάζονται συνήθως με έναν ή πολλούς 8 ή 16 bit μικροεπεξεργαστές ,που σημαίνει ότι μπορεί το ΕΣ να εκτελεί απλές εφαρμογές, έχουν λίγο υλικό και μικρό λογισμικό πράγμα που επιτρέπει στο σύστημα αυτό να τροφοδοτείται και μέσω μπαταρίας. Κατά την ανάπτυξη λογισμικού χρησιμοποιείται μετταγλωττιστής ανάλογα με το είδος του μικροεπεξεργαστή, συνήθως ο προγραμματισμός των μικροελεγκτών στην σημερινή εποχή γίνεται σε γλώσσα C ή C++ και ο κώδικας φορτώνεται στην μνήμη του μικροελεγκτή.

- **Μεσαίας κλίμακας ΕΣ:** (medium scale systems) Αυτά τα συστήματα σχεδιάζονται επίσης με έναν ή πολλους μικροεπεξεργαστές των 16 ή 32 bit, με σύστημα DSP (Digital Signal Processor) ή και με RISC (Reduced Instruction Set Computers) που είναι ένα μικρό σύνολο εντολών οι οποίες πραγματοποιούν ένα ελάχιστο πλήθος λειτουργιών , δηλαδή σε αυτή την κατηγορία ΕΣ έχουμε να κάνουμε με συστήματα λίγο ποιο περίπλοκα απο τα ΕΣ μικρής κλίμακας. Χρησιμοποιούν συγκεκριμένα και περίπλακα εργαλεία λογισμικού όπως προσομοιωτές ,εντοπιστές σφαλμάτων και ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) ,δηλαδή πρόγραμμα που βοηθάει στην ανάπτυξη κώδικα με την βοήθεια διαφόρων εργαλείων.

- **Εξελιγμένα ΕΣ:** (sophisticated systems) Σε αυτή την κατηγορία έχουμε να κάνουμε με συστήματα μεγάλης πολιπλοκότητας σε επίπεδο υλικού καθώς και λογισμικού. Χρησιμοποιούνται σε σύγχρονες τεχνολογίες όπως κινητά τηλέφωνα ,υπολογιστές και άλλες ηλεκτρονικές συσκευές που χρειάζονται ΕΣ για να εκτελούν πολύπλοκες λειτουργίες. Επιπλέον ορισμένες λειτουργίες σε επίπεδο λογισμικού όπως αλγόριθμοι, κρυπτογράφηση και αποκρυπτογράφηση ,αντίστροφοι αλγόριθμοι μετασχηματισμού και πρωτόκολλα επικοινωνίας όπως TCP/IP χρησημοποιούνται με σκοπό την αύξηση ταχύτητας των ΕΣ.

Με βάση την απόδοση και τις απαιτήσεις λειτουργικότητάς τους διακρίνουμε τις εξής κατηγορίες ΕΣ:

- **Αυτόνομα ΕΣ:** (stand alone systems) Τα συγκεκριμένα συστήματα λειτουργούν όπως αναφέρει η ονομασία απο μόνα τους, δηλαδή δέχονται στις μονάδες εισόδου κάποιο σήμα αναλογικό ή ψηφιακό, το επεξεργάζονται και αποδίδουν το αποτέλεσμα σε κάποια συνδεδεμένη συσκευή μέσω τις μονάδες εξόδου. Ένα μη-αυτόνομο ΕΣ θα έπρεπε να συνδεθεί με άλλα ενσωματωμένα ή μη συστήματα με σκοπό να διαχειριστούν κάποια απο τα δεδομένα ή άλλη λειτουργία.

- **ΕΣ Πραγματικού Χρόνου:** (real time systems) Ως ΕΣ πραγματικού χρόνου ορίζονται αυτά όπου το κάθε αποτέλεσμα πρέπει να επιστραφεί μέσα σε κάποιο συγκεκριμένο χρονικό πλαίσιο και ο χρήστης μέσω ηλεκτρονικού υπολογιστή μπορεί να ελέγχει ή και να ανταποκρίνεται σε διάφορες απαιτήσεις του συστήματος.

- **Δικτυωμέα ΕΣ:** (networked systems) Σε αυτή την κατηγορία έχουμε να κάνουμε με συστήματα τα οποία συνδέονται σε κάποιο δίκτυο, τοπικό (LAN), ευρείας περιοχής (WAN) ή στο διαδίκτυο. Αυτή η τεχνολογία επιτρέπει καλύτερες ενοποιημένες επικοινωνίες, ολοκληρωμένο τοπικό και παγκόσμιο έλεγχο

- **Κινητά ΕΣ:** (mobile systems) Τέλος τα κινητά ΕΣ χρησιμοποιούνται σε φορητές ηλεκτρονικές συσκευές με μοναδική αδυναμία την περιορισμένη παροχή τροφοδοσίας, καθώς δουλεύουν με μπαταρίες, και την μικρή σε μέγεθος μνήμη της τάξεως των GB.

1.5 Πλεονεκτήματα και μειωνεκτήματα των ΕΣ

Σε αυτή την ενότητα θα αναφερθούν ορισμένες ιδιαιτερότητες των ενσωματωμένων συστημάτων που τα κάνουν να ξεχωρίζουν σημαντικά απο απο τα υπόλοιπα ηλεκτρονικά συστήματα. Αρχικά ένα ΕΣ δεν μπορεί να χρησιμοποιηθεί για λειτουργίες διαφορετικές απο αυτές που σχεδιάστηκε να κάνει και αυτό οφείλεται στο γεγονός ότι αυτά τα συστήματα είναι δύσκολο να αναβαθμιστούν ως προς το υλικό καθώς και το λογισμικό.

Για αυτό το λόγο σε περίπτωση παρουσίασης προβλήματος το σύστημα θα πρέπει να τεθεί σε λειτουργία επανεκκίνησης και όλη αυτή η διαδικασία κάνει την αντιμτώπιση προβλημάτων (troubleshooting) δυσκολότερη. Επιπλέον έχουν μικρή αντοχή τροφοδοσίας που σημαίνει ότι σε περίπτωση υπερτάσεων υπάρχει κίνδυνος καταστροφής και γι'αυτό το λόγο απαιτείται η σχεδίαση κυκλώματος σταθεροποιειτή τάσης (voltage regulator), γεγονός που κάνει τα ΕΣ να έχουν υψηλότερες απαιτήσεις σχεδίασης και ανάπτυξης.

Ωστόσο καθώς ένα ΕΣ εκτελεί συνήθως μία απλή εφαρμογή που δεν αλλάζει, οι απαιτήσεις για το σύστημά του είναι λιγότερο επαχθείς, δηλαδή έχει μικρές πιθανότητες να εμφανίσει σφάλματα καθώς έχει λίγες διασυνδέσεις, είναι πιο σταθερό και αξιόπιστο και μπορεί να αντέξει σε μεταβολές του περιβάλλοντος, όπως υψηλές θερμοκρασίες, θόρυβο κ.α. Επιπρόσθετα έχουν μικρή κατανάλωση ισχύος, είναι πιο αποδοτικά και γρήγορα σε απλές εφαρμογές, αφού δεν υπάρχει μεγάλη αποθήκευση δεδομένων και έτσι γίνεται βελτιστοποίηση πόρων μνήμης και μικροεπεξεργαστή. Οι παραπάνω αναφορές στις ιδιαιτερότητες των ΕΣ οφείλονται κυρίως την κατάλληλη επιλογή μικροεπεξεργαστών, για τους οποίους θα αναφερθούμε αναλυτικά στο κεφάλαιο 2.

1.6 Επίλογος κεφαλαίου 1

Στο κεφάλαιο μελετήθηκαν τα Ενσωματομένα Συστήματα (ΕΣ) και δώθηκε ένας σύντομος ορισμός για το τί είναι ένα ΕΣ.Αναφέραμε λοιπόν ότι ΕΣ είναι κάθε συσκευή η οποία περιλαμβάνει έναν προγραμματιζόμενο επεξεργαστή με ειδική λειτουργία μέσα σε μεγαλύτερο μηχανικό ή ηλεκτρικό σύστημα καθώς επίσης και που τα συναντάμε στην καθημερινότητά μας. Τα ΕΣ χωρίζονται σε κατηγορίες ανάλογα με την απόδοση του μικροελεγκτή, όπου ξεχωρίζουμε τρείς κατηγορίες, τα ΕΣ **μικρής κλίμακας**, **μεσαίας κλίμακας** και **εξελιγμένα ΕΣ** και ανάλογα με την απόδοση των λειτουργιών διακρίνουμε τα **Αυτόνομα**, **Δικτυωμένα** ,**Κινητά ΕΣ** καθώς επίσης και ΕΣ **Πραγματικού χρόνου**. Τέλος έγινε αναφορά για την βασική δομή των ΕΣ καθώς αποτελούνται απο τις μονάδες Εισόδου/Εξόδου , την κεντρική μονάδα επεξεργασίας (CPU), την μνήμη και τους μετατροπείς A/D και D/A. Όλα τα παραπάνω συγκροτούν ένα ΕΣ που προορίζεται για μία συγκεκριμένη λειτουργία. Χαρακτηριστικό παράδειγμα είναι η μητρική πλακέτα των υπολογιστών που υπάρχουν σε όλους τους ηλεκτρονικούς υπολογιστές και κατα συνέπεια τους χρησιμοποιούμε στην καθημερινότητά μας. Περιέχει όσα αναφέραμε αλλά και πολλά περισσότερα καθώς πρόκειται για ένα σύνθετο και εξελιγμένο ΕΣ.

Εικόνα 1.3 Μητρική πλακέτα ηλεκτρονικού υπολογιστή

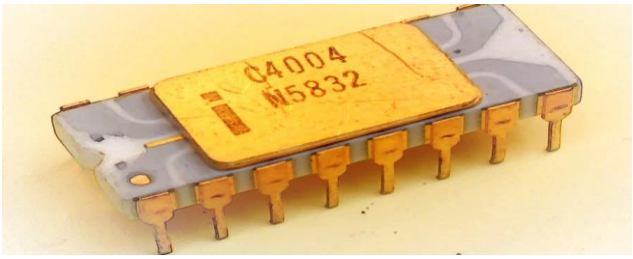
Κεφάλαιο 2: Μικροεπεξεργαστές

2.1 Εισαγωγή στους μικροεπεξεργαστές

Ο Μικροελεγκτής είναι ένα είδος επεξεργαστή, αποτελεί ουσιαστικά μια παραλλαγή του μικροεπεξεργαστή .Η λειτουργία του μπορεί να πραγματοποιηθεί με ελάχιστα εξωτερικά εξαρτήματα, λόγω των πολλών υποσυστημάτων που διαθέτει. Οι περισσότεροι μικροελεγκτές βασίζονται στην αρχιτεκτονική Von-Neuman, η οποία σαφώς καθόρισε τα τέσσερα βασικά συστατικά που απαιτούνται για ένα ψηφιακό σύστημα. Αυτά περιλαμβάνουν όπως αναφέραμε στην ενότητα 1.3 έναν επεξεργαστικό πυρήνα (CPU), τη μνήμη για το πρόγραμμα και τα δεδομένα (RAM), χώρο μόνιμης αποθήκευσης (Flash σε έναν μικροελεγκτή), καθώς επίσης και τις θύρες I/O για επικοινωνία με εξωτερικές περιφερειακές μονάδες. Τέλος, όπως ένας μικροϋπολογιστής έχει την δυνατότητα να εκτελεί διάφορα προγράμματα και να διαθέτει περιφερειακές συσκευές, επεξεργαστή και μνήμη, έτσι και ο μικροελεγκτής διαθέτει τα χαρακτηριστικά που προαναφέρθηκαν και μάλιστα, σε ένα μόνο chip. Ενώ η αποθήκευση των προγραμμάτων που εκτελούν οι μικροελεγκτές γίνεται πάντα στη μνήμη προγράμματος.

2.2 Ο πρώτος μικροεπεξεργαστής των 4bit (1ης γενιάς)

Ο πρώτος ολοκληρωμένος επεξεργαστής κυκλοφόρησε το 1971 απο την εταιρία Intel με την ονομασία C4004 που σχεδιάστηκε και κατασκευάστηκε απο τον Ted Hoff και τον Stan Mazor. Ο μικροεπεξεργαστής αυτός ήταν ένας επεξεργαστής των 4-bit που αποτελούταν απο 2.300 τρανζίστορ με συχνότητα ρολογιού στα 740KHz, μπορούσε να εκτελέσει 60.000 πράξεις το δευτερόλεπτο και να δει 640 bytes μνήμης. Πρέπει να σημειωθεί ότι κάθε μικροεπεξεργαστής έχει μερικές διαφορετικές εκδόσεις, οι οποίες διαφέρουν ως προς την ταχύτητα με την οποία αυτός εργάζεται. Μέτρο της ταχύτητας είναι η συχνότητα λειτουργίας, η οποία μετρείται σε Hz(Hertz). Όσο μεγαλύτερη είναι η συχνότητα, τόσο γρηγορότερος είναι ο μικροεπεξεργαστής, σε σχέση με μικροεπεξεργαστές της ίδιας κατηγορίας.



Εικόνα 2.1 Intel 4004 microchip(4-bit)

2.2.1 2ης γενιάς μικροεπεξεργαστές

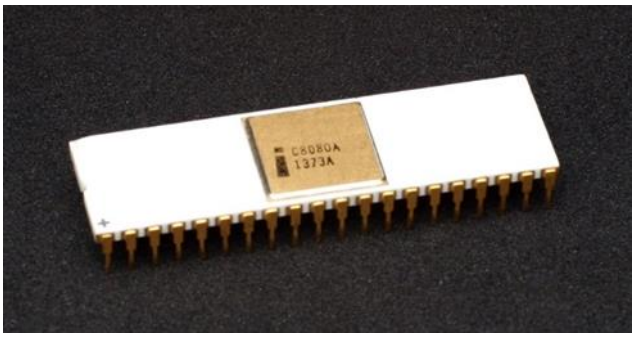
Τους 4 bits επεξεργαστές ακολούθησαν οι 8 bits, με τον σημαντικότερο από αυτούς τον Intel 8008 ο οποίος ήταν ο πρώτος εμπορικός 8 bit μικροεπεξεργαστής. Η ανάπτυξή του ξεκίνησε το 1971 με επικεφαλή τον Federico Faggin και ολοκληρώθηκε το 1972, αφού ξανασχεδιάστηκε καθώς η αρχική σχεδίαση είχε διαρροές ηλεκτρικού φορτίου από τις συσκευές μνήμης. Η συχνότητα ρολογιού ήταν στα 200 KHz, ενώ το chip χρησιμοποιούσε 3500 κρυσταλλολυχνίες και μπορούσε να δει 16 Kbytes μνήμης. Χρησιμοποιήθηκε στον Micral το 1973, ο οποίος ήταν ο πιο σύγχρονος υπολογιστής που τροφοδοτήθηκε από έναν μικροεπεξεργαστή 8008 ενώ ένα χρόνο αργότερα η Scelbi χρησιμοποιεί τον 8008 στον υπολογιστη 8H. Με τον 8008 ξέσπασε μεγάλο ενδιαφέρον για την ανάπτυξη μικροεπεξεργαστών το οποιο είχε αποτέλεσμα να αυξηθούν οι απαιτήσεις στην ταχύτητα, επικοινωνία με το περιβάλλον, και πιο πολλές εντολές και εισόδους δεδομένων.

Το 1974 ακολούθησε η κυκλοφορία του Intel 8080, ο οποίος έτρεχε στα 2 MHz, μπορούσε να δει 64 Kbytes μνήμης και περιείχε 6.000 τρανζίστορ. Σημαντικά γεγονότα που ακολούθησαν ήταν η ανάπτυξη του λειτουργικού συστήματος CP/M για τον Intel 8080 από τον Gary Kildall της εταιρίας Microcomputer Applications Associates, η σχεδίαση του υπολογιστή Altair 8800 ο οποίος χρησιμοποιούσε Intel 8080 με 256 bytes RAM και η ανάπτυξη της γλώσσας Microsoft Basic, από τον Bill Gates και τον Paul Allen, για τους μικροεπεξεργαστές της Intel. Τα γεγονότα αυτά θεωρούνται από πολλούς ότι οδήγησαν στη ραγδαία ανάπτυξη των προσωπικών υπολογιστών. Σύντομα μετά τη κυκλοφορία του 8080, η Motorola κυκλοφόρησε τον 6800, 8 bit επεξεργαστη. Είχε 4000 τρανζίστορ, 78 εντολές, σήμα χρονισμού στα 1 ή 2 MHz με 16 bit πλάτος διαύλου διευθύνσεων και ήταν ένας από τους πρώτους μικροεπεξεργαστές με καταχωρητή δείκτη (index register). Ο 6800 ήταν καλοσχεδιασμένος, παρόλα τα προβλήματα με την παραγωγή και γι' αυτό και χρησιμοποιήθηκε σε πολλά συστήματα. Έτσι η εταιρία MITS ξεκίνησε την σχεδίαση ενός Altair βασισμένου στον Motorola 6800 και λίγο αργότερα παρουσιάστηκε ο υπολογιστής Sphere I, με επεξεργαστή τον Motorola 6800, 4Kbytes RAM, πρόγραμμα ROM monitor, πληκτρολόγιο και διασύνδεση βίντεο. Προς τα τέλη του 1970 η Tektronix σχεδίασε τον 4051 βασισμένο στον επεξεργαστή της Motorola με 32 Kbytes RAM πληκτρολόγιο και high speed data cartridge. Ο 4051 προορίζονταν για χρήστες που προγραμμάτιζαν σε BASIC. Τον 6800 ακολούθησε ο 6809, ένα από τα πιο ισχυρά σχέδια μικροεπεξεργαστή και επίσης ένα από τον πιο σύνθετα σχέδια λογικής που έγιναν ποτέ στην παραγωγή οποιουδήποτε μικροεπεξεργαστή. Αρχικά χρησιμοποιούσε σήμα χρονισμού στο 1 MHz στο μοντέλο 68A09, ύστερα στο 1.5 MHz στο μοντέλο 68A09 και στα 2 MHz στο 68B09. Ο 6809 χρησιμοποιήθηκε, μεταξύ άλλων συστημάτων και στη κονσόλα παιχνιδιών Vectrex.

Το 1975, η ανταγωνίστρια εταιρεία MOS Technology κυκλοφορεί τον επεξεργαστή 6502, μια παραλλαγή του 6800, ο οποίος χρησιμοποιούσε δυο 8 bit καταχωρητές.Είχε 5000 τρανζίστορ, 56 εντολές, σήμα χρονισμού αρχικά 20 KHz μέχρι 4 MHz με 16 bit πλάτος διαύλου διευθύνσεων. Χρησιμοποιήθηκε σε πολλά συστήματα, μερικά από αυτά οι κονσόλες ηλεκτρονικών παιχνιδιών Atari

2600, Nintendo Entertainment System/NES και οι πρωτοι οικιακοί υπολογιστές Commodore 64 και Apple II. Το 1976 φτιάχνεται ο Z80 από την εταιρεία Zilog. Επίσης 8bit μικροεπεξεργαστής, βασισμένος στον 8080, του οποίου η γλώσσα μηχανής είναι υπερσύνολο αυτής του Intel 8080. Είχε σήμα χρονισμού στα 3.5 MHz με 16 bit πλάτος διαύλου διευθύνσεων, ενώ μπορούσε να δει 64 Kbytes μνήμης. Το χαμηλό κόστος, η μικρή συσκευασία καθώς και άλλα χαρακτηριστικά τον έκαναν πολύ δημοφιλή στη δεκαετία του 80.

Τέλος, το 1982, ήδη στην αρχή της εποχής των 16 bit επεξεργαστών κυκλοφορεί ο RCA(CDP)1802 της RCA και χρησιμοποιήθηκε για την κατασκευή των δορυφόρων Voyager, Viking και του διαστημοπλοίου Γαλιλαίος. Είχε πολύ μικρή κατανάλωση και ήταν ανθεκτικός στην κοσμική ακτινοβολία και τις ηλεκτροστατικές αποφορτίσεις.



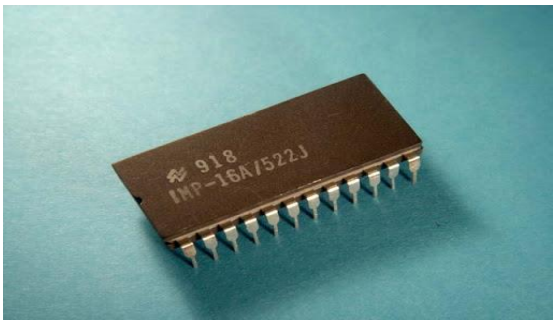
Εικόνα 2.2 Intel 8080 microchip (8-bit)

2.2.2 3ης γενιάς μικροεπεξεργαστές

Ο πρώτος 16bit μικροεπεξεργαστής multi-chip ήταν ο IMP 16 της National, που εισήχθη στις αρχές του 1973. Το 1974 εισήχθη η 8 bit έκδοσή του IMP-8.Δύο χρόνια αργότερα, η National εισήγαγε το πρώτο 16-bit single-chip μικροεπεξεργαστή, τον PACE, ο οποίος ακολουθήθηκε αργότερα από μια NMOS έκδοση, το INS8900. Το 1976, εμφανίζεται ο TMS 9900 της Texas Instruments, ένας από τους πρώτους καθαρά 16bit μικροεπεξεργαστές. Ο TMS 9900 δεν είχε καθόλου εσωτερικούς καταχωρητές, εκτός από έναν που όριζε την θέση των καταχωρητών του στην RAM, όπου αποθηκεύονται. Η σχεδίαση επέτρεπε την ταχύτατη αλλαγή context, αφού για να αλλάξουν όλοι οι καταχωρητές και να κληθεί μια συνάρτηση, πρέπει να αλλάξει ο μοναδικός εσωτερικός καταχωρητής. Η συγκεκριμένη σχεδίαση είχε νόημα για την εποχή της, διότι η εσωτερική μνήμη ήταν πιο αργή από την εξωτερική.

Η Intel επανέρχεται στο προσκήνιο αναβαθμίζοντας το σχέδιο του 8080 στον 16-bit Intel 8086, το πρώτο μέλος της x86 οικογένειας που χρησιμοποιούν οι περισσότεροι σύγχρονοι υπολογιστές. Ο 8086 είχε 10 φορές καλύτερη απόδοση από τον 8080. Η Intel εισήγαγε τον 8086 ως οικονομικώς αποδοτικό τρόπο μεταφοράς του λογισμικού από τον 8080, και πέτυχε κερδίζοντας την εμπιστοσύνη πολλών επιχειρήσεων με εκείνη την προϋπόθεση. Ο 8086 είχε 29.000 τρανζίστορ, ταχύτητα λειτουργίας στα 10 MHz, ενώ χρησιμοποιούσε καταχωρητές των 16 bit και δίαυλο δεδομένων των 16 bit. Επιπλέον, μπορούσε να δει 1 Mbyte μνήμης. Τον Ιούνιο του 1979, αποκαλύφθηκε ο μικροεπεξεργαστής 8088, που ήταν μια παραλλαγή του 8086. Πρόκειται για έναν 16bit επεξεργαστή εσωτερικά, του οποίου ο εξωτερικός δίαυλος δεδομένων ήταν των 8 bits. Η σχεδίαση αυτή είχε σκοπό τη χρήση των υπαρχόντων 8bit controller chips για συσκευές. Ο 8088 περιείχε 29000 τρανζίστορς και μπορούσε να δει 1 Mbyte μνήμης. Μετά τον 8088 η Intel απελευθέρωσε τους 16bit μικροεπεξεργαστές 80186 και 80286, παγιώνοντας την κυριαρχία της στην αγορά προσωπικών υπολογιστών. Ο 80286 παρουσιάστηκε το 1982, και είχε συχνότητα λειτουργίας αρχικά στα 6 MHz και έπειτα στα 12 MHz. Ο ιστορικός αυτός μικροεπεξεργαστής ανήκε στην οικογένεια x86 και περιελάμβανε δίαυλο δεδομένων 16 bit, δίαυλο διευθύνσεων 24 bit. Επιπλέον, μπορούσε να δει μέχρι 16 MBytes μνήμης και περιείχε 130000

τρανζίστορ. Ο 80286 αποτελεί τον πρώτο μικροεπεξεργαστής που είχε τη δυνατότητα να λειτουργεί στην κατάσταση Protected Mode (προστατευμένη κατάσταση λειτουργίας).



Εικόνα 2.3 IMP-16 microchip(16-bit)

2.2.3 4ης γενιάς μικροεπεξεργαστές

Στις αρχές της δεκαετίας του 80, η Motorola κυκλοφόρησε τους πρώτους 32-bit μικροεπεξεργαστές.Το 1979 κυκλοφόρησε τον διασημότερο 32-bit μικροεπεξεργαστη MC69000 καθώς ήταν το πρώτο μέλος της οικογένειας m68k που είχε 32 bit καταχωρητές αλλά χρησιμοποιούσε 16 bit διαδρομές δεδομένων, καθώς και έναν 16 bit εξωτερικό δίαυλο δεδομένων. Η Motorola το περιέγραψε γενικά ως 16bit επεξεργαστή, αν και είχε 32bit αρχιτεκτονική. Ο συνδυασμός της υψηλής ταχύτητας, του μεγάλου χώρου αποθήκευσης (16 Mbyte) και του αρκετά χαμηλού κόστους τον έκανε τον δημοφιλέστερο μικροεπεξεργαστή της κατηγορίας του, με αποτέλεσμα να χρησιμοποιηθεί στους υπολογιστές Apple Lisa και η Macintosh.

Το 1982 εμφανίζεται ο Intel 80286 ο οποίος έχει εσωτερικά 134.000 τρανζίστορ και λειτουργεί σε συχνότητα 12,5 MHz πράγμα που σημαίνει οτι είναι πιο γρήγορος επεξεργαστής απο τους προκατόχους του. Επιπλέον περιλάμβανε δίαυλο δεδομένων 16-bit, δίαυλο διευθύνσεων 24-bit και μπορούσε να δει μέχρι 16 Mbytes μνήμης.Είχε δυνατότητα να λειτουργεί σε προστατευόμενη κατάσταση (protected mode) όπου ο μικροεπεξεργαστής είχε την ικανότητα να χωρίζει την μνήμη σε τμήματα όπου ένα τμήμα μεγέθους 640 Kbytes, το όριζε για την αποθήκευση προγραμμάτων και λειτουργικού συστήματος για να διασφαλίσει ότι τα δεδομένα δεν αντικατασταθούν ακούσια.

Ένα άλλο ενδιαφέρον σχέδιο ήταν ο Zilog Z8000, ο οποίος εισήχθη πολύ αργά στην αγορά και εξαφανίστηκε από αυτή πολύ γρήγορα. Από το 1985 έως το 2003, η32-bit x86 αρχιτεκτονική έγινε η κυρίαρχη αρχιτεκτονική στην αγορά dektops, laptop, και server και οι επεξεργαστές έγιναν γρηγορότεροι και πιο ισχυροί. Η σειρά Pentium της Intel είναι πιθανότατα η πιο αναγνωρίσιμη σειρά 32 bit επεξεργαστών. Πριν τη σειρά Pentium προηγήθηκαν αρκετές σειρές επεξεργαστων. Ξεκινώντας τον Οκτώβριο του 1985, η Intel παρουσιάζει τον απόγονο του 80286 τον μικροεπεξεργαστή 80386. Είχε συχνότητα λειτουργίας αρχικά στα 16 MHz. και χρησιμοποιούσε καθαρούς 32bit καταχωρητές και 32bit διαύλους δεδομένων και διευθύνσεων. Περιείχε 275000 τρανζίστορ, υποστήριζε πολυδιεργασία (multitasking) εικονικής μνήμης με δυνατότητα σελιδοποίησης (paging). Το 1989, εμφανίζεται ο μικροεπεξεργαστής Intel 80486, ο οποίος είχε 1200000 τρανζίστορ και συχνότητα λειτουργίας 50 MHz.

Τη δεκαετία του 1990 έκαναν την εμφάνισή τους οι μικροεπεξεργαστές Intel Pentium, οι οποίοι αποτελούσαν τη συνέχεια του 80486 ενώ είχαν υπερβαθμωτή (superscalar) αρχιτεκτονική και 32bit δίαυλο δεδομένων. Το 1993 εμφανίζεται ο Intel Pentium της οικογένειας P5, ο οποίος περιείχε 3100000 τρανζίστορ και λειτουργούσε στα 60 και 66 MHz. Το 1995, η Intel παρουσιάζει τον Pentium Pro, τον πρώτο στην οικογένεια των P6. Είχε 5500000 τρανζίστορ και ανήκε στην έκτη γενιά των επεξεργαστών της οικογένειας x86. Δύο χρόνια αργότερα, η Intel εισάγει τον μικροεπεξεργαστή Pentium II, έναν

Pentium Pro με τεχνολογία MMX (MMX εντολές) για την υποστήριξη πολυμέσων. Ο Pentium II είχε 7500000 τρανζίστορ και η συχνότητα λειτουργίας του βρισκόταν στα 300 MHz.

Το 1999, ακολούθησε ο Pentium III με 9500000 τρανζίστορ και συχνότητα λειτουργίας στα 450 MHz. Την επόμενη χρονιά, εμφανίστηκε ο Pentium IV ο οποίος ήταν σχεδιασμένος σύμφωνα με την μικροαρχιτεκτονική NetBurst, η οποία συνεχίζει να αποτελεί την τεχνολογική καρδιά του Pentium IV και των παραλλαγών του κυκλοφόρησαν αργότερα. Ο Pentium D ήταν ο τελευταίος μικροεπεξεργαστής της σειράς Pentium, η οποία σταμάτησε να κυκλοφορεί το 2008. Λόγω του μεγάλου κόστους παραγωγής του Pentium II η Intel ανεπτυξε και κυκλοφόρησε μια νέα σειρά, τη σειρά Celeron. Ο Celeron είχε πιο προσιτή τιμή, αλλά δεν μπορούσε να λειτουργήσει σε πολύ υψηλές συχνότητες. Η ανταγωνίστρια της Intel, AMD μπήκε δυναμικά στο χώρο το 1997. Αρκετά αργότερα σε σχέση με την Intel αφου η Intel είχε ήδη πάρει τα δικαιώματα της αρχιτεκτονικής x86 οπότε όλες οι άλλες εταιρείες έπρεπε να απαντύξουν τα δικά τους σχέδια. Η αρχή έγινε με τους επεξεργαστές της σειράς K6, οι οποίοι ήταν εφάμιλλοι αυτών της Intel σε τιμή και επιδόσεις.

Το 1999, η AMD προώθησε την καινούργια οικογένεια μικροεπεξεργαστών, Athlon. Ο Athlon Classic, αποτελεί τον πρώτο επεξεργαστή της σειράς και μεγάλο ανταγωνιστή των Pentium, εισήγαγε την έβδομη γενιά επεξεργαστών της οικογένειας x86.

Εικόνα 2.4 Intel 80286(32-bit)

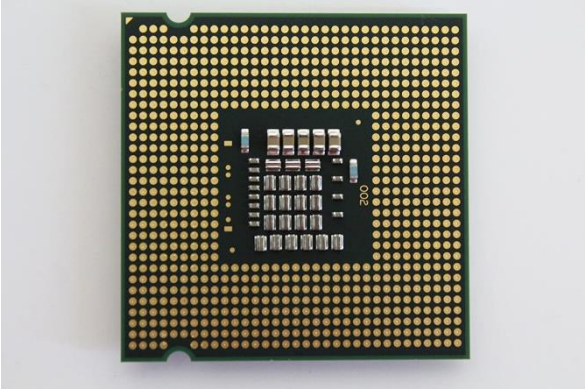
2.2.4 5ης γενιάς μικροεπεξεργαστές

Οι πρώτοι 64bit επεξεργαστές άρχισαν να εφαρμόζονται στους υπολογιστές γραφείου το 2003 και απευθύνονταν αποκλειστικά στην αγορά σταθμών εργασίας(workstation) και των διακομιστών. Τον Σεπτέμβριο του 2003 η AMD εισήγαγε τον Athlon 64 και ακολούθησε η Intel με τον Intel 64 όπου και οι 2 επεξεργαστές μπορούσαν να τρέξουν 32 και 64bit εντολές. Η κυκλοφορία των επεξεργαστών αυτών, είχε ως αποτέλεσμα να ενσωματωθούν σε ακαδημαϊκά και ερευνητικά προγράμματα καθώς εκεί χρειάζονταν μεγάλοι υπολογισμοί, γρήγορες προσβάσεις σε βάσεις δεδομένων αλλά και επίλυση σύνθετων προβλημάτων που μπορούσαν να προσφέρουν οι 64bit επεξεργαστές. Ο Athlon 64 αποτελεί διαφοροποίηση της αρχιτεκτονικής AMD64 για να μειωθεί το κόστος του. Το 2005, η AMD ανακοίνωσε τους διπλοπύρηνους επεξεργαστές Opteron για servers και workstations, καθώς και τους διπλοπύρηνους επεξεργαστές Athlon 64 για προσωπικούς υπολογιστές. Τον Φεβρουάριο του 2009, η AMD παρουσίασε τον τετραπύρηνο επεξεργαστή Phenom II με κωδικό όνομα Deneb.

Το 2006, η Intel αλλάζει τα δεδομένα στην απόδοσης και κατανάλωσης των υπολογιστών ανακοινώνοντας τις αρχιτεκτονικές Intel Core 2 Duo και Intel Core 2 Extreme. Οι αρχιτεκτονικές αυτές μπορούν να χρησιμοποιηθούν σε επεξεργαστές για σταθερούς, φορητούς και servers και παρέχουν

ισχυρή απόδοση με παράλληλη εξοικονόμηση ενέργειας. Η χρήση των 2 ή 4 πυρήνων διευκολύνει τη ομαλότερη και γρηγορότερη εκτέλεση περισσότερων διεργασιών.

Το 2008, κυκλοφόρησε ο Atom, ο μικρότερος σε μέγεθος επεξεργαστής της Intel που υλοποιήθηκε με τα μικρότερα τρανζίστορ του κόσμου. Δημιουργήθηκε ως μία εντελώς νέα σχεδίαση, ειδικά για φθηνές συσκευές, όπως πολύ μικρά notebooks και φορητές συσκευές με πρόσβαση στο Internet. Για τη σχεδίαση του Atom χρησιμοποιήθηκε η μικροαρχιτεκτονική Core, η ίδια δηλαδή τεχνολογία με την οποία η Intel κατασκευάζει τους γνωστούς Core 2 Duo επεξεργαστές για επιτραπέζιους και φορητούς υπολογιστές. Την ίδια χρονιά η Intel ανακοίνωσε τη νέα σειρά επεξεργαστών της με το όνομα Core i7.



Εικόνα 2.5 Intel Core 2 Duo(64-bit)

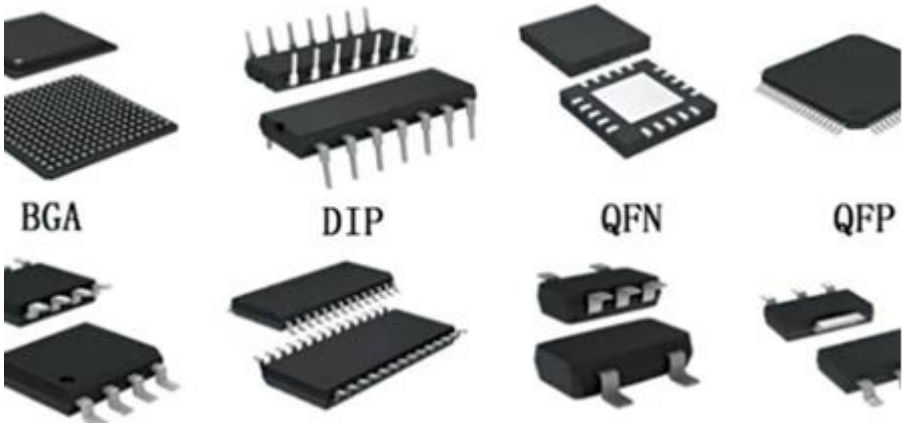
2.3 Οικογένεια μικροελεγκτών PIC της MicroChip

Ο PIC είναι ένας μικροελεγκτής περιφερειακής διασύνδεσης που αναπτύχθηκε το 1993 από τους μικροελεγκτές της General Instruments. Ελέγχεται από λογισμικό και προγραμματίζεται με τέτοιο τρόπο ώστε να εκτελεί διαφορετικές εργασίες και ελέγχει μια γραμμή παραγωγής. Οι μικροελεγκτές PIC χρησιμοποιούνται σε διάφορες νέες εφαρμογές όπως smartphone, αξεσουάρ ήχου και προηγμένες ιατρικές συσκευές. Τα πρώτα μέρη της οικογένειας ήταν διαθέσιμα το 1976 και μέχρι το 2013 η εταιρεία είχε αποστείλει περισσότερα από δώδεκα δισεκατομμύρια μεμονωμένα ανταλλακτικά, που χρησιμοποιήθηκαν σε μια μεγάλη ποικιλία ενσωματωμένων συστημάτων. Τα πρώτα μοντέλα PIC είχαν μνήμη μόνο για ανάγνωση (ROM) ή EPROM προγραμματιζόμενη με πεδίο για αποθήκευση προγραμμάτων, μερικά με πρόβλεψη για διαγραφή μνήμης. Όλα τα τρέχοντα μοντέλα χρησιμοποιούν μνήμη flash για αποθήκευση προγραμμάτων και τα νεότερα μοντέλα επιτρέπουν στο PIC να επαναπρογραμματίζει τον εαυτό του. Η μνήμη προγράμματος και η μνήμη δεδομένων διαχωρίζονται. Η μνήμη δεδομένων είναι 8-bit, 16-bit και, στα τελευταία μοντέλα, πλάτους 32-bit. Οι οδηγίες προγράμματος ποικίλλουν σε bit-count ανά οικογένεια PIC και μπορεί να έχουν μήκος 12, 14, 16 ή 24 bit. Το σετ εντολών ποικίλλει επίσης ανάλογα με το μοντέλο, με πιο ισχυρά chips προσθέτοντας οδηγίες για λειτουργίες επεξεργασίας ψηφιακού σήματος. Οι δυνατότητες υλικού των συσκευών PIC κυμαίνονται από 6-pin SMD, 8-pin DIP chip έως 144-pin SMD chip, με μονάδες I/O, ADC και DAC modules και θύρες επικοινωνίας όπως UART, I2C, CAN και ακόμη και USB. Ωστόσο σε αυτή την υποενότητα θα αναλύσουμε τον μικροελεγκτή PIC18F4550 με τον οποίο ξεκινήσαμε την υλοποίηση της πτυχιακής εργασίας με σκοπό την παρατήρηση των περιορισμών και των βελτιώσεων.

2.3.1 Ανάλυση PIC 18F4550

Ο μικροεπεξεργαστής αυτός ανήκει στην οικογένεια των 8-bit με αρχιτεκτονική Harvard,δηλαδή έχει ξεχωριστό χώρο μνήμης δεδομένων και προγραμμάτων.Είναι διαθέσιμος σε 40 pin DIP (Dual In Line), 44-pin QFP(Quad Flat Package) και QFN πακέτα (Quad Flat No leads) με τα εξής χαρακτηριστικά:

- Τροφοδοτείται από 2.5-5.5V DC, ως εκ τούτου κατάλληλο για εφαρμογές με χαμηλή ισχύ.
- Περιέχει έναν ελεγκτή USB 2.0 on-chip με ρυθμιστή τάσης. Το USB χρησιμοποιείται για τον προγραμματισμό του PIC.
- Ευέλικτη δομή ταλαντωτή. Διαθέτει γεννήτρια ρολογιού ενσωματωμένο στον PIC και μπορεί επίσης να χρησιμοποιήσει εξωτερικό ταλαντωτής κρυστάλλου.
- Τρεις εξωτερικές διακοπές, τέσσερει-alls χρονοδιακόπτες, διπλούς αναλογικούς συγκριτές με πολλαπλές εισόδους,κύρια σύγχρονη σειριακή θύρα (MSSP) που υποστηρίζει επικοινωνίες SPI και I2C.
- Ενσωματωμένο μετατροπέα A/D(Analog/Digital) των 10-bit με 13 κανάλια, καθώς επίσης μονάδα PWM.
- Ενσωματωμένη μνήμη προγραμματισμού στα 16KB και 2KB μνήμη δεδομένων.Υποστηρίζει δυνατότητα προγραμματισμού εντός συστήματος (ISP), με δυνατότητα εντοπισμού σφαλμάτων.

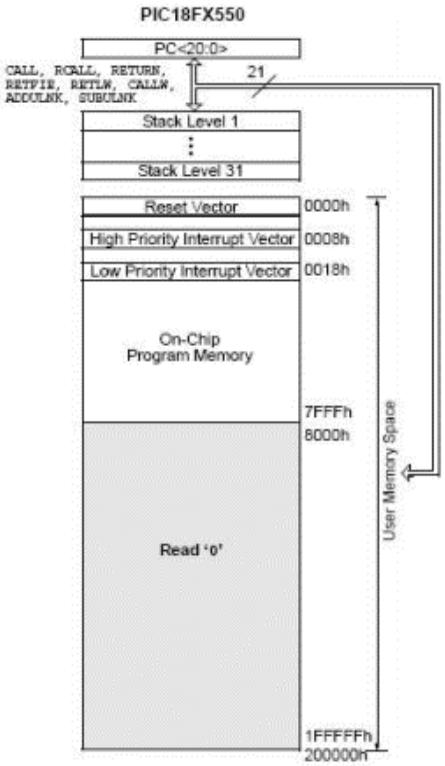


Εικόνα 2.6 Πακέτα μικροελεγκτών

2.3.2 Οργάνωση Μνήμης PIC18F4550

Η μνήμη του μικροελεγτή 18F4550 με βάση την αρχιτεκτονική Harvard, αποτελείται απο τρείς τύπους μνήμης που είναι η **μνήμη προγράμματος**, η **μνήμη δεδομένων** και η **μνήμη δεδομένων EEPROM**.

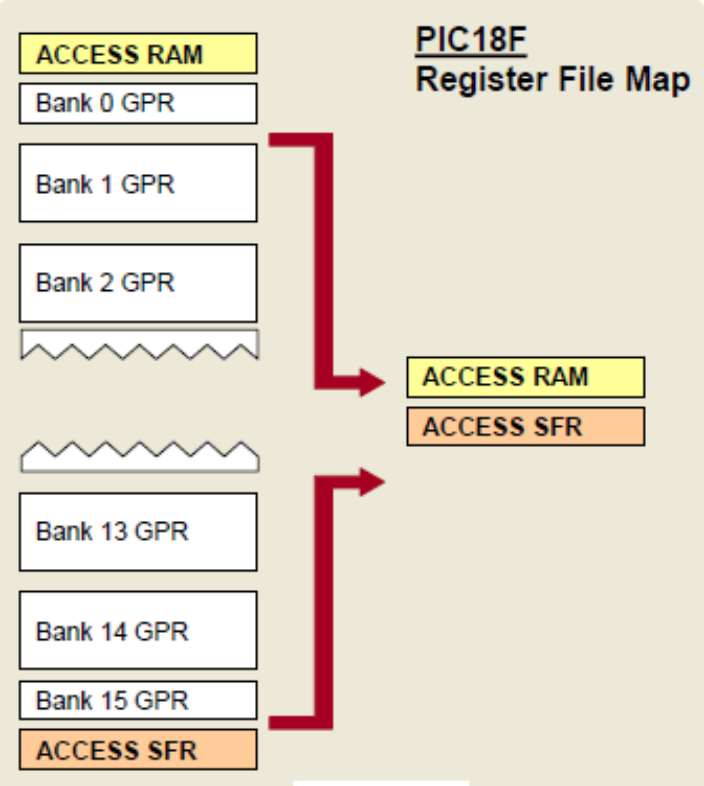
- **Οργάνωση μνήμης προγράμματος**: Ο PIC18F4550 περιέχει έναν μετρητή προγράμματος 21-bit που είναι ικανός να χειριστεί χώρο μνήμης προγράμματος 2MB. Έχει 32KB μνήμης Flash και μπορεί να αποθηκεύσει έως και 16.384 λέξεις. Το Σχήμα 2.7 δείχνει την οργάνωση μνήμης προγράμματος.



Σχήμα 2.7 Οργάνωση μνήμης προγράμματος

- **Οργάνωση μνήμης δεδομένων:** Η μνήμη δεδομένων στον 4550 υλοποιείται ως στατική μνήμη RAM. Κάθε καταχωρητής στη μνήμη δεδομένων έχει μια διεύθυνση 12-bit, επιτρέποντας έτσι έως και 4096 byte μνήμης δεδομένων. Ο χώρος μνήμης στην οικογένεια PIC18 χωρίζεται σε 16 τράπεζες (banks) που περιέχουν 256 bytes η καθεμία. Ο 4550 συγκεκριμένα περιέχει οκτώ πλήρεις banks, για συνολικά 2048 byte. Το Σχήμα 2.8 δείχνει τα δεδομένα οργάνωσης μνήμης για αυτόν τον μικροελεγκτή. Η μνήμη δεδομένων περιέχει καταχωρητές ειδικών λειτουργιών (SFR) και καταχωρητές γενικού σκοπού (GPR). Τα SFR χρησιμοποιούνται για τον έλεγχο και την κατάσταση των ελεγκτών και των περιφερειακών λειτουργιών, ενώ οι GPR χρησιμοποιούνται για αποθήκευση δεδομένων. Το αποτέλεσμα μιας λειτουργίας ανάγνωσης μιας μη εκτελεσμένης τοποθεσίας θα διαβάζεται ως "0". Το σετ εντολών και η αρχιτεκτονική επιτρέπουν τη λειτουργία σε όλες τις τράπεζες. Ολόκληρη η μνήμη δεδομένων μπορεί να προσεγγιστεί με άμεσες ή έμμεσες διευθύνσεις. Για να διασφαλίσουμε ότι καταχωρητές μπορούν να προσπελαστούν σε έναν μόνο κύκλο, οι συσκευές PIC18 χρησιμοποιούν μια τράπεζα Πρόσβασης(Access Bank). Αυτός είναι ένας χώρος μνήμης 256 byte που παρέχει γρήγορη πρόσβαση σε SFR και το χαμηλότερο τμήμα του GPR Bank 0 χωρίς τη χρήση του Bank Select Register (BSR).
- **Οργάνωση μνήμης EEPROM:** Η EEPROM είναι μια μη πτητική συστοιχία μνήμης, δηλαδή δεν χάνει τα δεδομένα της όταν διακοπεί η τροφοδοσία και είναι ξεχωριστή απο την μνήμη RAM και την μνήμη προγράμματος η οποία χρησιμοποιείται για μακροχρόνια αποθήκευση δεδομένων προγράμματος. Η EEPROM μπορεί να διαβαστεί και να γραφτεί κατα την διάρκεια λειτουργίας του μικροελεγκτή με την χρήση τεσσάρων SFR καταχωρητών(EECON1, EECON2, EEADR και EEDATA). Κατά την διασύνδεση με τα μπλόκ μνήμης ο καταχωρητής EEDATA κρατάει 8-bit για την ανάγνωση/εγγραφή, ο καταχωρητής EEADR κρατάει την διεύθυνση της EEPROM στην οποία γίνεται η πρόσβαση.

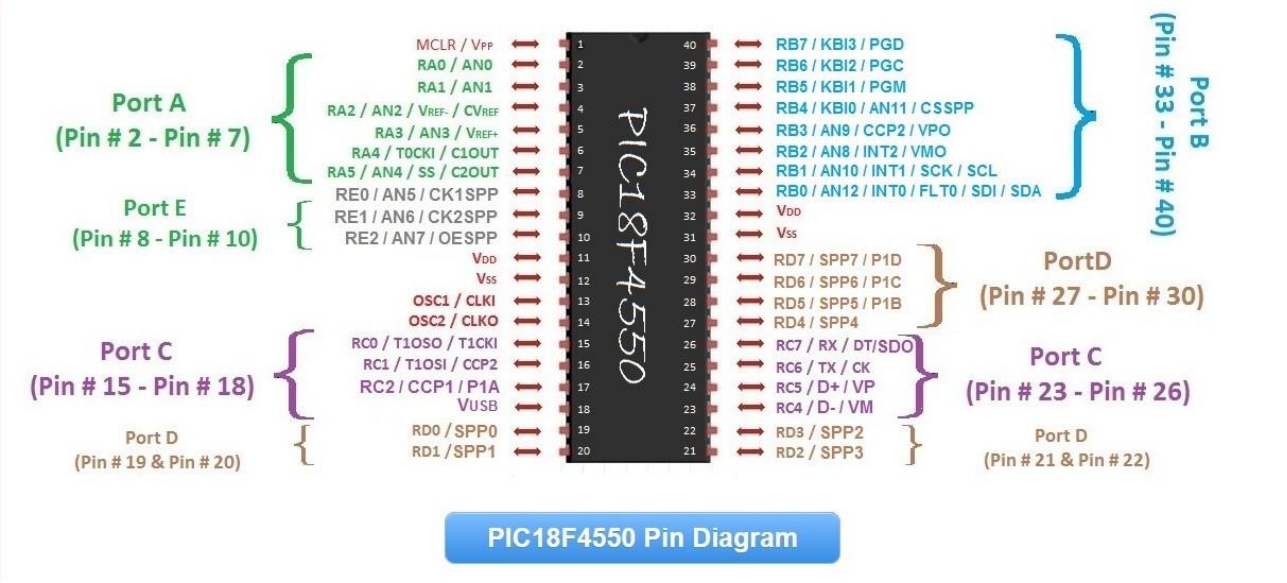
Πρέπει να σημειωθεί ότι όταν γράφουμε δεδομένα σε κάποια διεύθυνση της μνήμης EEPROM που έχει ήδη δεδομένα τότα κατα την εγγραφή διαγράφονται τα παλία δεδομένα και προστίθενται τα νέα δεδομένα.



Σχήμα 2.8 Οργάνωση μνήμης δεδομένων

2.3.3 Είσοδοι έξοδοι γενικού σκοπού

Ο μικροελεγτής PIC18F4550 αποτελείται απο 5 παράλληλες πόρτες που είναι PORTA , PORTB , PORTC, PORTE και PORTD ,οι οποίες χρησιμοποιούνται ως είσοδοι ή ως έξοδοι ανάλογα με την εφαρμογή που εκτελείται. Κάθε πόρτα περιέχει ένα σύνολο απο συγκεκριμένες λειτουργίες οι οποίες με βάση το datasheet μπορούν εύκολα να προσδιοριστούν. Στο Σχήμα 2.9 μπορούμε να δούμε αναλυτικά τα pins του μικροελεγκτή καθώς και τις επιπλέον λειτουργίες του.
Για παράδειγμα στην πόρτα C μπορούμε να διακρίνουμε το pin 23 το οποίο μπορεί να χρησιμοποιηθεί για τις εξής λειτουργίες RC4, D- και VM. Με βάση το datasheet[9] του μικροελεγκτή 18F4550 το pin 23 στην λειτουργία RC4 μπορεί να χρησιμοποιηθεί μόνο ως είσοδος δεδομένων,ενώ στις λειτουργίες D- και VM το συγκεκριμένο pin χρησιμοποιείται ως έξοδος δεδομένων (Data-) του USB και ως πομποδέκτης USB (VM) αντίστοιχα. Η κάθε λειτουργία των εισόδων-εξόδων μπορεί να προσδιοριστεί μόνο με βάση το datasheet του μικροελεγκτή.



Σχήμα 2.9 Πόρτες εισόδου-εξόδου

2.4 Οικογένεια μικροελεγκτών AVR της Atmel

Το AVR σημάινει ALF Vegard Risc και είναι μια οικογένεια μικροελεγκτών που αναπτύχθηκε το 1996 από την Atmel, η οποία αποκτήθηκε από την Microchip Technology το 2016. Πρόκειται για τροποποιημένη αρχιτεκτονική Harvard 8-bit RISC single-chip μικροελεγκτές. Το AVR ήταν μια από τις πρώτες οικογένειες μικροελεγκτών που χρησιμοποίησαν μνήμη flash on-chip για αποθήκευση προγραμμάτων, σε αντίθεση με μια προγραμματιζόμενη ROM, EPROM ή EEPROM που χρησιμοποιούνταν από άλλους μικροελεγκτές εκείνη τη στιγμή. Οι μικροελεγκτές AVR βρίσκουν πολλές εφαρμογές ως ενσωματωμένα συστήματα και είναι ιδιαίτερα συνηθισμένες σε ενσωματωμένες εκπαιδευτικές εφαρμογές.

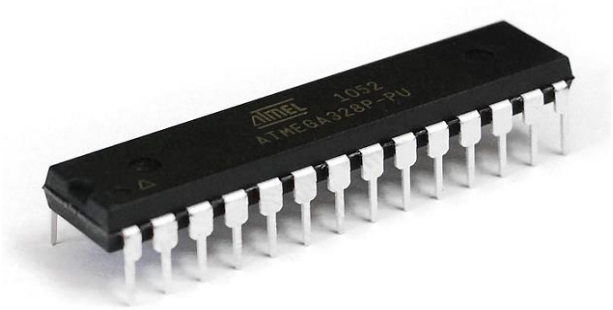
Οι AVR διακρίνονται σε 3 μεγάλες κατηγορίες τους **tinyAVR**, **megaAVR** και **xmegaAVR**, καθένα με συγκεκριμένες ιδιαιτερότητες και λειτουργίες.Συγκεκριμένα η σειρά tinyAVR μικροελγκτές έχουν στην διάθεσή τους 0.5 έως 16 KB μνήμη προγράμματος,είναι πακέτα που εχουν 6 και 32 πόρτες εισόδου/εξόδου και έχουν περιορισμένο σέτ περιφερειακών που είναι συμβατά με τον συγκεκριμένο μικροελεγκτή. Στην συνέχεια οι megaAVR σε αντίθεση με τους tinyAVR έχουν περισσότερη μνήμη προγράμματος που ξεκινάει απο τα 4 KB και φτάνει έως και τα 512 KB που εξασφαλίζει καλύτερες ταχύτητες.Επιπλέον τα πακέτα αυτών των μικροελεγκτών έχουν 28 και 100 pins εισόδου/εξόδου με μεγαλύτερο εύρος συμβατών περιφεφρειακών, πράγμα που σημαίνει ότι ο χρήστης έχει περισσότερες επιλογές στην δημιουργία κάποιας εφαρμογής και γι'αυτό η συγκεκριμένη οικογένεια μικρολεγκτών είναι η πιο διαδεδομένη στον κόσμο και προορίζεται κυρίως για εκπαιδευτικούς σκοπούς.

Τέλος η σειρά xmegaAVR χρησιμοποιείται για απαιτητικές εφαρμογές που καθώς οι xmega προσφέρουν ακόμη πιο γρήγορη μνήμη προγράμματος που φτάνει έως και τα 348 KB, περισσότερες πόρτες εισόδου/εξόδου που χωρίζονται σε 3 κατηγορίες, A4 με 44 pins, A3 με 64 pins και A1 με 100 pins. Επιπρόσθετα το εύρος των περιφερειακών είναι ακόμη μεγαλύτερο λόγο των απαιτήσεων της συγκεκριμένης κατηγορίας που προορίζεται κυρίως για επαγγελματική χρήση.

Εικόνα 2.10 Κατηγορίες των AVR μικροελεγκτών

2.4.1 Ανάλυση ATMEGA-328P

Ο συγκεκριμένος μικροελεγκτής χρησιμοποιείται κυρίως στην μητρική πλακέτα ανοικτού κώδικα Arduino, για την οποία θα αναφερθούμε αναλυτικά στην ενότητα Χ, και ανήκει στην κατηγορία των megaAVR. Είναι ένας 8 bit μικροελεγκτής εύκολος στην χρήση καθώς υπάρχει ενσωματωμένο πάνω σε μητρική που είναι έτοιμη για χρήση, υπάρχουν έτοιμες βιβλιοθήκες και ο κώδικας για τον προγραμματιμό του είναι σε C/C++ πράγμαα που το καθιστά εύκολο στην χρήση καθώς η συγκεκριμένες γλώσσες προγραμματισμού είναι πλέον οι πιο γνωστές και τις συναντάμε στις περισσότερες εφαρμογές που χρησιμοποιούν ΕΣ.



Εικόνα 2.11 Μικροελεγκτής ATmega328P

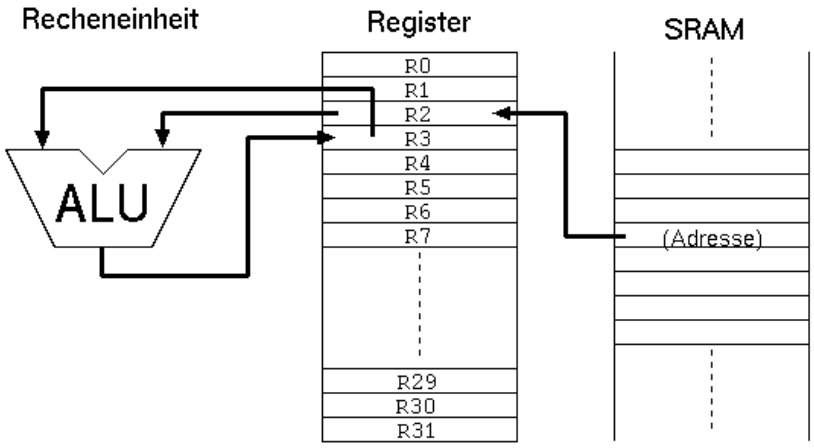
2.4.2 Αρχιτεκτονική του ATMEGA-328P

Όλοι οι AVR μικροελεγκτές ακολουθούν το ίδιο πρότυπο αρχιτεκτονικής όπως φαίνεται και στο Σχήμα 2.10. Συγκεκριμένα αποτελούνται απο τα εξής μέρη:

- **Data BUS 8-bit (Δίαυλος Δεδομένων)**: Είναι παράλληλες γραμμές δεδομένων των 8-bit από τις οποίες ταξιδεύουν τα δεδομένα μέσα στον μικροελεγτή. Η αναφορά του 8 bit στον δίαυλο δεδομένων γίνεται για να προσδιορίσουμε το μήκος των δεδομένων που ρέουν και έχεις ως αποτέλεσμα ο μικροελεγκτής να χαρακτηριστεί ως MCU των 8-bit.
- **ALU (Αριθμητική και Λογική Μονάδα)**: Όπως οι μικροελεγκτές PIC που αναφερθήκαμε στην ενότητα 2.3 έτσι και οι AVR περιέχουν την συγκεκριμένει μονάδα που τους επιτρέπει να χειριστούν αριθμητικές και λογικές πράξεις.
- **Μνήμη SRAM**: Χρησιμοποιείται για την προσωρινή αποθήκευση δεδομένων καθώς είναι πτητικές μνήμες,δηλαδή χάνουν τα δεδομένα τους όταν δεν υπάρχει τροφοδοσία.

Οι συγκεκριμένες μνήμες δεν έχουν απευθείας πρόσβαση στην ALU και γι'αυτό χρησιμοποιούνται καταχωρητές (registers) για την προσπέλαση δεδομένων.

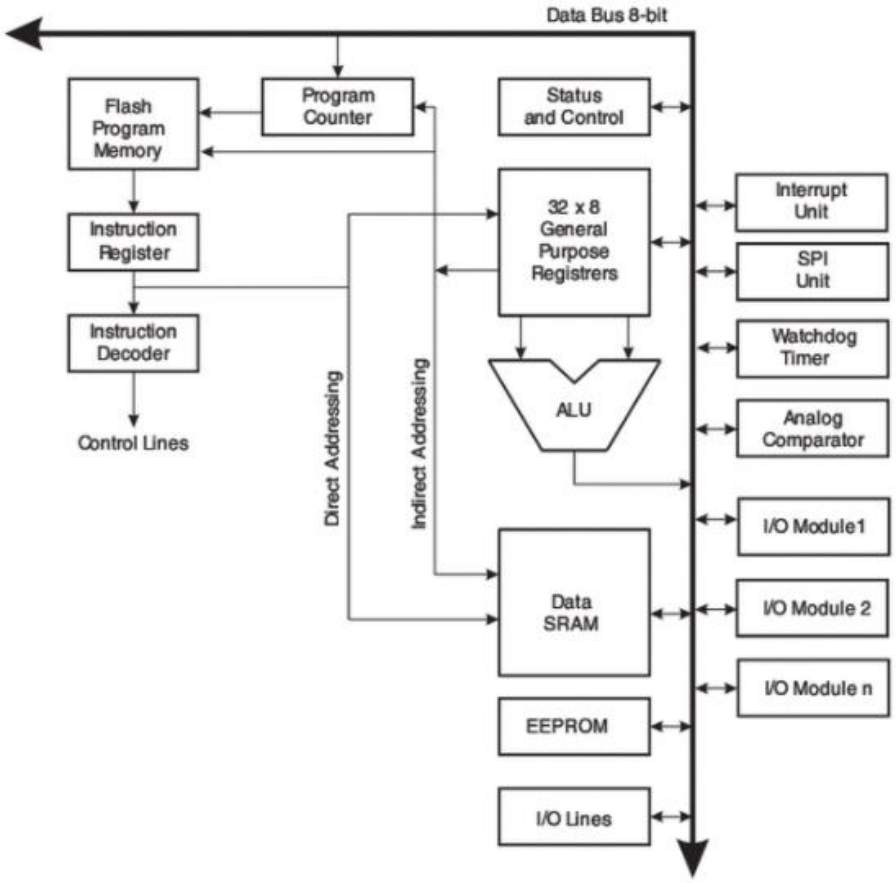
- **Καταχωρητές γενικού σκοπού (GPR 32x8)**: Όπως αναφερθήκαμε παραπάνω η προσπέλαση δεδομένων γίνεται με την βοήθεια των καταχωρητών οι οποίοι επικοινωνούν αμφίδρομα με την ALU και την SRAM . Οι AVR στο σύνολο έχουν 32 καταχωρητές των 8-bit ο καθένας στους οποίους αποθηκεύονται δεδομένα απο την SRAM και στην συνέχεια τα μεταφέρουν διαμέσου του δίαυλου δεδομένων στην αριθμητική και λογική μονάδα επεξεργασίας. Επιπλέον η ALU μπορεί και αυτή με την σειρά της να αποθηκεύσει δεδομένα στους καταχωρητές και να τα περάσουν μέσα στην SRAM ώστε να αποθηκευτούν προσωρινά.



Σχήμα 2.12 Διαδικασία μεταφοράς δεδομένων διαμέσου καταχωρητών

- **Μνήμη EEPROM**: Το συγκεκριμένο είδος μνήμης είναι μη πτητικό,δηλαδή δεν χάνει τα δεδομένα της όταν διακοπεί η τροφοδοσία, οπότε μπορούμε να αποθηκεύσουμε δεδομένα και να διατηρηθούν. Πρέπει να σημειώσουμε ότι η EEPROM στους AVR έχει χρόνο ζωής περίπου 100.000 εγγραφές / σελίδα EEPROM και γι'αυτό όταν προγραμματίζουμε έναν μικροελεγκτή πρέπει να προσέχουμε το μέγεθος πληροφοριών που γράφουμε κάθε φορά που αννεώνουμε την EEPROM.
- **Μνήμη προγράμματος (Flash Program Memory)**: Ανήκει στην κατηγορία μνημών, δηλαδή μπορούμε να αποθηκεύσουμε μόνιμα τα δεδομένα μας. Πρέπει να σημειώσουμε ότι στην συγκεκριμένη μνήμη αποθηκεύει αποκλειστικά και μόνο το πρόγραμμα λειτουργίας που γράψαμε.
- **Καταχωρητής κατάστασης και ελέγχου (Status & Control Register)**: Το είδος των καταχωρητών αυτών αποθηκεύει δεδομένα σχετικά με την κατάσταση του ΕΣ και τον έλεγχο των εντολών του προγράματος,για παράδειγμα εξετάζει τις εντολές if-else του προγράμματος και αποθηκεύει τα αποτελέσματά τους.
- **Καταχωρητής μετρητής προγράμματος(Program Counter Register)**: Είναι υπεύθυνος να παρακολουθεί την θέση του προγράμματος που εκτελείτε στον μικροελεκγτή.
- **Καταχωρητές Instruction και Decoder** : Στους χρήστες οι καταχωρητές αυτοί δεν είναι σημαντικοί, αλλά για έναν μικροελεγκτή είναι απαραίτητοι καθώς περιέχουν οδηγίες απο την CPU για τον τρόπο αποκωδικοποίησης του προγράμματος.

- **Μονάδες Εισόδου / Εξόδου (Input / Output Unit)**: Ένας AVR περιέχει ψηφιακές καθώς και αναλογικές μονάδες I/O που χρησιμοποιούνται για την εισαγωγή δεδομένων αλλά και την απόδοση αποτελεσμάτων αντίστοιχα.

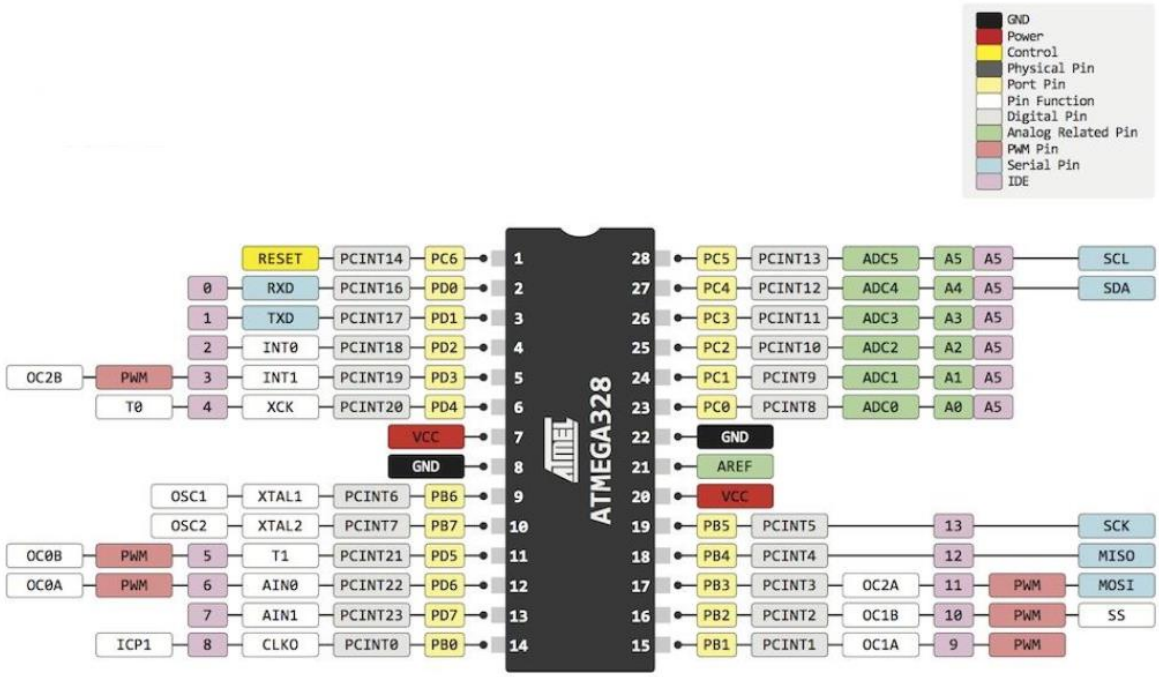


Σχήμα 2.13 Δομή ενός AVR μικροελεγκτή

2.4.3 Διαμόρφωση μονάδων Εισόδου / Εξόδου

Στην υποενότητα αυτή θα αναφέρουμε επιγραμματικά τις λειτουργίες των μονάδων που πρέπει να γνωρίζει ο χρήστης, απο τις οποίες αποτελείται ο μικροελεγκτής ATMEGA328P. Συγκεκριμένα περιέχει 3 πόρτες PBx ,PCx και PDx (κίτρινο χρώμα) όπου x ο αριθμός της πόρτα και καθεμία έχει ένα σύνολο απο συγκεκριμένες λειτουργίες τις οποίες κάθε χρήστης μπορεί να ενημερωθεί απο τα datasheet. Ωστόσο μπορούμε να παρατηρήσουμε απο το Σχήμα 2.14, όπως και κάθε μικροελεγκτής, έχει αφοσιώσει 4 pins για την τροφοδοσία και την γείωση (Power και GND) και περιέχει ένα σύνολο απο συγκεκριμένα pins που εκτελούν την λειτουργία **διαμόρφωσης εύρους παλμών** (PWM). Επιπλέον όπως αναφερθήκαμε στην υποενότητα 2.4.2 οι πόρτες του ATMEGA χωρίζονται σε ψηφιακές (**Digital Port**) και αναλογικές (**Analog Port**) και μπορούμε να διακρίνουμε ότι κάποια πόρτα μπορεί να λειτουρήσει ως ψηφιακή ή αναλογική, ανάλογα με το πρόγραμμα του χρήστη.

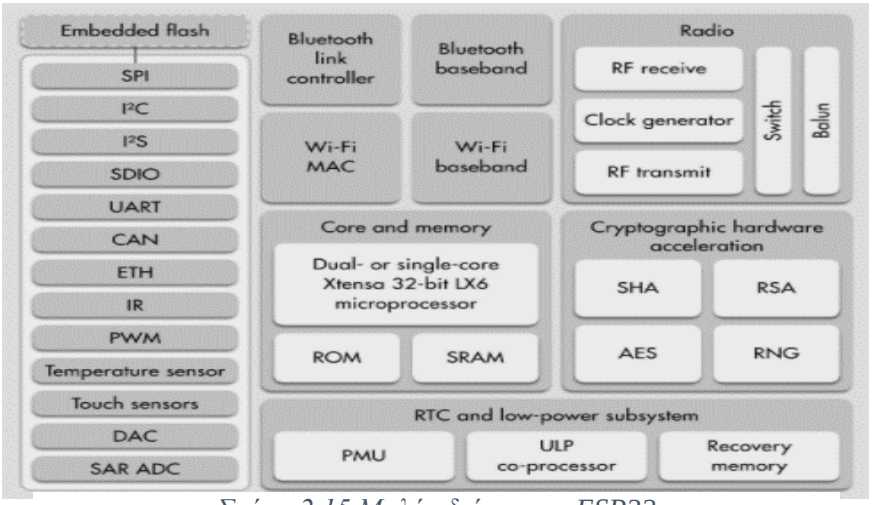
Τέλος θα πρέπει να σημειώσουμε ότι οι λειτουργίες κάποιων πορτών δεν μπορούν να προγραμματιστούν απο τον χρήστη καθώς έχουν κατασκευαστεί και προγραμματιστεί με τέτοιο τρόπο ώστε να εκτελούν μία συγκεκριμένη λειτουργία όπως αυτή των πορτών Power και GND.



Σχήμα 2.14 Πόρτες Εισόδοου / Εξόδου ATMEGA328

2.5 Οικογένεια μικροεπεξεργαστών ESP32

Είναι μικροεπεξεργαστές διπλού πυρήνα με χαμηλό κόστος και ισχύς, με ενσωματωμένο Wi-Fi και Bluetooth. Έχουν ταχύτητα που μπορεί να φτάσει έως και τα 240MHz πράγμα που σημαίνει ότι είναι πολύ πιο γρήγοροι επεξεργαστές απο τον ATMEGA328P και τον PIC18F4550, τους οποίους αναλύσαμε σε προηγούμενες ενότητες. Επιπλέον θα αναφερθούμε στον επεξεργαστή ESP WROOM-32 ο οποίος είναι 6^{η} γενιάς των 32-bit και χρησιμοποιείται κυρίως για επεξεργασία πολύπλοκων δεδομένων και εφαρμογές που απαιτούν Wi-Fi και Bluetooth. Ο επεξεργαστής αυτός εκτός απο το γεγονός ότι έχει μεγαλύτερες ταχύτητες απο τους προκατόχους του, έχει και μεγαλύτερη ευελιξία ως προς την χρήση του. Στην πραγματικότητα όμως το ESP είναι ένα σύστημα σε τσίπ (SoC) που χρησιμοποιείται ως μικροελεγκτής γενικού σκοπού με μια μεγάλη ποικιλία περιφερειακών όπως φαίνεται στο σχήμα 2.15.



Σχήμα 2.15 Μπλόκ διάγραμμα ESP32

Αν και είναι δυνατό να σχεδιάσουμε ένα προϊόν χρησιμοποιώντας το ESP32 SoC, αυτή δεν είναι μια κοινή προσέγγιση. Τα περισσότερα σχέδια με βάση το ESP32 χρησιμοποιούν προκατασκευασμένες μονάδες που αποτελούνται από ένα πραγματικό SoC ESP32, εξωτερική μνήμη flash, ένα κρύσταλλο και μια προρυθμισμένη κεραία PCB (ή μια υποδοχή κεραίας IPEX) και στην συνέχεια όλο αυτό το συγκρότημα τοποθετείται μέσα σε ένα θωρακισμένο δοχείο (Εικόνα 2.16).

Εικόνα 2.16 ESP WROOM-32

2.5.1 Βασικά περιφεφρειακά του ESP WROOM-32

Συγκεκριμένα το ESP32 τρέχει προγράμματα των 32 bit με μνήμη RAM στα 512 KB και περιέχει ένα μεγάλο εύρος απο περιφεριακά όπως είναι τα εξής:

- **Αισθητήρες αφής (Touch Sensors):** Το ESP32 διαθέτει 10 αισθητήρες αφής που μπορούν να ανιχνεύσουν αλλαγές σε οτιδήποτε έχει ηλεκτρικό φορτίο, όπως το ανθρώπινο δέρμα. Έτσι μπορούν να ανιχνεύσουν αλλαγές που προκαλούνται όταν αγγίζουμε οι ίδιοι τις ακίδες εισόδου / εξόδου. Αυτές οι ακίδες με την σειρά τους μπορούν να ενσωματωθούν σε χωρητικές επαφές και να αντικαταστήσουν τα μηχανικά κουμπιά, αλλά κυρίως χρησιμοποιούνται ως πηγή αφύπνισης όταν το ESP μπαίνει σε λειτουργία ύπνου (sleep mode).
- **Μετατροπείς ADC και DAC:** Περιέχει 18 κανάλια μετατροπείς αναλογικού σήματος σε ψηφιακό και 2 κανάλια DAC των 8-bit, δηλαδή μετατροπή σήματος απο ψηφιακό σε αναλογικό.
- **Δίαυλος I2C (Inter-Integrated Circuit):** Είναι ένας σειριακός δίαυλος και χρησιμοποιείται για την επικοινωνία τους ESP με τα περιφερειακά.
- **Δίαυλος SPI (Serial Peripheral Interface):** Είναι δίαυλος δεδομένων όμοιος με τον I2C με την μόνη διαφορά ότι προορίζεται για επικοινωνία του ESP με τα περιφεριακά σε μικρές ταχύτητες όπως καταχωρητές και SD μνήμες.
- **Σειριακή επικοινωνία UART (Universal Asynchronous Receiver/Transmitter):** Είναι ένα κύκλωμα των υπολογιστών το οποίο διαμεσολαβεί στην σειριακή επικοινωνία ενσωματωμένων υπολογιστικών συστημάτων όπως μικροελεγκτές και πραγματοποιείται διαμέσου θυρών RS.
  Η ταχύτητα επικοινωνίας ή αλλιώς ρυθμός μετάδοσης μπορεί να παραμετροποιηθεί και μετριέται σε baud rate (bits/sec). Ένα UART παίρνει bytes δεδομένων και τα στέλνει σε σειριακή μορφή ως bits. Στον προορισμό ένα δεύτερο UART συλλέγει τα bits και δημιουργεί τα bytes δεδομένων που αποστάλθηκαν. Κάθε UART περιέχει ένα καταχωρητή ολίσθησης (shift register) ο οποίος χρησιμοποιείται για την μετατροπή της σειριακής σε παράλληλη μορφή (εγγραφή/διάβασμα των δεδομένων μέσα ένα καταχωρητή από λήψη ή για αποστολή). Στην επικοινωνία με UART

υπάρχουν εξωτερικά κυκλώματα που μετατρέπουν τα επίπεδα των volt του σήματος του καλωδίου σε αυτό που χρησιμοποιεί το UART.

- **Δίαυλος I2S (Inter-IC Sound):** Είναι ένας δίαυλος που χρησιμοποιείται για την σύνδεση ψηφιακών ήχων μεταξύ τους. Διαχωρίζει σήματα ρολογιού και σειριακών δεδομένων με αποτέλεσμα να χρησιμοποιούνται απλούστεροι δέκτες σε σχέση με τους δέκτες που χρησιμοποιούν ασύγχρονα συστήματα επικοινωνίας (όπως αυτά των UART) που πρέπει να ανακτήσουν τους παλμούς ρολογιού απο την ροή δεδομένων.
- **PWM:** Οπως και στους αλλους μικροελεγκτές έτσι και εδώ μπορούμε να κάνουμε χρήση διαμόρφωσης εύρους παλμών.
- **Μνήμες:** Το ESP32 διαθέτει πολλαπλές μνήμες όπως **DRAM**, η οποία είναι μια πτητική μνήμη και χρησιμοποιείται για την προσωρινή αποθήκευση δεδομένων, **IRAM**, στην οποία αποθηκεύονται δεδομένα (των 32-bit) που πρόκειται να εκτελεστούν απο το πρόγραμμα και τέλος χρησιμοποιούν έναν συνδιασμό των παραπάνω **D/IRAM** η οποία μπορεί να λειτουργεί άλλοτε ως DRAM ή ως IRAM ανάλογα με την εφαρμογή που εκτελέι ο χρήστης.
- **Πρωτόκολλο CAN(Controller Area Network):** Υπάρχουν 4 είδη μηνυμάτων (πλαισίων) CAN που είναι, το πλαίσιο **δεδομένων**, **σφάλματος**, **υπερφόρτωσης** και **απομακρυσμένου ελέγχου** (Remote). Το πλαίιο δεδομένων είναι το βασικό μήνυμα που μεταδίδει δεδομένα απο τον πομπό προς του κόμβου, μέσα σε έναν δίαυλο. Ένα πλαίσιο σφάλματος μπορεί να μεταδοθεί απο οποιονδήποτε κόμβο που ανιχνευέι σφάλμα δικτύου, ενώ τα μηνύματα υπερφόρτωσης συνηθίζουν να μας παρουσιάζουν επιπλέον καθυστέρηση μεταξύ δεδομένων ή και πλαίσια απομακρυσμένου ελέγχου. Τέλος τα πλαίσια απομακρυσμένου ελέγχου μεταδίδεται απο έναν πομπό για να ζητήσει δεδομένα απο έναν συγκεκριμένο κόμβο.

2.5.2 Λειτουργία του Wi-Fi

Ένα ασύρματο δίκτυο Wi-Fi χρησιμοποιεί σήματα ραδιοσυχνοτήτων αντί καλωδίων για την σύνδεση των συσκευών μας σε αυτό. Οι ζώνες κύματος του Wi-Fi έχουν σχετικά υψηλή απορρόφηση και γι'αυτό πολλά κοινά εμπόδια όπως τοίχοι, κολώνες, οικιακές συσκευές κ.λ.π μπορούν να μειώσουν σημαντικά το εύρος, αλλά αυτό βοηθά επίσης στην ελαχιστοποίηση των παρεμβολών μεταξύ διαφορετικών δικτύων σε πολυσύχναστα περιβάλλοντα. Ένα σημείο πρόσβασης(Access Point-AP) έχει συχνά μια εμβέλεια περίπου 20 μέτρων σε εσωτερικούς χώρους, ενώ ορισμένα σύγχρονα σημεία πρόσβασης απαιτούν έως 150 μέτρα σε εξωτερικούς χώρους.Ωστώσο πρέπει να αναφερθούμε στο που βασίζεται πραγματικά το Wi-Fi και αυτό είναι το προτότυπο IEEE802.11 και χρησιμοποιούνται συνήθως για τοπική δικτύωση συσκευών, πρόσβαση στο Διαδίκτυο και στα πολυμέσα. Το ESP32 έχει ενσωματωμένο Wi-Fi που βασίζεται σε αυτού του είδους το πρωτότυπο και συγκεκριμένα τα 802.11 b/g/n/e/i ,όπου:

- **Το πρότυπο 802.11b**, έχει μέγιστο ρυθμό ακατέργαστων δεδομένων 11 Mbit / seconds (Megabits ανά δευτερόλεπτο) και χρησιμοποιεί την ίδια μέθοδο πρόσβασης πολυμέσων που ορίζεται στο αρχικό πρότυπο. Το συγκεκριμένο πρότυπο λειτουργεί με συχνότητα στα 2.4GHz.
- **Το πρότυπο 802.11g**, λειτουργέι και αυτό με συχνότητα 2.4 GHz με την διαφορά ότι έχει μέγιστο ρυθμό δεδομένων 54 Mbit/sec χωρίς κωδικούς διόρθωσης και μέση απόδοση περίπου στα 22Mbit/sec, γεγονός που σημαίνει ότι η πληροφορία μεταφέρεται πολύ πιο γρήγορα απο το πρότυπο 802.11b.
- **Το πρότυπο 802.11n**, είναι μία βελτιωμένη έκδοση απο τα προηγούμενα όπου έχουμε την προσθήκη υποστήριξης κεραιών πολλαπλών εισόδων-πολλαπλών εξόδων(MIMO) και λειτουργία

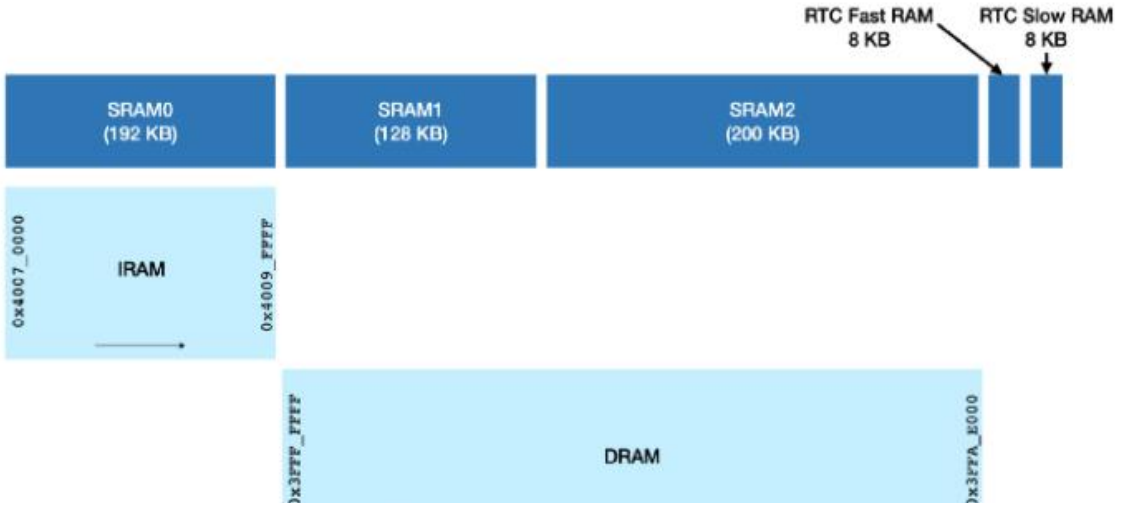
- στις ζώνες των 2.4 GHz και 5GHz (η υποστήριξη για ζώνες 5 GHz είναι προαιρετική) με ταχύτητα δεδομένων που κυμαίνεται απο 54 Mbit/sec έως 600Mbit/sec.
- **Το πρότυπο 802.11e**, αντιμετωπίζεται το πρόβλημα ποιότητας υπηρεσίας QoS. Για περιήγηση σε εφαρμογές όπως περιήγηση στο Διαδίκτυο, αποστολή email, καθυστερήσεις στη λήψη απαντήσεων ή αποστολή δεδομένων δεν έχει σημαντικό αντίκτυπο. Καταλήγει σε αργές λήψεις ή μικρές καθυστερήσεις στα email που αποστέλλονται. Παρόλο που μπορεί να προκαλέσει μικρή ενόχληση στον χρήστη, δεν υπάρχει πραγματικός επιχειρησιακός αντίκτυπος στην παρεχόμενη υπηρεσία. Ωστόσο, για εφαρμογές όπως η μετάδοση φωνής ή βίντεο, υπάρχει πολύ μεγαλύτερη επίδραση και αυτό δημιουργεί πολύ μεγαλύτερη ανάγκη για το 802.11e.
- **Το πρότυπο 802.11i**, προσφέρει έναν μακροπρόθεσμο έλεγχο ταυτότητας, εμπιστευτικότητας και ακεραιότητας ως λύση ασφάλειας για ασύρματα δίκτυα.

Το ESP32 έχει δύο λειτουργίες που βασίζονται στο Wi-Fi και αυτές είναι το AP και το Web Station. Στην λειτουργία του AP δημιουργούμε το δικό μας δίκτυο όπου μπορούν να συνδεθούν κοντινές συσκευές, οι οποίες μπορούν να μεταφέρουν δεδομένα απο και πρός το ESP32, ενώ στην λειτουργία του σταθμού το ESP32 συνδέεται στο τοπικό μας δίκτυο και παίρνει μια IP απο το ρούτερ μας. Με αυτό τον τρόπο μπορούμε να επικοινωνήσουμε με το ESP διαμέσου του ρούτερ μας.

2.5.3 Οργάνωση μνήμης

Η εσωτερική μνήμη του μικροεπεξεργαστή ESP32 είναι ο πιο πολύτιμος πόρος καθώς καταλαμβάνει το μεγαλύτερο κομμάτι ενός τσίπ. Σήμερα οι περισσότερες εφαρμογές που υλοποιούνται πάνω σε έναν μικροεπεξεργαστή έχουν συνεχώς την ανάγκη μνήμης με αποτέλεσμα οι χρήστες να πρέπει να αξιοποιήσουν στο έπακρο το υλικό που διαθέτουν καθώς επίσης και να κατανοήσουν την αρχιτεκτονική της μνήμης. Όπως αναφέραμε στην αρχή της ενότητας 2.5 το ESP32 βασίζεται σε μία δομή SoC που περιλαμβάνει υποσυστήματα επικοινωνίας (Wi-Fi, Bluetooth) και απαιτούν συγκεκριμένη μνήμη ώστε να λειτουργούν σωστά και ομαλά. Συγκεκριμένα θα αναφερθούμε στα είδη μνήμης που περιλαμβάνονται μέσα στον μικροεπεξεργαστή καθώς επίσης και την λειτουργία της κάθε μνήμης εξατομικευμένα.

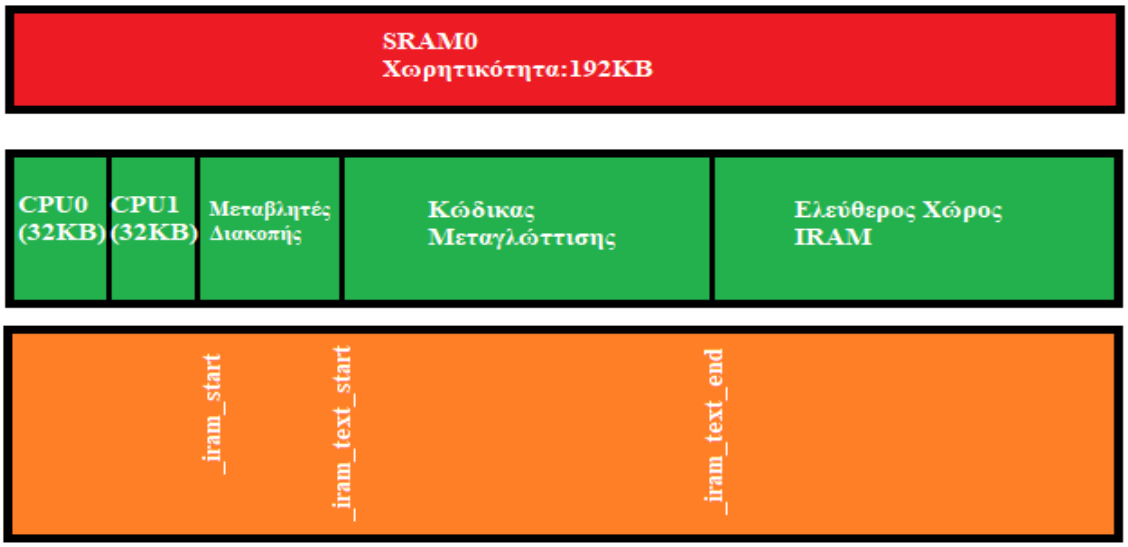
Παρακάτω θα αναφερθούμε στην δομή της SRAM η οποία χωρίζεται σε 3 μπλόκ μνήμης (SRAM0, SRAM1, SRAM2) και χρησιμοποιείται με δύο τρόπους, ως IRAM και DRAM.



Σχήμα 2.17 Οργάνωση μνήμης ESP32

Απο το σχήμα 2.17 μπορούμε να παρατηρήσουμε ότι η SRAM0 μπορεί να χρησιμοποιηθεί ως χώρος διευθύσεων IRAM πειρίπου στα 192 KB αποθηκευτικού χώρου ενώ η SRAM1 και SRAM2 ως DRAM στα 328 KB. Σε μερικές περιπτώσεις όπου αυτό είναι απαραίτητο η SRAM1 μπορεί να χρησιμοποιηθεί και ως αποθηκευτικός χώρος IRAM ωστόσο για τον συγκεκριμένο MCU λειτουργεί ως DRAM.πειρίπου στα 192 KB αποθηκευτικού χώρου ενώ η SRAM1 και SRAM2 ως DRAM στα 328 KB.

Η IRAM όπως αναφέραμε προηγουμένος έχει στην διάθεσή της 192 KB αποθηκευτικού χώρου στο ESP32, όπου τα 32 KB χρησιμοποιούνται ως κρυφή μνήμη της CPU0 και άλλα 32 KB για την CPU1(Το ESP είναι ένας MCU με δύο πυρήνες όπως αναφέραμε στην αρχή της ενότητας 2.5 και γι'αυτό γίνεται λόγος για CPU0 και CPU1). Στην συνέχεια μετά την μή διαθειμότητα των 64 KB ο κώδικας διασύνδεης(linker script), που υπάρχεχι ενσωματωμένο μέσα στο ES32, αρχίζει να τοποθετεί τα τα διανύματα διακοπής και το μεταγλωττισμένο κείμενο της εφαρμογής μέσα στην IRAM.



Σχήμα 2.18 Δομή της IRAM

Ενώ στις περισσότερες εφαρμογές ο κώδικας εκτελείται έξω απο την μνήμη flash, υπάρχουν κάποια τμήματα του κώδικα που εξαρτώνται απο τον χρόνο ή πρέπει να εκτελεστούν μέσα στην flash μνήμη και αυτό έχει ως αποτέλεσμα να πρέπει να τοποθετηθούν μέσα στην μνήμη IRAM. Αυτό επιτυγχάνεται με τον κώδικα διασύνδεσης που αναφέραμε παραπάνω. Απο το σχήμα 2.18 παρατηρούμε κάποιες εκφράσεις _iram_start , _iram_text_start και iram_text_end που τοποθετούνται απο τον κώδικα διασύνδεσης και είναι όριο που προσδιορίζουν απο που θα ξεκινάει ο κώδικας μεταγλώττισης, που θα τελειώνει και μέσα σε ποιά όρια θα τοποθετηθεί. Το υπόλοιπο κομμάτι Free IRAM τοποθετείτε στο υπόλοιπο της μνήμης και είναι διαθέσιμο για οποιαδήποτε αποθήκευση. Επιπλέον πρέπει να σημειωθεί ότι ο πυρήνας CPU1 του ESP32 απο κατασκευή δεν είναι διαθέσιμος, αλλά θα πρέπει να ενεργοποιηθέι απο τον χρήστη διαμέσου του κώδικα μεταγλώττισης και για τον λόγο αυτό τα 32KB μνήμης που καταλαμβάνει προστίθενται στον χώρο διαθέσιμη μνήμη IRAM (Free IRAM). Η IRAM αδρομερώς χρησιμοποιείται για την αποθήκευση δεδομένων με τους εξής δύο περιορισμούς:

- η διεύθυνση που χριημοποιείται για την πρόσβαση στα δεδομένα μέσα στην IRAM πρέπει να έχει μήκος 32-bit
- και το μέγεθος των δεδομένων προς αποθήκευσει πρέπει να είναι και αυτό 32-bit.

Εάν μία εφαρμογή δεν ακολουθεί και τους δύο περιορισμούς τότε δεν μπορεί να χρησιμοποιήσει την IRAM. Σε οποιαδήποτε άλλη περίπτωση μπορεί να γίνει πρόσβαση στην συγκεκριμένη μνήμη.Σε αντίθεση με την SRAM0 η οποία λειτουργεί ως IRAM, οι μνήμες SRAM1 και SRAM2 που λειτουργούν ως DRAM έχουν μία πιο περίπλοκη δομή καθώς οι διευθύνσεις της ξεκινούν στο τέλος της SRAM2 αυξάνωντας προς τα πίσω όπως στο παρακάτω Σχήμα 2.19. Αυτό συμβαίνει για τον γεγονός ότι η SRAM1 σε κάποιες περιπτώσεις μπορεί να χρησιμοποιηθεί ως IRAM οπότε όταν συμβεί κάτι τέτοιο τότε ως DRAM θα λειτουγεί μόνο η SRAM2 και με κάποιο τρόπο πρέπει να ξεχωρίζει κάθε φορά αν η SRAM1 λειτουργεί ως DRAM ή όχι.

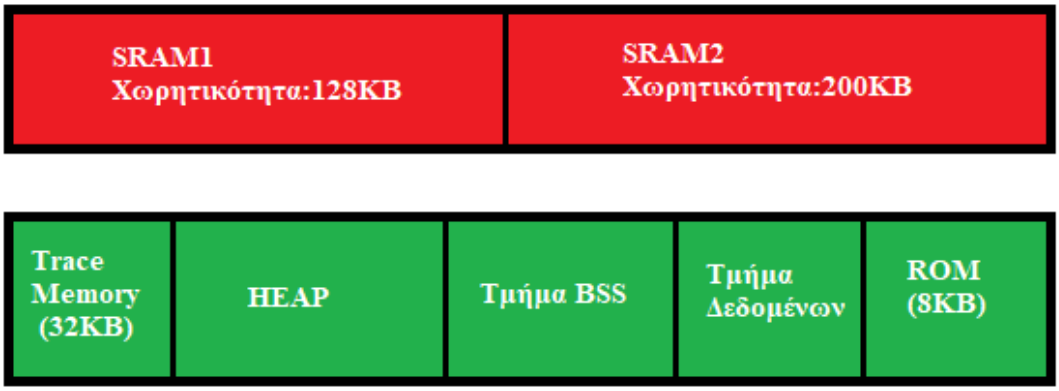
Σχήμα 2.19 Δομή DRAM με ελεγκτή BT

Σχήμα 2.20 Δομή της DRAM

Όπως και στην IRAM έτσι και εδώ η SRAM1 καταλαμβάνει χώρο 128 KB και η SRAM2 200 KB αποηθκευτικού χώρου όπου τα πρώτα 8 KB χρησιμοποιούνται για την αποθήκευση δεδομένων για τις διάφορες λειτουργίες της μνήμης ROM. Συναντάμε και έδω τον κώδικα διασύνδεσης ο οποίος τοποθετεί το αρχικό τμήμα δεδομένων μετά απο την μνήμη των 8KB. Στην συνέχεια παρατηρούμε το τμήμα BSS κάτι το οποίο δεν συναντήσαμε στην μνήμη IRAM. Το τμήμα αυτό είναι γνωτό και ως τμήμα μη αρχικοποιημένων δεδομένων και περιέχει όλες τις καθολικές μεταβλητές που αρχικοποιούνται στο μηδέν ή έχουν ρητή αρχικοποίηση στον πηγαίο κώδικα. Χαρακτηριστικό παράδειγμα είναι σε έναν κώδικα της C/C++ η στατική μεταβλητή static int x ή static int y = 0 θα αποθηκευτούν στο τμήμα της μνήμης BSS. Το υπόλοιπο κομμάτι της μνήμης που περισσέυει μετά την κατανομή δεδομένων προστίθεται ως σωρός (Heap), όπου είναι το τμήμα που μπορεί ο χρήστης να επεξεργαστεί διαμέσου του κώδικά του,δηλαδή να κάνει εγγραφή και διαγραφή δεδομένων απο την μνήμη όποτε εκείνος επιθυμεί. Επιπρόσθετα θα αναφερθούμε στον όρο ελεγκτή Burst Type ή BT ο οποίος είναι υπεύθυνος

για την μεταφορά πακέτων δεδομένων μεταξύ λογισμικού και υλικού. Όταν ενεργοποιηθεί αυτή η λειτουργία τότε ο κώδικας διασύνδεσης κρατάει 54 KB μνήμης για αυτό τον σκοπό, όπως φαίνεται και στο Σχήμα 2.20.

Εκτός απο την λειτουργία BT ,είτε αυτή είναι ενεργοποιημένη είτε όχι, θα συναντήσουμε στην DRAM την μνήμη Trace η οποία καταλαμβάνει τον δικό της χώρο των 32 KB και δίνει πληροφορίες σχετικά με χώρο που καταλαμβάνουν τα δεδομέναν μας μεσα στην DRAM συνολικά.



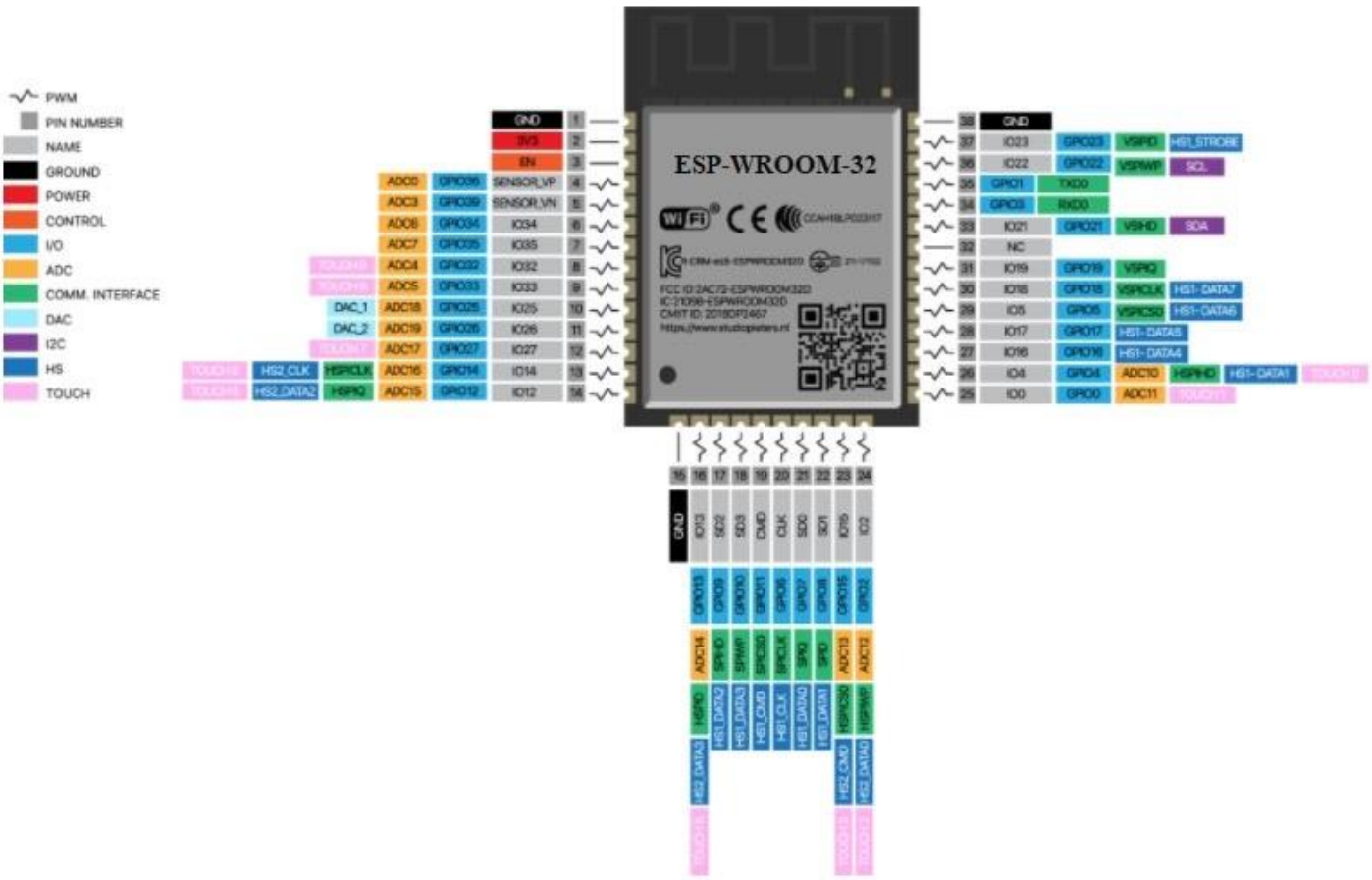
Σχήμα 2.21 Δομή DRAM με Trace Memory

Τέλος το ESP32 παρέχει μία ακόμη κατηγορία μνήμης με ονομασία SPIRAM (θα την συναντήσουμε και ως PSRAM – Pseudo Static RAM) η οποία είναι άμεσα προσβάσιμη διαμέσου της IRAM. Η SPIRAM μπορεί να αποθηκεύση δεδομένα μέχρι 4 MB και έχει εύρος διευθύνσεων απο 0x3F80_0000 έως 0x3FBF_FFFF. Μια εφαρμογή μπορεί να εκμεταλλευτεί την χρήση της SPIRAM με 3 τρόπους:

- Χρήση της SPIRAM ως τμήμα BSS για συγκεκριμένες λογισμικές λειτουργίες.
- Μπορούμε να χρησιμοποιήσουμε τον κατανεμητή heap για να οργανώσουμε δυναμικά την SPIRAM. Με τον όρο δυναμικά κάνουμε αναφορά στον κώδικα εφαρμογής γραμμένος σε C/C++, όπου οι μεταβλητές είναι ορισμένες με τέτοιο τρόπο που επιτρέπουν την καλύτερη διαχείρηση μνήμης και δομής του κώδικα όταν δεν είναι γνωστός ο χώρος μνήμης που θα καταλαμβάνει κάποια εφαρμογή. Για παράδειγμα για να ορίσουμε μια μεταβλητή δυναμικά που θα δέχεται ως όρισμα έναν ακέραιο αριθμό θα πρέπει να γράψουμε το εξής int *number = new int; όπου int number ορίζουμε τον αριθμό και με τον όρο new int κάνουμε δυναμική κατανομή του αριθμού, όπου δημιουργούμε ένα «κελί» μέσα στο οποίο θα τοποθετηθεί ο αριθμός number. Στην συνέχεια για να υπάρχει καλύτερος έλεγχος της μνήμης θα πρέπει κάθε φορά οι μεταβλητές που ορίζουμε να διαγράφονται απο την μνήμη μετά το πέρας της λειτουργίας και αυτό επιτυγχάνεται με την χρήση της εντολής delete number;.
- Πέρα απο την δυναμική κατανομή είναι προτιμότερο σε έναν κώδικα, για να υπάρχει καλύτερη διαχείρηση μνήμης απο το ίδιο πειβάλλον προγραμματισμού, να ορίζουμε τις μεταβλητές με στατική κατανομή. Σε αυτή την ΠΕ ο κώδικας που εφαρμόζεται έχει γραφτεί με τον συγκεκριμένο τρόπο καθώς δεν ήταν επιθυμητό να διαγράφουμε τις μεταβλητές μετά την εκτέλεση του κώδικα.

Πρέπει να σημειωθεί ότι η χρήση της SPI προκαλεί αρκετές καθυστερήσεις στην προσπέλαση δεδομένων και γι’αυτό το λόγο συνιστάται η χρήση της SRAM για την αποθήκευση των δεδομένων.

2.5.4 Διαμόρφωση μονάδων Εισόδου / Εξόδου



Σχήμα 2.22 Διαμόρφωση ακίδων ESP WROOM32

Το ESP32 περιλαμβάνει 38 ακίδες απο τις οποίες η ακίδα 1 και 38 χρησιμοποιούνται ως γείωση GND, η ακίδα 2 με ονομασία 3V3 παρέχει τροφοδοσία της τάξης των 3.3Volts διαμέσου σταθεροποιητή τάσης και η ακίδα 3 με ονομασία EN χρησιμοποιείται για τον έλεγχο του ESP αφού προορίζεται ως ακίδα ενεργοποίησης του σταθεροποιητή τάσης των 3.3V.Αυτό σημαίνει ότι εφαρμόζοντας ένα κουμπί στην ακίδα αυτή μπορούμε να επανεκκινήσουμε το ESP. Παρακάτω θα αναφερόμαστε στις ακίδες με την ονομασία GPIO (General Purpose Input/Output) για λόγους ευκολίας. Θα χωρίσουμε λοιπόν τα GPIO σε διάφορες κατηγορίες ανάλογα με την λειτουργία τους, έτσι ξεκινώντας έχουμε:

- **Ακίδες εισόδου:** GPIO34 ,GPIO35, GPIO36, GPIO39. Αυτές οι ακίδες μπορούν να χρησιμοποιηθούν αποκλειστικά και μόνο ώς είσοδοι .
- **Ακίδες SPI:** GPIO6(SCK/CLK), GPIO7(SDO), GPIO8(SDI), GPIO9(SHD), GPIO10(SWP), GPIO11(CSC/CMD).Οι ακίδες απο 6 έως 11 χρησιμοποιούντα έμμεσα απο το ESP για την λειτουργία της SPI και δεν προτείνονται για χρήση σε εφαρμογές είτε ως είσοδοι ή έξοδοι.
- **Ακίδες επαφής:** GPIO4, GPIO0, GPIO2, GPIO15, GPIO13, GPIO12, GPIO14, GPIO27, GPIO33, GPIO32. Το ESP32 διαθέτει 10 εσωτερικούς χωρητικούς αισθητήρες αφής που μπορούν να ανιχνεύσουν παραλλαγές σε οτιδήποτε έχει ηλεκτρικό φορτίο, όπως το ανθρώπινο δέρμα. Έτσι μπορούν να ανιχνεύσουν παραλλαγές που προκαλούνται όταν αγγίζουμε τα συγκεκριμένα GPIO. Μπορούν επίσης να χρησιμοποιηθούν και για γενική χρήση ως είσοδοι/έξοδοι σε κάποια εφαρμογή.

- **Ακίδες ADC:** GPIO36, GPIO39, GPIO34, GPIO35, GPIO32, GPIO33, GPIO25, GPIO26, GPIO27, GPIO14, GPIO12, GPIO13, GPIO15, GPIO2, GPIO0, GPIO4. Τα κανάλια εισόδου ADC έχουν ανάλυση 12 bit. Αυτό σημαίνει ότι μπορούμε να λάβουμε αναλογικές μετρήσεις που κυμαίνονται από 0 έως 4095, στις οποίες το 0 αντιστοιχεί στο 0V και 4095 στα 3.3V. Έχουμε επίσης τη δυνατότητα να ρυθμίσσουμε την ανάλυση των καναλιών μας διαμέσου του κώδικα, καθώς και το εύρος ADC. Επιπλέον οι ακίδες αυτές δεν έχουν γραμμική συμπεριφορά, δηλαδή θα παρατηρήσουμε ότι η τάση όταν η ακίδα είναι απενεργοποιημένη διακυμαίνεται απο 0 έως 0.1Volts, ενώ όταν είναι ενεργοποιημένη έχουμε διακύμανση απο 3.2 έως 3.3 Volts.
- **Ακίδες DAC:** GPIO25, GPIO26. Τα κανάλια αυτά των 8-bits έχουν την ικανότητα να μετατρέπουν το ψηφιακό σήμα εισόδου σε αναλογικό σήμα εξόδου.
- **Ακίδες RTC (Real Time Clock):** GPIO36, GPIO37, GPIO38, GPIO39, GPIO34, GPIO35, GPIO32, GPIO33, GPIO25, GPIO26, GPIO27, GPIO14, GPIO12, GPIO13, GPIO15, GPIO2, GPIO0, GPIO4. Το ESP παρέχει υποστήριξη Real Time Clock όπου οι ακίδες αυτές μπορούν να χρησιμοποιηθούν απο το ίδιο το ESP όταν αυτό βρίσκεται σε λειτουργία "βαθύ ύπνου". Οι ακίδες δρομολογούνται στο υποσύστημα RTC χαμηλής ισχύος έτσι ώστε να "ξυπνήσουν" το ESP όταν λειτουργεί ο συν-επεξεργαστής Ultra-Low-Power. Η λειτουργία αυτή βασίζεται στην καλύτερη διαχείρηση ισχύος του ESP όταν αυτό τροφοδοτείται αλλά δεν λειτουργεί, έτσι τα GPIO που αναφέρθηκαν μπορούν να χρησιμοποιηθούν ως εξωτερική πηγή αφύπνησης.
- **Ακίδες I2C:** GPIO21(SDA), GPIO22(SCL). Το ESP παρέχει δύο κανάλια I2C για την επικοινωνία των συσκευών που βρίσκονται πάνω στο τυπωμένο κύκλωμα. Το κανάλι 21 είναι η γραμμή δεδομένων μέσα απο την οποία ρέουν τα δεδομένα επικοινωνίας και το κανάλι 22 είναι η γραμμή ρολογιού η οποία είναι υπεύθυνη για τον συγχρονισμό των δεδομένων.

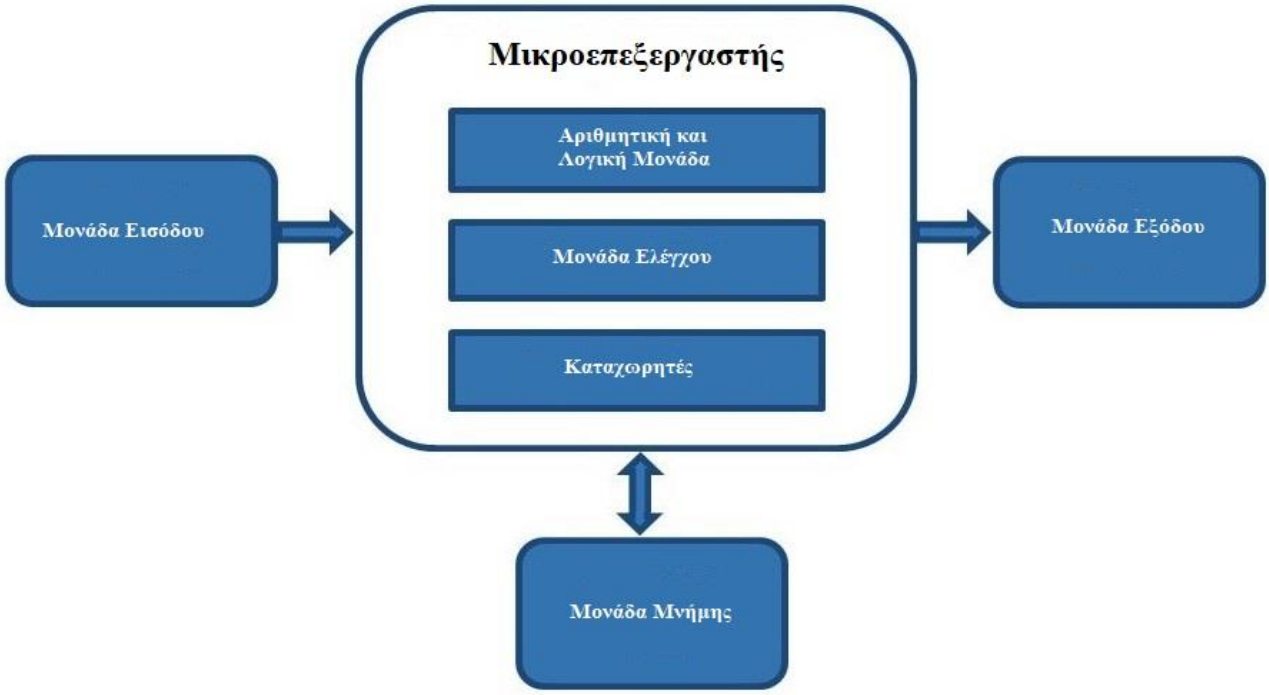
Με βάση τα παραπάνω παρατηρούμε ότι κάθε GPIO έχει πολλαπλές λειτουργίες ανάλογα με την εφαρμογή που επιθυμεί ο χρήστης. Ωστόσο στην παρούσα ΠΕ αναφέραμε τις βασικότερες λειτουργίες που αξιοποιήθηκαν κατα την υλοποίηση της εφαρμογής και για περισσότερες λεπτομέρειες θα πρέπει να αναζητήσουμε μέσα απο το datasheet [11] του ESP-WROOM-32.

2.6 Τρόπος δημιουργίας εφαρμογών με χρήση μικροελεγκτών

Ο πιο διαδεδομένος τρόπος ανάπτυξης εφαρμογών πάνω σε ΕΣ αρχικά είναι η κατάλληλη επιλογή μικροελεγκτή που θα ταιριάζει με τις απαιτήσεις των υλικών, αλλά εξίσου σημαντικό να είναι και συμβατό με το λογισμικό που πρόκειται να χρησιμοποιήσουμε. Για χάριν του παραδείγματός μας ας συμφωνήσουμε ότι ως επιλογή του μικροελεγκτή αποφασίσαμε να χρησιμοποιήσουμε τον ESP32. Στην συνέχεια ξεκινώντας την εφαρμογή, στόχος μας είναι να δημιουργήσουμε ένα περιβάλλον όπου μπορεί να αναπτυχθεί και δοκιμαστεί ο κώδικας της εφαρμογής για το συγκεκριμένο ΕΣ. Η διαδικασία αυτή απαιτεί την εγκατάσταση κάποιας πλατφόρμας πολλαπλής ανάπτυξης, όπου ο κώδικας μπορεί να γραφτεί, να επεξεργαστεί και να φορτωθεί στον μικροελεγκτή. Τέτοιου είδους πλατφόρμα είναι ένα **Ολοκληρωμένο Περιβάλλον Ανάπτυξης (Integrated Development Environment-IDE)** το οποίο περέχει, ένα πρόγραμμα επεξεργασίας κειμένου για την σύνταξη του κώδικα, έναν μεταγλωττιστή, linker script και έναν φορτωτή για την λήψη του μεταγλωττισμένου δυαδικού κώδικα στα σωστά τμήματα φυσικής διεύθυνσης. Η φυσική διεύθυνση είναι μία διεύθυνση μνήμης που αναπαρίσταται με μορφή δυαδικού συστήματος στον δίαυλο διευθύνσεων προκειμένου να επιτρέψει στον δίαυλο δεδομένων να έχει πρόσβαση σε κάποια θέση αποθήκευσης της κύριας μνήμης.Όταν επιλέγουμε κάποιο IDE πρέπει αυτό να μας προσφέρει την καλύτερη απόδοση όσο αναφορά τον κώδικα και την συμβατότητα του περιβάλλοντος με τον ESP32.

2.7 Επίλογος κεφαλαίου 2

Σε αυτό το κεφάλαιο αναφέρθηκε ένα απο τα βασικότερα στοιχεία των ΕΣ που είναι ο μικροεπεξεργαστής (πολλές φορές θα το συναντήσουμε και ως μικροελεγκτής) καθώς και την ραγδαία εξέλιξή του απο την δεκαετία του 90΄ έως και σήμερα. Στην συνέχεια παρατηρήσαμε αναλυτικά τρείς βασικούς μικροεπεξεργαστές που χρησιμοποιήθηκαν στην παρούσα ΠΕ, τον **PIC1F4550**, τον **ATMEGA328-P** και τον **ESP WROOM-32** και συγκεκριμένα παρουσιάσαμε τις οικογένειες, στις οποίες ανήκουν οι μικροεπεξεργαστές, τον τρόπο με τον οποίο οργανώνεται η μνήμη τους, καθώς επίσης την αρχιτεκτονική τους και την διαμόρφωση των ακίδων τους. Κάθε μικροεπεξεργαστής χρησιμοποιείται για συγκεκριμένες εφαρμογές ανάλογα με τις απαιτήσεις του συστήματος καθώς ποικίλουν σε ταχύτητα και αποθηκευτικό χώρο, γι'αυτό οι χρήστες πρέπει να εξοικειωθούν στο πως πρέπει να επιλέγουν ένα κατάλληλο μικροεπεξεργαστή ανάλογα με τις απαιτήσεις του συστήματος. Στοο κεφάλαιο αυτό περιγράφτηκαν τα βασικά μέρη ενός μικροεπεξεργαστή και πως μπορούν να χρησιμοποιηθούν, ανάλογα με την διαμόρφωση των ακίδων του, σε εφαρμογές και γενικά να γίνει αντιληπτή η εξέλιξη των μικροεπεξεργαστών.



Σχήμα 2.23 Βασική δομή του μικροεπεξεργαστή

Κεφάλαιο 3: Υλοποίηση διάταξης εμφάνισης μηνυμάτων με LCD και έλεγχος φωτεινότητας μέσω αισθητήρων φωτός

3.1 Εισαγωγή στο πρότζεκτ

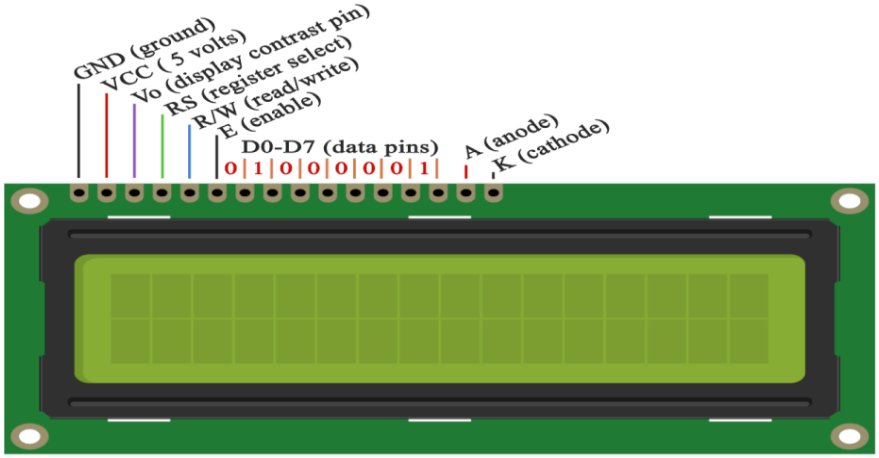
Στην παρούσα ΠΕ στόχος είναι η υλοποίηση και η κατασκευή ενός ΕΣ, στο οποίο ο χρήστης ελέγχει το κείμενο οθόνης LCD (Liquid Crystal Display), το οποίο μετακινείται απο δεξιά πρός τα αριστερά. Επιπλέον το ΕΣ διαθέτει κύκλωμα φωτοαντίστασης το οποίο επιτρέπει την αυτόματη ρύθμιση της φωτεινότητας LCD και διαμέσου ιστοσελίδας HTML (Hyper Text Markup Language) δίνεται στον χρήστη ο πλήρης έλεγχος του ΕΣ, δηλαδή έχει στην διάθεσή του λειτουργίες όπως, επανακίνηση του ΕΣ, έλεγχος του κειμένου και ένδειξη της φωτεινότητας. Σκοπός της υλοποίσης είναι η εξοικείωση με τους μικροεπεξεργαστές και η ενσωμάτωσή τους σε ΕΣ, καθώς επίσης και η εξοικείωση με τα διάφορα υλικά και λογισμικά που χρησιμοποιήθηκαν. Να σημειώσουμε ότι μετά το πέρας της ΠΕ αποκτήθηκαν γνώσεις πάνω στα ΕΣ και στους μικροεπεξεργαστές, δηλαδή ο τρόπος με τον οποίο μπορούμε να ξεχωρίσουμε τα ΕΣ και τους μιροεπεξεργαστές, καθώς και με ποια κριτήρια επιλέγουμε τον κατάλληλο μικροεπεξεργαστή για την εφαρμογή μας. Επιπρόσθετα αποκτήθηκαν γνώσεις σε διάφορες γλώσσες προγραμματισμού, όπως embedded C, για τον προγραμματισμό του ΕΣ, HTML5 και JavaScript που χρησιμοποιήθηκαν για την υλοποίηση της ιστοδελίδας. Τέλος σημαντικό ρόλο αποτέλεσαν τα προγράμματα που χρησιμοποιήθηκαν για την σχεδίαση και την προσομοίωση του ΕΣ.

3.2 Βασικά υλικά που χρησιμοποιήθηκαν

Παρακάτω θα περιγραφτούν τα υλικά που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής ξεκινώντας με την οθόνη LCD.

3.2.1 Οθόνη LCD

Η LCD είναι μία συσκευή εξόδου που μπορεί να ελέγχεται απευθείας απο κάποιον μικροεπεξεργαστή. Αποτελείται από δύο γραμμές των 16 χαρακτήρων η καθεμία και διαθέτει οπίσθιο φωτισμό LED και 16 ακροδέκτες για την σύνδεσή της με τον μικροεπεξεργαστή και συγκεκριμένα το ESP.



Εικόνα 3.2.1 Οθόνη LCD

Όπως φαίνεται στην εικόνα 3.2.1 οι ακροδέκτες της LCD έχουν συγκεκριμένες λειτουργίες που είναι:

- GND: Ο ακροδέκτης χρησιμοποιείται ως γείωση.
- VCC: Η τροφοδοσία της LCD γίνεται διαμέσου αυτού του ακροδέκτη και ανέρχεται στα 5V.
- Vo: Χρησιμοποιείται για τη ρύθμιση της αντίθεσης της οθόνης. Ο ακροδέκτης αυτός συνδέεται στην γείωση μέσω μιας αντίστασης, της οποίας το μέγεθος καθορίζει την αντίθεση της LCD.
- RS (Register Select): Η LCD διαθέτει εσωτερικά δύο καταχωρητές που χρησιμοποιούνται για την αποθήκευση εντολών που απευθύνονται στην οθόνη και για την αποθήκευση δεδομένων που πρόκειται να εμφανιστούν σε αυτή. Μέσω του ακροδέκτη RS ο μικροελεγκτής σηματοδοτεί αν αυτό που στέλνει κάθε φορά στην οθόνη είναι δεδομένα ή εντολές. Συγκερκιμένα αν το RS είναι “0” (LOW) τότε η εισερχόμενη πληροφορία γράφεται στον καταχωρητή εντολών καθώς πρόκειται για εντολή. Αντιθέτως εάν το RS είναι “1” (HIGH) τότε η εισερχόμενη πληροφορία γράφεται στον καταχωρητή δεδομένων αφού πρόκειται για δεδομένο που στάλθηκε απο τον μικροελεγκτή.
- R/W ή RW (Read/Write): Ο συγκεκριμένος ακροδέκτης καθορίζει αν η ενέργεια που θα εκτελεστεί στον καταχωρητή πρόκειται για ανάγνωση ή εγγραφή. Εάν το RW είναι “0” (LOW) τότε πρόκειται να εκτελεστεί εγγραφή στον καταχωρητή, ενώ εαν το RW είναι “1” (HIGH) τότε θα γίνει ανάγνωση από τον καταχωρητή.
- E (Enable): Ο ακροδέκτης αυτός συγχρονίζει την εγγραφή δεδομένων στον κατάλληλο καταχωρητή της οθόνης. Συγκεκριμένα όταν ο μικροελεγκτής θέλει να στείλει δεδομένα στην οθόνη, τότε γράφει τις κατάλληλες τιμές στους 8 ακροδέκτες δεδομένων (D0-D7) και στην συνέχεια στέλνει έναν παλμό πάνω στον ακροδέκτη E. Τα δεδομένα γράφονται στον κατάλληλο καταχωρητή όταν ο ακροδέκτης μεταβεί απο HIGH σε LOW (απο “1” σε “0”).
- D0-D7 (Ακροδέκτες Δεδομένων): Χρησιμοποιούνται για την μετάδοση δεδομένω 8-bit.
- A (Anode): Είναι συνδεδεμένος με την άνοδο του LED που παρέχει τον οπίσθιο φωτισμό στην LCD
- K (Cathode): Είναι συνδεδεμένος με την κάθοδο του LED που παρέχει οπίσθιο φωτισμό.

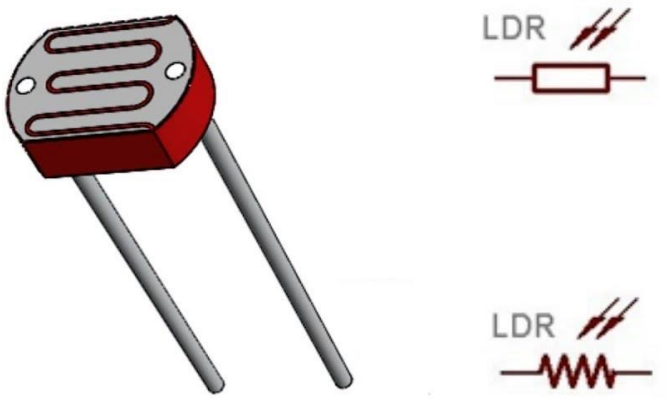
Σημειώνεται ότι υπάρχουν πολλοί κατασκευαστές LCD και στην συγκεκριμένη ΠΕ χρησιμοποιήθηκε η LCD 1602K η οποία λειτουργεί όπως ακριβώς αναφέραμε.

3.2.2 Φωτοαντίσταση (LDR)

Η φωτοαντίσταση είναι μία αντίσταση το μέτρο της οποίας μεταβάλλεται ανάλογα με την ένταση του φωτός που προσπίπτει σε αυτή. Όταν το φώς γίνεται έντονο τότε το μέτρο της αντίστασης μειώνεται, ενώ στο σκοτάδι παίρνει την μέγιστη τιμή της. Η φωτοαντίσταση κατασκευάζεται απο φωτοαγώγιμο υλικό με υψηλή αρχική αντίσταση και όταν το φώς προσπίπτει στην επιφάνειά της τότε η ενέργεια αυτή διεγείρει τα δεσμευμένα ηλεκτρόνια ώστε να διαφύγουν και να μετατραπούν σε ελεύθερα ηλεκτρόνια. Όσο μεγαλύτερη είναι η ενέργεια του φωτός, τόσο μεγαλύτερη η προσλαμβανόμενη ενέργεια και τόσο περισσότερα ελεύθερα ηλεκτρόνια. Με τον τρόπο αυτό η διέλεσυη του ρεύματος γίνεται ευκολότερη και συνεπώς η ωμική αντίσταση μικραίνει.

Με βάση τα υλικά απο τα οποία κατασκευάζεται μία φωτοαντίσταση μπορούμε να διακρίνουμε 2 περιπτώσεις. Τις ενδογενείς που χρησιμοποιούν μη αναδιπλούμενα υλικά όπως πυρίτιο και γερμάνιο και τα φωτόνια που πέφτουν διεγείρουν τα ηλεκτρόνια απο την ζώνη σθένους στην ζώνη αγωγιμότητας, όπως αναφέραμε παραπάνω, με αποτέλεσμα να έχουμε ροή ρεύματος και μικρή τιμή αντίστασης. Η δεύτερη περίπτωση αναφέρεται στις εξωγενείς φωτοαντιστάσεις που κατασκευάζονται απο προσμίξεις και δημιουργούν μία νέα ενεργειακή ζώνη πάνω απο την υπάρχουσα ζώνη σθένους. Αυτό έχει ως αποτέλεσμα η μετάβαση στην ζώνη αγωγημότητας να απαιτεί λιγότερη ενέργεια χάρη στο μικρό ενεργειακό κενό. Ωστόσο οι εξωγενείς φωτοαντιστάσεις έχουν περισσότερη ευαισθησία στις μεταβολλές τους φωτός σε σχέση με τις ενδογενείς. Μία άλλη ενδιαφέρουσα ιδιότητα των LDR είναι η

χρονική καθυστέρηση μεταξύ των μεταβολών της έντασης του φωτός που ονομάζεται ποσοστό ανάκτησης αντίστασης. Η ιδιότητα αυτή αναφέρει ότι χρειάζεται περίπου 10 ms για να πέσει εντελώς η αντίσταση όταν εφαρμόζεται φώς μετά απο σκοτάδι, ενώ μπορεί να χρειαστεί έως και 1 sec για να αυξηθεί η αντίσταση στην αρχική τιμή της μετά την πλήρη αφαίρεση του φωτός. Για το λόγο αυτό το LDR δεν μπορεί να χρησιμοποιηθεί σε εφαρμογές που απαιτούν ακρίβεια στην διακύμανση της αντίστασης ανάλογα με την ένταση του φωτός.



Εικόνα 3.2.2 Αναπαράσταση φωτοαντίστασης (LDR)

3.2.3 Ποτενσιόμετρο (POT)

Είναι ουσιαστικά μία αναλογική αντίσταση με τρείς ακροδέκτες που παρέχει μία μεταβλητή τιμή αντίστασης. Είναι παθητικές συσκευές, που σημαίνει ότι δεν απαιτούν τροφοδοσία ρεύματος ή πρόσθετα κυκλώματα για να εκτελέσουν τη βασική τους λειτουργία. Συγκεκριμένα αποτελούνται απο τρείς ακροδέκτες, οι δύο αποτελούν τα άκρα της αντίστασης ενώ ο μεσαίος ακροδέκτης χρησιμοποιείται για την ολίσθηση της τιμής.

Εικόνα 3.2.3 Αναλογικό περιστροφικό ποτενσιόμετρο

3.2.4 Πυκνωτές

Οι πυκνωτές είναι στοιχεία που έχουν την ικανότητα να αποθηκεύουν ενέργεια με την μορφή ηλεκτρικού φορτίου, που παράγει μία πιθανή διαφορά στις πλάκες τους. Λειτουργούν δηλαδή σαν επαναφορτιζόμενες μπαταρίες. Υπάρχουν πολλά είδη πυκνωτών που μπορούν να χρησιμοποιηθούν σε κυκλώματα συντονισμού έως και σε κυκλώματα για την διόρθωσης συντελεστή ισχύος, ωστόσο όλοι πραγματοποιούν την ίδια διαδικασία που είναι να αποθηκεύουν ενέργεια. Ένας πυκνωτής αποτελείται απο δύο ή περισσότερες αγώγιμες πλάκες που δεν αγγίζουν η μία την άλλη, αλλά διαχωρίζονται ηλεκτρικά είτε από τον αέρα είτε από κάποια άλλη μορφή μονωτικού υλικού. Το μονωτικό αυτό στρώμα θα το συναντήσουμε και ώς **διηλεκτρικό**. Λόγω του διηλεκτρικού στρώματος το DC ρεύμα δεν μπορεί να ρέει μέσω του πυκνωτή καθώς το εμποδίζει επιτρέποντας όμως την δημιουργία τάσης στις πλακέτες με την μορφή ηλεκτρικού φορτίου.

Υπάρχουν πολλά είδη πυκνωτών αλλά στην παρούσα ΠΕ χρησιμοποιήθηκαν **κεραμικοί** και **ηλεκτρολυτικοί** πυκνωτές. Στην πρώτη κατηγορία χρησιμοποιούν ως διηλεκτρικό κεραμικό υλικό. Οι τιμές τους κυμαίνονται μεταξύ 1nF και 100uF και δεν είναι πολωμένοι, δηλαδή δεν έχουν θετικό και αρνητικό πόλο οπότε μπορούν να συνδεθούν σε κυκλώματα AC. Έχουν μεγάλη απόκριση συχνοτήτων λόγω χαμηλής αντίστασης και επαγωγής που δημιουργούνται απο παρασιτικά φαινόμενα.

Εικόνα 3.2.4 Κεραμικοί πυκνωτές

Σε αντίθεση με τους κεραμικούς πυκνωτές οι ηλεκτρολυτικοί χρησιμοποιούν έναν ηλεκτρολύτη για την παροχή μεγαλύτερων τιμών χωρητικότητας. Είναι κατασκευασμένοι απο ταντάλιο ή αλουμίνιο και έχουν εύρος χωρητικότητας απο 1uF έως και 47mF, με τάση λειτουργίας έως μερικές εκατοντάδες DC Volt. Ιδιέταιρη προσοχή πρέπει να δώσουμε στην σύνδεση ηλεκτρολυτικών πυκνωτών με το κύκλωμα καθώς αυτού του είδους πυκνωτές είναι πολωμένοι, δηλαδή διαθέτουν θετικό και αρνητικό πόλο, συνεπώς είναι συμβατοί μόνο σε κυκλώματα DC που πρέπει να συνδεθούν σωστά. Στα AC κυκλώματα δεν είναι δυνατόν να χρησιμοποιήσουμε ηλεκτρολυτικούς πυκνωτές λόγω της πόλωσής τους καθώς το AC σήμα εισόδου αλλάζει συνεχώς την πόλωση της πηγής με αποτέλεσμα οι ηλεκτρολυτικοί πυκνωτές να καταστραφούν. Τα μειωνεκτήματα αυτών των πυκνωτών είναι τα μεγάλα ρεύματα διαρροής, οι ανοχές των τιμών τους που ανέρχεται στο 20% της αρχικής τιμής τους, η ισοδύναμη αντίσταση σειράς και ο περιορισμένος χρόνος ζωής τους. Οι ηλεκτρολυτικοί πυκνωτές χρησιμοποιούνται σε εφαρμογές που απαιτείται μεγάλη χωρητικότητα όπως για παράδειγμα στο φιλτράρισμα τροφοδοτικών για την απομάκρυνση του AC σήματος, για την εξομάλυνση της εισόδου-εξόδου καθώς επίσης χρησιμοποιούνται και ως φίλτρα χαμηλής διέλευσης για σήματα DC.

Εικόνα 3.2.5 Ηλεκτρολυτικός πυκνωτής

3.2.5 Ταλαντωτής κρυστάλλου

Ένας ταλαντωτής κρυστάλλου είναι ένα ηλεκτρονικό κύκλωμα ταλαντωτών που χρησιμοποιείται για την μηχανική αντήχηση ενός δονούμενου πιεζοηλεκτρικού κρυστάλλου. Οι ταλαντωτές αυτοί δημιουργούν ένα ηλεκτρικό σήμα με δεδομένη συχνότητα και χρησιμοποιούνται συνήθως για την παρακολούθηση χρόνου σε ψηφιακά κυκλώματα καθώς παρέχουν σταθερό σήμα ρολογιού και για την σταθεροποίηση συχνοτήτων σε ραδιοπομπούς και δέκτες. Ο ταλαντωτής αυτός λειτουργεί σύμφωνα με την αρχή του αντίστροφου πιεζοηλεκτρικού φαινομένου δηλαδή, το εφαρμοζόμενο ηλεκτρικό πεδίο θα προκαλέσει μηχανική παραμόρφωση σε ορισμένα υλικά. Με αυτό τον τρόπο χρησιμοποιεί το μηχανικό συντονισμό του δονούμενου κρυστάλλου, το οποίο κατασκευάζεται απο πιεζοηλεκτρικό υλικό για την δημιουργία ηλεκτρικού σήματος με συγκεκριμένη συχνότητα.

Εικόνα 3.2.6 Ταλαντωτής κρυστάλλου με συχνότητα στα 16MHz

3.2.6 Αντιστάσεις

Η αντίσταση είναι ένα παθητικό ηλεκτρικό στοιχείο που δημιουργεί αντίσταση στην ροή του ηλεκτρικού ρεύματος και βρισκονται σχεδόν σε όλα τα ηλεκτρονικά κυκλώματα. Μονάδα μέτρησης της αντίστασης είναι το Ohm (προφέρεται Ώμ) όπου ένα Ohm είναι η αντίσταση που εμφανίζεται όταν ένα ρεύμα ενός Αμπέρ (Ampere-A) περνάει απο μία αντίσταση με πτώση τάσης στα άκρα της ίση με ένα Βόλτ (Volt-V). Σύμφωνα με τον νόμο του Ohm: $\mathbf{R} = \frac{\mathbf{V}}{\mathbf{I}}$ **(1)** το ρεύμα είναι ανάλογο με την τάση στα άκρα των ακροδεκτών.

Εικόνα 3.2.7 Αναπαράσταση και συμβολισμός αντίστασης

Απο την Εικόνα 3.2.7 παρατηρούμε τέσσερα χρώματα επάνω στην αντίσταση. Τα τρία πρώτα χρώματα (διαβάζοντα πάντα απο δεξιά προς τα αριστερά) είναι τιμή της αντίστασης ενώ το τελευταίο χρώμα (Χρυσό) είναι η ανοχή της. Στο εμπόριο υπάρχουν και αντιστάσεις με τρία ή και πέντε χρώματα που προορίζονται για απλές εφαρμογές και για εφαρμογές που απαιτούν μεγαλύτερη ακρίβεια αντίστοιχα. Παρακάτω απεικονίζεται ο χρωματικός χώδικας των αντιστάσεων που είναι ένα απο τα σημαντικότερα εργαλεία ενός ηλεκτρονικού μηχανικού.

Χρώμα	1η λωρίδα	2η λωρίδα	3η λωρίδα	4η λωρίδα Ανοχή	θερμικός συντελεστής
Μαύρο	0	0	$\times 10^0$		
Καφέ	1	1	$\times 10^1$	±1% (F)	100 ppm
Κόκκινο	2	2	$\times 10^2$	±2% (G)	50 ppm
Πορτοκαλί	3	3	$\times 10^3$		15 ppm
Κίτρινο	4	4	$\times 10^4$		25 ppm
Πράσινο	5	5	$\times 10^5$	±0.5% (D)	
Μπλε	6	6	$\times 10^6$	±0.25% (C)	
Μωβ	7	7	$\times 10^7$	±0.1% (B)	
Γκρι	8	8	$\times 10^8$	±0.05% (A)	
Λευκό	9	9	$\times 10^9$		
Χρυσαφί			$\times 0.1$	±5% (J)	
Ασημί			$\times 0.01$	±10% (K)	

Σχήμα 3.2.8 Χρωματικός κώδικας αντιστάσεων

3.2.7 Φωτοδίοδος LED

Είναι μία δίοδος εκομπής φωτός με 2 ακρόδέκτες που συμβολίζουν την άνοδο (+) και την κάθοδο (-) της διόδου. Στην συγκεκριμένη εφαρμογή χρησιμοποιήθηκε ως ένδειξη παροχής τροφοδοσίας και γενικότερα σε ηλεκτρονικές εφαρμογές τα LED χρησιμοποιούνται συνήθως ως στοιχεία ένδειξης. Στην παρακάτω Εικόνα 3.2.9 οι δίοδοι εκπομπής φωτός φιλοξενούνται στο εσωτερικό του περιβλήματος, το οποίο ποικίλει σε χρώματα. Το χρώμα του φωτός που θα εκπέμπεται απο το LED εξαρτάται απο το χρώμα του περιβλήματος και άρα ένα κόκκινο περίβλημα θα δημιουργεί ένα κόκκινο φώς. Τέλος να πρέπει να σημειωθεί ότι σε ένα LED ο μεγαλύτερος σε μήκος ακροδέκτης συμβολίζει την άνοδο (+) της διόδου.

Εικόνα 3.2.9 Δίοδοι εκπομπής φωτός LED

3.2.8 PIC18F4550

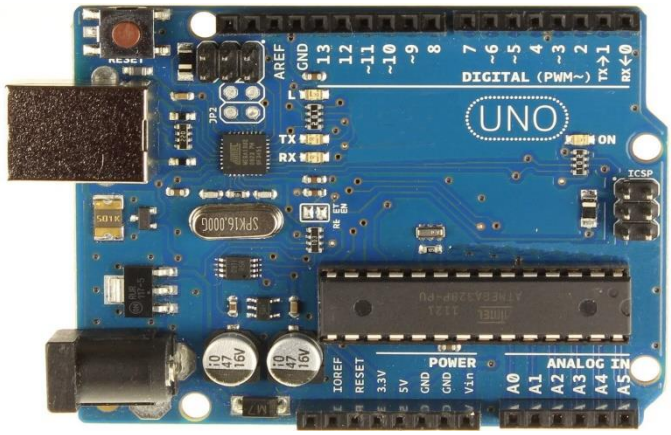
Στην συγκεκριμένη εφαρμογή χρησιμοποιήθηκαν 3 διαφορετικοί μικροελεγκτές με σκοπό την παρατήρηση σφαλμάτων και βελτιώσεων. Αρχικά η υλοποίηση έγινε με τον PIC18F4550 ο οποίος αναλύθηκε στην υποενότητα 2.3.1.

Εικόνα 3.2.10 PIC18F4550 από την MicroChip

3.2.9 Arduino Uno

Το Arduino Uno είναι ένα κύκλωμα που βασίζεται στον μικροελεγκτή ATMEGA328P που περιγράφτηκε στην υποενότητα 2.4.1. Το κύκλωμα αυτό διαθέτει 14 ψηφιακούς ακροδέκτες εισόδου/εξόδου εκ των οποίων οι 6 μπορούν να χρησιμοποιηθούν ως έξοδοι PWM, 6 αναλογικές

εισόδους, έναν κρύσταλλο χαλαζίας με συχνότητα σήματος στα 16MHz, μία σύνδεση USB για την παροχή τροφοδοσίας και ένα κουμπί επαναφοράς. Το Arduino Uno χρησιμοποιήθηκε για το χαμηλό κόστος αγοράς του, την συμβατότητα του λογισμικού του σε περιβάλλον Windows 10, την απλότητα του περιβάλλοντος ανάπτυξης και το επεκτάσιμο λογισμικού ανοιχτού κώδικα που παρέχει πλήθος βιβλιοθηκών C/C++ που μπορούν να εφαρμοστούν σε διάφορες εφαρμογές.



Εικόνα 3.2.11 Arduino Uno R3

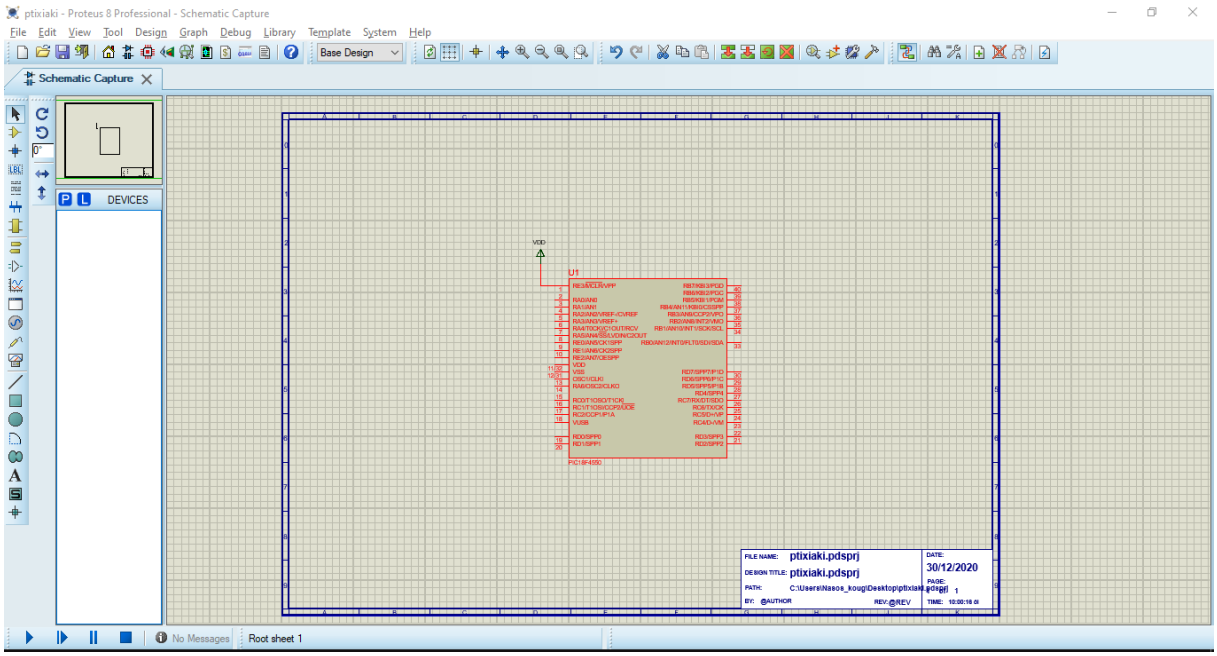
3.2.10 ESP32

Το ESP32 είναι ένα σύστημα χαμηλού κόστους και χαμηλής κατανάλωσης που βασίζεται στον μικροελεγκτή Tensilica Xtensa LX6 που μελετήσαμε στην υποενότητα 2.5. Το σύστημα αυτό χρησιμοποιείται κυρίως για το ενσωματωμένο Wi-Fi και για την δύναμη που παρέχει ο μικροελεγκτής αυτός καθώς έχει μεγαλύτερη ταχύτητα επεξεργασίας σε σχέση με τους άλλους δύο μικροελεγκτές. Με το σύστημα αυτό δημιουργήθηκε το τελικό προϊόν της εφαρμογής αφού ήταν επιθυμητή η χρήση του Wi-Fi για την δημιουργία ιστοσελίδας, απο την οποία ο χρήστης θα έχει τον πλήρη έλεγχο λειτουργίας της εφαρμογής.Κάτι που ο ATMEGA328P και PIC18F4550 δεν διαθέτουν. Η σχεδίαση του συστήματος αυτού θα αναλυθεί στην ενότητα 3.5.

Εικόνα 3.2.12 Σύστημα ESP32

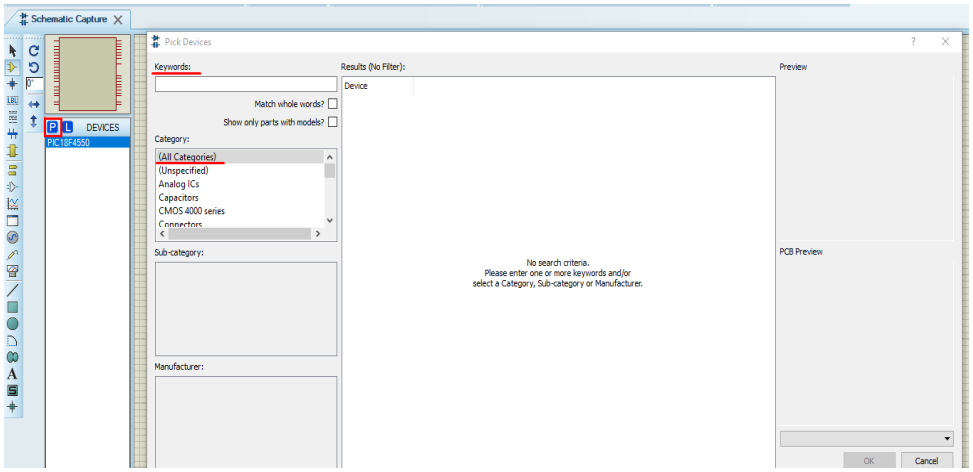
3.3 Σχεδίαση και κατασκευή με τον PIC18F4550

Στο αρχικό στάδιο της εφαρμογής χρησιμοποιήθηκε ο μικροελεγκτής PIC18F4550 για τον οποίο σχεδιάστηκε ηλεκτρονική πλακέτα και έγινε προσομοίωση του κώδικα με σκοπό να παρατηρήσουμε εάν ήταν δυνατό να υλοποιηθεί μία τέτοιου είδους εφαρμογή. Η σχεδίαση και προσομοίωση της εφαρμογής έγινε στο περιβάλλον **Proteus Design Suite** στο οποίο έπρεπε πρώτα να γίνει το σχηματικό. Τέλος αφού γίνει επαλήθευση των επιλογών απο το παράθυρο που θα εμφανιστεί, πατάμε Finish. Αφού ολοκληρωθεί η διαδικασία δημιουργίας νέου πρότζεκτ, το Proteus θα εμφανίσει το σχηματικό με τον μικροελεγκτή έτοιμο χωρίς να χρειαστεί να τον αναζητήσουμε και να τον τοποθετήσουμε, οπότε το επόμενο βήμα είναι η δημιουργία του σχηματικού (Εικόνα 3.3.1).



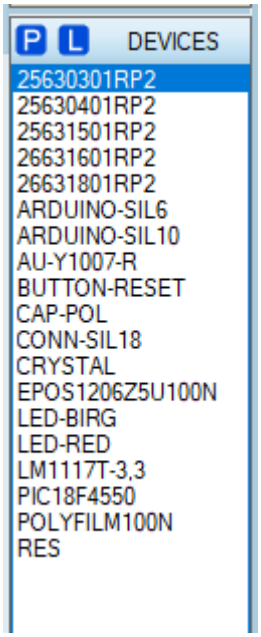
Εικόνα 3.3.1 Σχηματικό Proteus

Για την ανάπτυξη του σχηματικου χρειαζόμαστε όλα τα υλικά που απαιτούνται για την εφαρμογή δηλαδή, πυκνωτές, αντιστάσεις, LED, connectors, θύρα USB, ταλαντωτή κρυστάλλου, κουμπιά (buttons) και έναν ρυθμιστή τάσης. Όλα αυτά τα υλικά υπάρχουν στην βιβλιοθήκη των υλικών όπως φαίνεται στην Εικόνα 3.3.2



Εικόνα 3.3.2 Πίνακας αναζήτησης υλικών

Ο πίνακας με τα υλικά που χρησιμοποιήθηκαν είναι αυτός στην Εικόνα 3.3.3. Στην συνέχεια έγιναν οι κατάλληλες συνδέσεις στο σχηματικό ώστε να λειτουργεί σωστά το κύκλωμά. Για να επιτευχθεί, αυτό μέσα απο το datasheet [9] του μικροελεγκτή PIC18F4550 βρέθηκαν οι τις κατάλληλες συνδέσεις καθώς επίσης και οι σωστές τιμές των πυκνωτών και των αντιστάσεων. Αρχικά για τον ταλαντωτή κρυστάλλου των 20MHz χρησιμοποιήθηκε με βάση την τιμή δύο πυκνωτών σε κατάλληλη συνδεσμολογία με τον ταλαντωτή κρυστάλλου, όπως φαίνεται στο Σχήμα 3.3.4. Συγκεκριμένα για τον κρύσταλλο των 20MHz πρέπει να επιλεχθούν δύο πυκνωτές της τάξης των 15 pF σε αντιπαράλληλη σύνδεση με τον κρύσταλλο. Οι πυκνωτές είναι εκεί για να συντονιστούν με την κρυσταλλική επαγωγή και να προκαλέσουν την ταλάντωση του κρυστάλλου.



Εικόνα 3.3.3 Πίνακας υλικών

Osc Type	Crystal Freq	Typical Capacitor Values Tested:	
		C1	C2
XT	4 MHz	27 pF	27 pF
HS	4 MHz	27 pF	27 pF
	8 MHz	22 pF	22 pF
	20 MHz	15 pF	15 pF

Σχήμα 3.3.4 Πίνακας επιλογής πυκνωτών κρυστάλλου

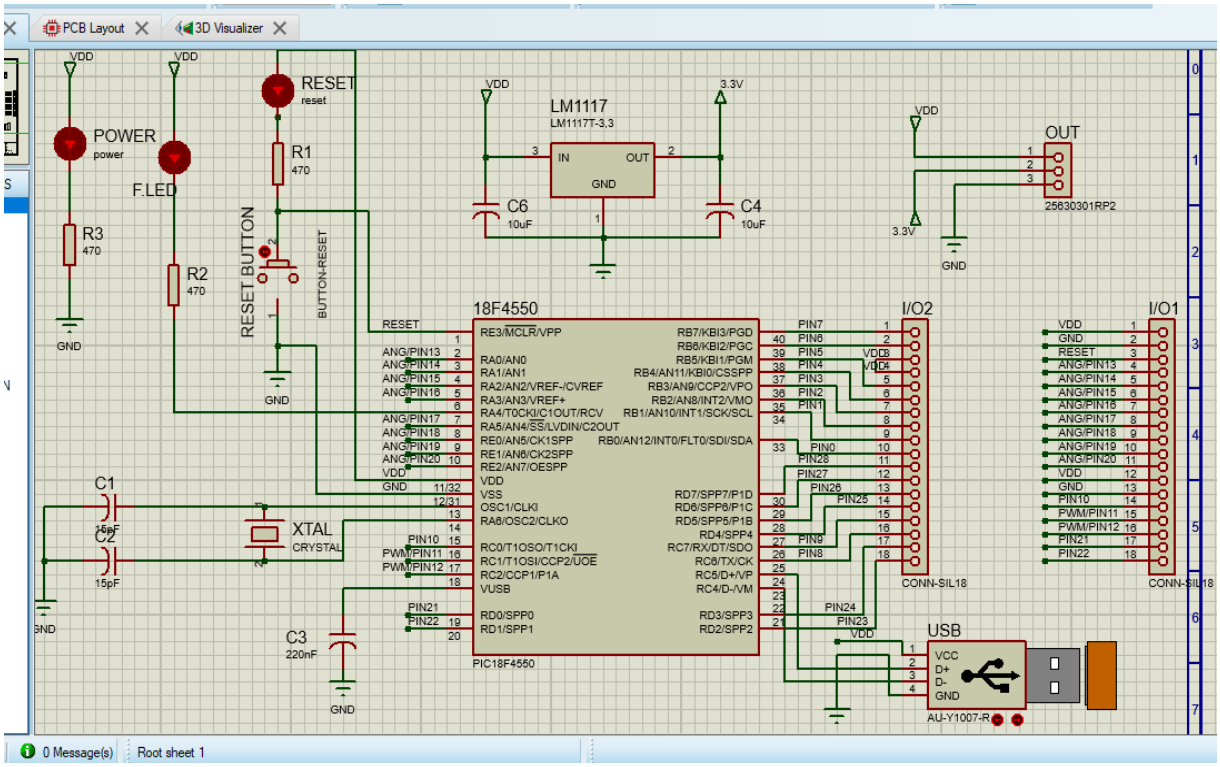
3.3.1 Σχεδίαση σχηματικού με τον PIC18F4550

Η συνδεσμολογία του κυκλώματος πρέπει να είναι αυτή στο Σχήμα 3.3.5. Χρησιμοποιήθηκαν 3 LED ως ενδείξεις, ένα για την ένδειξη ότι στο κύκλωμα παρέχεται τροφοδοσία και έχει ονομασία POWER, ένα LED με ονομασία RESET για την ένδειξη ότι το κύκλωμα βρίσκεται σε κατάσταση επανακκίνησης και τέλος ένα LED με ονομασία F.LED το οποίο χρησιμοποιείται ως ένδειξης προγραμματισμού. Στο κάθε LED τοποθετήθηκε σε σειρά μία αντίσταση των 470Ω με σκοπό την προστασία τους από υπερτάσεις και υπερ-ρεύματα. Ιδιαίτερα σηματνικό για το LED RESET είναι το κουμπί που εφαρμόστηκε και το οποίο κατα το πάτημά του θέτει το κύκλωμα σε λειτουργία επανακκίνησης. Η άνοδος του LED RESET εφαρμόζεται στον **ακροδέκτη 11** με ονομασία VDD, στον οποίο πρέπει να δωθεί κατάλληλη τροφοδοσία για να λειτουργήσει ο μικροελεγκτής, και ο **ακροδέκτης 1** με ονομασία $\overline{MCLR}$, συνδέεται ανάμεσα στην αντίσταση και ένα κουμπί το οποίο μετά το πάτημά του θα κλείσει το κύκλωμα RESET και θα τροφοδοτήσει τον **ακροδέκτη 1** με αποτέλεσμα να θέσει τον μικροελεγκτή σε

λειτουργία επανακκίνησης. Τέλος το F.LED προγραμματίστηκε μέσα απο τον κώδικα ώστε όταν ολοκληρωθεί ο γίνει εγγραφή του κώδικα στον μικροελεγκτή, αυτό να ανάψει ως ένδειξη ολοκλήρωσης προγραμματισμού.

Στην συνέχεια χωρίσαμε τον μικροελεγκτή σε 5 πόρτες με ονομασία RA, RB, RC, RD και RE με σκοπό τις πόρτες εξόδου να τις συνδέσουμε με συνδετήρες, όπου ο χρήστης θα μπορεί να συνδέσει με άνεση στις πόρτες αυτές τα περιφερειακά της εφαρμογής του. Έτσι λοιπόν οι πόρτες για αυτό τον σκοπό είναι οι RA0-RA5 (με εξαίρεση την πόρτα RA4 που την χρησιμοποιήσαμε ως ένδειξη προγραμματισμού), RB0-RB7, RC0-RC2 που χρησιμοποιούνται ώς πόρτες εξόδου για εφαρμογές που απαιτούν PWM, RC6 και RC7 ως αναλογικές πόρτες εξόδου, RD0-RD7 και RE0-RE2.

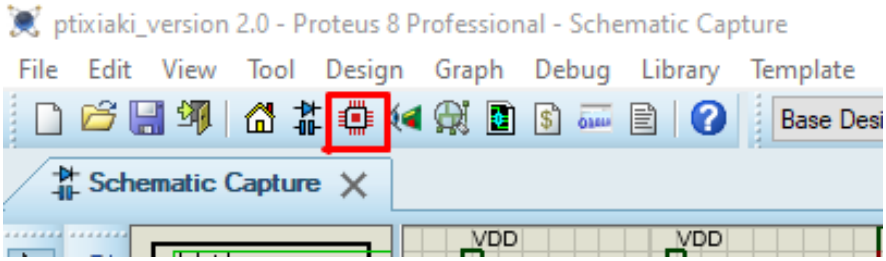
Ο PIC18F4550 έχει κάποιους ακροδέκτες που προορίζονται για συγκεκριμένες λειτουργίες και δεν μπορούν να χρησιμοποιηθούν ως πόρτες εισόδου/εξόδου γενικού σκοπού όπως στην περίπτωση με τον **ακροδέκτη 1** που χρησιμοποιείται για λειτουργία επανακκίνησης. Ξεκινώντας απο τους ακροδέκτες **OSC1** και **RA6** απο το Datasheet [9] του μικροελεγκτή συμπαιρένουμε ότι σε αυτούς τους δύο ακροδέκτες συνδέεται ο ταλαντωτής κρυστάλλου για τον συγχρονισμό του PIC18F4550. Στην συνέχεια ο ακροδέκτης **VUSB** που λειτουργεί ως έξοδος ενός εσωτερικού ρυθμιστή τάσης USB στα 3.3V δεν χρησιμοποιήθηκε στην εφαρμογή και γι'αυτό τον λόγο τον γειώνουμε διαμέσου ενός πυκνωτή ώστε να τον προστατέψουμε απο υπερτάσεις και υπερ-ρεύματα. Όσο αναφορά το USB, που χρησιμοποιήθηκε ως δίαυλος μεταφοράς δεδομένων απο τον ηλεκτρονικό υπολογιστή στον μικροελεγκτή, συνδέουμε τους δύο μεσαίους ακροδέκτες του USB (D+ και D-) με τους ακροδέκτες **RC5/D+** και **RC4/D-** που χρησιμοποιούνται ώς είσοδοι/έξοδοι μεταφοράς δεδομένων και οι άλλοι δύο ακροδέκτες (VCC και GND) του USB συνδέονται με την τροφοδοσία των 5 Volt και την γείωση αντίστοιχα. Τέλος εφαρμόστηκε ένας ρυθμιστής τάσης **LM1117T-3.3V** όπου μετατρέπει τα 5 Volt εισόδου σε 3.3 Volt εξόδου με σκοπό να προσφέρουμε στον κύκλωμα έξοδο που θα μπορεί να τροφοδοτεί υλικά που απαιτούν χαμηλότερη τροφοδοσία. Το **LM1117T** έχει 3 ακροδέτες IN, OUT και GND, όπου ο ακροδέκτης IN συνδέθηκε στην τροφοδοσία των 5V, αφού αυτή είναι η τάση που θέλουμε να υποβιβάσουμε στα 3.3V και στον ακροδέκτη OUT με κατάλληλη συνδεσμολογία δύο πυκνωτών (έναν ηλεκτρολυτικό 100nF και έναν κεραμικό 47uF) παίρνουμε την επιθυμητή τάση των 3.3V. Ο ακροδέκτης GND συνδέεται στην γείωση. Πρέπει να σημειωθεί ότι η συνδεσμολογία και οι τιμές των πυκνωτών γίνονται με βάση το datasheet [10] του LM1117T. Τέλος ο συνδετήρας με τις 3 εξόδους και ονομασία OUT συνδέεται με την τάση τροφοδοσίας των 5V για την παροχή τάσης εξόδου στα 5V, με την τάση εξόδου του υποβιβαστή τάσης LM1117T για την παροχή τροφοδοσίας της τάξης των 3.3V και με την κοινή γείωση του κυκλώματος για την παροχή ακροδέκτη γείωσης.



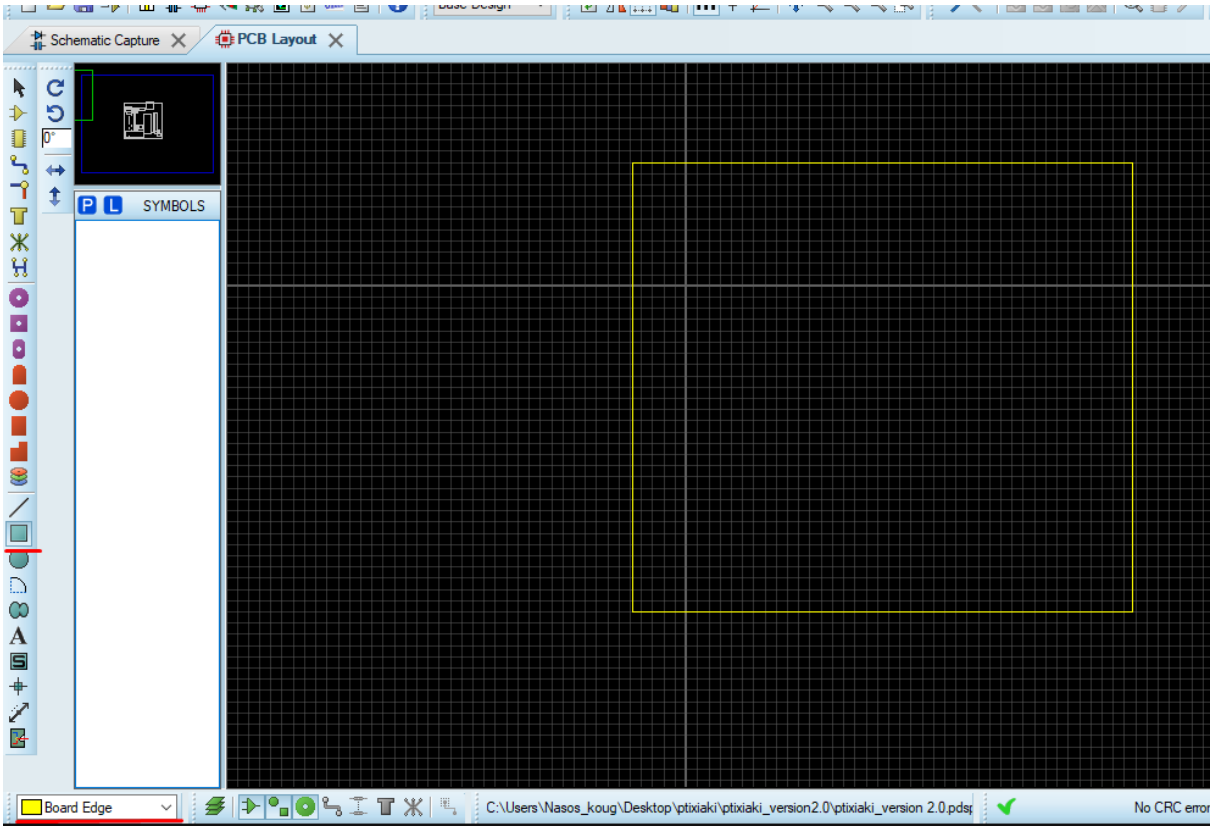
Εικόνα 3.3.5 Σχηματικό κυκλώματος με τον PIC18F4550

3.3.2 Σχεδίαση PCB με τον PIC184F4550

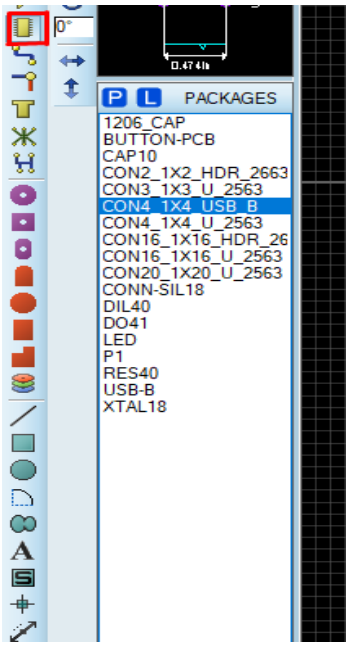
Σε αυτή την υποενότητα θα μελετηθεί ο τρόπος με τον οποίο σχεδιάζουμε σωστά ένα PCB και συγκεκριμένα το PCB του σχηματικού που έγινε στην υποενότητα 3.3.1. Η σχεδίαση του PCB έγινε επίσης στο περιβάλλον Proteus στην επιλογή PCB Layout όπως φαίνεται στην Εικόνα 3.3.6. Το επόμενο βήμα είναι να τοποθετθούν όλα τα υλικά μέσα στο κίτρινο πλαίσιο που δημιουργήθηκε σε κατάλληλη διάταξη ώστε να μην καταλαμβάνουν μεγάλο χώρο άσκοπα.



Εικόνα 3.3.6



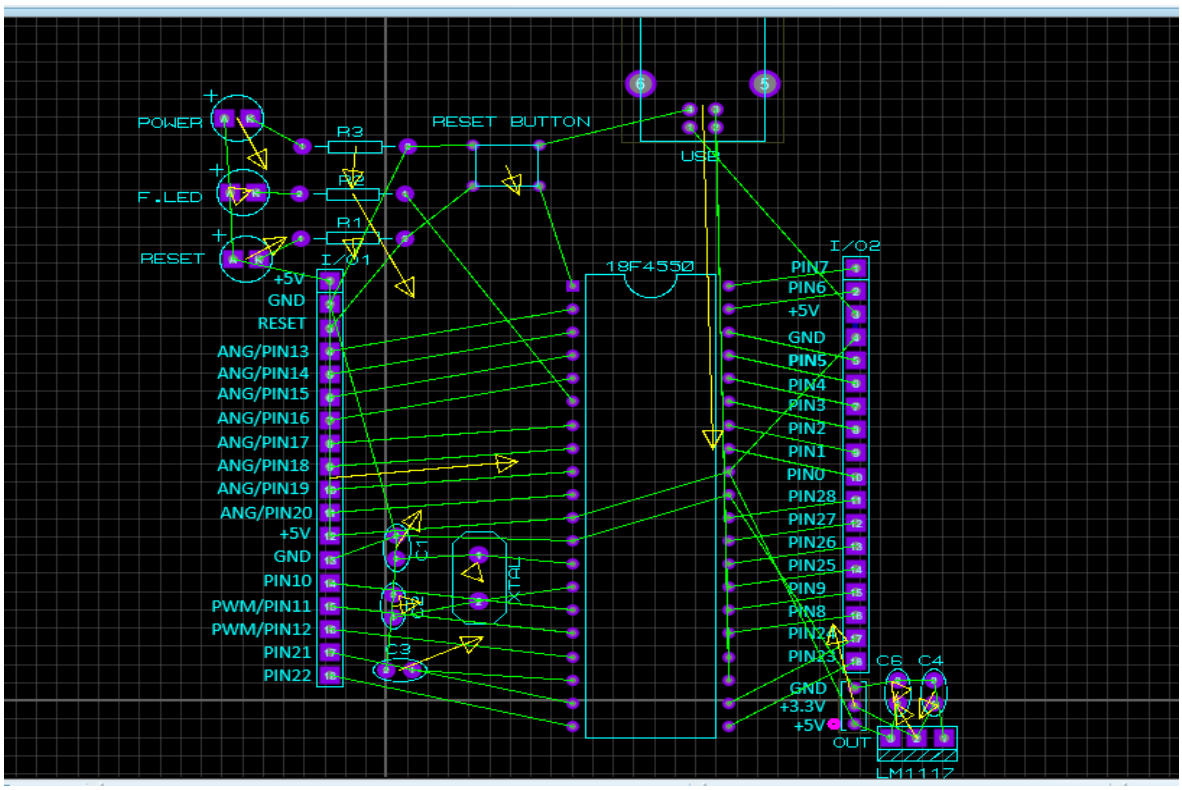
Εικόνα 3.3.7 Δημιουργία πλαισίου πλακέτας



Εικόνα 3.3.8 Πίνακας υλικών PCB

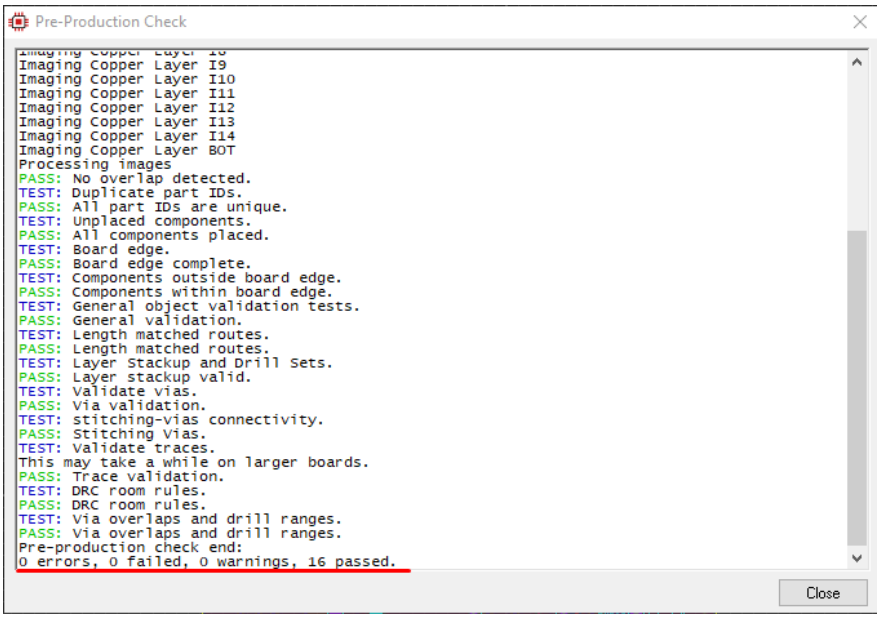
Μόλις τοποθετηθούν τα υλικά στο πλαίσιο, όπως φάινεται στην εικόνα 3.3.7, πρέπει να γίνουν οι κατάλληλες συνδέσεις των υλικών μεταξύ τους. Για την διασύνδεση τους είναι απαραίτητο να εφαρμοσοτύν κάποιοι κανόνες που ισχύουν σε όλες τις σχεδιάσεις PCB και είναι οι εξής:

- Οι αγωγοί που σχεδιάζονται δεν πρέπει να συμπίπτουν μεταξύ τους. Σε αντίθετη περίπτωση αυτό θεωρείται ως βραχυκύκλωμα.
- Οι αγωγοί κατα των πλείστων πρέπει να τοποθετηθούν στην κάτω πλευρά της πλακέτας. Όταν δεν υπάρχει διαθέσιμος χώρος στην κάτω πλευρά τότε για λόγους άνεσης και βελτίωσεις σχεδιάζουμε και στην πάνω πλευρά
- Το πάχος των αγωγών για λόγους ασφαλείας απο τις υπερτάσεις και τα υπερ-ρεύματα πρέπει να είναι τουλάχιστον 0.05cm και οι αγωγοί να απέχουν μεταξύ τους απόσταση τουλάχιστον 7mm.
- Κατά την σχεδίαση απαιραίτητη προϋπόθεση είναι η δημιουργία αμβλείας γωνίας του αγωγού καθώς οποιαδήποτε άλλη μορφή γωνίας μπορεί να δημιουργήσει βραχυκύκλωμα με την μεταπήδηση των ηλεκτρονίων απο την μία μεριά στην άλλη αφού τα ηλεκτρόνια ψάχνουν πάντα τον πιο εύκολο και γρήγορο δρόμο.
- Οι απόσταση μεταξύ των υλικών πρέπει να είναι τουλάχιστον 7mm για λόγους απομόνωσης.
- Η πλακέτα θα πρέπει να απομονώνει την πάνω πλευρά απο την κάτω. Το πάχος της απομόνωσης και η απόσταση των κενών υπολογίζεται αυτόματα απο το περιβάλλον Proteus.
- Τέλος μπορούμε να τοποθετήσουμε προαιρετικά τρύπες στα τέσσερα άκρα του PCB σε περίπτωση που θέλουμε να βιδώσουμε την πλακέτα σε κάποια επιφάνεια και να τοποθετήσουμε περιμετρικά της πλακέτας, τρύπες που δημιουργούν επικοινωνία μεταξύ των γειώσεων των δύο πλευρών. Αυτό γίνεται για να φτιάξουμε μία κοινή γείωση. Οι τρύπες που έγιναν στα άκρα της πλακέτας έχουν διάμετρο 3.5mm.

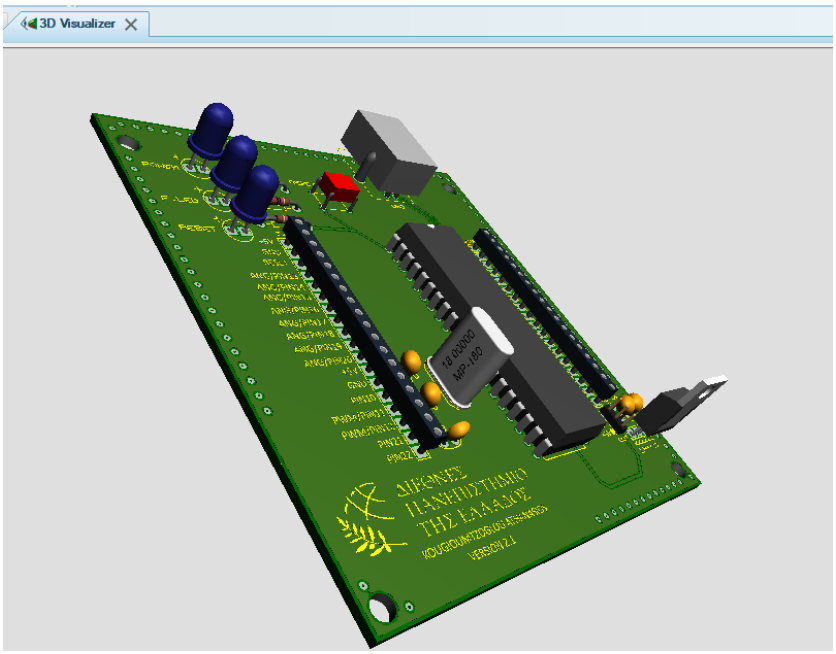


Εικόνα 3.3.9 Τοποθέτηση υλικών

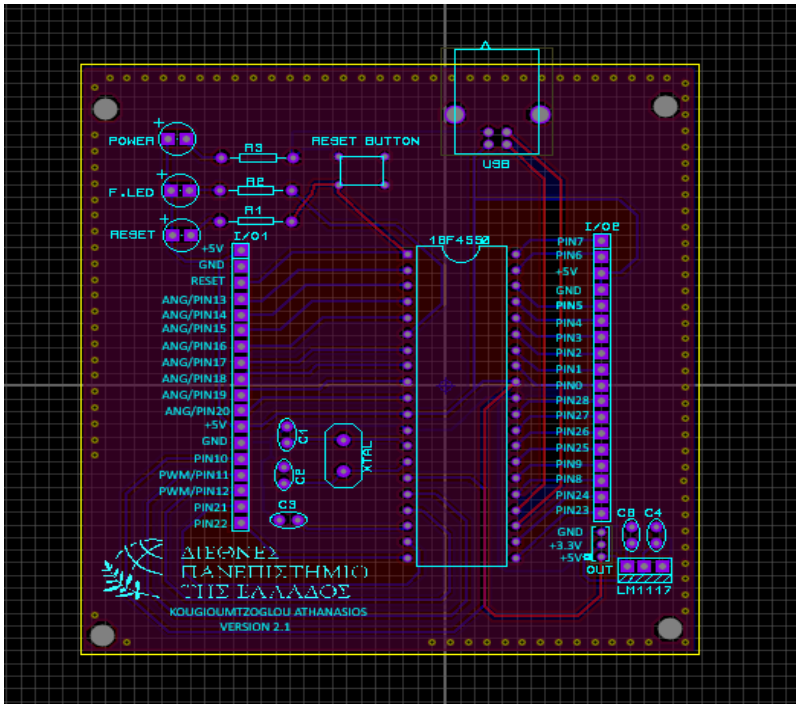
Αφού ολοκληρωθεί η σχεδίαση του PCB και έχουμε το τελικό αποτέλεσμα όπως φαίνεται στην Εικόνα 3.3.9 πρέπει να γίνει έλεγχος του PCB πριν στείλουμε το σχέδιό μας για εκτύπωση. Απο την πάνω μπάρα επιλέγουμε Output > Pre-Production Check και το Proteus κάνει αυτόματα τον έλεγχο σφαλμάτων. Σε περίπτωση που δεν υπάρχουν σφάλματα θα εμφανιστεί στο παράθυρο **0 errors** – **0 failed** – **0 warnings**. Στην συνέχεια μπορεί να γίνει εμφάνιση της πλακέτας σε τρισδιάστατη μορφή όπως φαίνεται στην Εικόνα 3.3.11.



Εικόνα 3.3.10 Αυτόματος ελεγχος PCB απο το Proteus



Εικόνα 3.3.11 Απεικόνιση πλακέτας σε μορφή 3D



Εικόνα 3.3.12 Τελικό αποτέλεσμα PCB

3.3.3 Προγραμματισμός της πλακέτας με τον PIC18F4550

Για τον προγραμματισμό του μικροελεκτή απαραίτητη προϋπόθεση είναι η διαθεσημότητα του εργαλείου PicKit3 (Εικόνα 3.3.13). Ο προγραμματιστής PicKit3 είναι μία συσκευή προγραμματισμού και ανίχνευσης σφαλμάτων (debugger), χαμηλού κόστους, ο οποίος ελέγχεται απο έναν υπολογιστή που διαθέτει περιβάλλον προγραμματισμού MPLAB IDE. Χρησιμοποιείται κυρίως για τον προγραμματισμό μικροελεκγτών της microchip PIC, με σκοπό τον προγραμματισμό των μικροελεγκτών διαμέσου του εργαλείου αυτού. Το PicKit3 μας παρέχει λειτουργίες και χαρακτηριστικά που βοηθάνε στην ανάπτυξη ενός ΕΣ όπως είναι η πλήρης συμβατότητά του με λειτουργικό Windows, ο γρήγορος προγραμματισμός του μικροελεγκτή σε πραγματικό χρόνο, η δυνατότητα επαναπρογραμματισμού μικροελεγκτών, καθώς και οι ενδείξεις που προσφέρουν δυνατότητα παρκολούθησης εγγραφής/ανάγνωσης.

Το PicKit3 διαθέτει 6 ακροδέκτες με βάση την Εικόνα 3.3.13 απο το τους οποίους:

- Ο **ακροδέκτης 1 ($\overline{MCLR}$)**: Συνδέεται με τον ακροδέκτη $\overline{MCLR}$ του μικροελεκγτή με σκοπό να τον επανακκινήσει πριν ξεκινήσει η διαδικασία προγραμματισμού του μικροελεγκτή.
- Ο **ακροδέκτης 2 (VDD)**: Συνδέεται με την τάση τροφοδοσίας του μικροελεγκτή που μπορεί να είναι στα 5V ή στα 3.3V.
- Ο **ακροδέκτης 3 (GND)**: Συνδέεται με την γείωση του συστήματος.
- Ο **ακροδέκτης 4 (PGD/ICSPDAT)**: Συνδέεται με τον ακροδέκτη PGD του μικροελεγτκή που χρησιμοποιείται για την μεταφορά σειριακών δεδομένων (διαδικασία προγραμματισμού του μικροελεγκτή).
- Ο **ακροδέκτης 5 (PGC/ICSPCLK)**: Συνδέεται με τον ακροδέκτη PGC του μικροελεγκτή που χρησιμοποιείται για τον συγχρονισμό των σειριακών δεδομένων κατά την μεταφορά τους.
- Ο **ακροδέκτης 6 (No connection)**: Δεν συνδέεται κάπου.

Εικόνα 3.3.13 PicKit3 ακροδέκτες

Ο επόμενος στόχος είναι η σύνδεση του PICKIT3 με τον υπολογιστή, ώστε να φορτωθεί ο κατάλληλος κώδικας, ο οποίος αργότερα θα ενσωματωθεί στον μικροελεγκτή. Ξεκινώντας την διαδικασία εγγραφής κώδικα ενεργοποίησης της επικοινωνίας του PIC με τν υπολογιστή διαμέσου θύρας USB, στο περιβάλλον PicKit3 v3.10, ώστε μόλις συνδεθεί στον υπολογιστή, θα αναγνωριστεί απο εκείνον και πλέον θα μπορούμε να προγραμματίσουμε την πλακέτα χωρίς την χρήση του PicKit3. Ο κώδικας που χρησιμοποιήθηκε για την ενεργοποίηση του USB της πλακέτας δίνεται απο την εταιρία Microchip μαζί με τα drivers του μικροελεγκτή PIC18F4550 που χρησιμοποιούνται με σκοπό, ο υπολογιστής να αναγνωρίζει την συσκευή. Ωστόσο σε αυτό το σημείο παρατηρήθηκε το πρόβλημα συμβατότητας των drivers με τον υπολογιστή που χρησιμοποιήθηκε για την υλοποίηση της Π.Ε δηλαδή τα drivers που διαθέτει η εταιρία δεν ήταν συμβατά με το λογισμικό του υπολογιστή και δεν μπορούσε να αναγνωρίσει τον PIC18F4550. Γι'αυτό τον λόγο στραφήκαμε στον ATMEGA328P, ο οποίος παρέχει συμβατότητα σε Windows 10 καθώς επίσης και drivers με την βοήθεια του περιβάλλοντος ανάπτυξης ARDUINO IDE.

3.4 Σχεδίαση και κατασκευή με τον ATMEGA328P (Arduino Uno)

Σε αυτή την υποενότητα θα αναλυθεί η πειραματική κατασκευή με χρήση Arduino Uno. Αρχικά επιλέξαμε τον ATMEGA328P (Arduino Uno) καθώς περιέχει ένα μεγάλο εύρος περιφερειακών που μπορούν να χρησιμοποιηθούν για μελλοντικές χρήσεις. Επιπλέον ο μικροελεγκτής αυτός έχει πλήθος βιβλιοθηκών, για τον προγραμματισμό του, που είναι συμβατές με τον μεταγλωττιστή του και διαθέτει καλή διαχείριση των πόρων. Πρέπει να σημειωθεί ότι σε σχέση με την εφαρμογή της υποενότητας 3.3 που δεν πραγματοποιήθηκε, καθώς υπήρξαν σφάλματα συμβατότητας με τον υπολογιστή, σε αυτή την εφαρμογή έγινε πρώτα έρευνα σχετικά με τους περιορισμούς του συστήματος.Δηλαδή ο μεταγλωττιστής που χρησιμοποιήθηκε με ονομασία Arduino IDE για τον σκοπό του προγραμματισμού

του μικροελεγκτή λειτούργησε καθώς επίσης λειτούργησε και η επικοινωνία της πλακέτας με τον υπολογιστή και τον μεταγλωττιστή.

Πρέπει να αναφερθεί ότι ο μικροελεγτής αυτός υπάρχει έτοιμος στην αγορά και ενσωματωμένος είδη σε πλακέτα, γι'αυτό το λόγο δεν χρειάζεται κάποιος να προβέι στην σχεδίαση τυπωμένου κυκλώματος αφού είναι πιο φτηνό να αγοραστεί έτοιμο. Ωστόσο στην παρούσα ΠΕ το τυπωμένο κύκλωμα δεν σχεδιάστηκε διότι στο τελικό προϊόν χρησιμοποιήθηκε το ESP32 για λόγους χρήσης του WiFi και γι'αυτό έγινε σχεδίαση με βάση το ESP32.

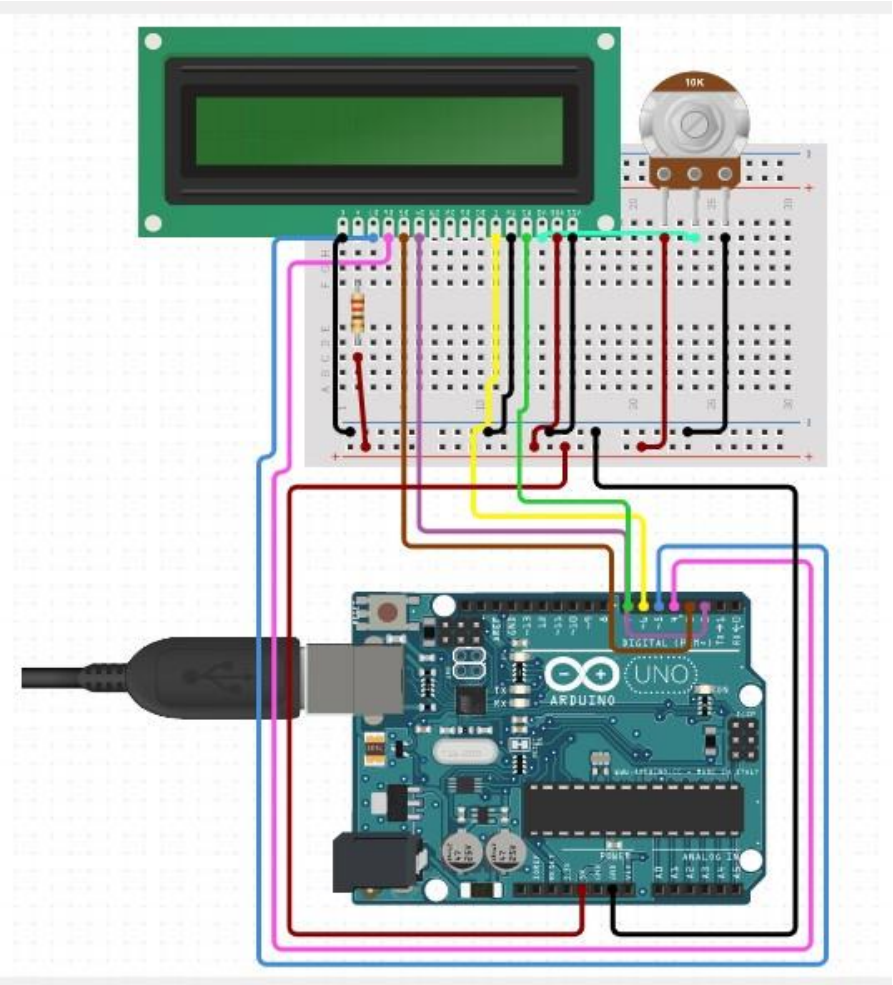
Στο Σχήμα 3.4 φαίνεται η συνδεσμολογία των υλικών με το Arduino Uno. Αναλυτικότερα έχουμε:

- Μία LCD 16X2, στην οποία θα εμφανίζεται το μήνυμα που θα κινείται απο δεξιά προς τα αριστερά (scrolling effect).
- Ένα ποτενσιόμετρο των 10K για την ρύθμιση της αντίθεσης της LCD.
- Μία αντίσταση 220Ω για την ένταση της φωτεινότητας.

Με βάση την παρακάτω συνδεσμολογία αυτό που πρέπει να προσέξουμε είναι η σύνδεση του ποτενσιομέτρου με την LCD καθώς μόνο ο ακροδέκτης Vo (της LCD) είναι αυτός που ρυθμίζει την αντίθεση την οθόνης. Σε περίπτωση που συνδεθεί το ποτενσιόμετρο σε κάποιον άλλο ακροδέκτη τότε θα έχει ως αποτέλεσμα, στην οθόνη να μην εμφανίζεται το μήνυμα που στέλνουμε. Στην συνέχεια η αντίσταση των 220Ω θα καθορίσει την ένταση της φωτεινότητας και συνδέεται με την άνοδο των LED της LCD που είναι ο ακροδέκτης A. Πρέπει να σημειωθεί ότι στην επιλογή της αντίστασης, η LCD τροφοδοτείται με τάση των 5V και γι'αυτό χρειαζόμαστε μία αντίσταση που μπορεί να χειριστεί τάσεις των 5V αλλά και να δίνει την επιθυμητή ένταση φωτεινότητας. Με βάση τα κριτήρια αυτά μία αντίσταση κάτω των 220Ω θα έχει ως αποτέλεσμα την παροχή μεγαλύτερης έντασης φωτεινότητας, ενώ όσο αυξάνουμε την αντίσταση αυτή τότε η ένταση θα μειώνεται όλο και περισσότερο.Το ίδιο αποτέλεσμα μπορούμε να το δημιουργήσουμε χρησιμοποιώντας ένα LDR σε διάταξη διαιρέτη τάσης, ώστε να έχουμε έλεγχο της φωτεινότητας με βάση την ένταση φωτός απο εξωτερική πηγή. Σε αυτή την ενότητα σκοπός είναι η ορθή λειτουργία του μεταγλωττιστή και η επικοινωνία του με το κύκλωμα.Η διάταξη του LDR θα αναλυθεί στην επόμενη ενότητα όπου θα παρουσιαστεί το τελικό προϊόν.

Στην συνέχεια ο ακροδέκτης RS συνδέεται με την ψηφιακή είσοδο/έξοδο 7 του Arduino Uno καθώς είναι υπεύθυνο για την επιλογή του εσωτερικού καταχωρητή της LCD και για την αποθήκευση των δεδομένων που απευθύνονται σε αυτήν. Στον καταχωρητή της LCD πρόκειται να εκτελεστεί η διαδικασία εγγραφής, δηλαδή να γράφτεί σε αυτόν δεδομένα με σκοπό να θέσουμε την οθόνη σε λειτουργία.Γι'αυτό ο ακροδέκτης RW πρέπει να συνδεθεί στην γείωση με σκοπό να πάρει την τιμή 0 όπως αναφέρθηκε στην υποενότητα 3.2.1. Ο ακροδέκτης E συνδέεται με την ψηφιακή είσοδο/έξοδο 6 η οποία πρέπει να είναι θύρα που υποστηρίζει παλμούς PWM και δημιουργεί ένα παλμό απο 0 σε 1 και το αντίστροφο. Όταν η μετάβαση του παλμού είναι απο HIGH σε LOW (απο 1 σε 0) τότε τα δεδομένα γράφονται στον κατάλληλο καταχωητή.

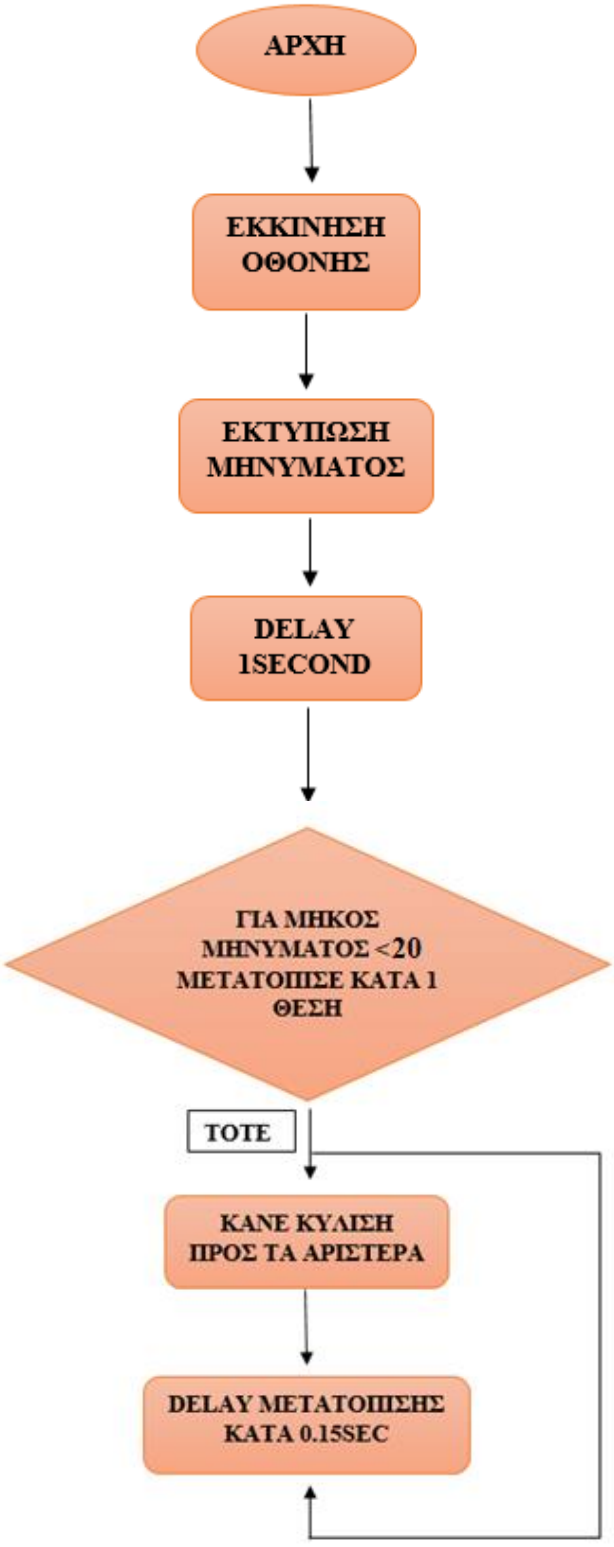
Τέλος οι ακδορέκτες απο D4 έως D7 μπορούν να συνδεθούν σε οποιαδήποτε ψηφιακή είσοδο του μικροελεκγτή, αφού είναι υπεύθυνοι για την μεταφορά δεδομένων των 8-bit. Σε αυτό το παράδειγμα οι ακροδέκτες συνδέθηκαν με την εξής διαμόρφωση: D4→2, D5→3, D6→4, D7→5.



Εικόνα 3.4.1 Διασυνέσεις υλικών με το Arduino Uno

3.4.1 Προγραμματισμός του μικροελεκγτή Arduino Uno

Παρακάτω θα γίνει αναφορά στον δοκιμαστικό κώδικα που έχει ώς σκοπό την εμφάνιση μηνυμάτων στην LCD και την κύλιση του μηνύματος απο δεξιά πρός τα αριστερά ή το αντίστροφο. Στο Σχήμα 3.4.2 φαίνεται το διάγραμμα του κώδικα με τις βασικές λειτουργίες.

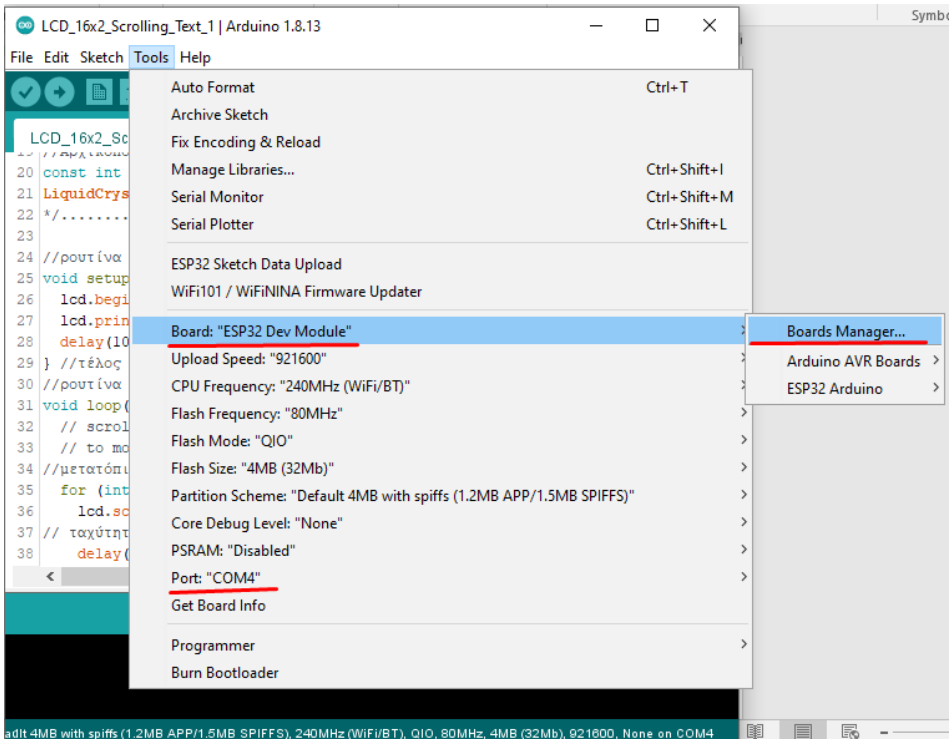


Σχήμα 3.4.2 Διάγραμμα κώδικα Arduino Uno

Ο παραπάνω κώδικας θέτει σε λειτουργία την οθόνη LCD, συγκεκριμένα ενεργοποιεί τις 16 στήλες και τις 2 γραμμές αφού πρόκειται για οθόνη LCD 16X2. Στην συνέχεια ο χρήστης διαμέσου του κώδικα πληκτρολογεί το μήνυμα που θέλει να εκτυπώσει στην οθόνη, η οποία με την σειρά της μετακινεί το μύνημα κατα 1 θέση προς τα αριστερά με μήκος χαρακτήρων 20 θέσεις. Αυτό σημαίνει ότι ανάλογα με τον αριθμό του κατχωρητή θέσης, έαν αυτός είναι μικρότερος του 16 (όσες είναι οι συνολικές θέσεις μιας γραμμής της LCD) για παράδειγμα ίσος με 13 τότε το μύνημα θα σταματάει κάθε φορά στην 4η θέση απο τα αριστερά. (16-13+1 = 4η θέση). Τέλος το μήνυμα παραμένει στην οθόνη για 1 δευτερόλεπτο πριν αρχίσει να εκτελεί ξανά την μετακίνηση.

Ο προγραμματισμός του μικροελεγκτή έγινε με την χρήση του μεταγλωττιστή ARDUINO IDE τον οποίο μπορούμε να κατεβάσουμε δωρεάν και προσφέρει μεγάλη ποικιλία βιβλιοθηκών για περιφερειακά όπως η οθόνη LCD. Αφού ανοίξουμε τον μεταγλωττιστή αυτόματα δημιουργεί ένα πρότζεκτ όπου μπορούμε να γράψουμε τον κώδικά. Όταν ανοίξουμε το νέο πρότζεκτ παρατηρούμε ότι ο μεταγλωττιστής έχει έτοιμους τους βρόγχους **void setup** και **void loop**, τους οποίους μπορούμε να διαγράψουμε εάν έχουμε έτοιμο το πρόγραμμά μας. Αφού γράψουμε τον κώδικα που χρειάζεται τότε το επόμενο βήμα είναι να ορίσουμε στον μεταγλωττιστή με ποιον μικροελεγκτή θα επικοινωνήσει. Συγκεκριμένα απο την μπάρα Tool→Board→Boards Manager, επιλέγουμε τον μικροελεγκτή που πρόκειται να συνδέσουμε, σε αυτή την περίπτωση επιλέγουμε Arduino Uno R3 και κάνουμε εγκατάσταση με σκοπό να κατεβάσει ο μεταγλωττιστής τις κατάλληλες βιβλιοθήκες. Ιδιαίτερη προσοχή απαιτείται στην επιλογή του μικροελεγκτή καθώς και στην θύρα Port που πρέπει να επιλέξουμε την θύρα εισόδου όπως φαίνεται στην Εικόνα 3.4.3.

Αφού γίνουν οι απαραίτητες ρυθμίσεις και φορτωθεί ο κώδικας στον μικροελεγκτή, τότε θα εμφανιστεί το μύνημα που δόθηκε, στην LCD.



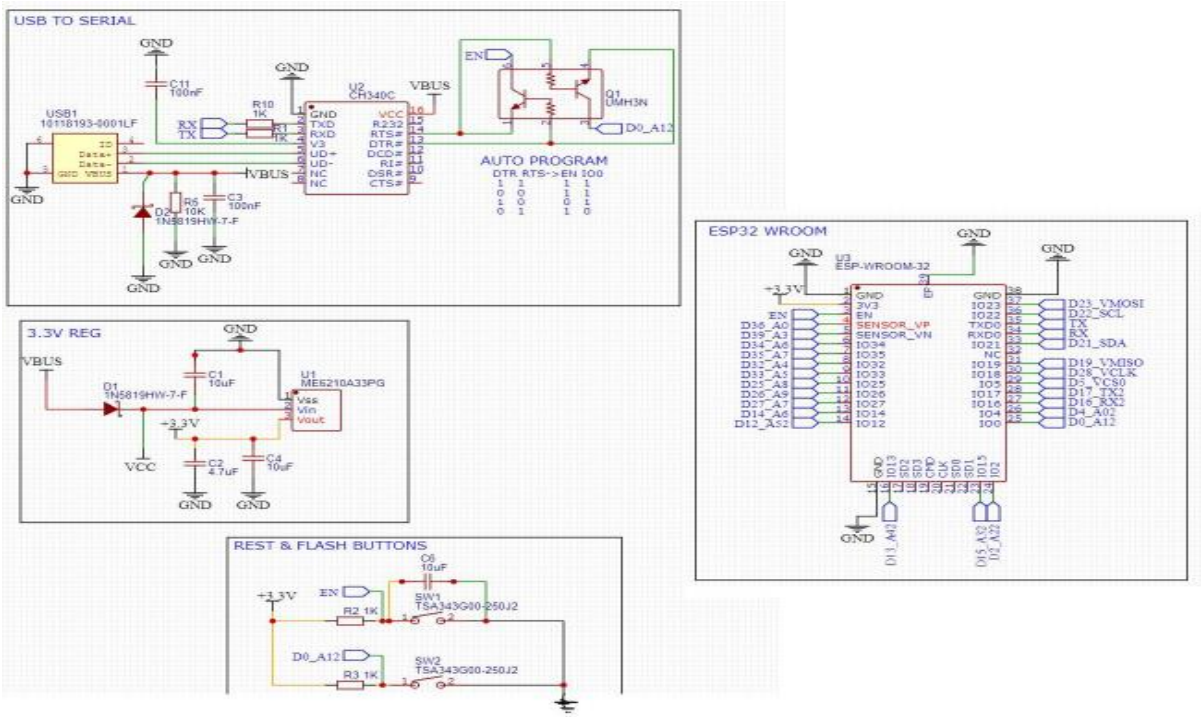
Εικόνα 3.4.3 Επιλογή μικροελεγκτή και θύρα επικοινωνίας

3.4.2 Παρατηρήσεις και βελτιώσεις με χρήση του ATMEGA328P

Με την παραπάνω υλοποίηση διαπιστώνουμε, σχετικά με την χρήση του Arduino Uno, την έλλειψη του ασύρματου δικτύου WiFi, κάτι το οποίο στην σημερινή εποχή των μικροεπεξεργαστών είναι αναγκαίο, τις ελάχιστες μονάδες εισόδου/εξόδου για μελλοντική χρήση και τον μεγάλο όγκο στην κατασκευή. Σχετικά με την επιλογή του Arduino Uno είναι μία φτηνή λύση για την ενασχόληση με μικροελεγκτές καθώς έχει πλήθος περιφερειακών και βιβλιοθηκών για χρήση. Ο κώδικας που εφαρμόστηκε έχει μειονεκτήματα και ελλείψης όπως αυτή του LDR καθώς και το γεγονός ότι τα αποτελέσματα του κώδικα, όπως το μύνημα που εκτυπώνεται στην LCD, καταλαμβάνει χώρο στην μνήμη του ,κάτι το οποίο δεν είναι επιθυμητό. Για την βελτίωση των παραπάνω χρησιμοποιήθηκε στο τελικό πρότζεκτ ο μικροελεγκτής ESP32 ο οποίος παρέχει δυνατότητα χρήσης WiFi, μεγαλύτερες ταχύτητες, περισσότερες μονάδες εισόδου/εξόδου, και διάφορες άλλες λειτουργίες που το Arduino Uno δεν προσφέρει.

3.5 Σχεδίαση και κατασκευή του ESP32

Στην Εικόνα 3.5.1 φαίνεται το σχηματικό του ESP32 που αποτελείται απο 4 βαθμίδες, την βαθμίδα του USB TO SERIAL που είναι υπεύθυνη για την επικοινωνία του ESP32 με τον υπολογιστή, την κύρια βαθμίδα ESP32-WROOM στην οποία γίνονται οι διασυνδέσεις του μικροελεγκτή με την γείωση,την τροφοδοσία και με τις κεφαλίδες ακίδων (pin headers). Στην συνέχεια η βαθμίδα τροφοδοσίας 3.3V REGULATOR είναι πρακτικά ένας σταθεροποιητής τάσης στα 3.3V για την παροχή τροφοδοσίας σε αυτή την επιθυμητή τάση. Τέλος για την επανεκκίνηση και την λειτουργία bootloader του ESP χρειάζονται 2 buttons αντίστοιχα και έτσι έχουμε την βαθμίδα RESET-BOOT.Στην ΠΕ χρησιμοποιήθηκε έτοιμη πλακέτα ESP32 καθώς είναι πιο οικονομική σε σχέση με την αγορά των υλικών και της σχεδίασης της πλακέτας.



Εικόνα 3.5.1 Σχηματικό ESP32

Στην συνέχεια έγιναν οι κατάλληλες συνδέσεις των υλικών με τον μικροελεγκτή. Αρχικά στο Σχήμα 3.5.2 παρατηρούνται οι συνδέσεις της LCD οθόνης με την εξής διαμόρφωση:

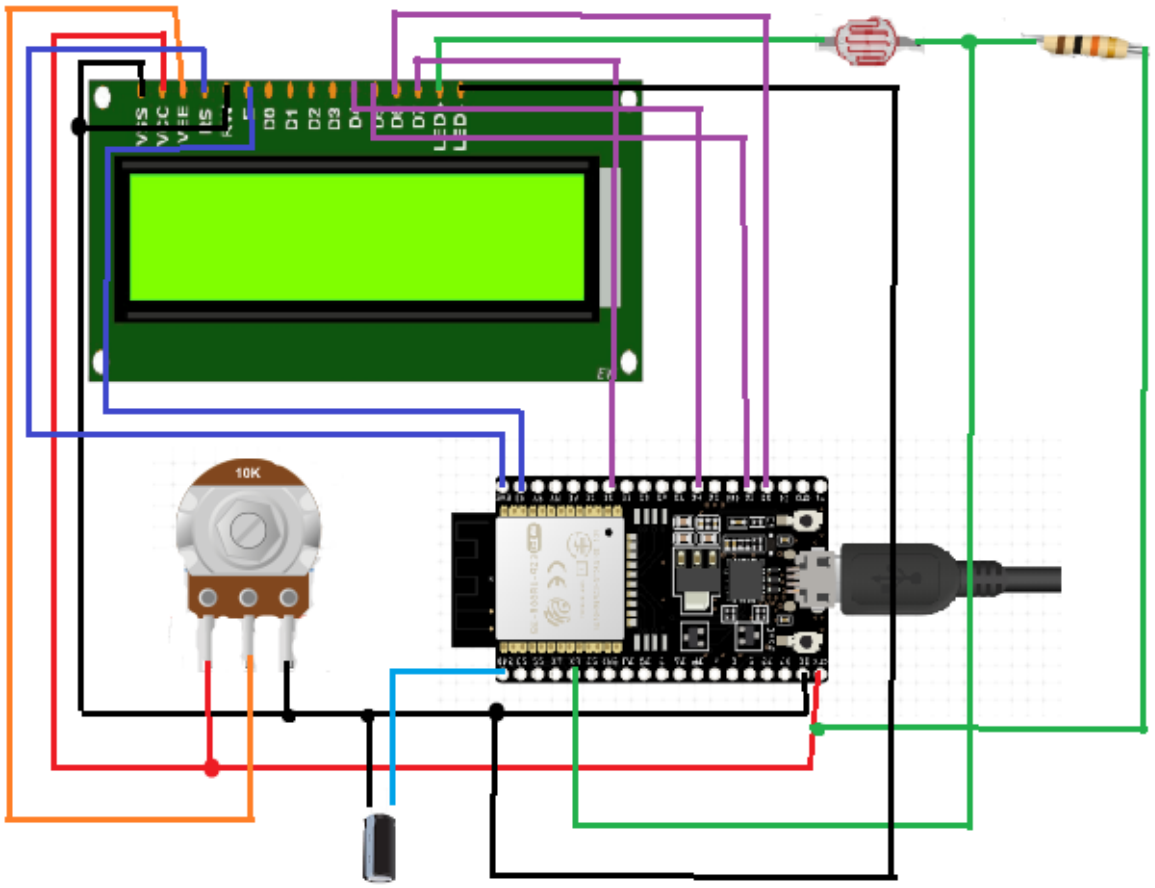
- Ακροδέκτης VSS→ GND
- Ακροδέκτης VDD→ +5V
- Ακροδέκτης Vo→ μεσαίος ακροδέκτης ποτενσιομέτρου για την ρύθμιση αντίθεσης της οθόνης
- Ακροδέκτης RS→ ακροδέκτης D23 του ESP
- Ακροδέκτης RW→ GND
- Ακροδέκτης E→ ακροδέκτης D22 του ESP
- Ακροδέκτης D4→ ακροδέκτης D5 του ESP
- Ακροδέκτης D5 → ακροδέκτης D4 του ESP
- Ακροδέκτης D6→ ακροδέκτης D2 του ESP
- Ακροδέκτης D7→ ακροδέκτης D18 του ESP
- Ακροδέκτης A→ ακροδέκτη LDR
- Ακροδέκτης K→ GND

Παρατηρούμε την χρήση του LDR που δίνει μία αναλογική τάση όταν συνδεθεί σε τροφοδοσία +5V, η οποία ποικίλει σε μέγεθος ανάλογα με την ένταση του φωτός που πέφτει στο LDR.Δηλαδή όσο μεγαλύτερη είναι η ένταση του φωτός, τόσο μεγαλύτερη θα είναι η αντίστοιχη τάση απο το LDR. Δεδομένου ότι το LDR παρέχει αναλογική τάση τότε το συνδέουμε με κάποιον ακροδέκτη του ESP που μπορεί να διαχειριστεί αναλογικά μεγέθη. Στην συγκεκριμένη περίπτωση συνδέθηκε με τον ακροδέκτη D32. Ο ακροδέκτης αυτός αφού διαβάσει την αναλογική τιμή τάσης απο τον διαιρέτη τάσης του LDR με την αντίσταση των 10ΚΩ, την μετατρέπει σε ψηφιακή τιμή στην περιοχή απο 0 έως 4095 που αντιστοιχούν σε τάσεις 0V και 3.3V αντίστοιχα. Πρέπει να σημειωθεί ότι η τάση τροφοδοσίας που παρέχεται στο ESP είναι 5V διαμέσου της θύρας USB και ο ακροδέκτης Vin λειτουργεί με δύο τρόπους. Ο πρώτος τρόπος είναι, αντί το ESP να τροφοδοτείται απο την θύρα USB, μπορούμε να συνδέσουμε στον ακροδέκτη Vin, όπου λειτουργεί ως είσοδος, μία εξωτερική πηγή διαφορετικής τάσης ώστε να τροφοδοτείται απο αυτή την πηγή. Ο δεύτερος τρόπος λειτουργίας είναι η τροφοδοσία του ESP απο το USB, όπως έχει γίνει στην εφαρμογή. Αυτό σημαίνει ότι ο ακροδέκτης Vin πλέον λειτουργεί ως έξοδο με τάση στον ακροδέκτη ίση με 5V. Αφού λοιπόν έχουμε τάση εξόδου στον ακροδέκτη Vin ίση με +5V, συνδέουμε το ένα άκρο του LDR με την άνοδο της LCD (ακροδέκτης A) και το άλλο άκρο με την αντίσταση των 10ΚΩ που καταλήγει στην τροφοδοσία των +5V (ακροδέκτης Vin). Το κοινό άκρο της LDR και της αντίστασης καταλήγει στον ακροδέκτη D32 του ESP όπου απο αυτό τον ακροδέκτη θα διαβάζουμε τις αναλογικές τιμές τάσης. Επειδή όμως το κανάλι ADC του ESP μπορεί να διαβάσει τιμές τάσης απο 0V έως 3.3V, αυτό σημαίνει ότι το εύρος των τιμών τάσης που παρέχει το LDR θα κυμαίνονται απο 0V έως και 3.3V. Η επιλογή της αντίστασης των 10ΚΩ έγινε πειραματικά καθώς στην αρχή δοκιμάστηκε αντίσταση των 220Ω και παρατηρήθηκε ότι ο διαιρέτης τάσης “συναντούσε” πολύ μικρή τιμή αντίστασης και έτσι οι μεταβολές της τάσης δεν γινόντουσαν αντιληπτές απο τον μικροεπεξεργαστή. Καταλήγουμε στο συμπέρασμα ότι στην επιλογή της αντίστασης σημαντικός παράγοντας είναι η τιμή τάσης τροφοδοσίας σε συνδιασμό με πειραματική εύρεση της τιμής της αντίστασης.

Στην συνέχεια συνδέεται ένα ποτενσιόμετρο των 10ΚΩ με τους δύο ακροδέκτες 1 και 3 να συνδέονται στην γείωση και την τροφοδοσία αντίστοιχα, ενώ τον μεσαίο ακροδέκτη 2 τον συνδέουμε στον ακροδέκτη Vo της LCD με σκοπό την ρύθμιση, της αντίθεση της οθόνης. Η επιλογή του ποτενσιομέτρου γίνεται με βάση τον βαθμό αντίθεσης που είναι επιθυμητό, δηλαδή όσο μικραίνει η τιμή του

ποτενσιομέτρου τόσο μικραίνει και η αντίθεση της LCD, οπότε η μέγιστη τιμή του ποτενσιομέτρου καθορίζει και το εύρος της αντίθεσης.

Τέλος συνδέουμε έναν ηλεκτρολυτικό πυκνωτή των 10uF στον ακροδέκτη EN του ESP και στην γείωση με στόχο τον ομαλό επαναπρογραμματισμό του ESP. Συγκεκριμένα κατα τον επαναπρογραμματισμό του ESP παρατηρήθηκε σφάλμα εγγραφής του κώδικα στην μνήμη του μικροεπεξεργαστή. Αυτό συμβαίνει διότι για την εγγραφή/διαγραφή του κώδικα στον μικροελεγκτή εφαρμόζεται κάποια τάση και για να μειώσουμε τον χρόνο επαναπρογραμματισμού χρησιμοποιούμε έναν πυκνωτή μεγαλύτερο απο 100nF. Συνδέοντας τον πυκνωτή με το κύκλωμα πετυχαίνετα η ομαλή λειτουργία επαναπρογραμματισμού του ESP χωρίς την εμφάνιση σφαλμάτων.



Εικόνα 3.5.2 Διασυνδέσεις υλικών με το ESP32

3.5.1 Προγραμματισμός του μικροελεγκτή ESP32

Στο ΕΣ αναπτύχθηκε κώδικας ο οποίος, συνδέει το ESP στο τοπικό δίκτυο, εμφανίζει στην LCD το μύνημα που πληκτρολογεί ο χρήστης, γίνεται έλεγχος του διαιρέτη τάσης για την αυτόματη φωτεινότητα και εμφανίζει τα αποτελέσματα αυτά σε μία ιστοσελίδα html. Συγκεκριμένα το ESP32 με βάση τον κώδικα λειτουργεί ως σταθμός δικτύου. Αυτό σημαίνει ότι ο μικροελεγκτής συνδέεται στο τοπικό δίκτυο και παίρνει πληροφορίες απο αυτό. Επιπρόσθετα δημιουργήθηκε ιστοσελίδα html για την εμφάνιση δεδομένων της LCD, δηλαδή το μύνημα που εμφανίζεται στην οθόνη και την ένταση της φωτεινότητας. Μέσα απο την ιστοσελίδα ο χρήστης θα έχει τον πλήρη έλεγχο του ESP καθώς θα μπορεί να επανναπρογραμματίζει τον μικροελεγκτή ασύρματα χωρίς να συνδεθεί στον υπολογιστή, και θα μπορεί να παίρνει πληροφορίες σχετικά με τον χώρο μνήμης Flash κα όπως φαίνεται στην Εικόνα 3.5.3. Δίνεται επίσης η δυνατότητα αποθήκευσης και διαγραφής αρχείων στην μνήμη SPI Flash File System

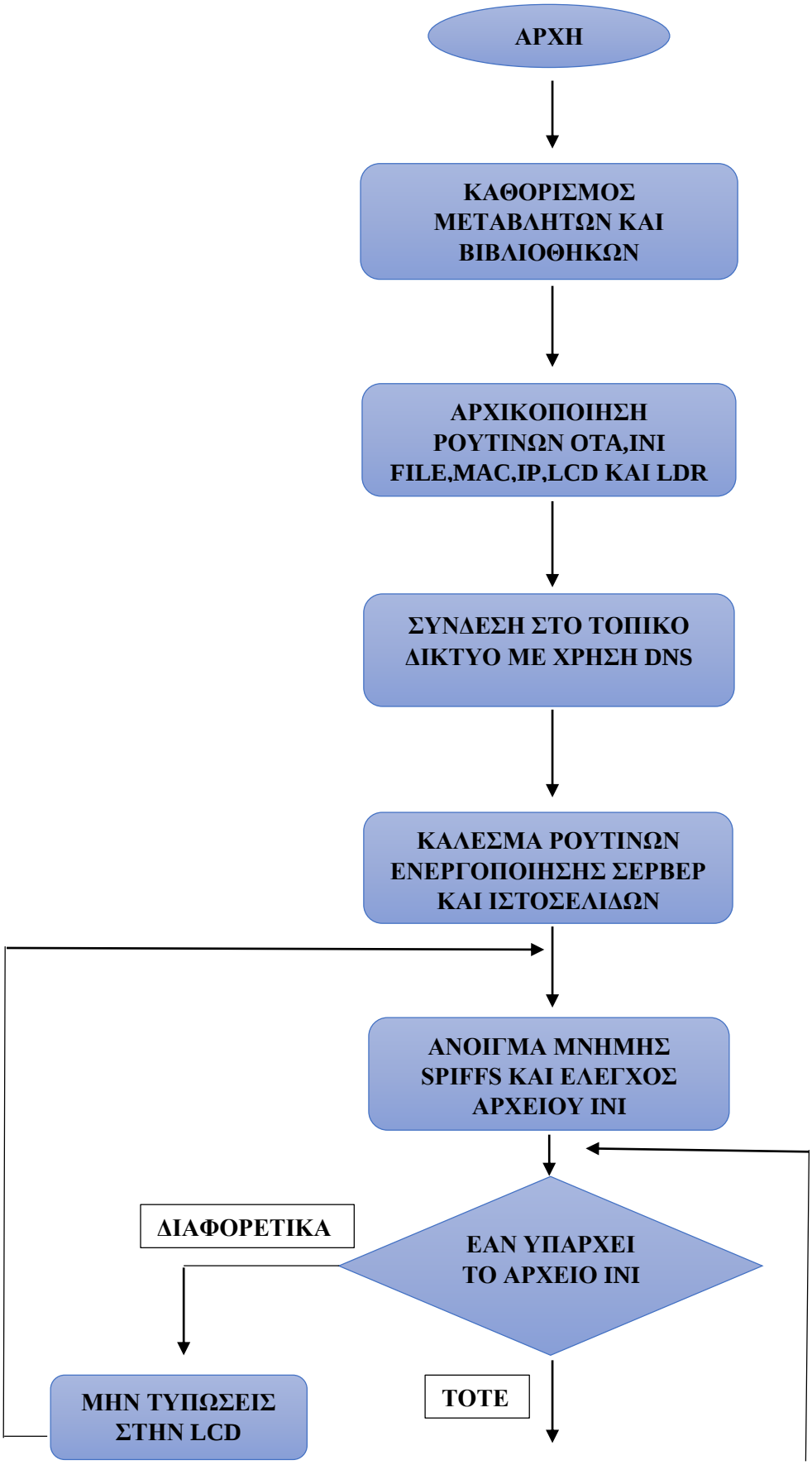
(SPIFFS).Ιδιαίτερα σημαντικό είναι η ασφάλεια και προστασία αυτών των δεδομένων απο ανεπιθύμητους χρήστες που έχουν πρόσβαση στο τοπικό δίκτυο και γιάυτό χρησιμοποιήθηκε κωδικός πρόσβασης κατά την είσοδο για την απεικόνιση των δεδομένων.

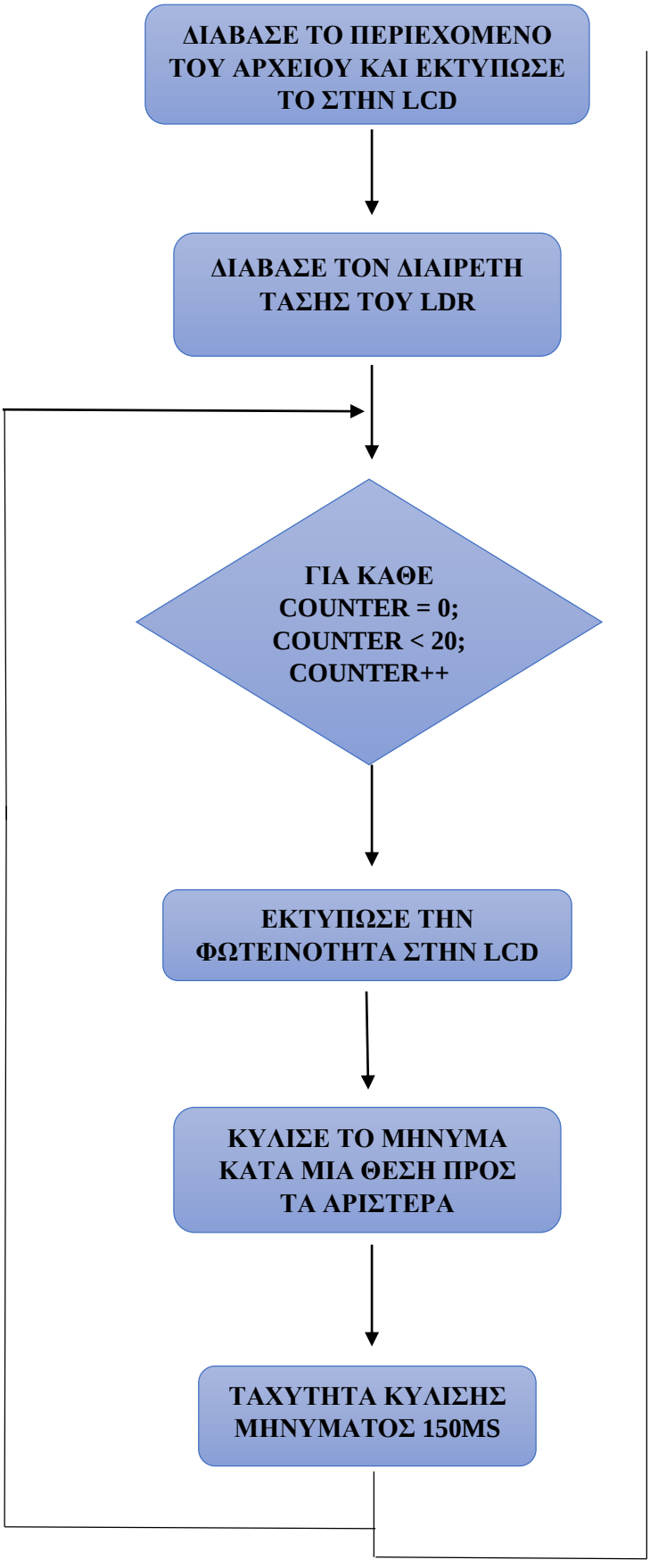
Το SPIFFS είναι ένα ελαφρύ σύστημα αρχείων που δημιουργήθηκε για μικροελεγκτές το οποίο συνδέεται με δίαυλο SPI όπως η μνήμη Flash.Το σύστημα αυτό επιτρέπει την πρόσβαση στην μνήμη Flash οποιαδήποτε στιγμή με σκοπό την ανάγνωση, την εγγραφή ή την διαγραφή αρχείων απο την μνήμη.

Εικόνα 3.5.3 SPIFFS και OTA

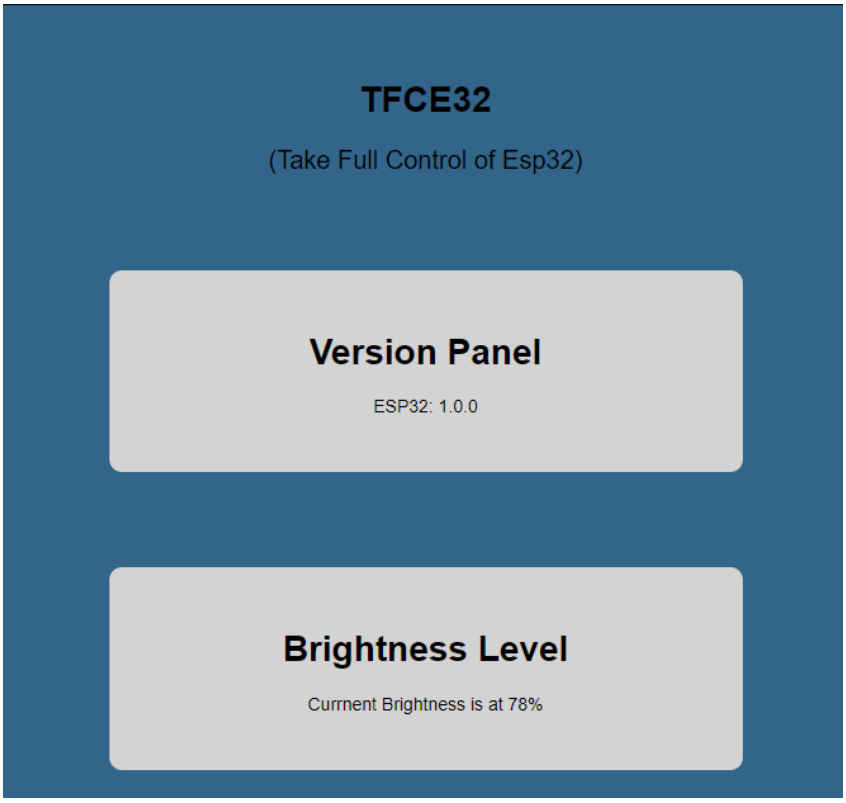
Στο Σχήμα 3.5.4 απεικονίζεται το διάγραμμα του κώδικα με τις βασικές λειτουργίες. Ιδιαίτερα σημαντική είναι η ρουτίνα δημιουργίας αρχείου Ini στο οποίο γίνεται εγγραφή του μηνύματος που πρόκειται να εμφανιστεί στην LCD και στην συνέχεια το αρχείο αυτό αποθηκεύεται στην μνήμη Flash. Στην συνέχεια ο μικροελεγκτής κάθε φορά ανοίγει την μνήμη και διαβάζει το περιεχόμενο αυτού του ini αρχείου, συγκεκριμένα διαβάζει την τιμή του κλειδιού Text το οποίο βρίσκεται στην ενότητα [Message].Εάν δεν υπάρχει το αρχείο το ESP δεν θα διαβάζει κάτι με αποτέλεσμα να μην εμφανίσει κάποιο μήνυμα στην οθόνη. Στην περίπτωση αυτή όταν πληκτρολογήσουμε το μήνυμα που θέλουμε να εμφανιστεί σστην οθόνη μέσα απο την ιστοσελίδα τότε αυτή με την σειρά της δημιουργεί το αρχείο ini και το αποθηκεύει στην SPIFFS.Τα αρχεία Ini είναι αρχεία διαμόρφωσης που αποτελούνται απο ενότητες που οργανώνουν ιδιότητες και απο ένα ζεύγη τιμών-κλειδιών, που περιέχονται στις διάφορες ενότητες του αρχείου. Ο τρόπος αποθήκευσης του μηνύματος έγινε με σκοπό την καλύτερη διαχείρηση της μνήμης Flash και για να έχει ο χρήστης τον έλεγχο του αρχείου σε περίπτωση που επιθυμεί να κάνει αλλαγές στο περιεχόμενο, πριν το αποθηκεύση στην μνήμη.Τέλος τρόπος με τον οποίο έχει

διαμορφωθεί ο κώδικας επιφέρει στην καλύτερη διαχείριση της μνήμης καθώς και στον αποδοτικό τρόπο εκτέλεσης των εντολών.

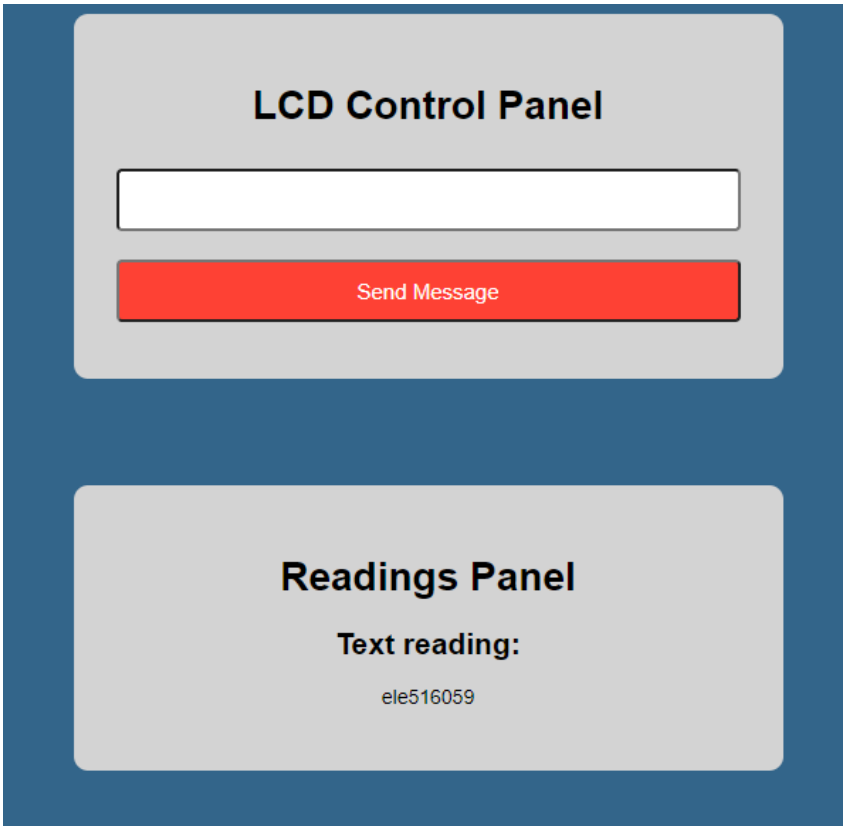




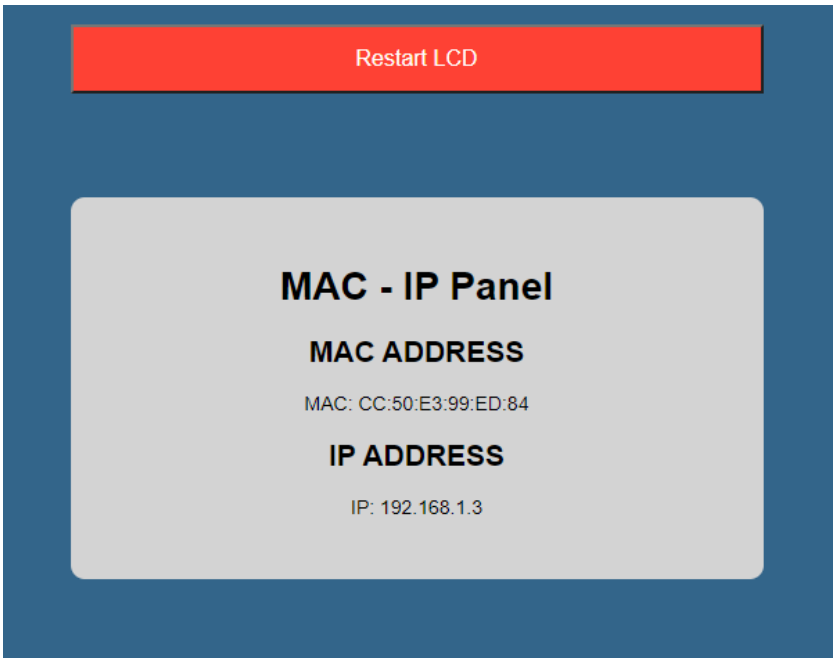
Σχήμα 3.5.4 Διάγραμμα κώδικα ESP32



3.5.5 Εμφάνιση έντασης φωτεινότητας μέσω html



Εικόνα 3.5.6 Είσοδος και εμφάνιση μηνύματος



Εικόνα 3.5.6 Εμφάνιση διευθύνσεων MAC και IP

3.5.2 Συμπεράσματα και βελτιώσεις στην τελική εφαρμογή

Απο τα παραπάνω διαπυστώθηκε η ορθή λειτουργία των υλικών και του λογισμικού που δέχονται ωστόσο βελτιώσεις κυρίως στο κομμάτι αναβάθμισης των περιφερειακών, στον κώδικα εμφάνισης μηνυμάτων στην LCD και στην εμφάνιση της ιστοσελίδας. Στον τομέα των περιφερειακών αστοχίες παρατηρήθηκαν στο LDR, όπου ο αισθητήρας δεν έχει ακρίβεια και δεν θεωρείται ιδανικός για τέτοιου είδους εφαρμογές. Συνεπώς η αλλαγή του LDR με κάποιον αισθητήρα φωτός υψηλής ακρίβειας μπορεί να διορθώσει αυτό το πρόβλημα. Στην συνέχεια η οθόνη LCD 16X2 (στήλες X γραμμές) μπορεί να αντικατασταθεί από κάποια άλλη LCD μεγαλύτερου μήκους, που αυτό όμως αυξάνει τον όγκο της κατασκευής.

Μέσα απο την ανάπτυξη του κώδικα η βιβλιοθήκη αρχικοποίησης της LCD και συγκεκριμένα η εντολή lcd.print(), που όταν καλείται εκτυπώνει το μήνυμα που δέχεται στην οθόνη, δεν μπορεί να πάρει τιμές σε πραγματικό χρόνο απο το ini αρχείο. Δηλαδή κάθε φορά που αποθηκεύουμε ένα μήνυμα απο την ιστοσελίδα στην μνήμη του ESP έχει οριστεί ένα κουμπί με ονομασία Restart LCD το οποίο θέτει σε επανεκκίνηση τον μικροελεγκτή με σκοπό να διαβάσει το νέο μήνυμα που αποθηκεύτηκε στο αρχείο.Η λύση στο πρόβλημα αυτό είναι η αναζήτηση και η δημιουργία νέας βιβλιοθήκης αρχικοποίησης της LCD όπου, όταν θα καλείται η εντολή εκτύπωσης του μηνύματος, αυτή θα συνεχίσει να “περιμένει” κάποιο νέο μήνυμα χωρίς να χρειαστεί η επανεκκίνηση του ESP.Επιπλέον κατά την λειτουργία ασύρματου σταθμού του μικροελεγκτή έχει δοθεί εντολή στον κώδικα που επιτρέπει την σύνδεση του ESP στο τοπικό δίκτυο σε συγκεκριμένη IP την 192.168.1.3 (Static IP).Αυτό έχει ως συνέπεια,σε περίπτωση που σε κάποιο router η συγκεκριμένη IP χρησιμοποιείται απο άλλη συσκευή,να μην μπορέσει να συνδεθεί ο μικροελεγκτής στο δίκτυο.Αυτό λύνεται πολύ απλά μέσα απο τον κώδικα διαγράφοντας της εντολές ανάθεσης στατικής IP.Ωστόσο εκτελώντας αυτή την ενέργεια το ESP θα παίρνει τυχαίες διευθύνσεις IP κάθε φορά που η προηγούμενη IP δεν είναι διαθέσιμη. Με την σειρά του, για να λύσουμε το πρόβλημα της τυχαίας IP εφαρμόζουμε πρωτόκολλο DNS (Domain Name Server) με το οποίο ορίζεται η IP σε οποιαδήποτε ονομασία επιθυμούμε.

Τέλος οι ιστοσελιδών, μπορούν να δεχτούν βελτιώσεις κυρίως στην εμφάνισή τους.Οι ιστοσελίδες Firmware Update και Update Tool δίνουν την δυνατότητα αναβάθμισης λογισμικού και ιστοσελιδών σε περίπτωση που πραγματοποιηθούν αυτές οιο βελτιώσεις.

3.6 Επίλογος κεφαλαίου 3

Σε αυτό το κεφάλαιο εφαρμόστηκε μεθοδολογία σχεδίασης και υλοποίησης ΕΣ με κύριο χαρακτηριστικό τρείς μικροελεγκτές. Οι μικροελγκτές PIC18F4550 και ATMEGA328P χρησιμοποιήθηκαν δοκιμαστικά για την παρατήρηση προβλημάτων και αστοχιών που θα προέκυπταν κατά την υλοποίηση. Συγκεκριμένα μέσα από τις αστοχίες αντιλήφθηκαν βελτιώσεις οι οποίες προέτρεπαν στην αναζήτηση νέου υλικού και λογισμικού.Κύριος παράγοντας της εφαρμογής αποτέλεσε η χρήση του WiFi το οποίο οδηγεί στην μελέτη και την έρευνα κώδικα html, JavaScript, που χρησιμοποιούνται για την σύνταξη ιστοσελίδας και κώδικα CSS που εφαρμόζεται για την εμφάνισή της. Επιπλέον στο επίπεδο του κώδικα ιδιαίτερα σημαντική είναι η γλώσσα προγραμματισμού C/C++ καθώς είναι το κύριο εργαλείο σε εφαρμογές αυτού του τομέα, δηλαδή στην υλοποίηση εφαρμογών με χρήση μικροελεγκτών. Τέλος μέσα απο την πλατφόρμα σχεδίασης PCB Proteus και μεταγλωττιστών πραγματοποιήθηκε εξοικείωση πάνω στα εργαλεία αυτά που αποσκοπούν στην σχεδίαση τυπωμένων κυκλωμάτων και στην σύνταξη κώδικα του μικροελεγκτή και της ιστοσελίδας.

Η υλοποίηση αυτή μπορεί να χρησιμοποιηθεί όπου απαιτείται η ένδειξη μηνυμάτων και η επεξεργασία τους όπως γίνεται σε τηλεματικές οθόνες. Επιπλέον πέρα απο τον επαγγελματικό τομέα, η συγκεκριμένη κατασκευή μπορεί να εφαρμοστεί για γεικούς σκοπούς όπως στην εμφάνιση μηνυμάτων υπενθύμισης που ορίζει ο χρήστης διαδικτυακά και στην αποθήκευση μικρών αρχείων για την δημιουργία αντιγράφων ασφαλείας ή για την μεταφορά τους απο συσκευή σε συσκευή ανάλογα απο τον υπολογιστή, απο τον οποίο συνδέεται ο χρήστης στην ιστοσελίδα.

Βιβλιογραφία

Βιβλία

[1] Μηνάς Δασυγένης, Δημήτριος Σούντρης, *Ενσωματωμένα Συστήματα: Ο αθέατος ψηφιακός κόσμος*, 2015

[2] Σ. Μπουλταδάκης, Γ. Πατουλίδης, Ν.Ασημόπουλος, *ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΙΚΡΟΕΛΕΓΚΤΩΝ ΓΙΑ ΜΗΧΑΝΙΚΟΥΣ*, Εκδόσεις ΤΖΙΟΛΑ, 2017

[3] James F. Kurose, Keith W. Ross, *Computer Networking A Top-Down Approach-Seventh Edition, Εκδόσεις Μ.Γκιούρδας, 2018*

[4] Παναγιώτης Παπάζογλου, Σπύρος-Πολυχρόνης Λιωνής, *Ανάπτυξη Εφαρμογών με το Arduino, Ένας πλήρης οδηγός για αρχάριους και προχωρημένους*, Εκδόσεις ΤΖΙΟΛΑ, 2015

[5] Sha Ga, *ESP8266 Arduino Tutorial: With Detailed Approach*, self published, 2018

[6] Catalin Batrinu, *ESP8266 Home Automation Projects*, Packt Publishing, 2017

[7] Agus Kurniawan, *Internet of Things Projects with ESP32*, Packt Publishing, 2019

[8] Peter Hoddie, Lizzie Prader, *IoT Development for ESP32 and ESP8266 with Javascript: A Practical Guide to XS and the Moddale SDK*, Apress, 2020

Datasheets

[9] Microchip, "*28/40/44-Pin, High-Performance, Enhanced Flash, USB Microcontrollers with nanoWatt Technology"*, PIC18F4550, 2006

[10] Texas Instruments, "*Wide Temperature Three-Pin Adjustable Regulator*", LM117, Jun. 2020

[11] Espressif, "*ESP32 Series*", Esp-Wroom32, Version 3.4, 2020

Internet Sites

[12] DIYI0T, *Arduino vs ESP8266 vs ESP32 Microcontroller Comparison*, Cdaviddav et al, 2020, [Online]. Available: https://diyi0t.com/technical-datasheet-microcontroller-comparison/

[13] Pantech Prolabs India Pvt Ltd, *Getting Started with pic kit 3*, [Online]. Available: https://www.pantechsolutions.net/getting-started-with-pic-kit-3

[14] Baselight, *LED Info*, [Online]. Available: http://www.baselight.se/led-info

[15] EEPower, *What is a resistor: Resistor fundamentals: Resistor Guide*, [Online]. Available: https://eepower.com/resistor-guide/resistor-fundamentals/what-is-a-resistor/#

[16] Electrical4U, *Crystal Oscillator: Circuit, Frequency & Working Principle*, Electrical4U, 2020, [Online]. Available: https://www.electrical4u.com/crystal-oscillator/

[17] Basic Electronics Tutorials, *Introduction to Capacitors, Capacitance and Charge*, 2020, [Online]. Available: https://www.electronics-tutorials.ws/capacitor/cap_1.html

[18] Random Nerd Tutorials, *Electronics Basics - How a Potentiometer Works*, 2019, [Online]. Available: https://randomnerdtutorials.com/electronics-basics-how-a-potentiometer-works/

[19] Photoresistors - an overview | Science Direct Topics, *Photoresistors*, [Online]. Available: https://www.sciencedirect.com/topics/engineering/photoresistors

[20] Electronic Circuits and Diagrams-Electronic Projects and Design, *Working of LCD (Liquid Crystal Display) with diagram and principle*, 2018, [Online]. Available: https://www.circuitstoday.com/liquid-crystal-displays-lcd-working

[21] Networked Embedded Systems & Real-Time Systems | Athena Research Center, *Networked Embedded Systems & Real-Time Systems*, [Online]. Available:

https://www.athena-innovation.gr/en/research-areas/networked-embedded-systems-real-time-systems

[22] DigitalThinkerHelp, *Advantages, Disadvantages, Characteristics of Embedded System*, Banger, 2020,[Online].Available:http://digitalthinkerhelp.com/advantages-disadvantages-characteristics-of-embedded-system/

[23] easysiteaussie, *Disadvantages Of Embedded Systems*, [Online]. Available:

https://easysiteaussie815.weebly.com/disadvantages-of-embedded-systems.html

[24] ElProCus, *Microcontrollers Types:Advantages, Disadvantages & Their Applications*, says: et al, 2020, [Online]. Available: https://www.elprocus.com/microcontrollers-types-and-applications/

[25] Electronics Hub, *Microcontroller Basics, Types and Applications*, Administrator et al, 2017, [Online]. Available: https://www.electronicshub.org/microcontrollers/

[26] Electronic Circuits and Diagrams-Electronic Projects and Design, *Microcontroller Invention History - Who Invented first Microcontroller*, Ruth et al, 2013, [Online]. Available:

https://www.circuitstoday.com/microcontroller-invention-history

[27] DroneBot Workshop, *Getting Started with the ESP32 - Using the Arduino IDE*, DroneBot Workshop, 2020, [Online]. Available: https://dronebotworkshop.com/esp32-intro/

[28] Last Minute Engineers, *Insight Into ESP32 Features & Using It With Arduino IDE (Easy Steps)*, 2020, [Online]. Available: https://lastminuteengineers.com/esp32-arduino-ide-tutorial/

[29] History Computer, *Microprocessor – Complete History of the Microprocessor*, [Online]. Available:

https://history-computer.com/microprocessor-complete-history-of-the-microprocessor/

[30] Engineers Garage, *AVR Microcontroller : All You Need To Know- (Part 1/46)*, Daga, 2019, [Online]. Available: https://www.engineersgarage.com/article_page/avr-microcontroller-all-you-need-to-know-part-1-46/

[31] The Engineering Projects, *Introduction to ATmega328*, Nasir, 2020, [Online]. Available: https://www.theengineeringprojects.com/2017/08/introduction-to-atmega328.html

[32] Wikipedia, *IEEE 802.11*, 2020, [Online]. Available: https://en.wikipedia.org/wiki/IEEE_802.11

[33] Techopedia.com, *What is IEEE 802.11i - Definition from Techopedia*, [Online]. Available: https://www.techopedia.com/definition/16835/ieee-80211i

Παράρτημα κώδικες

Κώδικας PIC18F4550

```
#include <18F4550.h>
#fuses HS,NOWDT,NOPUT,NOPROTECT,NOPBADEN
#use delay (clock=20M)
#define LCD_ENABLE_PIN  PIN_B0
#define LCD_RS_PIN      PIN_B1
#define LCD_RW_PIN      PIN_B2
#define LCD_DATA4       PIN_B4
#define LCD_DATA5       PIN_B5
#define LCD_DATA6       PIN_B6
#define LCD_DATA7       PIN_B7
#include <lcd.c>                        //Incluimos la libreria LCD
const char message[] = {"ele516059 dipae"};
void main (void)
{
  int y=1;
  signed int x=100;
  lcd_init();
  while(true)
  {
    lcd_gotoxy(x,y);
    lcd_putc(message);
    delay_ms(50);
    x--;
    if(x==0)
    {
      x=100;
```

```
        y++;
        if(y>2)
        y=1;
      }
     lcd_putc("\f");
   }
}
```

Κώδικας Arduino Uno R3

```
/*
  Pins configuration:
 * LCD RS pin to digital pin 7
 * LCD Enable pin to digital pin 6
 * LCD D4 pin to digital pin 2
 * LCD D5 pin to digital pin 3
 * LCD D6 pin to digital pin 4
 * LCD D7 pin to digital pin 5
 * LCD R/W pin to ground
 * 10K POT:
 * ends to +5V and ground
 * wiper to LCD VO pin (pin 3)
 * 220Ω resistor at pin A of LCD and +5V
*/


#include <LiquidCrystal.h> //εισαγωγή βιβλιοθήκης

//Αρχικοποίηση των pins της LCD:
const int rs = 7, en = 6, d4 = 2, d5 = 3, d6 = 4, d7 = 5;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

```
*/.....*/

//ρουτίνα setup:
void setup() {
  lcd.begin(16, 2); //ενεργοποίηση των στηλών και των γραμμών της LCD
  lcd.print("hello, world!"); //εκτύπωση του μηνύματος "This is a test" στην LCD
  delay(1000); //καθυστέρηση 1000ms(1second)
} //τέλος ρουτίνας setup
//ρουτίνα βρόγχου:
void loop() {
  // scroll 13 positions (string length) to the left
  // to move it offscreen left:
//μετατόπιση μηνύματος κατα 1 θέση προς τα αριστερά με μήκος μηνύματος 20 θέσεις:
  for (int positionCounter = 0; positionCounter < 20; positionCounter++) {
    lcd.scrollDisplayLeft(); //εντολή μετατόπισης προς τα αριστερά
// ταχύτητα μετατόπισης 150ms (0.15seconds).Μείωση του χρόνου delay έχει ως αποτέλεσμα την αύξηση της μετατόπισης του μηνύματος:
    delay(150);
  } //τέλος ρουτίνας for
        } //τέλος ρουτίνας βρόγχου
```

Κώδικας ESP32

Main κώδικας

```
/*
  The circuit:
 * LCD RS pin to digital pin 23
 * LCD Enable pin to digital pin 22
 * LCD D4 pin to digital pin 5
 * LCD D5 pin to digital pin 4
 * LCD D6 pin to digital pin 2
```

```
 * LCD D7 pin to digital pin 18
 * LCD R/W pin to ground
 * 10K potentiometer:
*/

//====================Βιβλιοθήκες================//
#include <LiquidCrystal.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <WebServer.h>
#include <Update.h>
#include <ArduinoOTA.h>
#include <ESPmDNS.h>
#include <ESPAsyncWebServer.h>
#include <AsyncTCP.h>
#include "mainpage.h"
#include "SPIFFSIniFile.h"
#include"SPIFFS.h"
//=============================================//
AsyncWebServer server(80);
#define FIRMWARE_VERSION "v1.0.1"
//===================Ορισμός ssid και password τοπικού δικτύου================//
const char* ssid = "(Το δικό σας SSID του router)";
const char* password = "(Ο κωδικός του router σας)";
//Ορισμός username και password για την σύνδεση στην ιστοσελίδα
const char* username = "(Ορίστε username)";
const char* pass = "(Ορίστε Password)";
//Ορισμός μεταβλητών
const char* PARAM_STRING1 = "inputString1"; //text input απο website
```

```
const char* PARAM_INPUT = "value";
const char *filename = "/config.ini"; //ονομασια αρχειου ini σε config
const int rs = 23, en = 22, d4 = 5, d5 = 4, d6 = 2, d7 = 18; //pins LCD οθόνης
static String getMacAddress(void);
uint8_t baseMac[6];
String  MAC; // αποθήκευση δεύθυνσης MAC
String message;
String logmessage;
float LDR; // pin LDR 32 LDR = AnalogRead(34)
int x; // μεταβλητή mapping των τιμών του LDR απο 0-4095 σε 0-100
const size_t bufferLen = 80;  //μεγεθος του buffer
char buffer1[bufferLen];  // αποθηκευση μηνύματος LCD στο buffer
size_t content_len;
size_t fwUpdateProgress;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7); //αρχικοποίηση των pins της LCD

//ρουτίνα εμφάνισης σφαλμάτων σε περίπτωση που δεν υπάρχει το ini αρχείο στην μνήμη
void printErrorMessage(uint8_t e, bool eol = true)
{
  switch (e) {
  case SPIFFSIniFile::errorNoError:
    Serial.print("no error");
    break;
  case SPIFFSIniFile::errorFileNotFound:
    Serial.print("file not found");
    break;
  case SPIFFSIniFile::errorFileNotOpen:
    Serial.print("file not open");
    break;
  case SPIFFSIniFile::errorSeekError:
```

```
    Serial.print("seek error");
    break;
  case SPIFFSIniFile::errorSectionNotFound:
    Serial.print("section not found");
    break;
  case SPIFFSIniFile::errorKeyNotFound:
    Serial.print("key not found");
    break;
  case SPIFFSIniFile::errorEndOfFile:
    Serial.print("end of file");
    break;
  case SPIFFSIniFile::errorUnknownError:
    Serial.print("unknown error");
    break;
  default:
    Serial.print("unknown error value");
    break;
  }
  if (eol)
    Serial.println();
}
//ρουτίνα ανάγνωσης αρχείου ini απο την μνήμη
void readFile(){

  if(!SPIFFS.begin(true)){
      Serial.println("An Error has occurred while mounting SPIFFS");
      return;
    }
   SPIFFSIniFile ini(filename);
```

```
  if (!ini.open()) {
    Serial.print("Ini file ");
    Serial.print(filename);
    Serial.println(" does not exist");
    // Cannot do anything else
    while (1);
  }

    Serial.println("Ini file exists");
   Serial.println("geting settings from ini file...");

   if(ini.getValue("Message", "Text", buffer1, bufferLen)){
    Serial.print("Text= ");
    Serial.println(buffer1);
  }

  else{
    Serial.print("Could not read 'Text' from section 'Message', error was ");
    printErrorMessage(ini.getError());
  }
}
//ρουτίνα εμφάνισης MAC διεύθυνσης
static String getMacAddress(void)
{
  esp_read_mac( baseMac, ESP_MAC_WIFI_STA ); // Get MAC address for WiFi station
  char macStr[18] = { 0 };
  sprintf(macStr, "%02X:%02X:%02X:%02X:%02X:%02X", baseMac[0], baseMac[1], baseMac[2], baseMac[3], baseMac[4], baseMac[5]);
  return String(macStr);
}
```

```
//αποθήκευση διεύθυνσης MAC και επιστροφή της τιμής σε μορφή String
String readMAC() {
  MAC = getMacAddress();
  return String (MAC);
}
//Αποθήκευση διεύθυνσης IP και επιστροφή σε μορφή String
String readIP() {
  String ipadd = WiFi.localIP().toString().c_str();
  return String (ipadd);
}
//AnalogRead pin LDR και mapping τιμής
String readLDR(){
  LDR = analogRead(32);
  x = map(LDR,4095,3580,0,100);
  return String(x);
  }
//αποθήκευση της τιμής του buffer στην μεταβλητή msg
String readText(){
  String msg = buffer1;
  return msg;
  }

//ρουτίνα ενεργοποίησης και σύνδεσης στο τοπικό δίκτυο
void wifiInit(){
WiFi.mode(WIFI_STA); //Connectto your wifi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to ");
  Serial.print(ssid);
```

```
//Wait for WiFi to connect
  while(WiFi.waitForConnectResult() != WL_CONNECTED){
      Serial.print(".");
    }

  //If connection successful show IP address in serial monitor
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());  //IP address assigned to your ESP
  }
//ρουτίνα ενεργοποίησης LCD και εμφάνιση μυνήματος
 void readLCD(){
  lcd.begin(16, 2); // set up the LCD's number of columns and rows
      message = lcd.print(buffer1);
      delay(1000);
  }



//ρουτίνα Setup
void setup() {
Serial.begin(115200);
pinMode(32,INPUT); //ορισμός pin 32 ώς είσοδος
//ενεργοποίηση SPIFFS
 if (!SPIFFS.begin()) {
    Serial.println("An Error has occurred while mounting SPIFFS");
    return;
  }
```

```
 wifiInit();  //ρουτίνα ενεργοποίησης WiFi
 readFile(); //ρουτίνα ανάγνωσης αρχείου
 readLCD(); //ρουτίνα μηνύματος LCD


 //OTA Update//
 ArduinoOTA.onStart([]() {
    String type;
    if (ArduinoOTA.getCommand() == U_FLASH)
      type = "sketch";
    else // U_SPIFFS
      type = "filesystem";
    SPIFFS.end(); // NOTE: if updating SPIFFS this would be the place to unmount SPIFFS using SPIFFS.end()
    Serial.println("Start updating " + type);
  })
  .onEnd([]() {
    Serial.println("\nEnd");
  })
  .onProgress([](unsigned int progress, unsigned int total) {
    Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
  })
  .onError([](ota_error_t error) {
    Serial.printf("Error[%u]: ", error);
    if (error == OTA_AUTH_ERROR) Serial.println("Auth Failed");
    else if (error == OTA_BEGIN_ERROR) Serial.println("Begin Failed");
    else if (error == OTA_CONNECT_ERROR) Serial.println("Connect Failed");
    else if (error == OTA_RECEIVE_ERROR) Serial.println("Receive Failed");
    else if (error == OTA_END_ERROR) Serial.println("End Failed");
  });
  ArduinoOTA.begin( );
```

```
   Update.onProgress(printProgress);
// Τέλος OTA Update//

//Ενεργοποίηση το DNS//
if (MDNS.begin("dipae")) {
    Serial.println("MDNS responder started");
  }
configureWebServers();//κάλεσμα ιστοσελιδών
server.begin(); //ενεργοποίηση Server
}

void loop() {
//κύλιση μυνήματος προς τα αριστερά κατα μία θέση με ταχύτητα 150ms
  for (int positionCounter = 0; positionCounter < 20; positionCounter++) {
    lcd.scrollDisplayLeft();
    // wait a bit:
    delay(150);

   lcd.setCursor(0,1);
     lcd.print("Brightness=   %");
     lcd.setCursor(11,1);
     lcd.print(x);  //εμφάνιση ποσοστού φωτεινότητας στην LCD

  }
 Serial.println(LDR);
  delay(2500);
}
//ρουτίνες εμφάνισης MAC,IP,Brightness και Text στην ιστοσελίδα
String functn(const String& var) {
```

```
 if (var == "mac") {
   return readMAC();
  }
  else if (var == "Ip") {
   return readIP();
  }
 else if (var=="VALUEP"){
  return readLDR();
  }
  else if (var == "TEXT"){
   return readText();
  }
  return String();
}
//ρουτίνα εμφάνισης δεδομένων αποθηκευτικού χώρου SPIFFS και διαχείρηση της μνήμης
String processor(const String& var) {
  if (var == "FIRMWARE") {
   return FIRMWARE_VERSION;
  }

  if (var == "FREESPIFFS") {
   return humanReadableSize((SPIFFS.totalBytes() - SPIFFS.usedBytes()));
  }

  if (var == "USEDSPIFFS") {
   return humanReadableSize(SPIFFS.usedBytes());
  }
  if (var == "TOTALSPIFFS") {
   return humanReadableSize(SPIFFS.totalBytes());
```

```
  }
}
String readFile(fs::FS &fs, const char * path){
  Serial.printf("Reading file: %s\r\n", path);
  File file = fs.open(path, "r");
  if(!file || file.isDirectory()){
    Serial.println("- empty file or failed to open file");
    return String();
  }
  Serial.println("- read from file:");
  String fileContent;
  while(file.available()){
    fileContent+=String((char)file.read());
  }
  Serial.println(fileContent);
  return fileContent;
}
void writeFile(fs::FS &fs, const char * path, const char * message){
  Serial.printf("Writing file: %s\r\n", path);
  File file = fs.open(path, "w");
  if(!file){
    Serial.println("- failed to open file for writing");
    return;
  }
  if(file.print(message)){
    Serial.println("- file written");
  } else {
    Serial.println("- write failed");
  }
```

```
}
void handleDoUpdate(AsyncWebServerRequest *request, const String& filename, size_t index, uint8_t
*data, size_t len, bool final)
{
  if (!index)
  {
    Serial.println("Update");
    content_len = request->contentLength();
    // if filename includes spiffs, update the spiffs partition
    int cmd = (filename.indexOf("spiffs") > -1) ? U_SPIFFS : U_FLASH;
    if (!Update.begin(UPDATE_SIZE_UNKNOWN, cmd))
    {
      Update.printError(Serial);
    }
  }

  if (Update.write(data, len) != len)
  {
    Update.printError(Serial);
#ifdef ESP8266
  }
  else
  {
    Serial.printf("Progress: %d%%\n", (Update.progress() * 100) / Update.size());
#endif
  }

  if (final)
  {
```

```
    AsyncWebServerResponse *response = request->beginResponse(302, "text/plain", "Please wait while the device reboots");
    response->addHeader("Refresh", "20");
    response->addHeader("Location", "/");
    request->send(response);
    if (!Update.end(true))
    {
      Update.printError(Serial);
    }
    else
    {
      Serial.println("Update complete");
      Serial.flush();
      ESP.restart();
    }
  }
}
static void printProgress(size_t prg, size_t sz)
{
  Serial.printf("Progress: %d%%\r\n", (prg * 100) / content_len);
  fwUpdateProgress = (prg * 100) / content_len;
}
String listFiles(bool ishtml) {
  String returnText = "";
  Serial.println("Listing files stored on SPIFFS");
  File root = SPIFFS.open("/");
  File foundfile = root.openNextFile();
  if (ishtml) {
    returnText+="<table><tr><thalign='left'>Name</th><th align='left'>Size</th><th></th><th></th></tr>";
```

```
  }
  while (foundfile) {
    if (ishtml) {
      returnText += "<tr align='left'><td>" + String("<a href='") + String(foundfile.name()) + String("'>") + String(foundfile.name()) + String("</a>") + String("</td><td>") + humanReadableSize(foundfile.size()) + "</td>";
      returnText += "<td><button onclick=\"downloadDeleteButton(\'" + String(foundfile.name()) + "\', \'download\')\">Download</button>";
      returnText += "<td><button onclick=\"downloadDeleteButton(\'" + String(foundfile.name()) + "\', \'delete\')\">Delete</button></tr>";
    } else {
      returnText += "File: "  + String(foundfile.name()) + " Size: " + humanReadableSize(foundfile.size()) + "\n";
    }
    foundfile = root.openNextFile();
  }
  if (ishtml) {
    returnText += "</table>";
  }
  root.close();
  foundfile.close();
  return returnText;
}

// Make size of files human readable
// source: https://github.com/CelliesProjects/minimalUploadAuthESP32
String humanReadableSize(const size_t bytes) {
  if (bytes < 1024) return String(bytes) + " B";
  else if (bytes < (1024 * 1024)) return String(bytes / 1024.0) + " KB";
  else if (bytes < (1024 * 1024 * 1024)) return String(bytes / 1024.0 / 1024.0) + " MB";
  else return String(bytes / 1024.0 / 1024.0 / 1024.0) + " GB";
```

```
}
//handler για το ανέβασμα αρχείων στο SPIFFS μέσω της default ιστοσελίδας
void handleUpload(AsyncWebServerRequest *request, String filename, size_t index, uint8_t *data, size_t len, bool final) {
 // make sure authenticated before allowing upload

 if (!index) {
  logmessage = "Upload Start: " + String(filename);
  // open the file on first call and store the file handle in the request object
  request->_tempFile = SPIFFS.open("/" + filename, "w");
  Serial.println(logmessage);
 }

 if (len) {
  // stream the incoming chunk to the opened file
  request->_tempFile.write(data, len);
  logmessage = "Writing file: " + String(filename) + " index=" + String(index) + " len=" + String(len);
  Serial.println(logmessage);
 }

 if (final) {
  logmessage = "Upload Complete: " + String(filename) + ",size: " + String(index + len);
  // close the file handle as the upload is now done
  request->_tempFile.close();
  Serial.println(logmessage);
  request->redirect("/");
 }
}
```

Κώδικας ενεργοποίησης και αποστολής των servers

```
void configureWebServers() {
   server.onFileUpload(handleUpload);
server.on("/upload",HTTP_GET,[](AsyncWebServerRequest * request){
 if(!request->authenticate(username, pass))
      return request->requestAuthentication();
  request->send_P(200, "text/html", index_html,processor);
 });
server.on("/logged-out", HTTP_GET, [](AsyncWebServerRequest * request) {
     if(!request->authenticate(username, pass))
      return request->requestAuthentication();
    request->send_P(401, "text/html", logout_html,processor);
  });
server.on("/listfiles", HTTP_GET, [](AsyncWebServerRequest * request){
    if(!request->authenticate(username, pass))
      return request->requestAuthentication();
     request->send(200, "text/plain", listFiles(true));
  });
server.on("/file", HTTP_GET, [](AsyncWebServerRequest * request) {
   if(!request->authenticate(username, pass))
      return request->requestAuthentication();
  if (request->hasParam("name") && request->hasParam("action")) {
        const char *fileName = request->getParam("name")->value().c_str();
        const char *fileAction = request->getParam("action")->value().c_str();
if (!SPIFFS.exists(fileName)) {

          request->send(400, "text/plain", "ERROR: file does not exist");
        } else {
```

```
        if (strcmp(fileAction, "download") == 0) {
          logmessage += " downloaded";
          request->send(SPIFFS, fileName, "application/octet-stream");
        } else if (strcmp(fileAction, "delete") == 0) {
          logmessage += " deleted";
          SPIFFS.remove(fileName);
          request->send(200, "text/plain", "Deleted File: " + String(fileName));
        } else {
          logmessage += " ERROR: invalid action param supplied";
          request->send(400, "text/plain", "ERROR: invalid action param supplied");
        }
        Serial.println(logmessage);
      }
    } else {
      request->send(400, "text/plain", "ERROR: name and action params required");
    }
   });
server.on("/", HTTP_GET,[](AsyncWebServerRequest * request){
 if(!request->authenticate(username, pass))
     return request->requestAuthentication();
  request->send(SPIFFS,"/ptixiaki_main.html");
 });
server.on("/BootLoaderIndex", HTTP_GET, [] (AsyncWebServerRequest * request) {
   if(!request->authenticate(username, pass))
     return request->requestAuthentication();
   request->send(SPIFFS, "/ptixiaki_boot.html");
  });
server.on("/doUpdate", HTTP_POST,
  [](AsyncWebServerRequest * request) {},
```

```
  [](AsyncWebServerRequest * request, const String & filename, size_t index, uint8_t *data,
     size_t len, bool final) {
    handleDoUpdate(request, filename, index, data, len, final);
  });
server.on("/update", HTTP_POST,
  [](AsyncWebServerRequest * request) {},
  [](AsyncWebServerRequest * request, const String & filename, size_t index, uint8_t *data,
     size_t len, bool final) {
    handleDoUpdate(request, filename, index, data, len, final);
  });
server.on("/mc", HTTP_GET, [](AsyncWebServerRequest *request){

    request->send_P(200, "text/plain", readMAC().c_str());
  });
server.on("/ip", HTTP_GET, [](AsyncWebServerRequest *request){

    request->send_P(200, "text/plain", readIP().c_str());
  });
server.on("/photo", HTTP_GET, [](AsyncWebServerRequest *request){

    request->send_P(200, "text/plain", readLDR().c_str());
  });
server.on("/txt", HTTP_GET, [](AsyncWebServerRequest *request){

    request->send_P(200, "text/plain", readText().c_str());
  });
server.on("/get", HTTP_GET, [] (AsyncWebServerRequest *request) {
 String inputMessage;
 String inputMessage1;
```

```
  String message1= "[Message]" "\n";
  String message2= "Text= ";
  if (request->hasParam(PARAM_STRING1)){
    inputMessage1 = request->getParam(PARAM_STRING1)->value();
    inputMessage  = message1 +message2+inputMessage1 ;
    writeFile(SPIFFS, "/config.ini", inputMessage.c_str());
  }
   request->send(200, "text/text", inputMessage);
 });
 server.on("/restart1", HTTP_GET, [](AsyncWebServerRequest * request){
  if(!request->authenticate(username, pass))
     return request->requestAuthentication();
   request->send(200, "text/plain","restarting");
   Serial.println("Restarting ESP32.Please wait for 10 sec");
   delay(10000);
   ESP.restart();
});
server.on("/restart", HTTP_GET, [](AsyncWebServerRequest * request){
 if(!request->authenticate(username, pass))
     return request->requestAuthentication();
   request->send(200, "text/plain","restarting");
   Serial.println("Restarting ESP32.Please wait for 5 sec");
   delay(5000);
   ESP.restart();

});
 server.onNotFound([](AsyncWebServerRequest * request) {
  request->send(404);
});}
```

Κώδικας ιστοσελίδας ptixiaki_main

```
<!DOCTYPE HTTP/1.1 200 OK>

<!DOCTYPE html><html>

<head>

<title>My ESP32</title>

<meta  name='viewport' content='width=device-width, initial-scale=1'>

<link rel='icon' href='data:,'>

</head><div1 id="mySidenav" class="sidenav">

  <a href="javascript:void(0)" class="closebtn" onclick="closeNav()">&times;</a>

  <a href='/logged-out'>Logout</a>

  <a href='#bottom'>Contact</a>

  <a href='https://www.iee.ihu.gr/'>University Website</a>

<a href='/upload'>Upload Tool</a>

<a href='/BootLoaderIndex'>Firmware Tool</a>

</div1>

<div1 id="main">

  <span style="font-size:30px;cursor:pointer" onclick="openNav()">&#9776; Menu</span>

</div1>

<a href='/'><img border='0' src='https://m.902.gr/sites/default/files/styles/902-grid-8/public/MediaV2/20200206/dipae.jpg?itok=kOvwnoTx' width='250' height='150'></a>

<div id=plaisio><div2 class='a'>

<h1>TFCE32</h1>

<body><p style='font-size:20px'>(Take Full Control of Esp32)</p>

<form name=BUILD_REVISIONS><h1>Version Panel</h1><p>ESP32: 1.0.0</p></form>

<form name=Brightness><h1>Brightness Level</h1><p>Currnent Brightness is at <span id="Pvalue">%VALUEP%</span>%</p></form>

<form  action="/get" id = "TxtFile" target="hidden-form"><h1>LCD Control Panel</h1>

<input type="text" id="TxtFile" value="" name="inputString1">

<input type="submit" id = 'bt' class='button' value='Send Message' onclick="readValues()" /></form>

<form name =Text_Reading><h1>Readings Panel</h1>
```

<h2>Text reading:</h2> %TEXT% </form>

<button class='buttons' onclick="restart()">Restart LCD</button>

<form name=MAC_IP><h1>MAC - IP Panel</h1><h2>MAC ADDRESS</h2>

MAC: %mac%

<h2>IP ADDRESS</h2>

IP: %Ip%

</form>

<div3 id= bottom><div2 class='a'>

Contact Information: Email: ele516059@el.teithe.gr </div2><button onclick='topFunction()' id='myBtn' title='Go to top'></button>

<meta name='viewport' content='width=device-width, initial-scale=1'><link rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css'><meta name='viewport' content='width=device-width, initial-scale=1'><link rel='icon' href='data:,'><style>

div2.a {

  text-align: center;

}

body{background:White;font-family:sans-serif;font-size:14px;color:black; width: auto;}

#file-input,input{width:100%;height:44px;border-radius:4px;margin:10px auto;font-size:15px}

#file-input{padding:0;border:1px solid #ddd;line-height:44px;text align:left;display:block;cursor:pointer}

form{background:LightGrey;max-width:445px;margin:75px auto;padding:30px;border-radius:10px;text-align:center}

div{background:#33658A ;max-width:500px;margin:75px auto;padding:100px;border-radius:100px;}

.button{background:#B3001E;color:#fff;cursor:pointer}

img {display: block;margin-left: auto;margin-right: auto;}

#myBtn {

  display: none;

  position: fixed;

```
  bottom: 20px;
  right: 0;
  z-index: 99;
  font-size: 18px;
  border: none;
  outline: none;
  background-color: Black;
  color: white;
  cursor: pointer;
  padding: 15px;
  border-radius: 30px;
}
#myBtn:hover {
  background-color: Grey;
}
.sidenav {
  height: 100%;
  width: 0;
  position: fixed;
  z-index: 1;
  top: 0;
  left: 0;
  background-color: #0C090B;
  overflow-x: hidden;
  transition: 0.5s;
  padding-top: 60px;
}

.sidenav a {
```

```
  padding: 8px 8px 8px 32px;
  text-decoration: none;
  font-size: 25px;
  color: #818181;
  display: block;
  transition: 0.3s;
}

.sidenav a:hover {
  color: #f1f1f1;
}

.sidenav .closebtn {
  position: absolute;
  top: 0;
  right: 25px;
  font-size: 36px;
  margin-left: 50px;
}

#main {
  transition: margin-left .5s;
  padding: 16px;
}

@media screen and (max-height: 450px) {
  .sidenav {padding-top: 15px;}
  .sidenav a {font-size: 18px;}
}
```

```
.buttons {
  display: block;
  width: 100%;
  border: yes;
  background-color: #B3001E;
  color:#fff;
  padding: 14px 28px;
  font-size: 16px;
  cursor: pointer;
  text-align: center;
}
</style><script>
function readValues() {
      alert("Saved values to ESP SPIFFS");
      setTimeout(function(){ document.location.reload(false); }, 500);
    }

//Get the button
var mybutton = document.getElementById("myBtn");

// When the user scrolls down 20px from the top of the document, show the button
window.onscroll = function() {scrollFunction()};

function scrollFunction() {
  if (document.body.scrollTop > 20 || document.documentElement.scrollTop > 20) {
    mybutton.style.display = "block";
  } else {
    mybutton.style.display = "none";
  }
```

```
}

// When the user clicks on the button, scroll to the top of the document
function topFunction() {
  document.body.scrollTop = 0;
  document.documentElement.scrollTop = 0;
}
</script>



<script>
function openNav() {
  document.getElementById("mySidenav").style.width = "250px";
  document.getElementById("main").style.marginLeft = "250px";
}

function closeNav() {
  document.getElementById("mySidenav").style.width = "0";
  document.getElementById("main").style.marginLeft= "0";
}
</script>
<script>
setInterval(function() {
  // Call a function repetatively with 2 Second interval
  getT();
}, 1000);
function getT() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
```

```
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("TXT").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "txt", true);
  xhttp.send();
}
setInterval(function() {
  // Call a function repetatively with 2 Second interval
  getP();
}, 1000); //1000mSeconds update rate

function getP() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("Pvalue").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "photo", true);
  xhttp.send();
}
setInterval(function() {
  // Call a function repetatively with 2 Second interval
  getM();
}, 1000);
```

```
function getM() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("MC").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "mc", true);
  xhttp.send();
}
</script>
<script>
setInterval(function() {
  // Call a function repetatively with 2 Second interval
  getI();
}, 1000);

function getI() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("IP").innerHTML =this.responseText;
    }
  };
  xhttp.open("GET", "ip", true);
  xhttp.send();
}
function restart(){
```

```
var xhr = new XMLHttpRequest();
  xhr.open("GET", "/restart", true);
  xhr.send();
  setTimeout(function(){ window.open("/","_self"); },5000);
}
</script>
```

Κώδικας ιστοσελίδας ptixiaki_boot

```
<title>OTA Update</title>
<meta  name='viewport' content='width=device-width, initial-scale=1'>
<div2 id="mySidenav" class="sidenav">
  <a href="javascript:void(0)" class="closebtn" onclick="closeNav()">&times;</a>
   <a href='/logged-out'>Logout</a>
<a href='/upload'>Upload Tool</a>
</div2>
<div2 id="main">
  <span style="font-size:30px;cursor:pointer" onclick="openNav()">&#9776; Menu</span>
</div2>
<a href='/'><img border='0' src='https://m.902.gr/sites/default/files/styles/902-grid-8/public/MediaV2/20200206/dipae.jpg?itok=kOvwnoTx' width='250' height='150'></a><script src='https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js'></script><form method='POST' action='#' enctype='multipart/form-data' id='upload_form'><input type='file' class='input' name='update' id='file' onchange='sub(this)' style=display:none><label id='file-input' for='file'> Choose file...</label><input type='submit' class=btn value='Update'><div1 id='prg'>progress: 0%</div1><br><div id='prgbar'><div id='bar'></div></div><br>


<script>function sub(obj){var fileName = obj.value.split('\\');document.getElementById('file-input').innerHTML = '  '+ fileName[fileName.length-1];};$('form').submit(function(e){e.preventDefault();var form = $('#upload_form')[0];var data = new FormData(form);$.ajax({url: '/update',type: 'POST',data: data,contentType: false,processData:false,xhr: function() {var xhr = new window.XMLHttpRequest();xhr.upload.addEventListener('progress', function(evt) {if (evt.lengthComputable) {var per = evt.loaded / evt.total;$('#prg').html('progress: ' + Math.round(per*100) + '%');$('#bar').css('width',Math.round(per*100) + '%');}}, false);return xhr;},success:function(d, s) {console.log('success!') },error: function (a, b, c) {}});});</script><script>

function openNav() {
```

```
  document.getElementById("mySidenav").style.width = "250px";
  document.getElementById("main").style.marginLeft = "250px";
}

function closeNav() {
  document.getElementById("mySidenav").style.width = "0";
  document.getElementById("main").style.marginLeft= "0";
}
</script>

<style>
.sidenav {
  height: 100%;
  width: 0;
  position: fixed;
  z-index: 1;
  top: 0;
  left: 0;
  background-color: #0C090B;
  overflow-x: hidden;
  transition: 0.5s;
  padding-top: 60px;
}

.sidenav a {
  padding: 8px 8px 8px 32px;
  text-decoration: none;
  font-size: 25px;
  color: #818181;
```

```
  display: block;
  transition: 0.3s;
}

.sidenav a:hover {
  color: #f1f1f1;
}

.sidenav .closebtn {
  position: absolute;
  top: 0;
  right: 25px;
  font-size: 36px;
  margin-left: 50px;
}

#main {
  transition: margin-left .5s;
  padding: 16px;
}

@media screen and (max-height: 450px) {
  .sidenav {padding-top: 15px;}
  .sidenav a {font-size: 18px;}
}

#file-input,input{width:100%;height:44px;border-radius:4px;margin:10px auto;font-
size:15px}input{background:#E2E2E2;border:0;padding:0 15px}body{background:White;font-
family:sans-serif;font-size:14px;color:black;}#file-input{background:#E2E2E2;padding:0;border:1px
solid #ddd;line-height:44px;text-align:left;display:block;cursor:pointer}#bar,#prgbar{background-
color:white;border-radius:10px}#bar{background-
```

color:#FE4134;width:0%;height:10px}form{background:#33658A;max-width:500px;margin:75px auto;padding:30px;border-radius:5px;text-align:center}.btn{background:#B3001E;color:#fff;cursor:pointer}img {display: block;margin-left: auto;margin-right: auto;}</style>

Κώδικας ιστοσελίδας index_html

```
<html lang="en">

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Add icon library -->

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<title>Upload Tool</title>

<style>

.buttons {

  background-color: #B3001E;

  border: none;

  color: white;

  padding: 10px 15px;

  text-align: center;

  text-decoration: none;

  display: inline-block;

  font-size: 16px;

}

div{background:#33658A;max-width:500px;margin:75px auto;padding:30px;border-radius:5px;text-align:center}

form.a{background:#33658A;max-width:445px;margin:75px auto;padding:30px;border-radius:5px;text-align:center}

form.b{background:#B6B3A5;max-width:350px;margin:75px auto;padding:20px;border-radius:5px;text-align:center}

body{background:White;font-family:sans-serif;font-size:14px;color:white; width: auto; text-align:center}

.btn {

  background-color: black;
```

```
  border: none;
  color: white;
  padding: 12px 16px;
  font-size: 16px;
  cursor: pointer;
}

/* Darker background on mouse-over */
.btn:hover {
  background-color: #33658A;
}
</style>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta charset="UTF-8">
</head>
<body>
 <button class="btn" onclick="homepage()"><i class="fa fa-home"></i> Home</button>
<div>

<p>Firmware: %FIRMWARE%</p>
  <p>Free Storage: <span id="freespiffs">%FREESPIFFS%</span> | Used Storage: <span id="usedspiffs">%USEDSPIFFS%</span>|TotalStorage:<span id="totalspiffs">%TOTALSPIFFS%</span></p>
  <p>
  <button class='buttons' onclick="logoutButton()">Logout</button>
  <button class='buttons' onclick="restart1()">Restart ESP32</button>
  <button class='buttons' onclick='listFilesButton()'>List Files</button>
  <button class='buttons' onclick='showUploadButtonFancy()'>Upload Files</button>
  </p>
  <p id="status"></p>
```

```
  <p id="detailsheader"></p>
  <p id="details"></p>
 </div>
<script>
function homepage() {
  location.replace("/")
}
function restart1(){
var xhr = new XMLHttpRequest();
  xhr.open("GET", "/restart1", true);
  xhr.send();
  setTimeout(function(){ window.open("/upload","_self"); }, 10000);
}
function logoutButton() {
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/logout", true);
  xhr.send();
  setTimeout(function(){ window.open("/logged-out","_self"); }, 1000);
}
function listFilesButton() {
  xmlhttp=new XMLHttpRequest();
  xmlhttp.open("GET", "/listfiles", false);
  xmlhttp.send();
  document.getElementById("detailsheader").innerHTML = "<h3>Files<h3>";
  document.getElementById("details").innerHTML = xmlhttp.responseText;
}
function downloadDeleteButton(filename, action) {
  var urltocall = "/file?name=" + filename + "&action=" + action;
  xmlhttp=new XMLHttpRequest();
```

```
  if (action == "delete") {
    xmlhttp.open("GET", urltocall, false);
    xmlhttp.send();
    document.getElementById("status").innerHTML = xmlhttp.responseText;
    xmlhttp.open("GET", "/listfiles", false);
    xmlhttp.send();
    document.getElementById("details").innerHTML = xmlhttp.responseText;
  }
  if (action == "download") {
    document.getElementById("status").innerHTML = "";
    window.open(urltocall,"_blank");
  }
}
function showUploadButtonFancy() {
  document.getElementById("detailsheader").innerHTML = "<h3>Upload HTML Files<h3>"
  document.getElementById("status").innerHTML = "";
  var uploadform = "<form method = \"POST\" action = \"/\" enctype=\"multipart/form-data\"><input type=\"file\" name=\"data\"/><input type=\"submit\" name=\"upload\" value=\"Upload\" title = \"Upload File\"></form>"
  document.getElementById("details").innerHTML = uploadform;
  var uploadform =
  "<form class= 'b'><form id=\"upload_form\" enctype=\"multipart/form-data\" method=\"post\">" +
  "<input type=\"file\" name=\"file1\" id=\"file1\" onchange=\"uploadFile()\"><br>" +
  "<progress id=\"progressBar\" value=\"0\" max=\"100\" style=\"width:300px;\"></progress>" +
  "<h3 id=\"status\"></h3>" +
  "<p id=\"loaded_n_total\"></p>" +
  "</form>";
  document.getElementById("details").innerHTML = uploadform;
}
function _(el) {
```

```
  return document.getElementById(el);
}
function uploadFile() {
  var file = _("file1").files[0];
  // alert(file.name+" | "+file.size+" | "+file.type);
  var formdata = new FormData();
  formdata.append("file1", file);
  var ajax = new XMLHttpRequest();
  ajax.upload.addEventListener("progress", progressHandler, false);
  ajax.addEventListener("load", completeHandler, false); // doesnt appear to ever get called even upon success
  ajax.addEventListener("error", errorHandler, false);
  ajax.addEventListener("abort", abortHandler, false);
  ajax.open("POST", "/");
  ajax.send(formdata);
}
function progressHandler(event) {
  //_("loaded_n_total").innerHTML = "Uploaded " + event.loaded + " bytes of " + event.total; // event.total doesnt show accurate total file size
  _("loaded_n_total").innerHTML = "Uploaded " + event.loaded + " bytes";
  var percent = (event.loaded / event.total) * 100;
  _("progressBar").value = Math.round(percent);
  _("status").innerHTML = Math.round(percent) + "% uploaded... please wait";
  if (percent >= 100) {
    _("status").innerHTML = "Please wait, writing file to filesystem";
  }
}
function completeHandler(event) {
  _("status").innerHTML = "Upload Complete";
  _("progressBar").value = 0;
```

```
  xmlhttp=new XMLHttpRequest();
  xmlhttp.open("GET", "/listfiles", false);
  xmlhttp.send();
  document.getElementById("status").innerHTML = "File Uploaded";
  document.getElementById("detailsheader").innerHTML = "<h3>Files<h3>";
  document.getElementById("details").innerHTML = xmlhttp.responseText;
}
function errorHandler(event) {
  _("status").innerHTML = "Upload Failed";
}
function abortHandler(event) {
  _("status").innerHTML = "inUpload Aborted";
}
</script>
</body>
</html>
```

Κώδικας ιστοσελίδας logout

```
<!DOCTYPE HTML>
<html lang="en">
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta charset="UTF-8">
</head>
<body>
  <p><a href="/">Go to home page</a></p>
</body>
</html>
```