ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

# ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

# ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

## ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

**«IoT εφαρμογή Μετεωρολογικού Σταθμού με την πλατφόρμα ESP32 για συλλογή δεδομένων από αισθητήρα περιβάλλοντος και παρουσίαση μέσω δυναμικής ιστοσελίδας»**


**Του φοιτητή**
**Σκουλαρόπουλου Ιωάννη**
**Αρ. Μητρώου: 2019243**

**Επιβλέπων**
**Γιακουμής Άγγελος**
**Επίκουρος Καθηγητής**


**Ημερομηνία 23 / 05 / 2024**

Τίτλος Δ.Ε. “ IoT εφαρμογή Μετεωρολογικού Σταθμού με την πλατφόρμα ESP32 για συλλογή δεδομένων από αισθητήρα περιβάλλοντος και παρουσίαση μέσω δυναμικής ιστοσελίδας ”

Κωδικός Δ.Ε. 24145

Ονοματεπώνυμο φοιτητή: Σκουλαρόπουλος Ιωάννης

Ονοματεπώνυμο εισηγητή: Γιακουμής Άγγελος

Ημερομηνία ανάληψης Δ.Ε. 11/03/2024

Ημερομηνία περάτωσης Δ.Ε. 23/05/2024

*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.*

*«Στην σύζυγό μου Κωνσταντία και στα παιδιά μου Ανδρέα, Αλέξιο και Ιφιγένεια, για την υπομονή και την αγάπη τους.»*

# Πρόλογος

Ονομάζομαι Σκουλαρόπουλος Ιωάννης και έχοντας ήδη το πτυχίο του Ηλεκτρονικού Μηχανικού Τ.Ε του Τ.Ε.Ι Θεσσαλονίκης, η εγγραφή μου στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε, έγινε τον Μάρτιο του 2020, μετά από επιτυχείς κατατακτήριες εξετάσεις τον Δεκέμβριο του 2019. Φτάνοντας λοιπόν στο τέλος της νέας μου προσπάθειας, επέλεξα το συγκεκριμένο θέμα διπλωματικής εργασίας γιατί θεωρώ ότι συνδυάζει με έναν πολύ αποδοτικό τρόπο την επιστήμη των Ηλεκτρονικών και της Πληροφορικής. Για την εκπόνηση της εργασίας χρειάζονται γνώσεις από πλήθος μαθημάτων που διδάσκονται στο τμήμα και είναι ο Δομημένος Προγραμματισμός (1102), ο Αντικειμενοστρεφής Προγραμματισμός (1205), οι Γλώσσες και Τεχνολογίες Ιστού (1405), τα Συστήματα Διαχείρισης Βάσεων Δεδομένων (1401), η Εισαγωγή στα Λειτουργικά Συστήματα (1403), τα Ενσωματωμένα Συστήματα (1602), το Διαδίκτυο των Πραγμάτων (1803) και η Ανάπτυξη Διαδικτυακών Συστημάτων και Εφαρμογών (1941). Το όφελος, λοιπόν, από την εκπόνηση της εργασίας είναι η δυνατότητα να συνδυάσω τις γνώσεις από τα παραπάνω μαθήματα για την δημιουργία μιας IoT εφαρμογής.

# Περίληψη

Η εργασία αφορά μια IoT εφαρμογή Μετεωρολογικού Σταθμού και έχει σχεδιαστεί για παρακολούθηση των τιμών της Θερμοκρασίας, της Υγρασίας και της Ατμοσφαιρικής Πίεσης, που μετράει ο αισθητήρας περιβάλλοντος BME280. Καρδιά του συστήματος είναι η υπολογιστική πλατφόρμα ESP32, που διαβάζει τις τιμές των μεγεθών από τον BME280, και τις αποστέλλει μέσω HTTPS POST αιτημάτων για αποθήκευση στη βάση δεδομένων του server που χρησιμοποιεί η εφαρμογή. Στη συνέχεια μέσω Web API και συγκεκριμένα στο Frontend οι χρήστες μπορούν να δουν τις 30 πιο πρόσφατες μετρήσεις μέσω διαγραμμάτων, που έχουν σχεδιαστεί με την βιβλιοθήκη Highcharts της Javascript, σε μια δυναμική ιστοσελίδα. Για τον προγραμματισμό στο Backend χρησιμοποιήθηκε η γλώσσα PHP. Η εφαρμογή υποστηρίζει ενημέρωση των χρηστών σε περίπτωση μη αποθήκευσης νέων δεδομένων για ορισμένο χρονικό διάστημα, και ειδικά για τον administrator ενημέρωση και μέσω email. Ακόμη δίνεται η δυνατότητα παραμετροποίησης της κατασκευής και σύνδεσης του ESP32 σε νέο δίκτυο Wi-Fi ή δίκτυο κινητής τηλεφωνίας μέσω ενός Access Point.

« IoT Weather Station Application using the ESP32 platform for collecting data from an environmental sensor and presenting it through a dynamic website »

«Ioannis Skoularopoulos»

## Abstract

"The project concerns an IoT application for a Weather Station designed to monitor the values of Temperature, Humidity, and Atmospheric Pressure, measured by the BME280 environmental sensor. The heart of the system is the ESP32 computational platform, which reads the values of the parameters from the BME280 and sends them via HTTPS POST requests for storage in the application's server database. Subsequently, through a Web API, specifically in the frontend, users can view the 30 most recent measurements via charts, designed using the Highcharts JavaScript library, on a dynamic website. For backend programming, PHP language was used. The application supports user notifications in case of failure to store new data for a certain period, and specifically for the administrator, notification via email. Additionally, users have the option to configure the setup and connect the ESP32 to a new Wi-Fi network or a mobile network via an Access Point."

# Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον επιβλέποντα καθηγητή μου, κ. Γιακουμή Άγγελο, για την αμέριστη συμπαράστασή του σε όλη τη διάρκεια εκπόνησης της εργασίας.

# Περιεχόμενα

# Κατάλογος Σχημάτων

# Κατάλογος Πινάκων

# Συντομογραφίες

| | |
|---|---|
| Δ.Ε. | Διπλωματική Εργασία |
| ΔΙΠΑΕ | Διεθνές Πανεπιστήμιο Ελλάδος |
| Administrator | Ο δημιουργός της εφαρμογής |
| AP | Access Point |
| API | Application Programming Interface |
| BME280 | Ο αισθητήρας περιβάλλοντος BME280 |
| ES | Embedded Systems |
| ESP32 | Υπολογιστική Πλατφόρμα ESP32 |
| ESP32_WS | ESP32 Weather Station, είναι η εφαρμογή της παρούσης εργασίας |
| IoT | Internet of Thinks |
| I2C | Το πρωτόκολλο I2C |
| LCD 16x2 RGB | Η οθόνη 16x2 Character LCD – RGB 3.3V / 5V της WAVESHARE |
| TI | Texas Instruments |
| User | Ο χρήστης της εφαρμογής |

# Κεφάλαιο 1ο: Εισαγωγή - IoT – ES - WebAPI

## 1.1 Εισαγωγή

Το IoT έχει επανασχεδιάσει την καθημερινότητά μας, επιτρέποντάς μας να παρακολουθούμε αλλά και να ελέγχουμε από απόσταση μέσω του Διαδικτύου, τις «έξυπνες» συσκευές που μας περιβάλλουν. Η παρούσα εργασία πραγματεύεται την σχεδίαση και την κατασκευή μιας IoT εφαρμογής Μετεωρολογικού Σταθμού -η οποία πλέον θα αναφέρεται ως ESP32_WS- συνδυάζοντας τομείς τεχνολογίας όπως:

- Embedded Systems και
- Web API

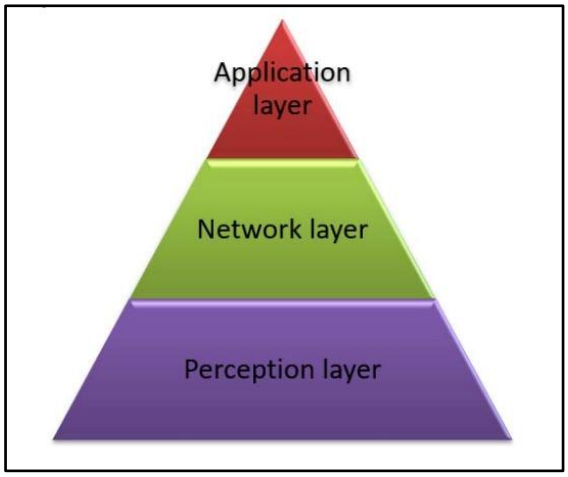
Αρχικά και στο 1ο Κεφάλαιο θα μιλήσουμε για το IoT, την αρχιτεκτονική των IoT εφαρμογών, τα χαρακτηριστικά τους, αλλά και τις εφαρμογές του IoT στην καθημερινότητά μας. Θα μιλήσουμε επίσης για την γενική δομή, τα χαρακτηριστικά και τις εφαρμογές των Ενσωματωμένων Συστημάτων -Embedded Systems- και τέλος θα δούμε το Web API -Front End και Back End-. Στη συνέχεια στο κεφάλαιο 2 δίνεται η γενική περιγραφή του συστήματος. Στο κεφάλαιο 3 θα δούμε τις γλώσσες προγραμματισμού που χρησιμοποιούνται για την εκπόνηση της εργασίας, δίνοντας και από ένα παράδειγμα κώδικα για την κάθε μια, αλλά και τα κατάλληλα εργαλεία – προγράμματα που χρησιμοποιήθηκαν κατά την ανάπτυξη της. Στα κεφάλαια 4, 5, και 6 θα μιλήσουμε για τα τρία επίπεδα της IoT εφαρμογής -Perception Layer, Network Layer και Application Layer- ενώ στο κεφάλαιο 7 θα δούμε το κατασκευαστικό κομμάτι, το ηλεκτρονικό κύκλωμα και τις συνδέσεις των εξαρτημάτων. Τέλος, στο κεφάλαιο 8 η εργασία τελειώνει με τα συμπεράσματα και τις προτάσεις για επέκταση – βελτίωσή της.

## 1.2 Διαδίκτυο των Πραγμάτων (IoT)

Αν θέλαμε να δώσουμε έναν ορισμό για το IoT θα λέγαμε ότι είναι η εξάπλωση της τεχνολογίας χαμηλού κόστους, η οποία έχει οδηγήσει στη δυνατότητα ύπαρξης διαφόρων συσκευών και αισθητήρων που μπορούν να επικοινωνούν και να αλληλοεπιδρούν μεταξύ τους αλλά και με τους ανθρώπους δημιουργώντας ένα έξυπνο περιβάλλον [23].

### 1.2.1 Αρχιτεκτονική IoT εφαρμογών

Συνηθισμένη -αλλά όχι μοναδική- αρχιτεκτονική των εφαρμογών IoT, είναι συνήθως αυτή των τριών επιπέδων όπως παρουσιάζεται στην Εικόνα 1.1




Εικόνα 1.1: Η αρχιτεκτονική IoT τριών επιπέδων [7]

Η αρχιτεκτονική αυτή και από κάτω προς τα πάνω περιλαμβάνει:

- **Το επίπεδο αντίληψης/αίσθησης (Perception layer):** Αποτελείται από μικρές, φθηνές και ισχυρές συσκευές που είναι σε θέση να ανιχνεύουν, να επεξεργάζονται και να επικοινωνούν [7].
- **Το επίπεδο δικτύου/μετάδοσης (Network Layer):** Είναι υπεύθυνο για τη μετάδοση δεδομένων από το επίπεδο αντίληψης/ανίχνευσης στο ανώτερο επίπεδο ενώ ταυτόχρονα μπορεί να λαμβάνει δεδομένα από το ανώτερο επίπεδο.
- **Το επίπεδο εφαρμογής (Application Layer):** Παρέχει αποθήκευση δεδομένων, επεξεργασία ή ανάλυση και έξυπνες, υψηλής ποιότητας και εξατομικευμένες εφαρμογές στον τελικό χρήστη (που αναπτύσσονται μαζί με τις λειτουργίες ενδιάμεσου λογισμικού). Οι πληροφορίες που συλλέγονται από τους αισθητήρες μπορούν να αποθηκευτούν τόσο σε τοπικές όσο και σε απομακρυσμένες βάσεις δεδομένων. Το επόμενο βήμα είναι το φιλτράρισμα δεδομένων και η εξαγωγή χρήσιμων πληροφοριών [7].

### 1.2.2 Χαρακτηριστικά IoT εφαρμογών

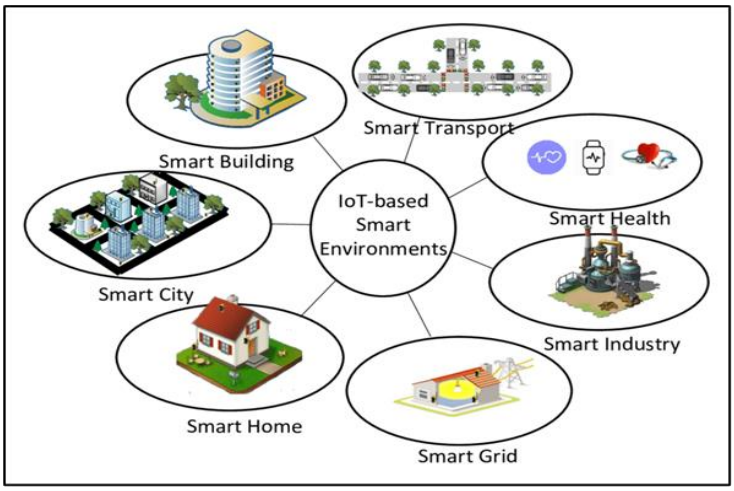
Οι IoT εφαρμογές συνήθως εμφανίζουν κάποιο/α από τα παρακάτω χαρακτηριστικά:

- **Συνδεσιμότητα:** Οι τεχνολογίες IoT θα πρέπει να επιτρέπουν τη συνδεσιμότητα πολλών και πολλαπλών συσκευών διαφόρων τύπων, καθώς και τη μετάδοση δεδομένων σε διάφορες μορφές.
- **Ασφάλεια Δεδομένων και Ατόμων:** Οι εφαρμογές IoT πρέπει να είναι ασφαλείς, προστατεύοντας τα δεδομένα που διακινούνται μέσα από αυτές και μπορεί να αφορούν π.χ χρηματικές συναλλαγές ή ιατρικό απόρρητο ασθενών. Υπάρχει λοιπόν η απαίτηση κρυπτογράφησης. Επίσης, σε κάποιες περιπτώσεις, οι IoT εφαρμογές θα πρέπει να δίνουν την δυνατότητα παρακολούθησης και ανίχνευσης της κίνησης των ατόμων, όπως για παράδειγμα στις περιπτώσεις εναλλακτικού τουρισμού ή τις περιπτώσεις θαλάσσιων καθώς και χειμερινών σπορ.
- **Ενεργειακή Αυτονομία – Συντήρηση:** Το αδύναμο σημείο μιας IoT εφαρμογής, μπορεί να είναι η διάρκεια ζωής, καθώς πολύ συχνά, οι κόμβοι αισθητήρων είναι συσκευές που λειτουργούν με μπαταρία. Η παράταση της διάρκειας ζωής του κόμβου - αισθητήρα και ολόκληρου του συστήματος IoT είναι κρίσιμης σημασίας και μπορεί να πραγματοποιηθεί με τη μείωση της κατανάλωσης π.χ sleep mode.

- **Ευκολία χρήσης:** Μια IoT εφαρμογή θα πρέπει να είναι εύκολη στη χρήση και να παρέχει ένα φιλικό περιβάλλον χρήστη.
- **Επεκτασιμότητα:** Με αυτόν τον όρο εννοούμε την ικανότητα μιας IoT εφαρμογής να μπορεί να προσαρμοστεί και να επεκταθεί στο μέλλον, ώστε να ανταποκριθεί στην εξέλιξη της τεχνολογίας.
- **Αξιοπιστία:** Είναι ίσως η βασικότερη απαίτηση και αναφέρεται τόσο στην ίδια την εφαρμογή, όσο και στα επιμέρους συστήματα που αυτή περιλαμβάνει. Με τον όρο αξιοπιστία εννοούμε την ικανότητα της εφαρμογής να λειτουργεί όπως σχεδιάστηκε και να παρέχει τα επιθυμητά αποτελέσματα.

### 1.2.3 Εφαρμογές IoT

Το IoT έχει εφαρμογές σε πολλούς τομείς της καθημερινότητας όπως μπορούμε να παρατηρήσουμε και στην Εικόνα 1.2




Εικόνα 1.2: Εφαρμογές IoT [24]

Οι τομείς αυτοί μεταξύ άλλων περιλαμβάνουν:

- **Έξυπνες Πόλεις (Smart Cities):** Μέσω του IoT, οι πόλεις μπορούν να εφαρμόσουν λύσεις για τη διαχείριση των κινήσεων, των υποδομών και των ενεργειακών πόρων για να βελτιώσουν την ποιότητα ζωής των κατοίκων.
- **Έξυπνο Σπίτι (Smart Home):** Με το IoT, τα σπίτια μπορούν να είναι εξοπλισμένα με έξυπνες συσκευές όπως θερμοστάτες, φώτα, και ηλεκτρικές συσκευές που μπορούν να ελέγχονται από απόσταση μέσω εφαρμογών.
- **Έξυπνη Υγεία και Ιατρική (Smart Health):** Στον τομέα της υγείας, το IoT μπορεί να χρησιμοποιηθεί για την παρακολούθηση της υγείας του ασθενούς, τη διαχείριση των χρόνων νοσηλείας και την παροχή αυτόματων ειδοποιήσεων σε περίπτωση έκτακτης ανάγκης.
- **Έξυπνη Βιομηχανία (Smart Industry):** Στη βιομηχανία, το IoT μπορεί να χρησιμοποιηθεί για τη βελτιστοποίηση των διεργασιών παραγωγής, την παρακολούθηση των μηχανημάτων και την πρόβλεψη βλαβών.
- **Έξυπνες Μεταφορές (Smart Transport):** Στις μεταφορές, το IoT μπορεί να χρησιμοποιηθεί για την παρακολούθηση και τον εντοπισμό φορτίων, την βελτιστοποίηση διαδρομών και μεταφορικών μέσων αλλά και την ασφάλεια των μεταφορών.

- **Έξυπνα κτίρια (Smart Building):** Στον τομέα των έξυπνων κτιρίων (Smart Buildings), το IoT μπορεί να παίξει κρίσιμο ρόλο στη βελτιστοποίηση της λειτουργίας και της διαχείρισής τους, πετυχαίνοντας ενεργειακή απόδοση, ασφάλεια, και προληπτική συντήρηση.

Ο αριθμός των έξυπνων συσκευών που δημιουργούν το Διαδίκτυο των Πραγμάτων (IoT) από το 2019 μέχρι σήμερα (2024), έχει αυξηθεί από 8,6 δισεκατομμυρία σε πάνω από 17 δισεκατομμυρία και υπολογίζεται να ξεπεράσει τα 29 δισεκατομμυρία το 2030.

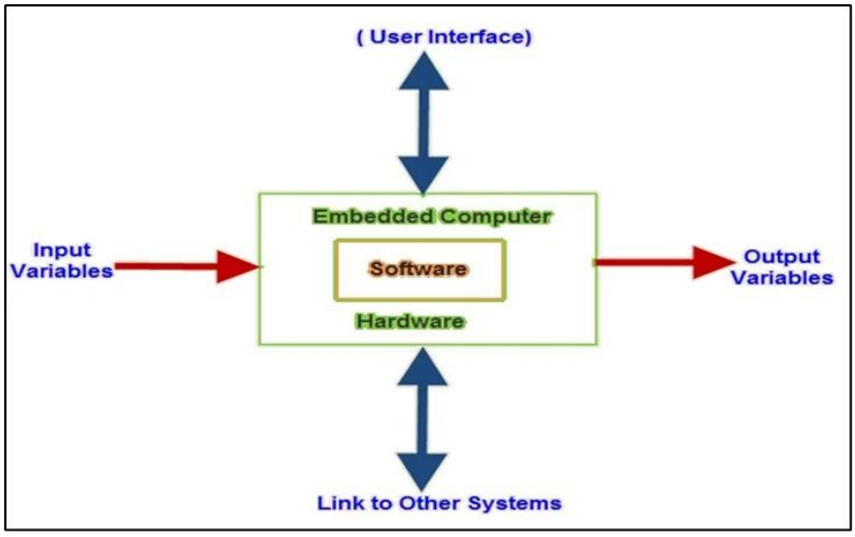
## 1.3 Embedded Systems

Παραδοσιακά, τα ES ορίζονται ως ηλεκτρονικά συστήματα που αποτελούνται από διάφορα στοιχεία υλικού και λογισμικού που μπορούν μαζί να εκτελέσουν μια αποκλειστική λειτουργία, είτε ανεξάρτητα είτε ως μέρος ενός μεγαλύτερου συστήματος. Τα τελευταία χρόνια, ο όρος «ES» τείνει να αντικατασταθεί σε δημοτικότητα από το IoT. Ωστόσο, οι συσκευές IoT είναι ουσιαστικά ES με συνδεσιμότητα στο Διαδίκτυο. Ένας άλλος σχετικός όρος είναι τα CPS (Cyber-Physical Systems) που επεκτείνει τα ES για να λάβει επιπλέον υπόψη το φυσικό περιβάλλον, κατά τη διάρκεια του σχεδιασμού του συστήματος [26].

Ανεξάρτητα από την ορολογία που χρησιμοποιείται, τα ES αντιπροσωπεύουν ένα διεπιστημονικό πεδίο που συνδυάζει πολλούς τομείς στην επιστήμη των υπολογιστών και την ηλεκτρική μηχανική. Καθώς τα ES γίνονται πιο περίπλοκα, κατανεμημένα και δικτυωμένα, οι προκλήσεις της εκπαίδευσης μελλοντικών σχεδιαστών ES που είναι ικανοί σε όλους τους κλάδους αποκτούν νέα επείγουσα ανάγκη [26].

### 1.3.1 Γενική Δομή ES

Η γενική δομή ενός ES φαίνεται παρακάτω στην Εικόνα 1.3




Εικόνα 1.3: Γενική Δομή Embedded System [28]

Όπως έχουμε αναφέρει σε ένα ενσωματωμένο σύστημα έχουμε συνδυασμένη χρήση του υλικού ES και του λογισμικού ES.

Στο υλικό του ES ανήκουν:

**Επεξεργαστής:** Ο επεξεργαστής είναι «ο εγκέφαλος» που ελέγχει ολόκληρο το σύστημα. Διαθέτει τη λογική κυκλωμάτων που ανταποκρίνεται σε βασικές εντολές και επεξεργάζεται τα δεδομένα για την επίτευξη της αναμενόμενης λειτουργικότητας.

- **Μνήμες:** Οι μνήμες υποστηρίζουν τον επεξεργαστή να κρατά τα δεδομένα προσωρινά ή μόνιμα για άμεση χρήση ή για μετέπειτα χρήση.
- **Είσοδοι/Εξοδοι (I/Os):** Ένα ενσωματωμένο σύστημα ανταποκρίνεται σε γεγονότα από τον εξωτερικό κόσμο και προκαλεί τη λήψη συγκεκριμένων μέτρων. Αυτό επιτυγχάνεται με τις εισόδους και εξόδους (I/Os) [32].
- **Παροχή Ενέργειας:** Η παροχή ενέργειας είναι ένα από τα πιο σημαντικά στοιχεία του σχεδιασμού ειδικά σε περιπτώσεις όπου η κατανάλωση ενέργειας είναι κρίσιμης σημασίας για τη λειτουργία του ES.
- **Περιφερειακές συσκευές εισόδου/εξόδου:** Αυτές οι συσκευές επιτρέπουν την αλληλεπίδραση με το εξωτερικό περιβάλλον. Περιλαμβάνουν αισθητήρες (όπως αισθητήρες θερμοκρασίας, πίεσης, επιτάχυνσης κ.λπ.) και ενεργοποιητές (όπως κινητήρες, μοτέρ, LED κ.λπ.).

Όσο αφορά το λογισμικό ενός ES, αυτό γράφεται για να εκτελεί μια συγκεκριμένη λειτουργία. Συνήθως γράφεται σε υψηλό επίπεδο και στη συνέχεια μεταγλωττίζεται για να παρέχει κώδικα που μπορεί να εκτελεστεί στο υλικό [33].

Περιλαμβάνει το σύνολο των προγραμμάτων και των διεργασιών που εκτελούνται στον επεξεργαστή του συστήματος και ελέγχουν τη λειτουργία του. Αυτό το λογισμικό συνήθως περιλαμβάνει τα εξής:

- **Εφαρμογή/Προγράμματα εφαρμογής:** Αυτά είναι τα προγράμματα που εκτελούνται στο ενσωματωμένο σύστημα και παρέχουν τις επιθυμητές λειτουργίες σύμφωνα με τις απαιτήσεις του συστήματος. Για παράδειγμα, εάν το ενσωματωμένο σύστημα είναι ένας ψηφιακός θερμοστάτης, το πρόγραμμα εφαρμογής θα περιλαμβάνει τον κώδικα που διαβάζει τη θερμοκρασία από τον αισθητήρα και ελέγχει τον κύκλωμα θέρμανσης/ψύξης ανάλογα.
- **Λειτουργικό σύστημα (RTOS):** Αν το ενσωματωμένο σύστημα χρειάζεται να λειτουργεί σε πραγματικό χρόνο ή να υποστηρίξει πολλαπλές διεργασίες, τότε χρησιμοποιεί ένα λειτουργικό σύστημα πραγματικού χρόνου (RTOS). Αυτό το λογισμικό διαχειρίζεται την εκτέλεση των διάφορων εργασιών και διασφαλίζει ότι οι χρονικές απαιτήσεις τους προστατεύονται.
- **Προγραμματιστικά εργαλεία και βιβλιοθήκες:** Αυτά τα εργαλεία και οι βιβλιοθήκες παρέχουν τα μέσα για την ανάπτυξη και τη διαχείριση του λογισμικού του ενσωματωμένου συστήματος. Αυτά μπορεί να περιλαμβάνουν περιβάλλοντα ανάπτυξης, μεταγλωττιστές, βιβλιοθήκες λογισμικού, εργαλεία διαχείρισης μνήμης κ.λπ.

Η διεπαφή χρήστη - User Interface - αντιπροσωπεύει την αλληλεπίδραση μεταξύ του χρήστη και του συστήματος και μπορεί να είναι τόσο υλικό όσο και λογισμικό στο ενσωματωμένο σύστημα, ανάλογα με την υλοποίηση και τις απαιτήσεις της εφαρμογής.

- **Υλική διεπαφή χρήστη:** Περιλαμβάνει φυσικές συσκευές όπως οθόνες LCD, οθόνες αφής, πλήκτρα, διακόπτες, LEDs κ.λπ. Αυτές οι συσκευές επιτρέπουν στον χρήστη να αλληλεπιδρά με το ES μέσω φυσικών ενεργειών, όπως πατήματα πλήκτρων ή αφή της οθόνης.
- **Λογισμική διεπαφή χρήστη:** Περιλαμβάνει γραφικά περιβάλλοντα, μενού, διάφορα είδη κειμένου, εικονίδια κ.λπ. Αυτά τα στοιχεία εμφανίζονται στην οθόνη και επιτρέπουν στον χρήστη να αλληλεπιδρά με το σύστημα μέσω ποντικιού, πληκτρολογίου, οθόνης αφής ή άλλων μεθόδων εισόδου.

Συχνά, ένα ενσωματωμένο σύστημα μπορεί να έχει και υλική και λογισμική διεπαφή χρήστη για να προσφέρει την καλύτερη εμπειρία χρήστη ανάλογα με τις ανάγκες και τις δυνατότητες του συστήματος.

Τέλος, το "link to other systems", της Εικόνας 1.3, χρησιμοποιείται για να περιγράψει το σύνολο των δυνατοτήτων για επικοινωνία με άλλα συστήματα, είτε αυτό γίνεται μέσω υλικού είτε μέσω λογισμικού.

Συνολικά θα λέγαμε ότι η σχεδίαση ενός ES καθορίζεται από διάφορες απαιτήσεις, όπως οι λειτουργικές απαιτήσεις, οι δυνατότητες επεξεργασίας, οι απαιτήσεις παροχής ισχύος, και οι απαιτήσεις αξιοπιστίας.

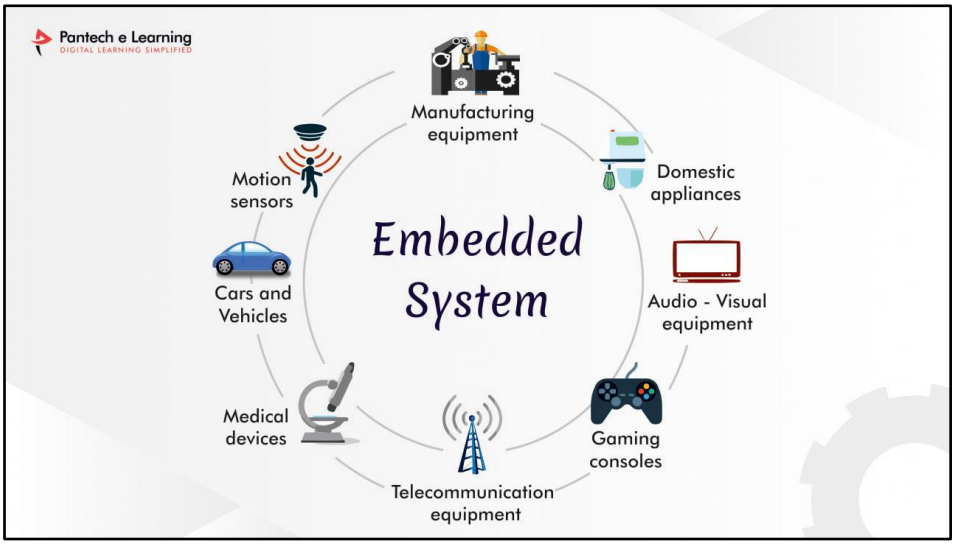
### 1.3.2 Χαρακτηριστικά των ES

Μερικά από τα βασικά χαρακτηριστικά των ES είναι:

- **Περιορισμένοι Πόροι:** Τα ES συνήθως έχουν περιορισμένους πόρους σε σχέση με τους υπολογιστές γενικού σκοπού. Αυτό μπορεί να περιλαμβάνει περιορισμένη μνήμη, επεξεργαστική ισχύ, ενέργεια και χώρο.
- **Χαμηλή Κατανάλωση Ενέργειας:** Πολλά ES πρέπει να λειτουργούν με χαμηλή κατανάλωση ενέργειας, ειδικά αν λειτουργούν με μπαταρίες ή αν χρησιμοποιούνται σε φορητές συσκευές.
- **Συνδεσιμότητα:** Πολλά ES πρέπει να είναι σε θέση να επικοινωνούν με άλλες συσκευές ή συστήματα, είτε μέσω δικτύου (όπως Ethernet, Wi-Fi) είτε μέσω άλλων πρωτοκόλλων (όπως UART, SPI, I2C).
- **Λειτουργία Πραγματικού Χρόνου:** Πολλά -όχι όλα- ES λειτουργούν σε πραγματικό χρόνο εκτελώντας εργασίες και ανταποκρινόμενα σε εισόδους εντός αυστηρών χρονικών περιορισμών. Είναι σχεδιασμένα να επεξεργάζονται πληροφορίες και να παράγουν εξόδους με ελάχιστη καθυστέρηση ή κυρίως με αυστηρές προθεσμίες. Ένα παράδειγμα ενσωματωμένου συστήματος πραγματικού χρόνου είναι ένα σύστημα αυτόματου ελέγχου σε ένα αυτοκίνητο, όπου οι αισθητήρες πρέπει να ανιχνεύουν τις αλλαγές στο περιβάλλον και να λαμβάνουν αποφάσεις για τον έλεγχο του οχήματος σε πραγματικό χρόνο.
- **Αξιοπιστία:** Τα ES πρέπει να λειτουργούν αξιόπιστα σε διάφορες συνθήκες και για μεγάλα χρονικά διαστήματα. Αποτυχίες σε αυτά τα συστήματα μπορεί να έχουν σημαντικές επιπτώσεις, ειδικά σε κρίσιμους τομείς όπως οι ιατρικές συσκευές ή οι εφαρμογές αεροδιαστημικής. Η χρήση αυστηρών διαδικασιών δοκιμών, χρησιμοποιείται για να διασφαλιστεί η αξιοπιστία και να εξασφαλιστεί η ασφάλεια [29].
- **Χαμηλό Κόστος:** Τα ενσωματωμένα συστήματα συνήθως σχεδιάζονται για να είναι οικονομικά αποδοτικά. Αυτό συμβαίνει επειδή συχνά χρησιμοποιούνται σε μεγάλες ποσότητες, και το κόστος ανά μονάδα πρέπει να είναι χαμηλό για να καθιστά το προϊόν οικονομικά βιώσιμο [30].
- **Εκτελούν συγκεκριμένες εργασίες:** Τα ενσωματωμένα συστήματα σχεδιάζονται για να εκτελούν συγκεκριμένες εργασίες ή λειτουργίες. Είναι βελτιστοποιημένα για τη συγκεκριμένη εργασία που πρέπει να εκτελέσουν, γεγονός που τα καθιστά πιο αποδοτικά και αξιόπιστα [30].

### 1.3.3 Εφαρμογές ES

Τα ES χρησιμοποιούνται σε πληθώρα εφαρμογών σε διάφορους τομείς, επιτρέποντας την αυτοματοποίηση και την εκτέλεση διαφόρων λειτουργιών. Ορισμένες από τις κύριες εφαρμογές των ενσωματωμένων συστημάτων φαίνονται στην Εικόνα 1.4




Εικόνα 1.4: Κύριες Εφαρμογές ES [27]

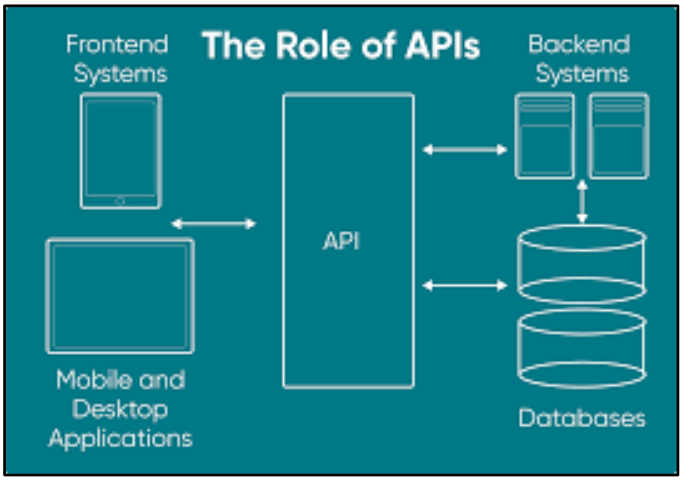
Οι εφαρμογές αυτές μεταξύ άλλων περιλαμβάνουν:

- **Αυτοκινητοβιομηχανία:** Τα ES χρησιμοποιούνται στα αυτοκίνητα για την εκτέλεση λειτουργιών όπως η αυτοματοποιημένη διαχείριση κινητήρα, η ασφάλεια, ο έλεγχος του συστήματος ενημέρωσης και ψυχαγωγίας, και η αυτόνομη οδήγηση.
- **Ηλεκτρονικές συσκευές:** Τα ES ενσωματώνονται σε ηλεκτρονικές συσκευές όπως τηλεοράσεις, κλιματιστικά, ψυγεία και πλυντήρια, για τη διευκόλυνση της λειτουργίας τους και την παροχή πρόσθετων λειτουργιών.
- **Ιατρική τεχνολογία:** Στην ιατρική, τα ES χρησιμοποιούνται για την παρακολούθηση της λειτουργίας ιατρικών συσκευών, της διάγνωσης ασθενειών αλλά και της υγείας των ασθενών.
- **Ενέργεια:** Στη βιομηχανία ενέργειας, τα ES ελέγχουν την παραγωγή και τη διανομή ενέργειας, καθώς και την εξοικονόμηση ενέργειας μέσω αυτόματων συστημάτων διαχείρισης.
- **Τηλεπικοινωνίες:** Στην τηλεπικοινωνία, τα ES χρησιμοποιούνται σε δρομολογητές, δορυφορικά συστήματα και άλλες δικτυακές συσκευές για τη διαχείριση και την επεξεργασία των δεδομένων.

## 1.4 WEB API

Ένα API είναι ένα σύνολο ορισμένων υπορουτίνων, πρωτοκόλλων και εργαλείων για τη δημιουργία λογισμικού εφαρμογών. Γενικά, πρόκειται για ένα σύνολο καθορισμένων μεθόδων επικοινωνίας μεταξύ διαφόρων συστατικών λογισμικού.

Ένα Web API είναι μια διεπαφή επεξεργασίας εφαρμογών μεταξύ ενός server και ενός browser. Όλες οι υπηρεσίες ιστού είναι APIs. Γενικά ένα Web API αποτελείται από δύο βασικά μέρη: το Frontend και το Backend όπως βλέπουμε στην Εικόνα 1.5




Εικόνα 1.5: Ο ρόλος του Web API

### 1.4.1 FrontEnd

Το Frontend αναφέρεται στο τμήμα του λογισμικού που είναι ορατό στον user και αλληλεπιδρά μαζί του. Σε ένα Web API, το Frontend μπορεί να είναι η διεπαφή χρήστη (User Interface) που χρησιμοποιείται για να κάνει αιτήσεις στο API. Το Frontend μπορεί να αποτελείται από HTML, CSS και JavaScript που επικοινωνούν με το Web API μέσω HTTPS αιτημάτων.

Δύο δημοφιλή στοιχεία που χρησιμοποιούνται συχνά στο Frontend των ιστοσελίδων και εφαρμογών είναι τόσο το jQuery όσο και η τεχνολογία Ajax.

- **jQuery:** Το jQuery είναι μια ελαφριά, γρήγορη και ευέλικτη βιβλιοθήκη JavaScript που διευκολύνει τη διαχείριση του DOM, τη δημιουργία εντυπωσιακών εφέ και την ανταλλαγή δεδομένων με τον server. Χρησιμοποιείται ευρέως για την απλοποίηση του κώδικα JavaScript και την επίτευξη συμβατότητας μεταξύ διαφορετικών browser.
- **Ajax (Asynchronous JavaScript and XML):** Το Ajax είναι μια τεχνική που επιτρέπει την ανταλλαγή δεδομένων μεταξύ του browser και του server χωρίς να χρειάζεται ανανέωση της σελίδας. Χρησιμοποιείται για την ασύγχρονη φόρτωση δεδομένων και τη δυναμική ενημέρωση του περιεχομένου της σελίδας.

Παρακάτω δίνεται ένα παράδειγμα AJAX call με χρήση της JQuery:

```
$.ajax({
//Εδώ ορίζεται η διεύθυνση URL προς την οποία θα γίνει το αίτημα AJAX. Στο παράδειγμα, η
//διεύθυνση URL είναι esp32-visualize.php.
  url: 'esp32-visualize.php'
//Εδώ ορίζεται η μέθοδος HTTP που θα χρησιμοποιηθεί για το αίτημα. Χρησιμοποιείται η μέθοδος
//GET.
  type: 'GET',
//Ο τύπος των δεδομένων που αναμένονται από τον server ως απάντηση είναι σε μορφή JSON.
  dataType: 'json',
// περίπτωση επιτυχούς ανάκτησης δεδομένων
```

```
   success: function(data) {
//κώδικας για την περίπτωση επιτυχούς ανάκτησης δεδομένων
   },
// περίπτωση ανεπιτυχούς ανάκτησης δεδομένων
   error: function(xhr, status, error) {
  //κώδικας για την περίπτωση ανεπιτυχούς ανάκτησης δεδομένων
   }
 });
```

### 1.4.2 BackEnd

Το Backend είναι το τμήμα του λογισμικού που τρέχει στον server και χειρίζεται τα αιτήματα που λαμβάνει το Web API. Το Backend είναι υπεύθυνο για την επεξεργασία των αιτημάτων, την πρόσβαση στη βάση δεδομένων (εάν απαιτείται) για την ανάκτηση, ενημέρωση ή διαγραφή δεδομένων, την επιστροφή των δεδομένων στο Frontend και την απάντηση στα αιτήματα με τη μορφή αποτελεσμάτων.

Στο Backend ανήκουν και τα API Endpoints. Ένα API Endpoint είναι ένα URL που λειτουργεί ως σημείο επαφής μεταξύ ενός client API και ενός server API. Οι client API αποστέλλουν αιτήσεις στα API endpoints για να έχουν πρόσβαση στη λειτουργικότητα και τα δεδομένα του API.

### 1.4.3 JSON

Η ανταλλαγή των δεδομένων γίνεται συνήθως σε μορφή JSON (JavaScript Object Notation). Το JSON είναι μια ελαφριά και αναγνώσιμη μορφή ανταλλαγής δεδομένων. Αποτελεί ένα πολύ δημοφιλές format, επειδή είναι εύκολο να κατανοηθεί και να δημιουργηθεί τόσο από ανθρώπους όσο και από υπολογιστές. Το JSON χρησιμοποιείται ευρέως για την ανταλλαγή δεδομένων μεταξύ του browser του χρήστη και του server, κυρίως σε περιβάλλοντα Web API. Με τη χρήση του JSON, δεδομένα όπως κείμενο, αριθμοί, λίστες και αντικείμενα μπορούν να μεταφερθούν από τον server στον browser και αντίστροφα. Το JSON έχει μια απλή σύνταξη που αποτελείται από ζεύγη "key-value" που ονομάζονται "properties". Οι τιμές μπορούν να είναι strings, αριθμοί, λίστες, αντικείμενα, boolean, ή ακόμα και null. Το JSON είναι ανεξάρτητο από τη γλώσσα προγραμματισμού, πράγμα που σημαίνει ότι μπορεί να χρησιμοποιηθεί σε πολλές γλώσσες προγραμματισμού για τη μετατροπή των δεδομένων.Παρακάτω στην Εικόνα 1.6 βλέπουμε ένα σύνολο από JSON Object και στην Εικόνα 1.7 έναν JSON Array.


Εικόνα: 1.6 JSON Objects

Στην Εικόνα 1.6 κάθε JSON object περιλαμβάνει δύο ζεύγη key-value -άρα δύο properties- όπου το πρώτο key είναι το "id" και η αντίστοιχη value είναι ο αριθμός του tweet, ενώ το δεύτερο key είναι το "full_text" και η αντίστοιχη value είναι το περιεχόμενο του tweet. Έτσι, κάθε JSON object αναπαριστά ένα tweet με βάση αυτά τα δύο ζεύγη key-value. Για παράδειγμα για το πρώτο έχουμε:

"key": "id", "value": 90101,

"key": "full_text", "value": "A beautiful tweet."

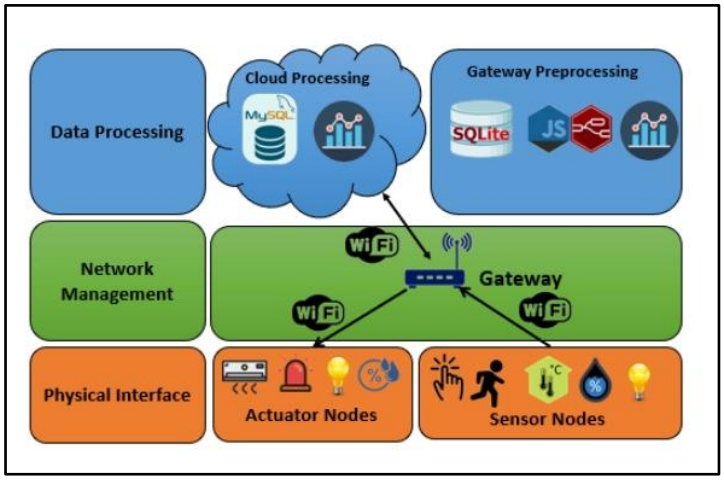

Εικόνα 1.7: JSON Array

Οι πίνακες JSON βρίσκονται μέσα σε αγκύλες. Μέσα σε έναν πίνακα, μπορούμε να δηλώσουμε οποιοδήποτε αριθμό αντικειμένων, τα οποία διαχωρίζονται με κόμμα. Στο συγκεκριμένο παράδειγμα έχουμε τα ονόματα των νικητών και τον αριθμό νικών του καθενός σε ένα παιχνίδι.

# Κεφάλαιο 2ο: ESP32_WS

## 2.1 Εισαγωγή

Ο ESP32_WS αποτελεί μια IoT εφαρμογή Μετεωρολογικού Σταθμού της οποίας ο σχεδιασμός και η υλοποίηση αποτελούν το αντικείμενο της παρούσης εργασίας. Στηρίζεται στην αρχιτεκτονική IoT τριών επιπέδων που περιγράφεται στην παράγραφο 1.5 και βλέπουμε πιο αναλυτικά στην Εικόνα 2.1, όπου στο Perception Layer -Physical Interface- συναντάμε τον BME280, την LCD 16x2 RGB και το push_button, στο Network Layer -Network Management- τον ESP32 ενώ στο Application Layer -Data Processing- έχουμε την αποθήκευση των μετρήσεων, την επεξεργασία και την εμφάνισή τους μέσω μιας δυναμικής ιστοσελίδας.




Εικόνα 2.1: Αρχιτεκτονική τριών επιπέδων για τον ESP32_WS

Στα επόμενα κεφάλαια γίνεται αναλυτική περιγραφή του κάθε επιπέδου.

## 2.2 Βασικά Χαρακτηριστικά του ESP32_WS

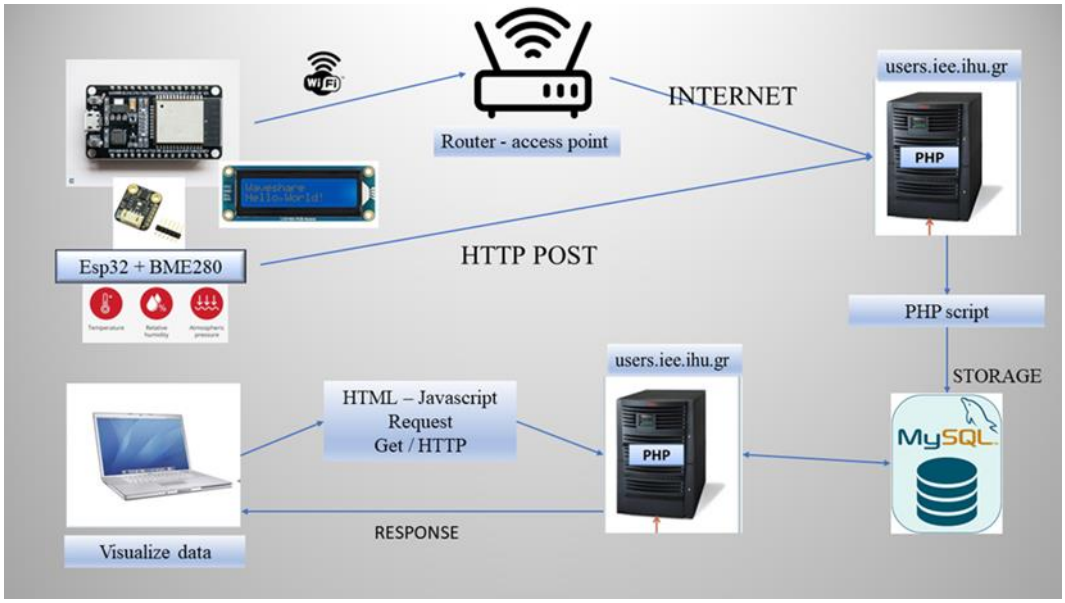
Τα βασικά χαρακτηριστικά του ESP32_WS είναι:

- Καταγραφή και απεικόνιση των τιμών των μεγεθών -θερμοκρασία, υγρασία και πίεση- με τη μορφή διαγραμμάτων σε δυναμική ιστοσελίδα, ώστε να είναι διαθέσιμα από οπουδήποτε μέσω διαδικτύου.
- Απεικόνιση των τιμών και τοπικά μέσω της LCD 16x2 RGB. Με τη βοήθεια ενός push-button γίνεται εναλλαγή της πληροφορίας στην οθόνη, δηλαδή με κάθε πάτημα αλλάζει και το φυσικό μέγεθος που εμφανίζεται στην οθόνη. Αρχικά και με τη σύνδεση στην τροφοδοσία εμφανίζεται η θερμοκρασία.
- Εμφάνιση στην LCD 16x2 RGB διαγνωστικών μηνυμάτων όπως π.χ στην περίπτωση αποσύνδεσης του BME280 -BME280 disconnected!- ή μετατροπής του ESP32 σε Access Point -AP-
- Παραμετροποίηση του ESP32. Ο ESP32_WS μπορεί να μεταφερθεί και να λειτουργήσει σε οποιοδήποτε χώρο αρκεί να υπάρχει πρόσβαση στο internet. Ο ESP32, μπορεί πολύ εύκολα να συνδεθεί στο καινούριο Wi – Fi δίκτυο ή δίκτυο κινητής τηλεφωνίας που θα του υποδείξουμε, μέσω μιας φορητής συσκευής (smartphone ή laptop). Αναλυτικά η διαδικασία περιγράφεται στην παράγραφο 5.6

- Μήνυμα σφάλματος στην ιστοσελίδα σε περίπτωση που για κάποιο λόγο π.χ αποσύνδεση BME280 ή αποσύνδεση από το Wi-Fi, δεν καταγράφονται για χρονικό διάστημα που ορίζεται από τον administrator, καινούριες τιμές στην βάση δεδομένων -παράγραφος 6.4-
- Αποστολή email στον administrator. Παράλληλα με το μήνυμα στην οθόνη ο administrator λαμβάνει email, που τον ενημερώνει ότι ο ESP32_WS για κάποιο λόγο δεν λειτουργεί -παράγραφος 6.5-

## 2.3 Γενική Περιγραφή της Εφαρμογής

Στο Σχήμα 2.2 δίνονται τα επιμέρους τμήματα της εφαρμογής.




Σχήμα 2.2: Αρχιτεκτονική του Συστήματος ESP32_WS [2],[3],[4],[5],[6]

Μια γενική περιγραφή της εφαρμογής με βάση το Σχήμα 2.2 είναι η παρακάτω:

Αρχικά ο ESP32 αναγνωρίζει τον BME280 και στη συνέχεια αφού συνδεθεί στο Wi-Fi που έχει αποθηκευμένο στη μνήμη του, διαβάζει τις τιμές, για την θερμοκρασία του περιβάλλοντος, την υγρασία και την ατμοσφαιρική πίεση από τον BME280, και τις στέλνει μέσω διαδικτύου με το HTTP πρωτόκολλο, στον users.iee.ihu.gr. Εκεί μέσω ενός PHP script αποθηκεύονται σε μία βάση δεδομένων MySql.

Οι 30 πιο πρόσφατες τιμές από την βάση δεδομένων είναι διαθέσιμες στον user σε μορφή διαγραμμάτων μέσω μιας δυναμικής ιστοσελίδας, η οποία ανανεώνεται σε τακτά χρονικά διαστήματα που ορίζονται από τον administrator.

# Κεφάλαιο 3ο: Γλώσσες Προγραμματισμού – Ανάπτυξη Προγραμμάτων

## 3.1 Εισαγωγή

Στο κεφάλαιο αυτό θα δούμε τις γλώσσες προγραμματισμού που χρειαζόμαστε για την υλοποίηση του ESP32_WS αλλά και τα κατάλληλα περιβάλλοντα για την συγγραφή - ανάπτυξη των προγραμμάτων.

Χρησιμοποιούμε γλώσσα “Arduino” ( ή wiring C) για να προγραμματίσουμε τον ESP32. Αυτή η γλώσσα είναι μια απλοποιημένη έκδοση της C/C++, σχεδιασμένη για την ανάπτυξη εφαρμογών για τις πλακέτες Arduino. Χρησιμοποιούμε HTML – Javascript -μαζί με Jquery και AJAX- για το Frontend και PHP για το Backend της εφαρμογής. Για την αποθήκευση και διαχείριση των μετρήσεων σε αυτή την εφαρμογή, θα χρησιμοποιηθεί η MySQL ως βάση δεδομένων.

Για την συγγραφή του προγράμματος του ESP32 θα χρησιμοποιηθεί το περιβάλλον ανάπτυξης Arduino IDE το οποίο περιγράφεται στο κεφάλαιο 5. Για την συγγραφή των προγραμμάτων σε HTML – Javascript και σε PHP θα χρησιμοποιηθεί το Visual Studio Code. Όλα τα αρχεία της εργασίας -εκτός από το auth.php- αποθηκεύονται στο GitHub και η αποθήκευσή τους γίνεται με τη βοήθεια του GitHub Desktop.

## 3.2 Wiring C

Όπως είπαμε ο ESP32 μπορεί να προγραμματιστεί σε περιβάλλον Arduino IDE χρησιμοποιώντας τη γλώσσα Wiring C και μπορεί να υποστηρίξει ένα πλήθος από βιβλιοθήκες.

Για την εκπόνηση της παρούσης εργασίας χρησιμοποιείται το παρακάτω αρχείο γραμμένο στο Arduino IDE :

1. esp32_wifi_postdata_LCD.ino

Ένα παράδειγμα κώδικα σε Wiring C, για τη λειτουργία ενός φαναριού δίνεται στον Πίνακα 3.1

Πίνακας 3.1: Παράδειγμα κώδικα Wiring C

```
#include <Arduino.h>
#define GREEN_LED_PIN 13
#define RED_LED_PIN 12
#define ORANGE_LED_PIN 14
void setup() {
  pinMode(GREEN_LED_PIN, OUTPUT);
  pinMode(RED_LED_PIN, OUTPUT);
  pinMode(ORANGE_LED_PIN, OUTPUT);
}
void loop() {
  // Πράσινο φως
```

```
  digitalWrite(GREEN_LED_PIN, HIGH);
  digitalWrite(RED_LED_PIN, LOW);
  digitalWrite(ORANGE_LED_PIN, LOW);
  delay(10000);
  // Πορτοκαλί φως
  digitalWrite(GREEN_LED_PIN, LOW);
  digitalWrite(RED_LED_PIN, LOW);
  digitalWrite(ORANGE_LED_PIN, HIGH);
  delay(2000);
  // Κόκκινο φως
  digitalWrite(GREEN_LED_PIN, LOW);
  digitalWrite(RED_LED_PIN, HIGH);
  digitalWrite(ORANGE_LED_PIN, LOW);
  delay(5000);
}
```

Ο παραπάνω κώδικας θα αναβοσβήνει τρία LED που έχουν συνδεθεί στα pin 13 το πράσινο, 12 το κόκκινο και 14 το πορτοκαλί του ESP32, προσομοιώνοντας τη λειτουργία ενός φαναριού που εναλλάσσεται ανάμεσα σε πράσινο, πορτοκαλί και κόκκινο φως για χρονικό διάστημα 10sec, 2sec και 5sec αντίστοιχα.

## 3.3 PHP

Η PHP (αναδρομική συντομογραφία για PHP: Hypertext Preprocessor) είναι μια ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού γενικού σκοπού με ανοικτό κώδικα που είναι ιδιαίτερα κατάλληλη για την ανάπτυξη ιστοσελίδων και μπορεί να ενσωματωθεί στο HTML.

Η PHP είναι μια γλώσσα προγραμματισμού που εκτελείται στην πλευρά του server (server-side). Αυτό σημαίνει ότι ο κώδικας PHP εκτελείται στον server πριν η σελίδα φορτωθεί στον client. Ο κώδικας PHP χρησιμοποιείται συνήθως για τη διαχείριση δεδομένων, την αλληλεπίδραση με βάσεις δεδομένων και τη δημιουργία δυναμικού περιεχομένου που στέλνεται στον client. Ο client θα λάβει τα αποτελέσματα της εκτέλεσης του σεναρίου, αλλά δεν θα γνωρίζει ποιος ήταν ο υποκείμενος κώδικας.

Το καλύτερο μέρος στη χρήση της PHP είναι ότι είναι σχετικά απλή για έναν νέο χρήστη, και προσφέρει πολλά προηγμένα χαρακτηριστικά για έναν επαγγελματία προγραμματιστή.

Για την εκπόνηση της παρούσης εργασίας χρησιμοποιούνται τα παρακάτω PHP αρχεία:

1 esp32-postdata.php
2. esp32-visualize.php
3. esp32-cronjob.php
4. esp32-mail.php
5. esp32-statusclear.php
6. auth.php

Ένα παράδειγμα κώδικα σε PHP, με το αρχείο esp32-postdata.php δίνεται στον Πίνακα 3.2. Τα υπόλοιπα php αρχεία δίνονται στο Παράρτημα Α.

Πίνακας 3.2: Παράδειγμα κώδικα σε PHP

```
<?php
require_once 'auth.php';
$servername = "localhost";
$dbname = "esp32";
$username = $user;
$password = $pass;

// Η τιμή του κλειδιού API πρέπει να είναι συμβατή με τον κώδικα esp32
$api_key_value = $api_postdata;

$api_key = $value1 = $value2 = $value3 = "";

// Έλεγχος αν η αίτηση είναι τύπου POST
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $api_key = test_input($_POST["api_key"]);
   if($api_key == $api_key_value) {
      // Λήψη των τιμών αισθητήρων από το POST request
      $value1 = test_input($_POST["value1"]);
      $value2 = test_input($_POST["value2"]);
      $value3 = test_input($_POST["value3"]);

      // Δημιουργία σύνδεσης με τη Βάση δεδομένων
      $conn = new mysqli($servername, $username, $password, $dbname,null,'/home/student/iee/2019/iee2019243/mysql/run/mysql.sock');
      // Έλεγχος Σύνδεσης
      if ($conn->connect_error) {
         die("Connection failed: " . $conn->connect_error);
      }
      // Εισαγωγή των τιμών στον πίνακα sensor της βάσης δεδομένων
```

```
        $sql = "INSERT INTO sensor (value1, value2, value3)
        VALUES ('" . $value1 . "', '" . $value2 . "', '" . $value3 . "')";


        if ($conn->query($sql) === TRUE) {
            echo "The values were saved correctly";
        }
        else {
            echo "Error: " . $sql . "<br>" . $conn->error;
        }
        $conn->close();
    }
    else {
        echo "Wrong API Key."; // Εμφάνιση μηνύματος λάθους για λάθος κλειδί API
    }
}
else {
    echo "No data posted.";
}
// Συνάρτηση για επεξεργασία εισόδου δεδομένων
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data; }
?>
```

Το αρχείο esp32_wifi_postdata_LCD.ino που τρέχει στον ESP32 και το esp32-postdata.php που τρέχει στον server -users.iee.ihu.gr- συνεργάζονται για να δημιουργήσουν ένα API.

Το αρχείο esp32_wifi_postdata_LCD.ino και μέσω της συνάρτησης sendSensorData() -που δίνεται στην παράγραφο 5.5.4- δημιουργεί τα δεδομένα και τα αποστέλλει μέσω HTTPS POST αιτημάτων, ενώ ο server με το esp32-postdata.php λαμβάνει τα αιτήματα, ελέγχει την αυθεντικότητά τους μέσω του κλειδιού API και αποθηκεύει τα δεδομένα στη βάση δεδομένων MySQL.

Το API έχει σχεδιαστεί έτσι ώστε να επιτρέπει την εύκολη και ασφαλή επικοινωνία μεταξύ των δύο συστημάτων, εξασφαλίζοντας την αποτελεσματική μετάδοση και αποθήκευση των δεδομένων του BME280.

## 3.4 HTML – Javascript

Για τον σχεδιασμό της δομής της ιστοσελίδας που θα χρησιμοποιηθεί για την παρακολούθηση των διαγραμμάτων της θερμοκρασίας, της υγρασίας και της ατμοσφαιρικής πίεσης, θα χρησιμοποιηθεί η γλώσσα HTML, στην οποία θα ενσωματώσουμε javascript για να προσθέσουμε διαδραστικά στοιχεία στην ιστοσελίδα. Θα χρησιμοποιηθεί η βιβλιοθήκη Highcharts η οποία είναι μια δημοφιλής βιβλιοθήκη JavaScript για τη δημιουργία διαγραμμάτων και γραφημάτων σε ιστοσελίδες.

Για την εκπόνηση της παρούσης εργασίας χρησιμοποιείται το παρακάτω HTML αρχείο:

1. esp32-visualize.html

Στον Πίνακα 3.3 δίνεται ένα παράδειγμα κώδικα σε HTML – Javascript, όπου γίνεται χρήση της Highcharts.

Πίνακας 3.3: Παράδειγμα κώδικα HTML – Javascript

```
<!DOCTYPE html>
<html>
<head>
 <!-- Καθορισμός της κλίμακας προβολής για φορητές συσκευές, responsive -- >
  <meta name="viewport" content="width=device-width, initial-scale=1">
<!-- Σύνδεση με τη βιβλιοθήκη jQuery -->
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<!-- Σύνδεση με τη βιβλιοθήκη Highcharts -->
  <script src="https://code.highcharts.com/highcharts.js"></script>
  <style>
    body {
      min-width: 310px;
      max-width: 1280px;
      height: 500px;
      margin: 0 auto;
    }
    h2 {
      font-family: Arial;
      font-size: 2.5rem;
      text-align: center;
    }
  </style>
</head>
```

```
<body>
    <h2>Weather Station</h2>
   <div id="chart-temperature" class="container"></div>
   <script>
// Ορισμός συνάρτησης για ανάκτηση δεδομένων από τον εξυπηρετητή με ajax call
    function fetchData() {
  $.ajax({
    url: 'esp32-visualize.php',
    type: 'GET',
    dataType: 'json',
// περίπτωση επιτυχούς ανάκτησης δεδομένων
    success: function(data) {
      let readings_time = data.map(a => a.reading_time);
      let value1 = data.map(a => parseFloat(a.value1));
// Εμφάνιση του γραφήματος
      document.getElementById("chart-temperature").style.display = 'block';
// Ενημέρωση του γραφήματος με τα νέα δεδομένα
      updateChart(chartT, value1, readings_time);
    },
// περίπτωση ανεπιτυχούς ανάκτησης δεδομένων
    error: function(xhr, status, error) {
      console.error("Error fetching data: " + error);
    }
  });
}
// Δημιουργία γραφήματος
function createChart(container, title, yAxisTitle, zonesConfig, name) {
      return Highcharts.chart(container, {
        chart: { type: 'line' },
        title: { text: title },
        series: [{
          name:name,
          data: [],
```

```
        zones: zonesConfig
      }],
      xAxis: {
        type: 'datetime'
        },
      yAxis: {
        title: { text: yAxisTitle }
      },
      credits: { enabled: false }
    });
  }
// Συνάρτηση για την ενημέρωση γραφήματος με νέα δεδομένα
  function updateChart(chart, data, categories) {
    chart.series[0].setData(data);
    chart.xAxis[0].setCategories(categories);
  }
// Δημιουργία γραφήματος θερμοκρασίας
  let chartT = createChart('chart-temperature', 'BME280 Temperature', 'Temperature (Celsius)', [{
    value: 24,
    color: '#0000ff'
  }, {
    value: 25,
    color: '#00ff00'
  }, {
    color: '#ff0000'
  }], 'Temperature');
// Ανάκτηση δεδομένων από τον εξυπηρετητή
  fetchData();
// Επαναλαμβανόμενη ανάκτηση δεδομένων κάθε 30 δευτερόλεπτα
  setInterval(fetchData, 30000);
</script>
</body>
</html>
```

Το παραπάνω παράδειγμα κώδικα είναι μόνο ένα μέρος του αρχείου esp32-visualize.html και εμφανίζει μόνο το γράφημα της θερμοκρασίας. Το esp32-visualize.html περιέχει τον πλήρη κώδικα για την δημιουργία της ιστοσελίδας και υπάρχει ολόκληρο στο Παράρτημα Α.

Ας κάνουμε κάποια σχόλια στον παραπάνω κώδικα.

Η γραμμή

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

είναι ένα σημαντικό μέρος της διαδικασίας δημιουργίας μιας responsive ιστοσελίδας. Με τη χρήση αυτής της γραμμής, η ιστοσελίδα προσαρμόζεται αυτόματα στο πλάτος της οθόνης της συσκευής που την προβάλει, επιτρέποντας έτσι μια βέλτιστη εμφάνιση σε φορητές συσκευές όπως κινητά τηλέφωνα και tablet.

Με την δομή

```
$.ajax({
  url: 'esp32-visualize.php',
  type: 'GET',
  dataType: 'json',
```

δημιουργούμε μια κλήση AJAX η οποία αποστέλλει ένα αίτημα στο URL του αρχείου 'esp32-visualize.php' χρησιμοποιώντας τη μέθοδο GET και αναμένει να λάβει απάντηση σε μορφή JSON. Ουσιαστικά το API βρίσκεται στο αρχείο με το όνομα 'esp32-visualize.php'. Όταν ο server λαμβάνει αυτό το αίτημα, το αρχείο 'esp32-visualize.php' εκτελείται στον server. Το αρχείο αυτό περιέχει php κώδικα που συλλέγει τις τιμές -30 πιο πρόσφατες- των μεγεθών από τη βάση δεδομένων και επιστρέφει αυτές τις τιμές σε μορφή JSON. Τέλος μπορούμε να παρατηρήσουμε και τη δημιουργία γραφημάτων καθώς και την ενημέρωσή τους με νέα δεδομένα με τη χρήση της βιβλιοθήκης Highcharts.

## 3.5 MySql

Ως μία από τις πιο δημοφιλείς σχεσιακές βάσεις δεδομένων σήμερα, η MySQL έχει γίνει και η προτιμώμενη σχεσιακή βάση δεδομένων για τις περισσότερες εταιρείες του Διαδικτύου λόγω του δωρεάν και ανοιχτού κώδικα, του μικρού μεγέθους, της υψηλής αποδοτικότητας, της απλότητας, της ευκολίας χρήσης και των πλούσιων πόρων [21]. Από τον Ιούλιο του 2014 η MySql είναι το 2$^{ο}$ πιο ευρέως διαδεδομένο σύστημα διαχείρισης βάσεων δεδομένων (RDBMS).

Η MySql χρησιμοποιείται και για την δημιουργία της Βάσης Δεδομένων στην παρούσα εργασία.

1. esp32_schema.sql

Στον Πίνακα 3.4 δίνεται ως παράδειγμα κώδικα SQL η δημιουργία του πίνακα sensor της εφαρμογής.

Πίνακας 3.4: Παράδειγμα κώδικα SQL [9]

```
CREATE TABLE IF NOT EXISTS `sensor` (
 `id` int(6) unsigned NOT NULL AUTO_INCREMENT,
 `value1` varchar(10) DEFAULT NULL,
 `value2` varchar(10) DEFAULT NULL,
```

```
  `value3` varchar(10) DEFAULT NULL,
  `reading_time` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp(),
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Με τον παραπάνω κώδικα που βρίσκεται στο αρχείο esp32_schema.sql, θα δημιουργήσουμε τον πίνακα sensor για την αποθήκευση των τιμών που καταγράφει ο BME280, δηλαδή της θερμοκρασίας -value1- της υγρασίας -value2- και της ατμοσφαιρικής πίεσης -value3-. Η κάθε γραμμή του πίνακα έχει το δικό της id που αποτελεί και το κύριο κλειδί του πίνακα και τη δική της χρονοσφραγίδα -reading_time- πληροφορία απαραίτητη για την ανάπτυξη της εφαρμογής, αφού από αυτήν καταλαβαίνουμε αν και για ποιο χρονικό διάστημα έχει σταματήσει η εγγραφή νέων τιμών στον πίνακα, ώστε να εμφανιστεί το κατάλληλο μήνυμα στην ιστοσελίδα και να αποσταλεί email στον administrator. Τμήμα του πίνακα sensor της MySql στον server φαίνεται παρακάτω στην Εικόνα 3.1

```
MariaDB [esp32]>
MariaDB [esp32]> select * from sensor;
+-----+--------+--------+---------+---------------------+
| id  | value1 | value2 | value3  | reading_time        |
+-----+--------+--------+---------+---------------------+
|   1 | 23.45  | 51.91  | 1013.78 | 2024-05-01 15:01:16 |
|   2 | 23.87  | 51.46  | 1013.70 | 2024-05-01 15:01:46 |
|   3 | 24.19  | 49.35  | 1013.72 | 2024-05-01 15:02:16 |
|   4 | 24.41  | 48.36  | 1013.71 | 2024-05-01 15:02:46 |
|   5 | 24.99  | 53.34  | 1013.71 | 2024-05-01 15:03:16 |
|   6 | 25.03  | 48.38  | 1013.74 | 2024-05-01 15:03:46 |
|   7 | 25.04  | 47.43  | 1013.74 | 2024-05-01 15:04:16 |
|   8 | 25.14  | 48.07  | 1013.71 | 2024-05-01 15:04:46 |
|   9 | 25.17  | 46.77  | 1013.73 | 2024-05-01 15:05:16 |
|  10 | 25.19  | 46.70  | 1013.74 | 2024-05-01 15:05:46 |
|  11 | 25.22  | 47.23  | 1013.74 | 2024-05-01 15:06:16 |
|  12 | 25.25  | 47.14  | 1013.74 | 2024-05-01 15:06:46 |
|  13 | 25.28  | 46.70  | 1013.81 | 2024-05-01 15:07:16 |
|  14 | 25.30  | 46.35  | 1013.76 | 2024-05-01 15:07:46 |
|  15 | 25.27  | 46.89  | 1013.74 | 2024-05-01 15:08:16 |
|  16 | 25.24  | 47.52  | 1013.75 | 2024-05-01 15:08:46 |
|  17 | 25.27  | 47.18  | 1013.77 | 2024-05-01 15:09:16 |
|  18 | 25.28  | 46.38  | 1013.78 | 2024-05-01 15:09:46 |
|  19 | 25.32  | 46.68  | 1013.71 | 2024-05-01 15:10:16 |
|  20 | 25.35  | 46.38  | 1013.72 | 2024-05-01 15:10:46 |
```

Εικόνα 3.1: Τμήμα του πίνακα sensor της MySql στον server.

## 3.6 Visual Studio Code

Το Visual Studio Code είναι ένας επεξεργαστής κειμένου και περιβάλλον ανάπτυξης (IDE) που αναπτύχθηκε από την Microsoft. Είναι δωρεάν, ανοικτού κώδικα και υποστηρίζει πολλές γλώσσες προγραμματισμού, όπως JavaScript, TypeScript, Python, C++, PHP, κ.ά. Το Visual Studio Code προσφέρει πληθώρα λειτουργιών, όπως επεξεργασία κειμένου με υπογράμμιση σύνταξης, αυτόματη συμπλήρωση κώδικα, οργάνωση αρχείων και φακέλων, ενσωματωμένη διαχείριση της έκδοσης ελέγχου, επεξεργασία και εκτέλεση εντολών τερματικού, και επέκταση μέσω πρόσθετων για περαιτέρω λειτουργικότητα. Επιπλέον, το Visual Studio Code είναι εξαιρετικά ευέλικτο, επιτρέποντας στους χρήστες να προσαρμόσουν το περιβάλλον εργασίας τους σύμφωνα με τις ανάγκες τους μέσω πολλών διαθέσιμων επεκτάσεων και θεμάτων. Η λήψη του, μπορεί να γίνει από την επίσημη ιστοσελίδα του Visual Studio Code, https://code.visualstudio.com [36]. Όπως έχουμε ήδη αναφέρει, στην παρούσα εργασία, θα χρησιμοποιηθεί για την ανάπτυξη των προγραμμάτων σε HTML – Javascript καθώς και σε PHP.

## 3.7 Το GitHub

Το GitHub θα μπορούσε να θεωρηθεί «το δεξί χέρι του προγραμματιστή».

«Αποτελεί μία cloud-based υπηρεσία, που δημιουργήθηκε το 2008, η οποία χρησιμοποιείται για την αποθήκευση κώδικα, όσο και για την αποθήκευση αλλαγών που πραγματοποιούνται σε ένα project, καθώς και την κοινοποίηση του εκάστοτε project σε άλλα άτομα.

Το GitHub, εκτός από υπηρεσία αποτελεί και μία κοινότητα, ένα social network προγραμματιστών, το οποίο δίνει την ευκαιρία σε προγραμματιστές από όλο τον κόσμο να επικοινωνούν μεταξύ τους. Αυτή η δυναμική της κοινότητας, δίνει την ευκαιρία στους προγραμματιστές, να προωθήσουν και οι ίδιοι την δουλειά τους, φτιάχνοντας στην ουσία ένα portfolio με τα projects τους, το οποίο έχει παγκόσμια ορατότητα, άρα και πιθανές επαγγελματικές ευκαιρίες σε όλο τον κόσμο» [37].

### 3.7.1 Repository

Ένα Repository -αποθετήριο- στο GitHub είναι ένα χώρος όπου μπορούμε να αποθηκεύουμε και να διαχειριζόμαστε τον κώδικα για ένα έργο λογισμικού. Μπορεί να περιέχει τα αρχεία που απαιτούνται για την εκτέλεση του λογισμικού, καθώς και οποιαδήποτε άλλη τεκμηρίωση, αρχεία δεδομένων ή εργαλεία ανάπτυξης που σχετίζονται με το έργο. Τα Repositories GitHub παρέχουν επίσης λειτουργίες διαχείρισης εκδόσεων (version control) που επιτρέπουν στους συνεργάτες να συνεργαστούν στην ανάπτυξη του λογισμικού, να παρακολουθούν τις αλλαγές και να διαχειρίζονται την εξέλιξή του.

### 3.7.2 Πως Λειτουργεί

Το GitHub λειτουργεί:

- **Στο version control:** Όσο δουλεύει ένας ή περισσότεροι προγραμματιστές πάνω σε ένα project , οι αλλαγές στον κώδικα είναι πολλές. Έτσι οι αλλαγές που πιθανόν να χρειαστεί να γίνουν, δεν είναι ασφαλές να πραγματοποιηθούν στον επίσημο κώδικα του project. Αντί γι' αυτό, οι προγραμματιστές χρησιμοποιούν δύο άλλες τεχνικές, το branching και το merging. Μέσα από το branching στην ουσία ο κάθε προγραμματιστής δημιουργεί ένα διπλότυπο του κώδικα που θέλει να επεξεργαστεί και έτσι κάνει ό,τι αλλαγή χρειάζεται. Μέσω του merging τοποθετεί τον κώδικα στο σημείο του project όπου ανήκει [37].
- **Στο Git:** Το Git αποτελεί ένα σύστημα ελέγχου του version control. Με άλλα λόγια, καταγράφει τις αλλαγές που γίνονται στο εκάστοτε project, συμβάλλοντας στον συγχρονισμό όλων των προγραμματιστών της ομάδας [37].

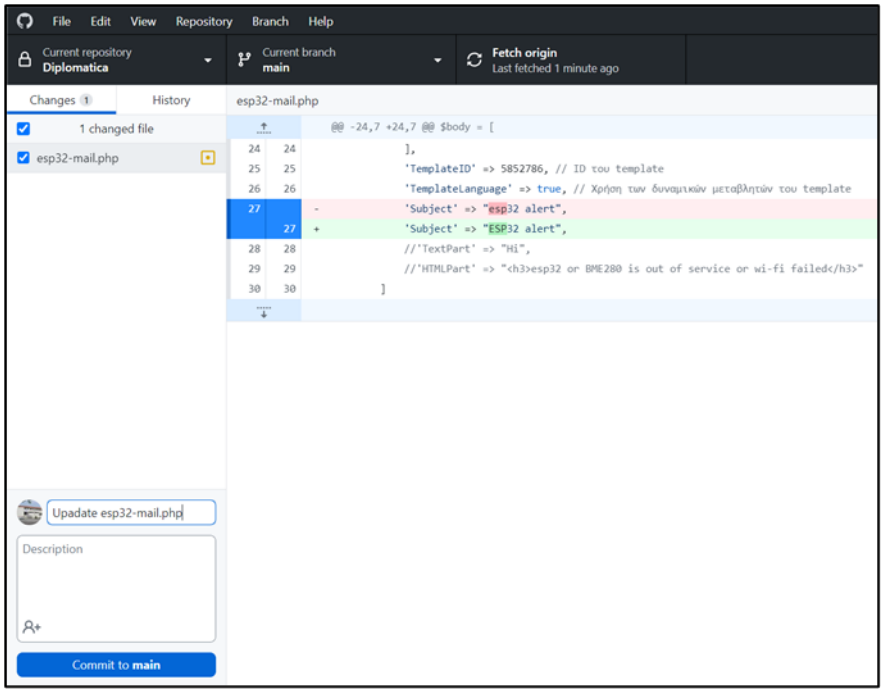
### 3.7.3 Πλεονεκτήματα του GitHub

Τα βασικότερα πλεονεκτήματα του GitHub θα μπορούσαμε να πούμε ότι είναι τα εξής:

- **Εύκολο project management:** Σε ένα project που εμπεριέχει κώδικά, εμπλέκονται τόσο προγραμματιστές όσο και project managers. Η συνεργασία των δύο, κάποιες φορές μπορεί να έχει δυσκολίες**,** που όμως μπορούν να αντιμετωπιστούν μέσω του GitHub**.**
- **Αποτελεσματικότητα Ομάδας:** Το GitHub περιέχει διάφορα εργαλεία, τα οποία συμβάλλουν στην καλύτερη οργάνωση της ομάδας, με αποτέλεσμα όλα τα μέλη να έχουν κοινή πορεία και να μένουν οργανωμένοι.
- **Ασφαλείς Αλλαγές:** «Μέσω του GitHub, δεν χρειάζεται να συμβαίνουν αλλαγές άμεσα στον επίσημο κώδικα. Έτσι, δεν τίθεται σε κίνδυνο το project όποτε συμβαίνει μία διόρθωση» [37].

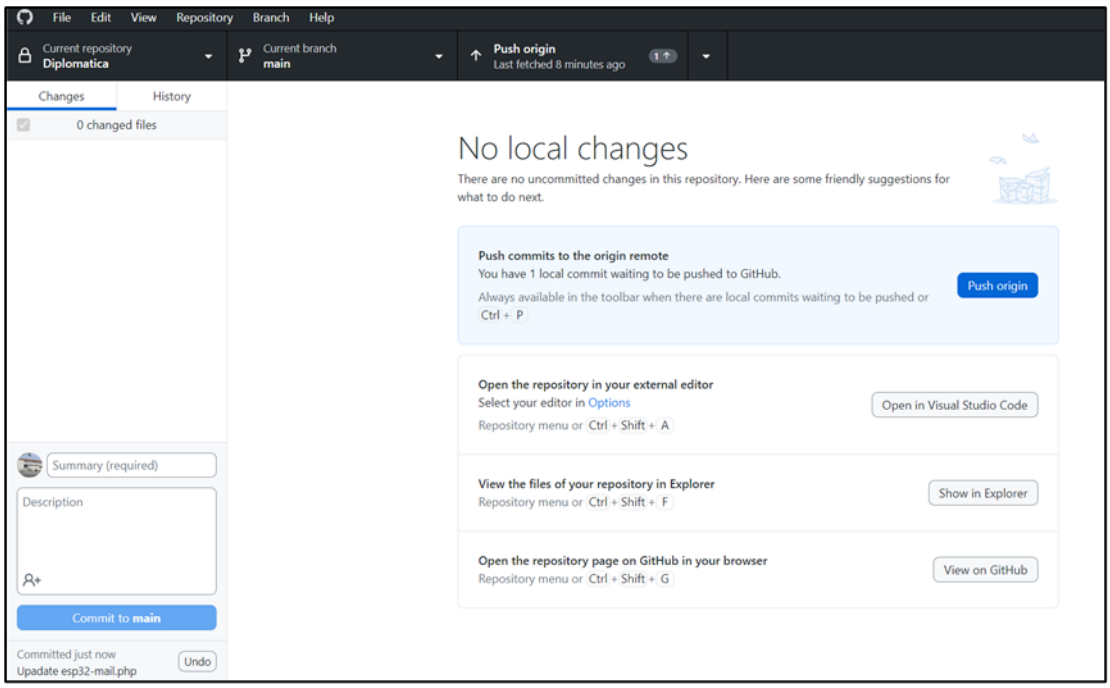
## 3.8 Το GitHub Desktop

Το GitHub Desktop είναι μια εφαρμογή γραφικού περιβάλλοντος χρήστη που παρέχει μια ευκολότερη διαχείριση των Repositories στο GitHub. Επιτρέπει την διαχείριση των Repositories, την πραγματοποίηση αλλαγών, την πραγματοποίηση αναζητήσεων στο ιστορικό των αλλαγών, καθώς και την εκτέλεση άλλων διαχειριστικών εργασίων σχετικά με τα Repositories. Είναι ένα εργαλείο που κάνει τη χρήση του GitHub πιο προσβάσιμη για αρχάριους χρήστες και παρέχει μια ευκολότερη εμπειρία διαχείρισης έργων ανοικτού κώδικα. Η λήψη του μπορεί να γίνει από την επίσημη ιστοσελίδα του GitHub, https://desktop.github.com. Στην Εικόνα 3.2 δίνεται με ένα απλό παράδειγμα το περιβάλλον εργασίας του GitHub Desktop.



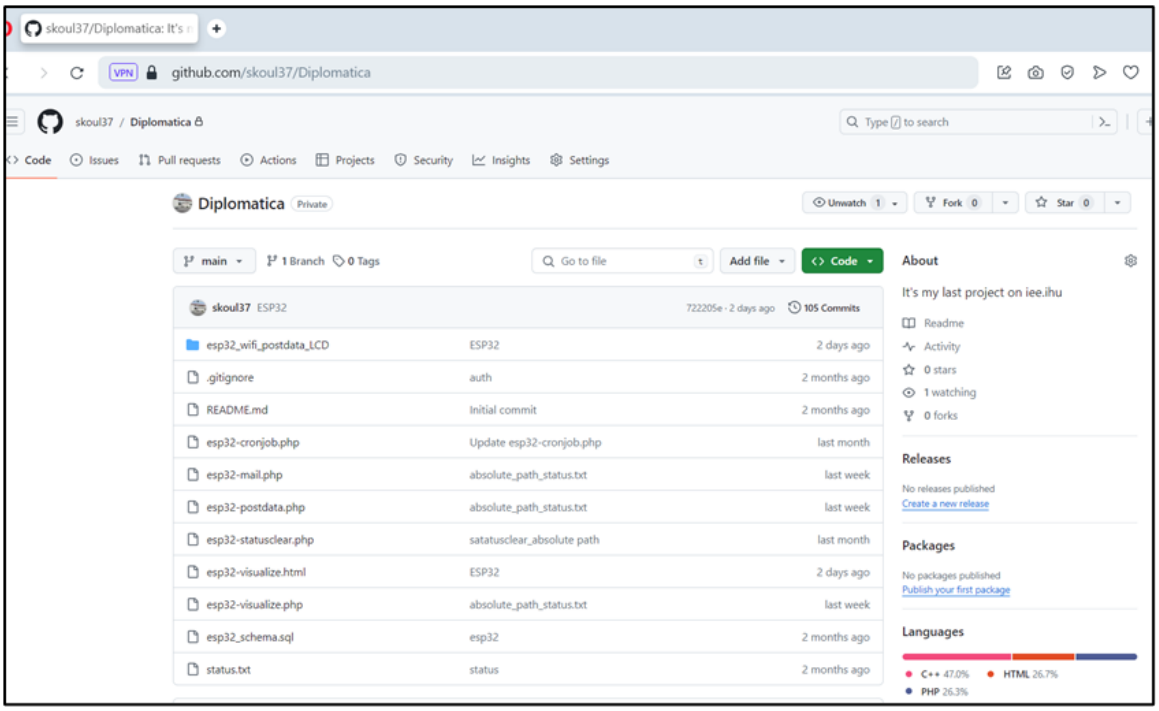

Εικόνα 3.2: Περιβάλλον εργασίας GitHub Desktop

Στην παραπάνω Εικόνα 3.2 μπορούμε να δούμε με ροζ χρώμα την αρχική γραμμή κώδικα και με πράσινο χρώμα, πως έγινε ο κώδικάς μας μετά την αλλαγή. Πατώντας Commit, αποθηκεύονται οι αλλαγές που έχουμε κάνει στο Repository, σε ένα τοπικό Repository και στη συνέχεια θα δούμε την Εικόνα 3.3.




Εικόνα 3.3: Αποστολή των αλλαγών -Push origin- στο κεντρικό Repository στο GitHub

Πατώντας Push origin αποστέλλονται οι τελευταίες αλλαγές που έχουμε κάνει στο τοπικό Repository, στο κεντρικό Repository στο GitHub. Αυτό επιτρέπει σε άλλους χρήστες να δουν και να αλληλεπιδράσουν με τις αλλαγές μας, μέσω του κεντρικού Repositoty. Οι αλλαγές μας ενσωματώνονται στο κοινό ιστορικό εργασιών του Repository. Στην Εικόνα 3.4 φαίνεται το Repository της παρούσης εργασίας στο GitHub.

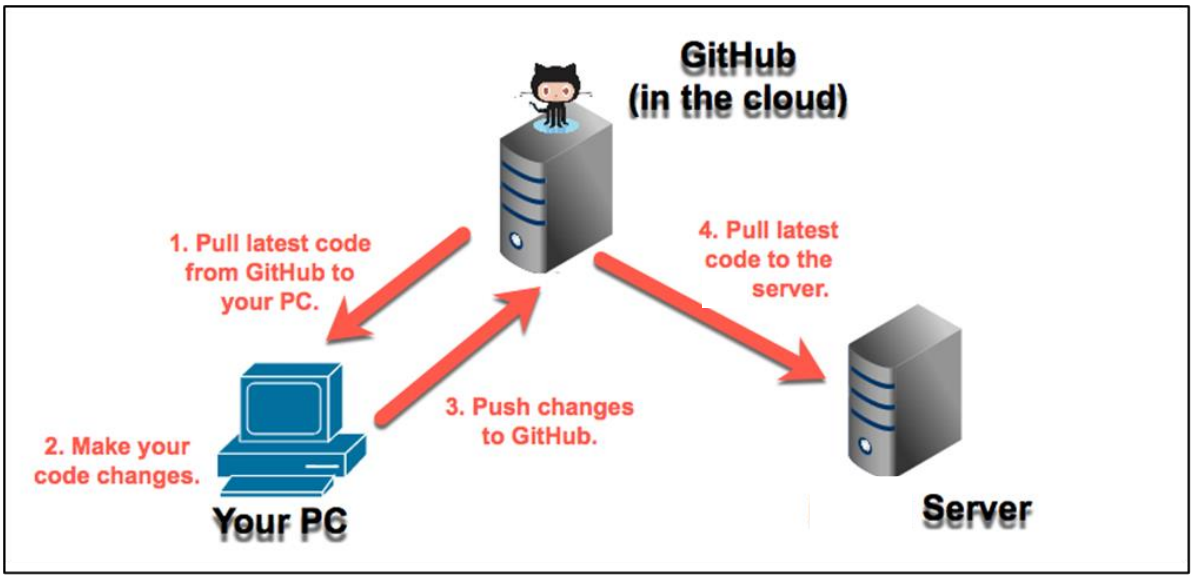



Εικόνα 3.4 Το Repository της εφαρμογής στο GitHub

## 3.9 Clone στον Server

Το Repository μπορεί να γίνει clone στον server στον φάκελο που επιθυμούμε με την εντολή gh repo clone <URL του Repository>. Αυτό θα δημιουργήσει έναν φάκελο με το όνομα του Repository και θα κλωνοποιήσει το Repository μέσα σε αυτόν τον φάκελο.

Στη συνέχεια με την git pull μέσα στον φάκελο του Repository, μπορούμε κάθε φορά να ανακτούμε από τον απομακρυσμένο κλάδο -remote branch- το πλήρες Repository και τις αλλαγές που έχουν γίνει σε αυτό το Repository και να τις ενσωματώνουμε στον τοπικό κλάδο. Στην Εικόνα 3.5 φαίνεται η σύνδεση ανάμεσα στον τοπικό υπολογιστή, το GitHub και τον Server.




Εικόνα 3.5 Σύνδεση ανάμεσα στον τοπικό υπολογιστή το GitHub και τον Server

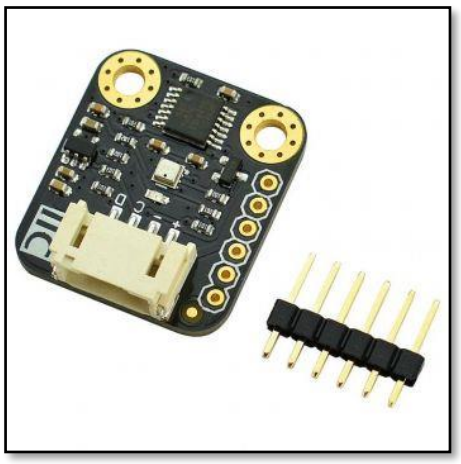
# Κεφάλαιο 4ο: Perception Layer

## 4.1 Εισαγωγή

Στο κεφάλαιο αυτό θα μιλήσουμε για το πρώτο επίπεδο της αρχιτεκτονικής IoT τριών επιπέδων, που είναι το επίπεδο αντίληψης/αίσθησης -Perception Layer- . Στο επίπεδο αυτό συναντάμε τον BME280 και το push_button, που με βάση την Εικόνα 2.1, είναι sensor nodes, και την LCD 16x2 RGB που επίσης που με βάση την Εικόνα 2.1, είναι ένας actuator node. Ο MBE280 και η LCD 16x2 RGB συνδέονται με τον ESP32 και επικοινωνούν με το I2C το οποίο περιγράφεται στην παράγραφο 4.4.Το push_button θα συνδεθεί σε μια είσοδο του ESP32 και συγκεκριμένα στο GPIO 2.

## 4.2 Ο αισθητήρας BME280

Ο BME280 ο οποίος φαίνεται παρακάτω στην Εικόνα 4.1, είναι ένας αισθητήρας περιβάλλοντος που ενσωματώνει αισθητήρα θερμοκρασίας, αισθητήρα υγρασίας και βαρόμετρο. Ο αισθητήρας είναι υψηλής ακρίβειας, πολλαπλών λειτουργιών και μικρού μεγέθους κ.λπ. Παρέχει διεπαφές SPI και I2C, οι οποίες διευκολύνουν τη δημιουργία γρήγορων πρωτοτύπων. Μπορεί να χρησιμοποιηθεί ευρέως στην παρακολούθηση του περιβάλλοντος, καταγραφή ιστορικού θερμοκρασίας και σε IoT εφαρμογές [2].



Εικόνα 4.1: Ο αισθητήρας Περιβάλλοντος BME280 [2]

**Προδιαγραφές BME280**

- Working Voltage：3.3V~5.0V
- Working Current：2mA
- Working Temperature：-40℃~ 85℃
- Temperature Measuring Range: -40℃~ 85℃, resolution of 0.1℃, deviation of ±0.5℃
- Humidity Measuring Range: 0~100%RH, resolution of 0.1%RH, deviation of ±2%RH
- Pressure Measuring Range: 300~1100hPa
- Humidity Sampling Time: 1s
- Dimension: 22 * 25 mm/ 0.87 * 0.98 inches
- Weight: 12g [17]

## 4.3 Η Οθόνη 16x2 Character LCD – RGB 3.3V / 5V

Πρόκειται για μια LCD – RGB οθόνη, η οποία φαίνεται παρακάτω στην Εικόνα 4.2, που μπορεί να εμφανίσει 2 σειρές χαρακτήρων με 16 χαρακτήρες η κάθε σειρά και διαθέτει ρυθμιζόμενο χρώμα RGB φωτισμού πίσω φόντου, με θεωρητική δυνατότητα έως 16 εκατομμύρια (256^3) χρώματα. Υποστηρίζει κύλιση οθόνης, κίνηση δρομέα και άλλες λειτουργίες, διαθέτει διεπαφή ελέγχου I2C που σημαίνει ότι απαιτούνται μόνο δύο ακίδες σήματος, εξοικονομώντας τους πόρους IO και είναι συμβατή με τάση λειτουργίας 3.3V/5V [18].


Εικόνα 4.2: Η Οθόνη 16x2 Character LCD – RGB 3.3V / 5V [2]

**Προδιαγραφές**

- Operating voltage: 3.3V/5V
- Interface: I2C
- LCD type: character LCD
- Controller: AiP31068L PCA9633DP2
- Display: 64.5 x 16.0 mm
- Dimension: 87.0 × 32.0 × 13.0mm
- Operating current: 26mA (5V), 13mA (3.3V) [18]

## 4.4 Το πρωτόκολλο I2C

Η επικοινωνία μεταξύ μικροελεγκτών και διαφόρων περιφερειακών συσκευών απαιτεί κάποιο είδος ψηφιακού πρωτοκόλλου. Το I2C, μια τυπική υλοποίηση του οποίου βλέπουμε στην Εικόνα 4.3, είναι ένα κοινό πρωτόκολλο επικοινωνίας που χρησιμοποιείται σε ποικιλία συσκευών από πολλές διαφορετικές οικογένειες προϊόντων.

**Επισκόπηση I2C**

Το I2C είναι ένα πρωτόκολλο σειριακής επικοινωνίας δύο καλωδίων που χρησιμοποιεί μια σειριακή γραμμή δεδομένων (SDA) και μια σειριακή γραμμή ρολογιού (SCL). Το πρωτόκολλο υποστηρίζει πολλαπλές συσκευές-στόχους σε έναν επικοινωνιακό δίαυλο και μπορεί επίσης να υποστηρίξει πολλαπλούς ελεγκτές που αποστέλλουν και λαμβάνουν εντολές και δεδομένα. Η επικοινωνία μεταδίδεται σε πακέτα byte με μοναδική διεύθυνση για κάθε συσκευή-στόχο.

**Ιστορία**

Το I2C, που συχνά αποκαλείται I ‘two’ C, σημαίνει Inter-Integrated Circuit protocol. Το I2C αναπτύχθηκε το 1982 από την Philips Semiconductor (τώρα NXP Semiconductor) ως ένα πρωτόκολλο χαμηλής ταχύτητας για τη σύνδεση ελεγκτικών συσκευών όπως μικροελεγκτές και επεξεργαστές με συσκευές-στόχους όπως μετατροπείς δεδομένων και άλλες περιφερειακές συσκευές. Από το 2006, η

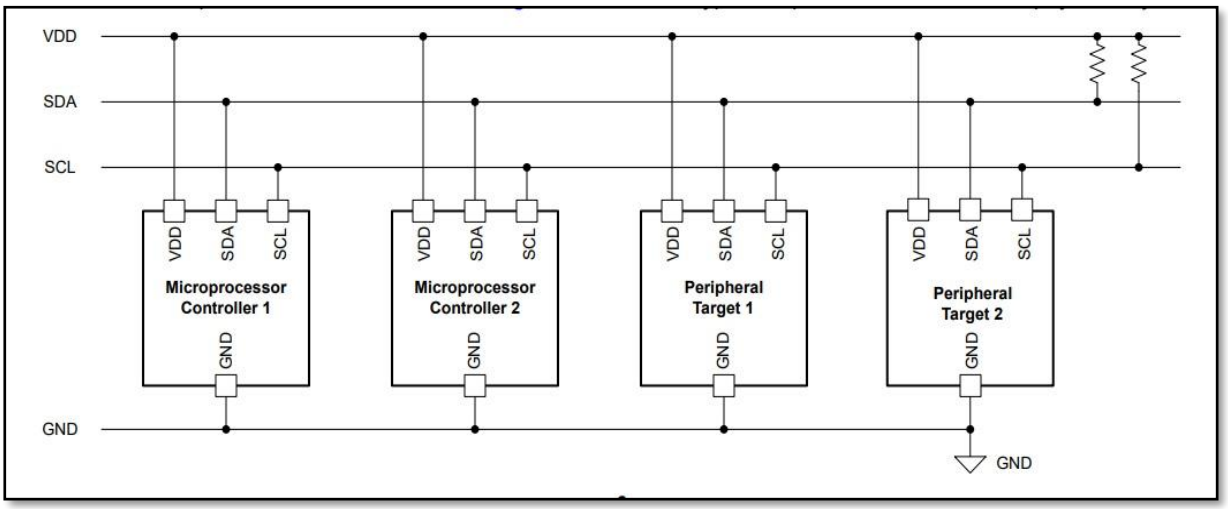
υλοποίηση του πρωτοκόλλου I2C δεν απαιτεί άδεια, και πολλές εταιρείες ημιαγωγών, έχουν εισάγει συσκευές συμβατές με I2C.

**Ταχύτητες I2C**

Το I2C διαθέτει αρκετούς τρόπους ταχύτητας ξεκινώντας με τον Κανονικό τρόπο (Standard-mode - Sm), ο οποίος είναι ένα σειριακό πρωτόκολλο που λειτουργεί έως 100 kilobits per second (kbps). Ακολουθεί ο Ταχύς τρόπος (Fast-mode - Fm) που φτάνει έως 400 kilobits per second (kbps) . Ο Ταχύς τρόπος Plus (Fm+) επιτρέπει επικοινωνία έως 1 megabit per second (Mbps). Αυτοί οι τρεις τρόποι είναι σχετικά παρόμοιοι, χρησιμοποιώντας μια δομή επικοινωνίας που είναι η ίδια. Ωστόσο, όλοι έχουν διαφορετικές προδιαγραφές χρονισμού για κάθε μία από τις λειτουργίες και η υλοποίηση υλικού του I2C στις συσκευές διαφέρει για να προσαρμοστεί στις διαφορετικές ταχύτητες.

**Επικοινωνία Δύο Καλωδίων**

Ένα σύστημα I2C διαθέτει δύο κοινόχρηστες γραμμές επικοινωνίας για όλες τις συσκευές στον δίαυλο. Αυτές οι δύο γραμμές χρησιμοποιούνται για αμφίδρομη, half-duplex επικοινωνία. Το I2C επιτρέπει την ύπαρξη πολλαπλών ελεγκτών και πολλαπλών συσκευών-στόχων. Απαιτούνται αντιστάσεις ανύψωσης (pullup resistors) και στις δύο από αυτές τις γραμμές.




Εικόνα 4.3: Τυπική Υλοποίηση I2C

Ένας από τους λόγους που το I2C είναι ένα κοινό πρωτόκολλο είναι επειδή χρησιμοποιούνται μόνο δύο γραμμές για την επικοινωνία. Η πρώτη γραμμή είναι το SCL, το οποίο είναι ένα σειριακό ρολόι που ελέγχεται κυρίως από την ελεγκτική συσκευή. Το SCL χρησιμοποιείται για τον συγχρονισμό των δεδομένων που εισάγονται ή εξάγονται από τη συσκευή-στόχο. Η δεύτερη γραμμή είναι το SDA, που είναι η σειριακή γραμμή δεδομένων. Το SDA χρησιμοποιείται για τη μετάδοση δεδομένων προς ή από τις συσκευές-στόχους. Για παράδειγμα, μια ελεγκτική συσκευή μπορεί να στείλει δεδομένα ρύθμισης και κωδικούς εξόδου σε έναν ψηφιακό προς αναλογικό μετατροπέα (DAC) στόχο, ή ένας στόχος αναλογικό προς ψηφιακό μετατροπέα (ADC) μπορεί να στείλει δεδομένα μετατροπής πίσω στην ελεγκτική συσκευή [19].

## 4.5 Push_Button

Πρόκειται για ένα push_button N.O (Normally Open) το οποίο φαίνεται παρακάτω στην εικόνα 4.4

Εικόνα 4.4: Το push_button

**Προδιαγραφές**

- Rating :  3 A 125 VAC; 1 A 250 VAC
- Contact Resistance: 20 mΩ max.
- Insulation Resistance: 500 VDC 100 MΩ min.
- Dielectric strength: 1000 VAC, 1 minute
- Electrical Life: 50000 cycles  [35]

# Κεφάλαιο 5ο: Network Layer

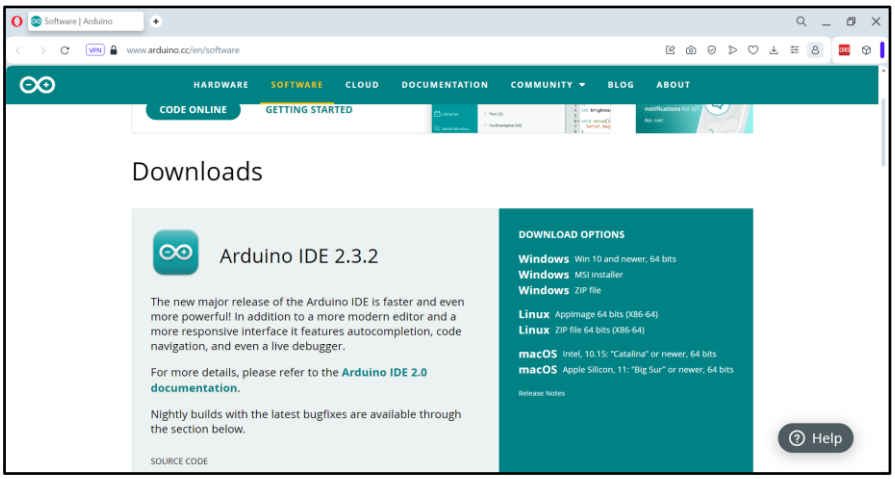
## 5.1 Εισαγωγή

Στο κεφάλαιο αυτό θα μιλήσουμε για το δεύτερο επίπεδο της αρχιτεκτονικής IoT τριών επιπέδων, που είναι το επίπεδο δικτύου/μετάδοσης -Network Layer- και είναι υπεύθυνο για τη μετάδοση δεδομένων από το επίπεδο αντίληψης/ανίχνευσης στο ανώτερο επίπεδο. Στο επίπεδο αυτό συναντάμε τον ESP32 που με βάση την Εικόνα 2.1, είναι το Gateway της εφαρμογής. Επικοινωνεί με sensor nodes, που στην περίπτωσή μας είναι ο BME280 και το push_button, αλλά και με actuator node, που στην περίπτωσή μας είναι η LCD 16x2 RGB.

## 5.2 Arduino IDE

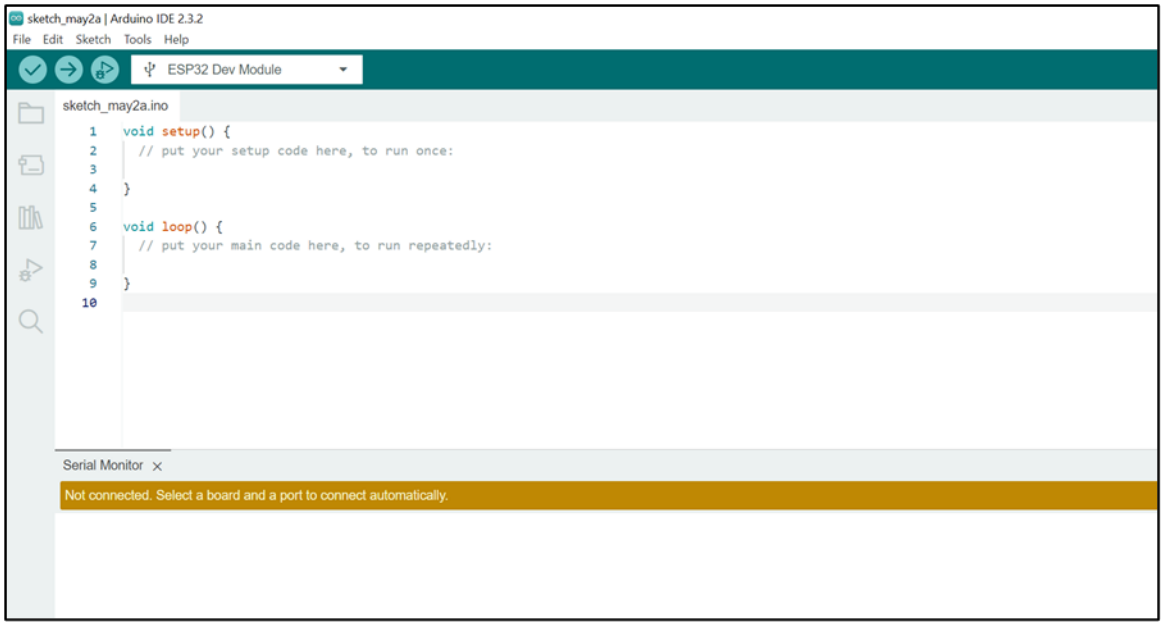
Το περιβάλλον εργασίας Arduino IDE είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης προγραμμάτων για τις πλακέτες Arduino, αλλά και για μια σειρά από άλλες πλακέτες όπως αυτές των οικογενειών ESP32 και ESP8266. Με μια φιλική προς τον χρήστη διεπαφή, το Arduino IDE επιτρέπει την εύκολη δημιουργία και επεξεργασία κώδικα μέσω ενός απλού περιβάλλοντος προγραμματισμού. Περιλαμβάνει μια πληθώρα βιβλιοθηκών και παραδειγμάτων κώδικα που καθιστούν εύκολη την εκκίνηση και την ανάπτυξη έξυπνων συσκευών με Arduino. Επιπλέον, προσφέρει δυνατότητες ανάλυσης κώδικα και ενσωματώνει εργαλεία για την διαχείριση των συσκευών Arduino, καθιστώντας το, ιδανικό εργαλείο για όλους όσους αναζητούν μια εύκολη και αποδοτική λύση για την ανάπτυξη προτύπων και πρωτότυπων.

Η εγκατάσταση του Arduino IDE -έκδοση 2.3.2 για την παρούσα εργασία- μπορεί να γίνει από την επίσημη ιστοσελίδα του Arduino, https://www.arduino.cc/en/software όπως φαίνεται και παρακάτω στην Εικόνα 5.1.




Εικόνα 5.1: Επίσημη Ιστοσελίδα Arduino [22]

Στο σημείο αυτό θα πούμε λίγα λόγια για τη χρήση του Arduino IDE, το περιβάλλον του οποίου παρουσιάζεται παρακάτω στην Εικόνα 5.2.




Εικόνα 5.2: Το περιβάλλον του Arduino IDE

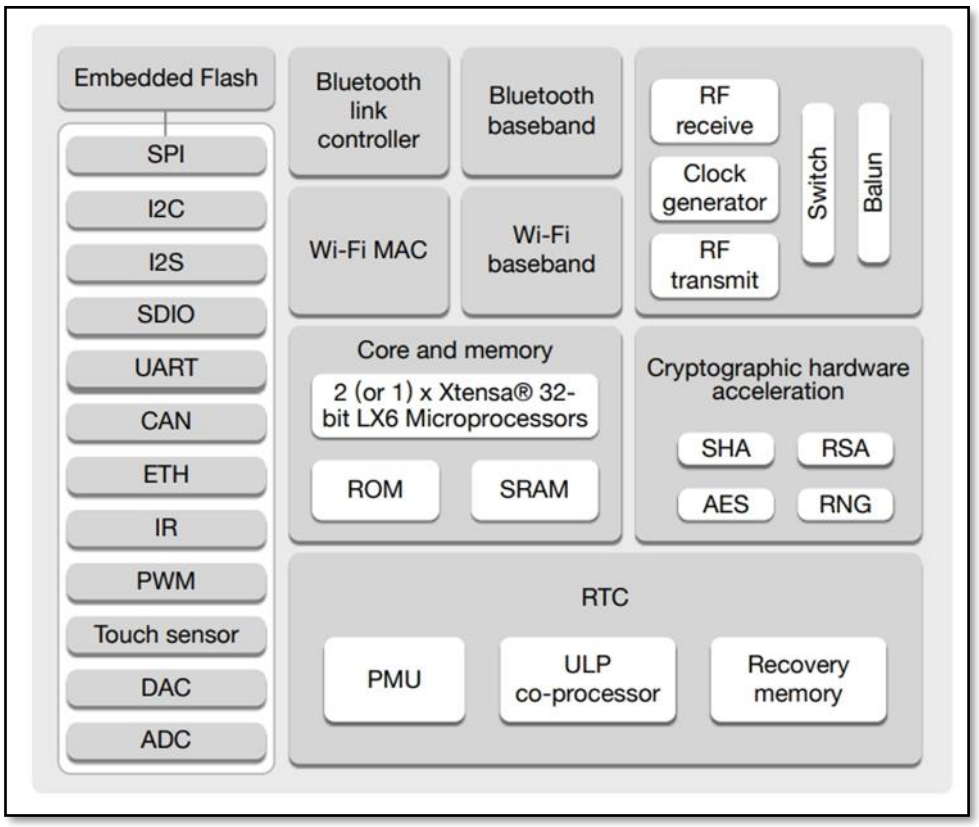
Η διεπαφή χωρίζεται σε τέσσερα (4) διακριτά τμήματα.

- **Ο συντάκτης κειμένου:** Εδώ μέσα γράφονται οι εντολές του προγράμματος. Κάθε πρόγραμμα αποτελείται από δύο τουλάχιστον συναρτήσεις, την **setup** η οποία περιλαμβάνει κώδικα που εκτελείται μόνο μια φορά κατά την εφαρμογή της τροφοδοσίας ή μετά από Reset και την **loop** η οποία περιλαμβάνει κώδικα που εκτελείται συνέχεια όσο υπάρχει τροφοδοσία.
- **Serial Monitor:** Είναι το πλαίσιο στην κάτω πλευρά και χρησιμοποιείται για την εμφάνιση πληροφοριακών μηνυμάτων π.χ εμφάνιση σφαλμάτων.
- **Menu:** Η γραμμή μενού βρίσκεται στην επάνω πλευρά και παρέχει πρόσβαση σε όλες τις λειτουργίες του IDE. Πέρα από τις τυπικές λειτουργίες, υπάρχουν κάποιες επιλογές που έχουν ιδιαίτερο ενδιαφέρον όπως η Preferences στο μενού file, που επιτρέπει την παραμετροποίηση του IDE. Ένα πρόγραμμα που αναπτύσσεται με το Arduino Ide ονομάζεται sketch και το ομώνυμο μενού παρέχει λειτουργίες όπως η επικύρωση/μεταγλώττιση του κώδικα -Verify/Compile- που πραγματοποιεί συντακτικό έλεγχο και τον μεταγλωττίζει παράγοντας εκτελέσιμο αρχείο, αλλά και η φόρτωση -upload- που εκτελεί επίσης συντακτικό έλεγχο και μεταγλώττιση του κώδικα και στην συνέχεια φορτώνει το εκτελέσιμο αρχείο στη συνδεμένη πλακέτα. Το μενού Tools δίνει πρόσβαση σε μια σειρά από εργαλεία όπως το Library Manager και το Serial Monitor.
- **Κουμπιά:** Τα κουμπιά που υπάρχουν πάνω από την περιοχή συγγραφής του κώδικα αποτελούν συντομεύσεις για κάποιες επιλογές του menou που χρησιμοποιούνται συχνά όπως οι λειτουργίες Verify/Compile και Upload του μενού Sketch [34].

## 5.3 Η αναπτυξιακή πλατφόρμα ESP32

Ο ESP32, αναπτυγμένος από την Espressif Systems, κυκλοφόρησε τον Σεπτέμβριο του 2016. Αυτός ο μικροελεγκτής είναι ο διάδοχος του ESP8266, φέρνοντας βελτιωμένες δυνατότητες όπως την υποστήριξη τόσο για Bluetooth όσο και για Wi-Fi, καθώς και μεγαλύτερη επεξεργαστική ισχύ και περισσότερους πόρους υλικού. Ο ESP32 είναι ένα σύστημα χαμηλού κόστους και χαμηλής κατανάλωσης και έχει σχεδιαστεί για έργα που σχετίζονται με το IoT και τα Embedded Systems [10].

Στην Εικόνα 5.3 φαίνεται το λειτουργικό μπλοκ διάγραμμα του ESP32.




Εικόνα 5.3: Λειτουργικό Μπλοκ Διάγραμμα ESP32 [11]

**Σύστημα και Μνήμη**: Ένας διπύρηνος, 32-bit, μικροελεγκτής με αρχιτεκτονική Harvard, Tensilica LX6, με ρυθμιζόμενη ταχύτητα CPU από 160 έως 240 MHz ενσωματώνεται στον ESP32. Τα ονόματα των δύο CPUs είναι "PRO_CPU" για πρωτόκολλα και "APP_CPU" για εφαρμογές.

Η αρχιτεκτονική διπλού πυρήνα του ESP32 του προσδίδει μεγαλύτερη ικανότητα επεξεργασίας, αφού οι δύο επεξεργαστές μπορούν να λειτουργούν ανεξάρτητα ή μαζί, και ο ESP32 είναι σε θέση να χειρίζεται πιο σύνθετες εργασίες και να επεξεργάζεται δεδομένα με μεγαλύτερη ταχύτητα.

Επίσης του προσφέρει μεγαλύτερη ευελιξία, καθώς επιτρέπει στους προγραμματιστές να αναθέτουν συγκεκριμένες εργασίες σε κάθε επεξεργαστή, δίνοντάς τους μεγαλύτερη ευελιξία στο σχεδιασμό των εφαρμογών τους. Ο ESP32 περιλαμβάνει επιπλέον χαρακτηριστικά όπως κλιμάκωση ισχύος και λειτουργίες ενέργειας.

Ο ESP32 περιλαμβάνει on-chip memory (εσωτερική μνήμη) και on-board memory (εξωτερική μνήμη). Η SRAM και η ROM παρέχουν την εσωτερική μνήμη· και οι δύο χρησιμοποιούνται για συγκεκριμένες λειτουργίες: 448 KB ROM χρησιμοποιούνται για την εκκίνηση και τις βασικές λειτουργίες, ενώ 520 KB SRAM χρησιμοποιούνται για δεδομένα και οδηγίες. Ο ESP32 υποστηρίζει έως και 16 MB ISP flash RAM [14].

**Είσοδοι και Έξοδοι**: Ο ESP32 περιέχει συνήθως 48 ακίδες (pins) στο τσιπ του με πολλαπλές λειτουργίες. Ωστόσο, ο ακριβής αριθμός των ακίδων που είναι προσβάσιμες μπορεί να διαφέρει ανάλογα με τον συγκεκριμένο ESP32 που χρησιμοποιείται. Για παράδειγμα, κάποιες πλακέτες μπορεί να εκθέτουν λιγότερες από τις 48 ακίδες για χρήση -για την εκπόνηση της παρούσης εργασίας χρησιμοποιείται ESP32 με διαθέσιμες 38 ακίδες- . Έτσι λοιπόν έχουμε:

- 18 κανάλια Μετατροπέα Αναλογικού σε Ψηφιακό (ADC)
- διεπαφές SPI
- διεπαφές UART
- διεπαφές I2C
- 16 κανάλια εξόδου PWM
- Μετατροπείς Ψηφιακού σε Αναλογικό (DAC)
- διεπαφές I2S
- 10 GPIOs για ανίχνευση χωρητικότητας

Εδώ θα πρέπει να σημειώσουμε πως η τοποθέτηση των ακίδων μπορεί να αλλάζει ανάλογα με τον κατασκευαστή [15].

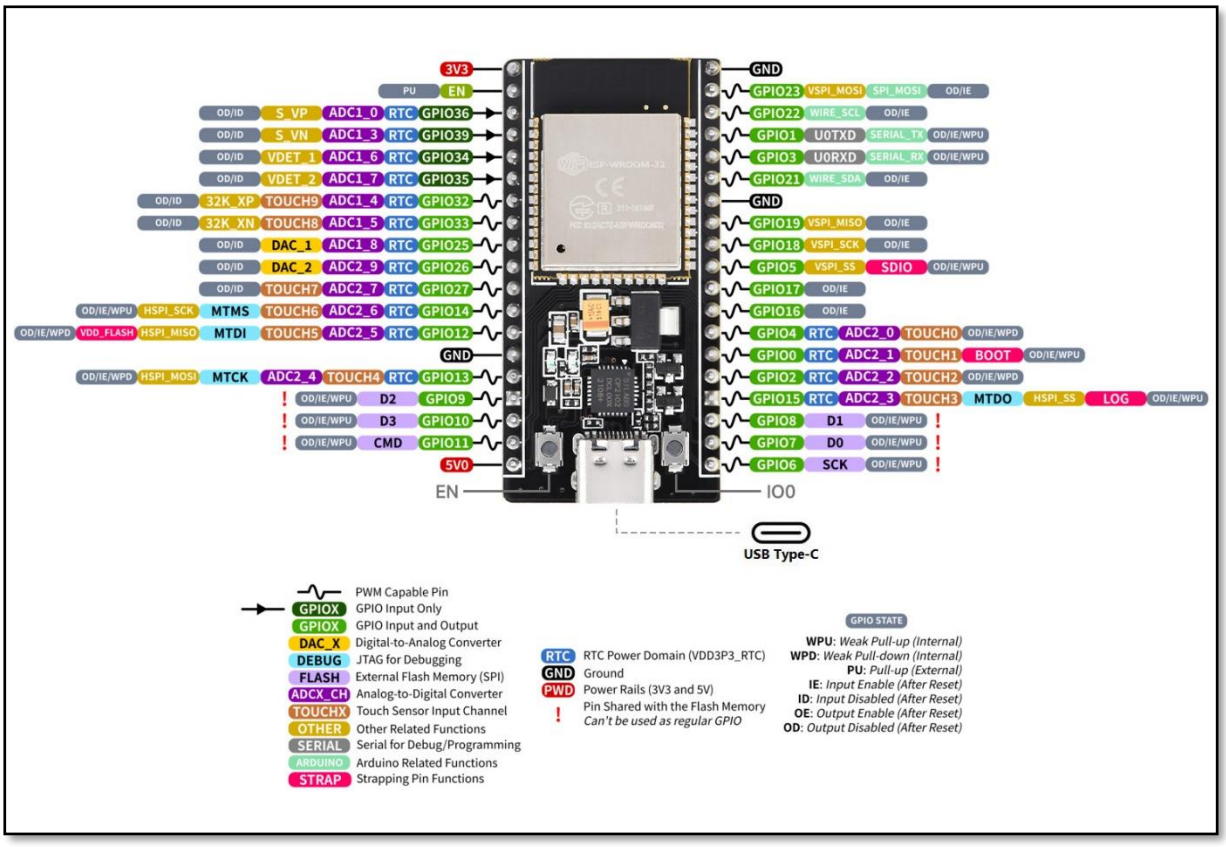
**Αρχές επικοινωνίας**: Ο ESP32 μπορεί να επικοινωνήσει με τον έξω κόσμο χρησιμοποιώντας δύο κύριες προσεγγίσεις: ενσύρματη και ασύρματη επικοινωνία. Η ασύρματη επικοινωνία επιτυγχάνεται με Wi-Fi και Bluetooth. Ο ESP32 περιλαμβάνει πλήρες 802.11 b/g/n/e/i WLAN MAC. Επομένως, μπορεί να λειτουργήσει ως σταθμός και να συνδεθεί στο Internet ή στο διακομιστή και ως σημείο πρόσβασης για να παρέχει μια διεπαφή που επιτρέπει στον χρήστη να ελέγχει ένα ενσωματωμένο σύστημα από μια εφαρμογή για κινητά. Όσον αφορά το Bluetooth, ο ESP32 παρέχει v4.2 BR/EDR και Bluetooth Low-Energy (BLE) με ταχύτητα λειτουργίας έως 4,0 Mbps [14].

**Sleep Modes**: Ιδιαίτερο χαρακτηριστικό του ESP32 είναι ότι μπορεί και υποστηρίζει λειτουργίες εξοικονόμησης ενέργειας, όπως οι λειτουργίες βαθύ ύπνου (deep sleep) και ελαφρού ύπνου (light sleep), γεγονός που τον καθιστά ιδανικό για εφαρμογές όπου η κατανάλωση ενέργειας είναι ιδιαίτερα κρίσιμη.

Στη λειτουργία βαθέως ύπνου, ο ESP32 απενεργοποιεί σχεδόν όλα τα εσωτερικά του κυκλώματα και διατηρεί μόνο ένα μικρό αριθμό υποσυστημάτων σε λειτουργία, κυρίως το Real Time Clock (RTC), το οποίο μπορεί να χρησιμοποιηθεί για να το ξυπνήσει από τον ύπνο μετά από προκαθορισμένο χρονικό διάστημα.

Σε αντίθεση με τη λειτουργία βαθέως ύπνου, η λειτουργία ελαφρού ύπνου (light sleep) του ESP32 διατηρεί περισσότερα υποσυστήματα σε λειτουργία. Κατά τη διάρκεια του ελαφρού ύπνου, οι CPU και οι περισσότεροί της περιφερειακοί ελεγκτές είναι απενεργοποιημένοι, αλλά η μνήμη RAM και τα κυκλώματα διατήρησης του RTC είναι ενεργά. Αυτό επιτρέπει στο ESP32 να ξυπνήσει πιο γρήγορα από ό,τι θα έκανε από την κατάσταση βαθέως ύπνου, καθώς δεν χρειάζεται να επαναφορτώσει το περιεχόμενο της μνήμης. Οι εφαρμογές της λειτουργίας ελαφρού ύπνου είναι ιδανικές για σενάρια όπου η συσκευή χρειάζεται να μειώσει την κατανάλωση ενέργειας, αλλά παράλληλα να μπορεί να ανταποκρίνεται γρήγορα σε εξωτερικά ερεθίσματα, όπως διακοπτικά σήματα ή άλλα σήματα αισθητήρων [15].

Για την εκπόνηση της παρούσης εργασίας θα χρησιμοποιηθεί ο ESP32 NodeMCU-32S. Η λειτουργία των ακροδεκτών του φαίνεται παρακάτω στην Εικόνα 5.4

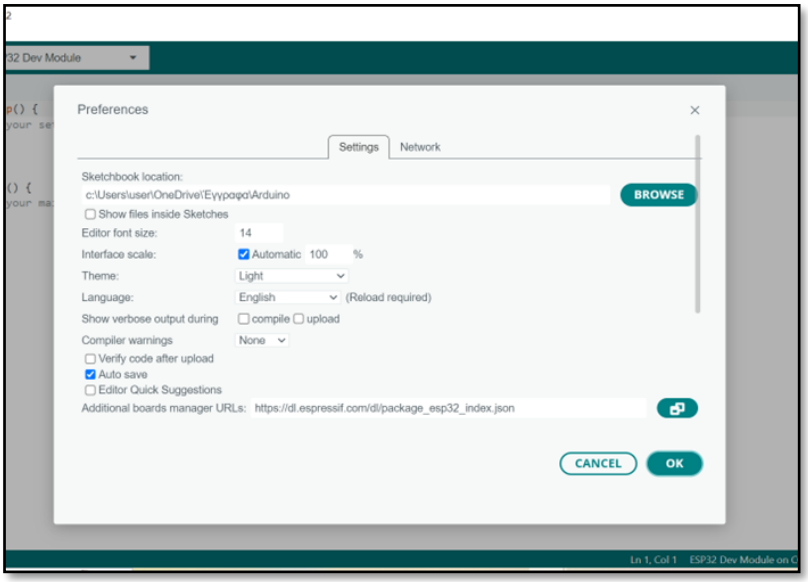



Εικόνα 5.4: Ο ESP32 NodeMCU-32S [8]

Στο σημείο αυτό θα δούμε πως θα εγκαταστήσουμε τον ESP32 στο Arduino IDE.

Πηγαίνουμε στο File → Preferences και προσθέτουμε το URL για τον διαχειριστή Πινάκων

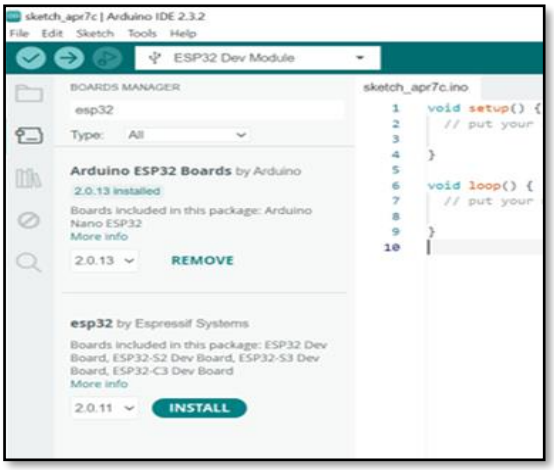
https://dl.espressif.com/dl/package_esp32_index.json και πατάμε OK, όπως φαίνεται και παρακάτω στην Εικόνα 5.5.




Εικόνα 5.5: Προσθήκη URL στον διαχειριστή Πινάκων

Στη συνέχεια στο μενού Tools → Board:ESP32 Dev Module → Boards Manager και πληκτρολογούμε ESP32.

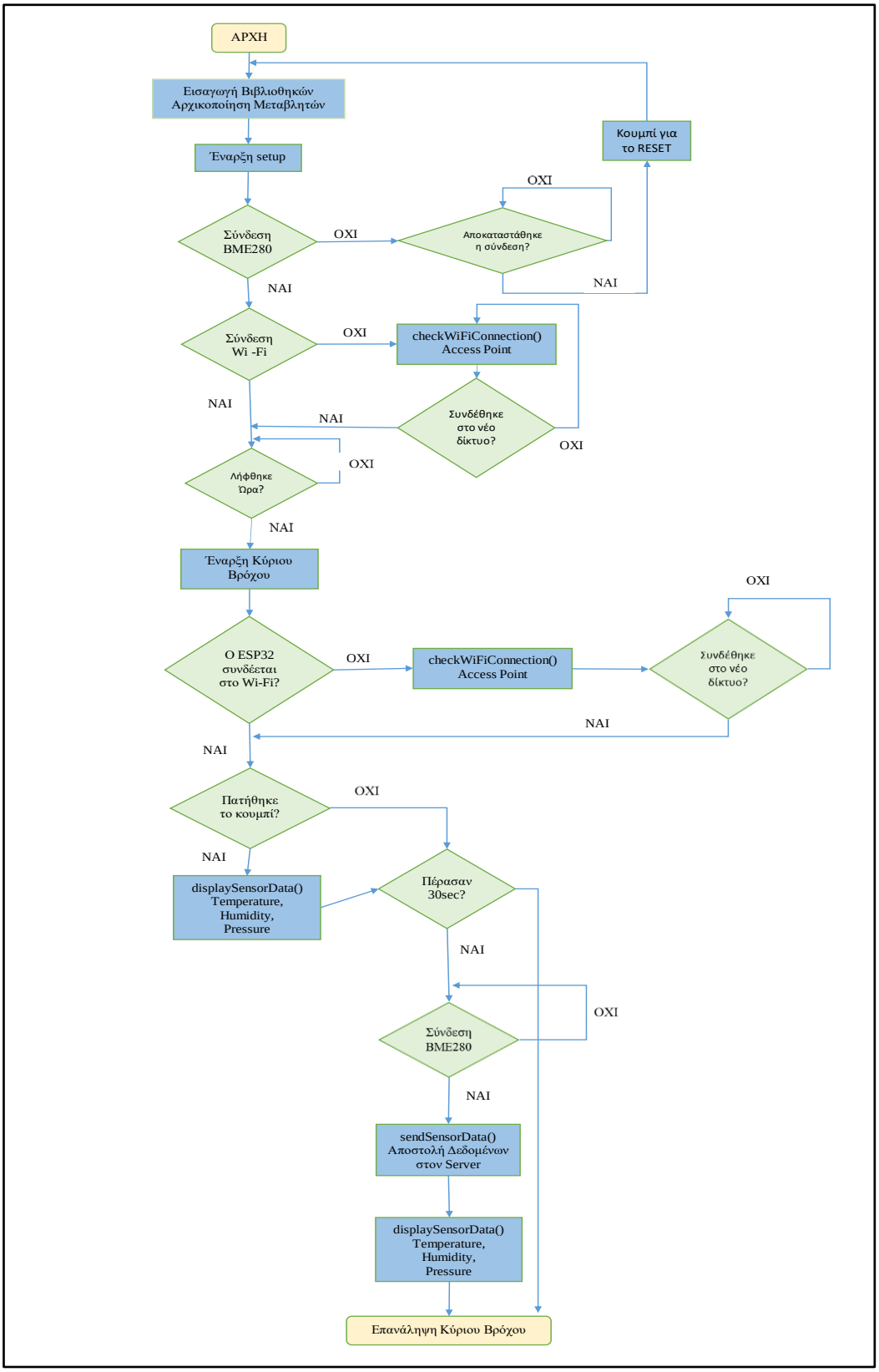
Στις επιλογές που θα δούμε, επιλέγουμε INSTALL στο esp32 by Espressif Systems όπως στην Εικόνα 5.6




Εικόνα 5.6: Εγκατάσταση ESP32 στο Arduino IDE

## 5.4 Διάγραμμα ροής (Flow Chart) Προγράμματος ESP32

Παρακάτω στο Σχήμα 5.7 δίνεται το διάγραμμα ροής για τον προγραμματισμό του ESP32 που βασίζεται στο αρχείο esp32_wifi_postdata_LCD.ino




Σχήμα 5.7: Διάγραμμα Ροής Προγράμματος ESP32

## 5.5 Εξήγηση Διαγράμματος Ροής

### 5.5.1 Εισαγωγή Βιβλιοθηκών – Αρχικοποίηση Μεταβλητών

Αρχικά εισάγουμε τις απαραίτητες βιβλιοθήκες όπως στην Εικόνα 5.8, για να μπορέσει ο ESP32 να διαχειριστεί τις περιφερειακές συσκευές όπως η οθόνη LCD 16x2 RGB <Waveshare_LCD1602_RGB.h> και ο BME280 <Adafruit_Sensor.h> και <Adafruit_BME280.h>, να επικοινωνήσει με αυτές τις συσκευές με το I2C <Wire.h> και να εκτελέσει λειτουργίες όπως να συνδεθεί στο Wi – Fi <WiFi.h> να δημιουργήσει HTTP αιτήσεις <HTTPClient.h>, να μετατραπεί σε AP <WiFiManager.h> όταν δεν μπορέσει να συνδεθεί στο αποθηκευμένο WiFi και να μπορεί να ενημερώνεται για την τρέχουσα ώρα <time.h> .

```
//Εισαγωγή των απαραίτητων βιβλιοθηκών
#include <Waveshare_LCD1602_RGB.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <HTTPClient.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <WiFiManager.h>
#include <time.h>
```

Εικόνα 5.8: Εισαγωγή απαραίτητων βιβλιοθηκών

Στη συνέχεια ακολουθεί ένα παράδειγμα εγκατάστασης μιας βιβλιοθήκης.

Στο μενού Tools επιλέγουμε Manage Libraries και πληκτρολογούμε Adafruit Unified Sensor.

Στη συνέχεια επιλέγουμε INSTALL όπως στην Εικόνα 5.9.


Εικόνα 5.9: Εγκατάσταση Βιβλιοθήκης Adafruit_Sensor

Η βιβλιοθήκη Adafruit Unified Sensor εγκαθίσταται ως ενδιάμεσο στρώμα πριν την εγκατάσταση της Adafruit_BME280.

Παραμένουμε στο Manage Libraries και αυτή τη φορά πληκτρολογούμε Adafruit BME280 Library και στη συνέχεια επιλέγουμε INSTALL όπως στην Εικόνα 5.10


Εικόνα 5.10: Εγκατάσταση Βιβλιοθήκης Adafruit_BME280

Με αντίστοιχο τρόπο εγκαθιστούμε και τις υπόλοιπες βιβλιοθήκες που χρειαζόμαστε.

Στη συνέχεια αρχικοποιούμε τις απαραίτητες μεταβλητές όπως φαίνεται στο τμήμα κώδικα στην Εικόνα 5.11

```
String apiKeyValue = "********************"; // Κλειδί API για την αποστολή δεδομένων μέσω HTTP

Adafruit_BME280 bme; // Δήλωση του αισθητήρα
Waveshare_LCD1602_RGB lcd(16, 2); // Διεύθυνση I2C και διαστάσεις οθόνης

const int buttonPin = 2;  // Η ακίδα που είναι συνδεδεμένο το κουμπί
volatile bool buttonPressed = false; // Μεταβλητή για τον έλεγχο του πατήματος του κουμπιού
int displayMode = 0; // Μετρητής για το τι να εμφανίζει η οθόνη

const char* ntpServer = "pool.ntp.org";  // Οριζμός NTP server από τον οποίο θα λαμβάνεται η ώρα
const long  gmtOffset_sec = 7200;  // Οριζμός απόκλισης ώρας σε δευτερόλεπτα από το GMT, εδώ GMT+2 ώρες για Ανατολική Ευρωπαϊκή Ζώνη
const int   daylightOffset_sec = 3600;  // Οριζμός απόκλισης ώρας για τη θερινή ώρα, προσθέτοντας 3600 δευτερόλεπτα ή μία ώρα

//Δημιουργία πινάκων που χρησιμοποιούνται για να αποθηκεύσουν τις τιμές των μετρήσεων
//που παίρνονται από τον αισθητήρα BME280 σε μορφή κειμένου, αλλά και την ώρα
char bufferT[6];
char bufferH[6];
char bufferP[10];
char timeString[20];

//Μεταβλητές πάνω και κάτω ορίων για θερμοκρασία - υγρασία - πίεση
float TempH = 25;
float TempL=24.0;
float HumH=45;
float HumL=40;
float PresH=1016.7;
float PresL=1000;

void IRAM_ATTR changeDisplayMode() {
  buttonPressed = true; // Αλλαγή της κατάστασης όταν πατηθεί το κουμπί
}
```

Εικόνα 5.11: Αρχικοποίηση Απαραίτητων Μεταβλητών

Στο σημείο αυτό να πούμε ότι ιδιαίτερη σημασία έχει το String apiKeyValue = ¨**********¨ καθώς όπως βλέπουμε είναι το κλειδί API για την αποστολή των δεδομένων μέσω HTTP. Το ίδιο κλειδί θα πρέπει να υπάρχει και στο esp32-postdata.php αρχείο που βρίσκεται στον SERVER και αποθηκεύει τα δεδομένα στη MySql.

### 5.5.2 Void setup( )

Στη συνέχεια γίνεται η έναρξη του setup όπως φαίνεται στο τμήμα κώδικα στην Εικόνα 5.12

```
void setup() {
  Serial.begin(921600);
  pinMode(buttonPin, INPUT_PULLUP); // Ορισμός της ακίδας του κουμπιού ως είσοδο με εσωτερικό pull-up resistor
  attachInterrupt(digitalPinToInterrupt(buttonPin), changeDisplayMode, FALLING); // Ρυθμιση διακοπής για την αλλαγή του display mode
                                                                                 // με το πάτημα του κουμπιού
  lcd.init(); // Αρχικοποίηση της οθόνης LCD
  lcd.setRGB(255, 255, 255); // Ενεργοποίηση του φωτισμού σε χρώμα μπλε by default
  //lcd.backlight(255); // Ενεργοποίηση του φωτισμού

  bool status = bme.begin(0x77); // Αρχικοποίηση του BME280 στη διεύθυνση (0x77)
  if (!status) {
  Serial.println("Could not find a valid BME280 sensor, check wiring!");
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.setRGB(255,255,255);
  lcd.send_string("BME280 ");
  lcd.setCursor(0, 1);
  lcd.send_string("   disconnected!");
  while (1);
  }
```

Εικόνα 5.12: void setup ( )

Μέσα στην void setup() αρχικά ορίζουμε την ακίδα στην οποία συνδέεται το κουμπί ως είσοδο, ρυθμίζουμε την διακοπή που καλείται κάθε φορά που πατιέται το push_button και αρχικοποιούμε την LCD 16x2 RGB οθόνη. Μέσα στην setup γίνεται και η αρχικοποίηση του BME280. Σε περίπτωση που ο ESP32 δεν αρχικοποιήσει τον BME280 στην LCD 16x2 RGB θα εμφανιστεί το μήνυμα “BME280 disconnected!” όπως βλέπουμε στην Εικόνα 5.13 και το πρόγραμμα **ΔΕΝ** θα προχωρήσει. Αν αποκαταστήσουμε τη σύνδεση με τον BME280 **θα πρέπει να πατήσουμε το κουμπάκι reset στον ESP32**.

**Να σημειώσουμε ότι η LCD 16x2 RGB προφανώς και δεν ανήκει στο Network Layer. Δίνονται οι παρακάτω εικόνες της, για καλύτερη κατανόηση της εργασίας από τον αναγνώστη.**


Εικόνα 5.13: Ο BME280 είναι αποσυνδεμένος

Στη συνέχεια και ενώ παραμένουμε στη συνάρτηση void setup () γίνεται η προσπάθεια σύνδεσης στο Wi – Fi που ο ESP32 έχει αποθηκευμένο στη μνήμη του, σύμφωνα με το τμήμα κώδικα που βλέπουμε στην Εικόνα 5.14

```
WiFi.begin();
  unsigned long startConnection = millis(); // Αποθήκευση της τρέχουσας χρονικής στιγμής

  // Περιμένουμε μέχρι να συνδεθούμε στο WiFi ή να περάσουν 30 δευτερόλεπτα
  while (WiFi.status() != WL_CONNECTED && millis() - startConnection < 30000) {
    delay(500);
    Serial.print(".");
    //lcd.clear();
    lcd.setCursor(0, 0);
    lcd.setRGB(255,255,255);
    lcd.send_string("connection...");
  }
  // // Έλεγχος αν η σύνδεση απέτυχε
  if(WiFi.status() != WL_CONNECTED) {
    Serial.println("Απέτυχε η σύνδεση στο WiFi");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.setRGB(255,255,255);
    lcd.send_string("connection ");
    lcd.setCursor(7, 1);
    lcd.send_string("failed");
    delay(3000);
    checkWiFiConnection(); // Καλούμε την checkWiFiConnection σε περίπτωση αποτυχίας
  } else {
    Serial.println("Επιτυχία Σύνδεσης");
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP()); // Εμφάνιση IP διεύθυνσης
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.setRGB(255,255,255);
    lcd.send_string("connection ");
    lcd.setCursor(7, 1);
    lcd.send_string("success!!!");
  }
```

Εικόνα 5.14: void setup () σύνδεση Wi - Fi

Στην LCD 16x2 RGB βλέπουμε το μήνυμα “connection…” όπως στην Εικόνα 5.15


Εικόνα 5.15: Προσπάθεια σύνδεσης στο Wi -Fi

Αν η σύνδεση είναι επιτυχημένη στην LCD 16x2 RGB θα δούμε το μήνυμα “connection success!!!” όπως φαίνεται στην Εικόνα 5.16


Εικόνα 5.16: Επιτυχημένη Σύνδεση

Στην αντίθετη περίπτωση και αφού περάσουν 30sec θα δούμε στην LCD 16x2 RGB το μήνυμα “connection failed” όπως στην Εικόνα 5.17 και θα κληθεί η συνάρτηση checkWiFiConnection ( ) που θα ελέγξει ξανά την σύνδεση και θα οδηγήσει τον ESP32 σε λειτουργία AP, περιμένοντας ρυθμίσεις για να συνδεθεί σε καινούριο δίκτυο. Στην LCD 16x2 RGB θα δούμε το μήνυμα “ACCESS POINT” όπως στην Εικόνα 5.18


Εικόνα 5.17: Η Σύνδεση Απέτυχε


Εικόνα 5.18: Ο ESP32 λειτουργεί ως AP

Εδώ πρέπει να σημειώσουμε, ότι το πρόγραμμα **ΔΕΝ** μπορεί να προχωρήσει μέχρι ο ESP32 να συνδεθεί στο καινούριο δίκτυο.

Τέλος ο ESP32 θα προσπαθήσει να λάβει και την ώρα σύμφωνα με το τμήμα κώδικα που βλέπουμε στην Εικόνα 5.19. Αν η διαδικασία είναι επιτυχής, η void setup () κλείνει και το πρόγραμμα συνεχίζει στη void loop (). Αν αποτύχει να λάβει ώρα το πρόγραμμα **ΔΕΝ** μπορεί να προχωρήσει.

```
// Ρύθμιση NTP
configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);

// Αναμονή για τη λήψη ώρας
// Αυτός ο βρόχος εκτελείται επαναληπτικά όσο η συνάρτηση time επιστρέφει null, δηλαδή όσο δεν υπάρχει διαθέσιμη ώρα.
while (!time(nullptr)) {
  Serial.print(".");
  delay(1000);
}
Serial.println("Time received!");
}
```

Εικόνα 5.19: void setup () λήψη ώρας.

### 5.5.3 Void loop ( )

Στη συνέχεια γίνεται η έναρξη της void loop ( ) σύμφωνα με το τμήμα κώδικα που βλέπουμε στην Εικόνα 5.20 όπου αρχικά ορίζονται οι απαραίτητες static μεταβλητές καθώς και η currentMillis, για την αποθήκευση της τρέχουσας χρονικής σήμανσης. Στην συνέχεια καλείται η checkWiFiConnection ( ) για τον έλεγχο της σύνδεσης. Όπως αναφέρθηκε και πιο πάνω, αν η σύνδεση έχει πέσει θα οδηγήσει τον ESP32 σε λειτουργία AP. Όπως και πριν στην void setup( ), το πρόγραμμα **ΔΕΝ** μπορεί να προχωρήσει μέχρι ο ESP32 να συνδεθεί στο καινούριο δίκτυο.

Στη συνέχεια γίνεται ο έλεγχος αν έχει πατηθεί το push_button, που θα προκαλέσει διακοπή και θα κληθεί η displaySensorData ( ), για να ενημερώσει την οθόνη με νέα δεδομένα.

```
void loop() {

static unsigned long lastDisplayUpdate = 0; // Τελευταία ενημέρωση οθόνης
static unsigned long lastDataSend = 0; // Τελευταία αποστολή δεδομένων
unsigned long currentMillis = millis(); // Τρέχον χρονικό σημείο

checkWiFiConnection();

if (buttonPressed) { //Έλεγχος εάν το κουμπί έχει πατηθεί
displayMode = (displayMode + 1) % 3; // Κυκλική αλλαγή μεταξύ των λειτουργιών εμφάνισης
buttonPressed = false; // Επαναφορά της σημαίας για το πάτημα του κουμπιού
delay(100);
displaySensorData(); // Ενημέρωση της οθόνης με τα νέα δεδομένα
}
```

Εικόνα 5.20: void loop ( )

Τέλος και ενώ βρισκόμαστε ακόμη στην loop ( ) -Εικόνα 5.21- θα γίνει ο έλεγχος αν υπάρχει η σύνδεση στο Wi- Fi, αν έχει περάσει ο χρόνος που έχει οριστεί από τον administrator -στην περίπτωσή μας 30sec- και αν ο BME280 παραμένει σε σύνδεση και θα κληθεί η sendSensorData( ) για να στείλει τις μετρήσεις στον server και στη συνέχεια πάλι μετά τους κατάλληλους ελέγχους, θα κληθεί η displaySensorData( ) που θα εμφανίσει τις μετρήσεις στην LCD 16x2 RGB.

```
if (WiFi.status() == WL_CONNECTED && currentMillis - lastDataSend >= 30000) {
    lastDataSend = currentMillis;
    if (!bme.begin(0x77)) {
    Serial.println("BME280 disconnected!");
    }else {
       sendSensorData(); //Αποστολή δεδομένων στον server
    }
}
if (currentMillis - lastDisplayUpdate >= 30000) {
    lastDisplayUpdate = currentMillis;
    if (!bme.begin(0x77)) {
    Serial.println("BME280 disconnected!");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.setRGB(255,255,255);
    lcd.send_string("BME280 ");
    lcd.setCursor(0, 1);
    lcd.send_string("  disconnected!");
    }else{
      displaySensorData();  //Εμφάνιση δεδομένων στην οθόνη
    }
    }
}
```

Εικόνα 5.21: void loop ( ) αποστολή μετρήσεων στον server, εμφάνιση στην LCD 16x2 RGB.

Εδώ κλείνει και η void loop ( ).

### 5.5.4 Συναρτήσεις

Εδώ θα γίνει η περιγραφή των τριών συναρτήσεων, checkWiFiConnection ( ), sendSensorData( ) και displaySensorData( ).

Στην Εικόνα 5.22 βλέπουμε τον κώδικα για την συνάρτηση checkWiFiConnection ( ).

```
void checkWiFiConnection() {
  if (WiFi.status() != WL_CONNECTED) { // Έλεγχος εάν δεν υπάρχει σύνδεση WiFi
    lcd.clear();
    lcd.setCursor(2, 0);
    lcd.setRGB(255,255,255);
    lcd.send_string("ACCESS POINT");
    WiFiManager wifiManager;     // Δημιουργία Αντικειμένου της κλάσσης WiFiManager
    wifiManager.autoConnect("ESP32_AP"); //Δημιουργία access point με όνομα "ESP32_AP"
  }
}
```

Εικόνα 5.22: Η συνάρτηση checkWiFiConnection ( )

Στην συνάρτηση αυτή αρχικά γίνεται ο έλεγχος εάν δεν υπάρχει σύνδεση Wi-Fi και αν αυτό ισχύει, με τη βοήθεια ενός αντικειμένου της κλάσσης WiFiManager, ο ESP32 μετατρέπεται σε AP.

Στην Εικόνα 5.23 βλέπουμε τον κώδικα για την συνάρτηση sendSensorData( ).

```
void sendSensorData(){
  if (WiFi.status() == WL_CONNECTED) { // Αποστολή δεδομένων μόνο εάν το WiFi είναι συνδεδεμένο
  WiFiClientSecure client; // Δημιουργία ενός ασφαλούς WiFiClient
  client.setInsecure();  // Ρύθμιση του πελάτη να μην ελέγχει την εγκυρότητα
  HTTPClient https; //Δημιουργία αντικειμένου για την διαχείριση HTTP POST

  String serverPath = "https://users.iee.ihu.gr/~iee2019243/Diplomatica/esp32-postdata.php"; //Διεύθυνση URL για αποστολή μετρήσεων

  https.begin(client, serverPath); // Αρχίζει μια HTTPS σύνδεση
  https.addHeader("Content-Type", "application/x-www-form-urlencoded"); // Ορισμός του τύπου του περιεχομένου της αίτησης

  String httpRequestData = "api_key=" + apiKeyValue + "&value1=" + String(bme.readTemperature()) +
                           "&value2=" + String(bme.readHumidity()) + "&value3=" + String(bme.readPressure() / 100.0F);
  int httpResponseCode = https.POST(httpRequestData); //Αποστολή των δεδομένων POST

  if (httpResponseCode > 0) {
    Serial.print(httpRequestData);
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
  } else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
  https.end();  // Λήξη της σύνδεσης
}
}
```

Εικόνα 5.23: Η συνάρτηση sendSensorData()

Η sendSensorData( ), είναι η συνάρτηση που χειρίζεται την αποστολή των μετρήσεων για την αποθήκευση στον server. Αφού ελέγξει την σύνδεση στο Wi – Fi, δημιουργεί μια σύνδεση HTTPS με τον server και αποστέλλει ένα αίτημα HTTP POST περιέχοντας τις τιμές από τον BME280 αλλά και το apiKeyValue. Έπειτα ελέγχει και εμφανίζει τον κωδικό απόκρισης HTTP στην περιοχή σειριακής επικοινωνίας (Serial Monitor) για ενημέρωση του administrator για την επιτυχία ή την αποτυχία της αποστολής και κλείνει την σύνδεση.

Τέλος έχουμε την συνάρτηση displaySensorData( ), για την εμφάνιση των μετρήσεων στην LCD 16x2 RGB. Η displaySensorData( ) δίνεται στην Εικόνα 5.24.

```
//Συνάρτηση απεικόνισης τιμών στην οθόνη
void displaySensorData() {

float temperature = bme.readTemperature();
float humidity = bme.readHumidity();
float pressure = bme.readPressure() / 100.0F; // Μετατροπή σε hPa

// Λήψη της τρέχουσας ώρας
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("Failed to obtain time");
    return;  // Αποτυχία λήψης ώρας, διακοπή της λειτουργίας
  }

lcd.clear(); // Καθαρισμός της οθόνης πριν την εμφάνιση δεδομένων
```

Εικόνα 5.24: Αρχίζει η συνάρτηση displaySensorData( ).

Αρχικά ορίζονται τρεις float μεταβλητές για να αποθηκεύσουν τις μετρήσεις από τον BME280. Στη συνέχεια λαμβάνεται η ώρα και καθαρίζει η οθόνη.

Στη συνέχεια με μια **switch – case** και ανάλογα με την τιμή του μετρητή displayMode εμφανίζεται το αντίστοιχο μέγεθος στην LCD 16x2 RGB. Με case 0 η Θερμοκρασία όπως φαίνεται στην Εικόνα 5.25

παρακάτω, και με αντίστοιχο τρόπο -παραλείπεται ο κώδικας- με case 1 η Υγρασία και με case 2 η Ατμοσφαιρική Πίεση.

```
switch (displayMode) {
case 0: // Θερμοκρασία
lcd.setCursor(0, 0);
if (temperature >= TempL && temperature <= TempH)
{
  lcd.setRGB(0,255,0);
}
  else if(temperature > TempH)
{
    lcd.setRGB(255,0,0);
}
else lcd.setRGB(255,255,255);
lcd.send_string("Temp: ");
dtostrf(temperature, 4, 2, bufferT);// Μορφοποίηση της θερμοκρασίας σε string
lcd.send_string(bufferT);
lcd.send_string((" "+String(char(223)) + "C").c_str());
strftime(timeString, sizeof(timeString), "%H:%M:%S", &timeinfo);
lcd.setCursor(4, 1);
lcd.send_string(timeString); // Εμφάνιση της ώρας
break;
```

Εικόνα 5.25: case 0 εμφανίζεται η Θερμοκρασία στην LCD 16x2 RGB

## 5.6 Ο ESP32 ως Access Point

Είδαμε παραπάνω, ότι τόσο κατά την void setup ( ) που καλείται κατά την εκκίνηση του προγράμματος, όσο και κατά την void loop ( ) που εκτελείται συνεχώς, αν ο ESP32 δεν μπορέσει να συνδεθεί στο Wi – Fi που έχει αποθηκευμένο στη μνήμη του, τότε μετατρέπεται σε AP με την κλήση της συνάρτησης checkWiFiConnection ( ).

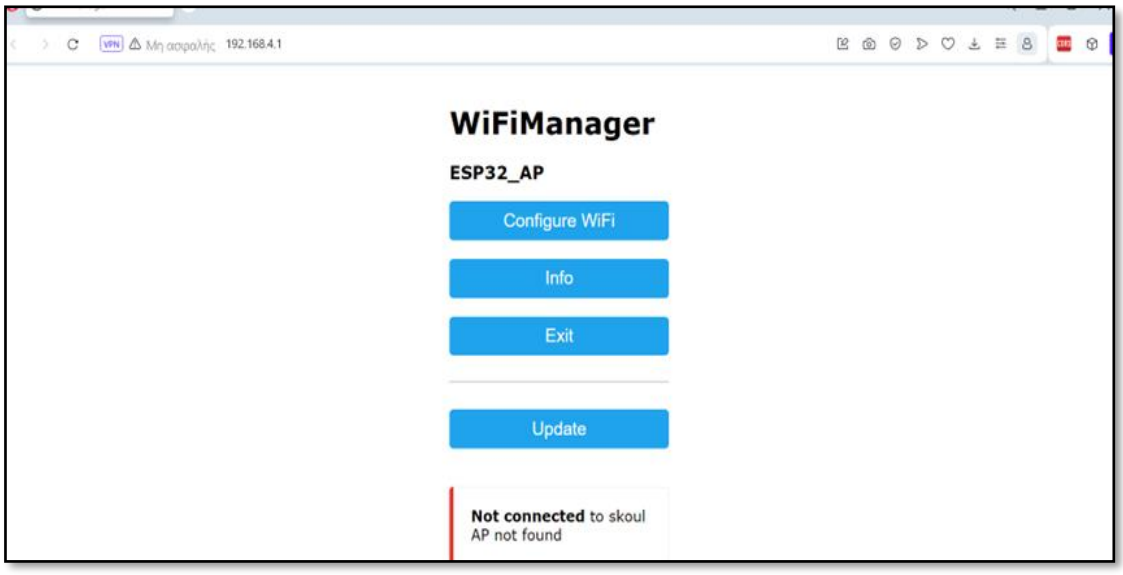
Αυτή είναι μια ιδιαίτερα χρήσιμη λειτουργία, γιατί κάνει το σύστημά μας παραμετροποιήσιμο, δηλαδή επιτρέπει στον απλό χρήστη να μεταφέρει τον ESP32_WS οπουδήποτε θέλει και μέσα από το smartphone ή το laptop του, να συνδέσει τον ESP32 στο υπάρχον Wi -Fi ή δίκτυο κινητής τηλεφωνίας.

Όταν λοιπόν ο ESP32 γίνει AP, με μια σάρωση για εύρεση δικτύων από τη φορητή συσκευή μας θα δούμε μια εικόνα όπως η παρακάτω Εικόνα 5.26


Εικόνα 5.26: Ο ESP32 έγινε AP

Βλέπουμε λοιπόν ότι ο ESP32 δημιουργεί ένα τοπικό ασύρματο δίκτυο με SSID ESP32_AP.

Συνδεόμαστε λοιπόν στο δίκτυο ESP32_AP -δεν χρειάζεται κωδικός πρόσβασης- και βλέπουμε την παρακάτω εικόνα 5.27




Εικόνα 5.27: Σύνδεση από τη φορητή συσκευή μας στο AP του ESP32

Ας πούμε λίγα λόγια για τις επιλογές που εμφανίζονται.

- **WiFi Manager:** Αυτός είναι ο τίτλος της διεπαφής, δηλώνει ότι βρίσκόμαστε στο μενού διαχείρισης του WiFi Manager.
- **ESP32_AP:** Όπως αναφέραμε και πιο πάνω, αυτό είναι το SSID του δικτύου που έχει δημιουργήσει ο ESP32. Δείχνει το όνομα του δικτύου WiFi στο οποίο είμαστε συνδεδεμένοι.
- **Configure WiFi:** Αυτή η επιλογή μας επιτρέπει να αλλάξουμε τα διαπιστευτήρια Wi - Fi που θέλουμε ο ESP32 να χρησιμοποιεί για να συνδεθεί σε άλλο δίκτυο, όπως το SSID και τον κωδικό πρόσβασης. Αυτή την επιλογή θα χρειαστούμε στην περίπτωσή μας.
- **Info:** Αυτή η επιλογή παρέχει γενικές πληροφορίες για τη συσκευή.
- **Exit:** Αυτή η επιλογή μας επιτρέπει να βγούμε από το μενού WiFi Manager.
- **Update:** Εδώ έχουμε τη δυνατότητα να ανεβάσουμε και να εγκαταστήσουμε μια νέα έκδοση του λογισμικού που ελέγχει τον ESP32.
- Τέλος κάτω – κάτω βλέπουμε ότι ο ESP32 δεν μπόρεσε να συνδεθεί στο δίκτυο που έχει αποθηκευμένο στη μνήμη του -**Not connected** to skoul-.

Επιλέγοντας λοιπόν Configure WiFi βλέπουμε την παρακάτω Εικόνα 5.28:


Εικόνα 5.28: Συνδέουμε τον ESP32 στο νέο δίκτυο

Δίνοντας το SSID και το Password συνδέουμε τον ESP32 στο νέο δίκτυο που θέλουμε. Επιλέγουμε Save και πλέον ο ESP32_WS λειτουργεί κανονικά έχοντας αποθηκεύσει τα στοιχεία -SSID και Password- του νέου δικτύου.

# Κεφάλαιο 6ο: Application Layer

## 6.1 Εισαγωγή

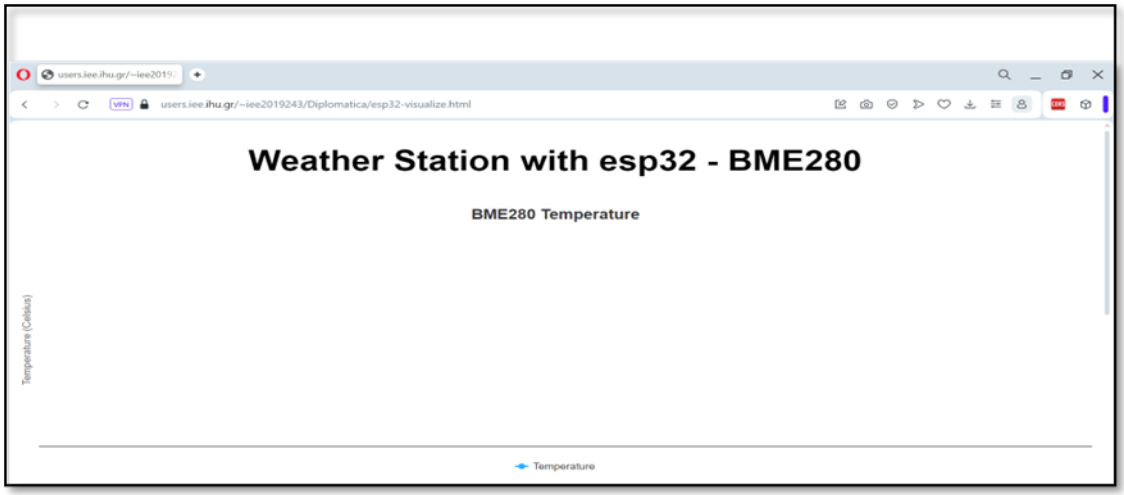
Στο κεφάλαιο αυτό θα μιλήσουμε για το τρίτο επίπεδο της αρχιτεκτονικής IoT τριών επιπέδων, που είναι το επίπεδο εφαρμογής -Application Layer- και είναι υπεύθυνο για την αποστολή της πληροφορίας μέσω διαδικτύου και την αποθήκευση της, στην απομακρυσμένη βάση δεδομένων, στον users.iee.ihu.gr. Το επόμενο βήμα είναι οι πληροφορίες αυτές να είναι διαθέσιμες στον χρήστη από οπουδήποτε μέσω μιας δυναμικής ιστοσελίδας. Η απεικόνιση δεδομένων -Visualization Data- επιτρέπει στον απλό χρήστη να διακρίνει πρότυπα, τάσεις και σχέσεις στα δεδομένα που μπορεί να μην είναι αμέσως εμφανή. Μέσω της απεικόνισης καθιστούμε τα δεδομένα πιο αντιληπτά και κατανοητά, επιτρέποντας μας να αποκτήσουμε μια βαθύτερη κατανόηση σε περίπλοκες πληροφορίες και συστήματα.

## 6.2 Frontend

Στο Frontend το URL της ιστοσελίδας είναι https://users.iee.ihu.gr/~iee2019243/Diplomatica/esp32-visualize.html.Με αυτό το URL ουσιαστικά ζητάμε από τον server το αρχείο esp32-visualize.html. Είναι αυτό το οποίο θα εμφανίσει τα διαγράμματα με τις μετρήσεις των μεγεθών.

### 6.2.1 Αρχική Οθόνη

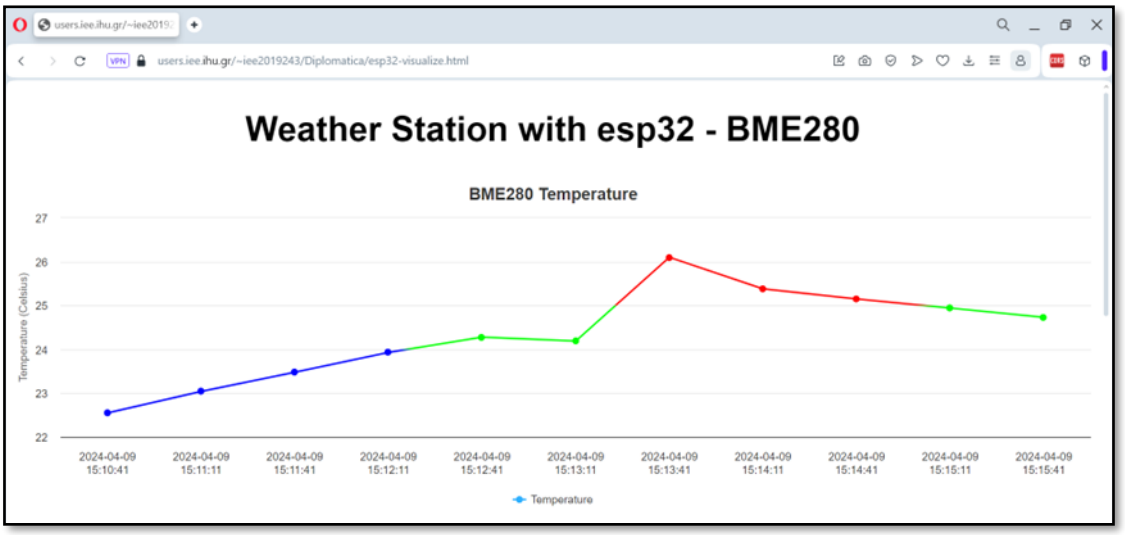
Στην Εικόνα 6.1 φαίνεται η αρχική οθόνη που θα δει ο user στο HTML – Javascript αρχείο, και συγκεκριμένα όταν ακόμη δεν έχει σταλεί καμία μέτρηση στη βάση δεδομένων, δηλαδή όταν ο πίνακας sensor που αποθηκεύει τις μετρήσεις στη MySql είναι άδειος.



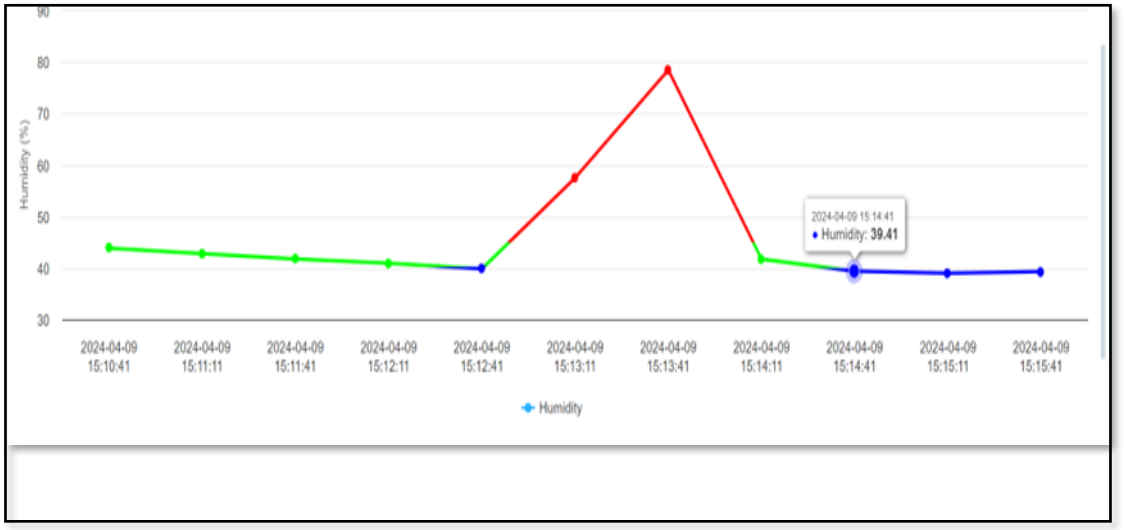

Εικόνα 6.1: Αρχική οθόνη χωρίς αποθηκευμένες τιμές μεγεθών

### 6.2.2 Διαγράμματα Μετρήσεων

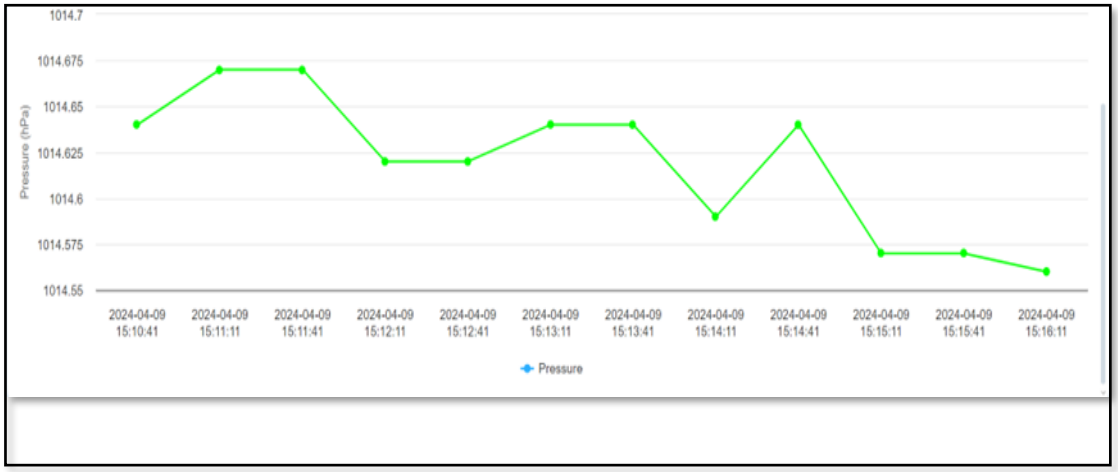
Στη συνέχεια και καθώς αρχίζουν να αποθηκεύονται οι τιμές των μεγεθών -θερμοκρασία, υγρασία και ατμοσφαιρική πίεση- αρχίζουν να σχηματίζονται δυναμικά και τα αντίστοιχα διαγράμματα. Της θερμοκρασίας όπως στην Εικόνα 6.2, της Υγρασίας όπως στην Εικόνα 6.3 και της Ατμοσφαιρικής Πίεσης όπως στην Εικόνα 6.4




Εικόνα 6.2: Διάγραμμα Θερμοκρασίας



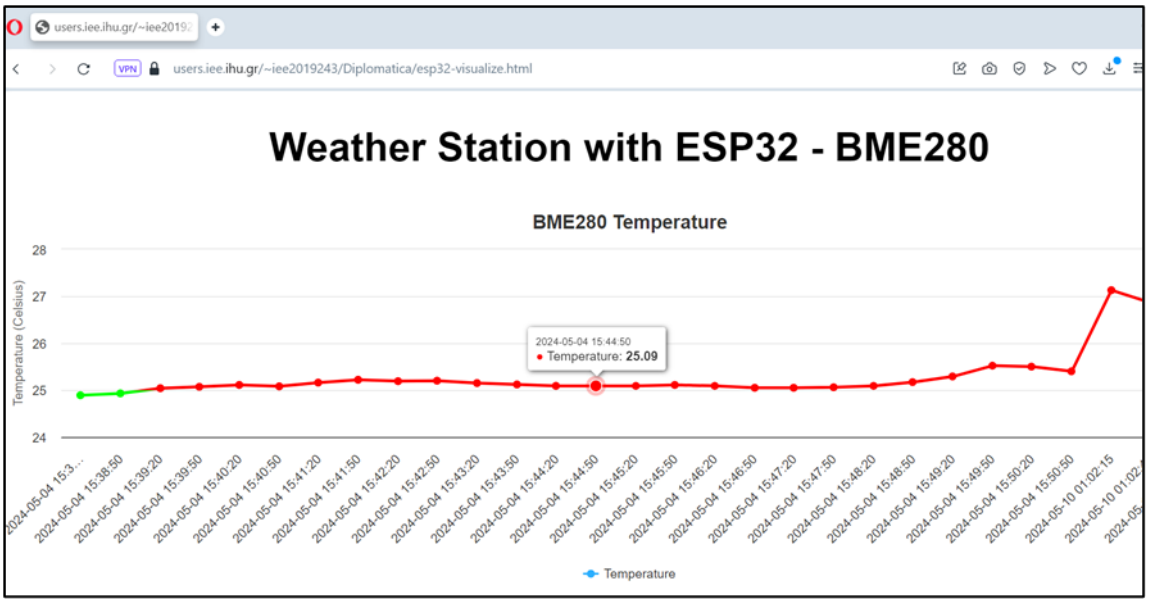

Εικόνα 6.3: Διάγραμμα Υγρασίας




Εικόνα 6.4 Διάγραμμα Ατμοσφαιρικής Πίεσης

Στο κάθε διάγραμμα αποθηκεύονται δυναμικά οι 30 πιο πρόσφατες τιμές και ανάλογα με το χρονικό διάστημα που έχουμε ορίσει για να γίνεται η μέτρηση. Για τις ανάγκες της παρουσίασης της εργασίας, το διάστημα αυτό έχει οριστεί στα 30sec.

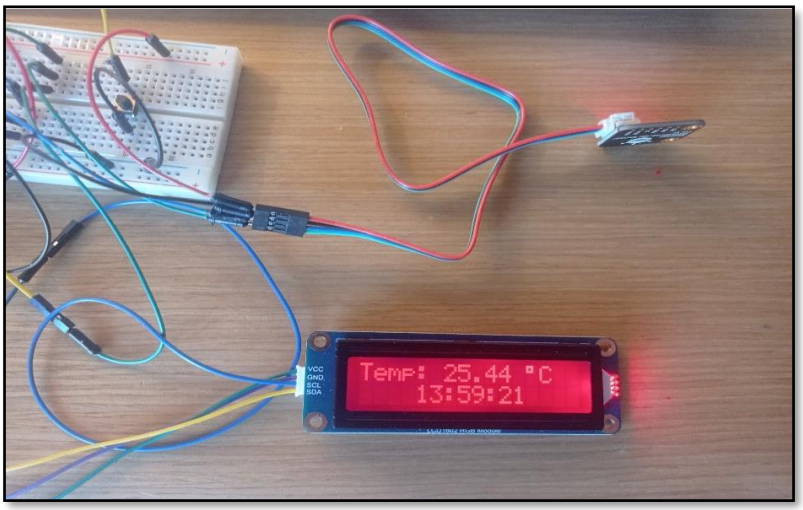
Στον οριζόντιο άξονα κάθε διαγράμματος φαίνεται η χρονική στιγμή λήψης της μέτρησης, ενώ στον κατακόρυφο έχουμε την τιμή του κάθε μεγέθους, σε $^{o}$C για την θερμοκρασία, σε % για την υγρασία και σε hPa -hectopascals- για την Ατμοσφαιρική Πίεση. Μετακινώντας τον κέρσορα του υπολογιστή μας σε κάποιο από τα σημεία – κόμβους μιας από τις καμπύλες ή ακουμπώντας ένα από αυτά τα σημεία στην touch οθόνη του smartphone μας , βλέπουμε τις πληροφορίες για την τιμή του μεγέθους που θέλουμε και για την συγκεκριμένη χρονική στιγμή, όπως φαίνεται παρακάτω στην Εικόνα 6.5




Εικόνα 6.5 Ο κέρσορας μεταφέρεται σε κάποιο σημείο – κόμβο της καμπύλης

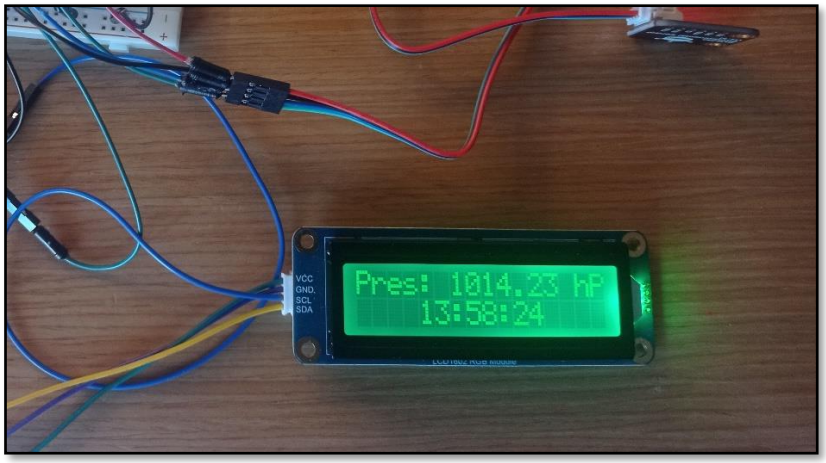
Βλέποντα τις παραπάνω εικόνες, πολύ εύκολα μπορούμε να παρατηρήσουμε ότι στα διαγράμματα εμφανίζονται τρία (3) χρώματα. Αυτό οφείλεται στο γεγονός ότι υπάρχουν πάνω και κάτω όρια κατά την απεικόνιση των μεγεθών. Σε περίπτωση που η τιμή σε κάποιο από τα τρία μεγέθη ξεπεράσει το επάνω όριο, το αντίστοιχο διάγραμμα χρωματίζεται κόκκινο. Αν η τιμή σε κάποιο από τα τρία μεγέθη πέσει κάτω από το κάτω όριο, το αντίστοιχο διάγραμμα χρωματίζεται μπλε και όταν η μετρούμενη τιμή βρίσκεται ανάμεσα στο κάτω και το επάνω όριο το διάγραμμα χρωματίζεται πράσινο.

Αντίστοιχα αλλάζει και το χρώμα στο background της LCD 16x2 RGB, όπως βλέπουμε στην Εικόνα 6.6 για την Θερμοκρασία, στην Εικόνα 6.7 για την Υγρασία και στην Εικόνα 6.8 για την Ατμοσφαιρική Πίεση. **Να σημειώσουμε ότι η LCD 16x2 RGB προφανώς και δεν ανήκει στο Application Layer. Δίνονται οι παρακάτω εικόνες της, για καλύτερη κατανόηση της εργασίας από τον αναγνώστη.**




Εικόνα 6.6: Η θερμοκρασία ξεπέρασε το πάνω όριο

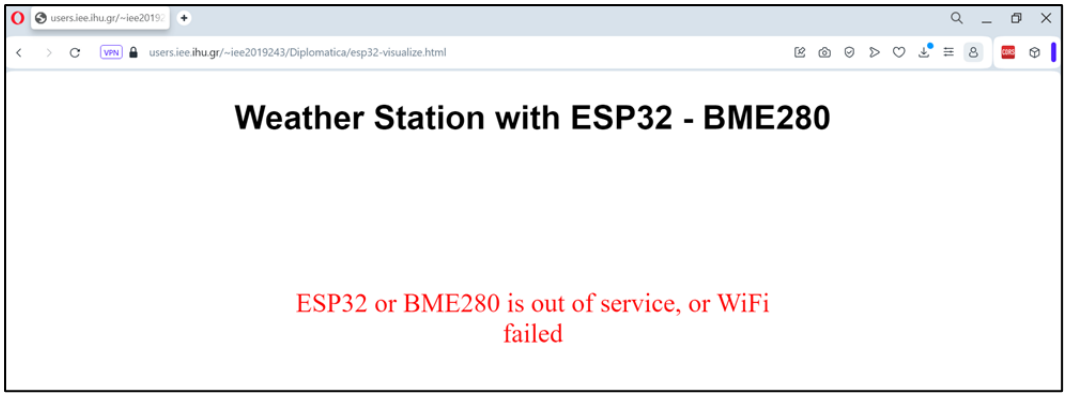

Εικόνα 6.7: Η υγρασία έπεσε από το κάτω όριο




Εικόνα 6.8: Η Ατμοσφαιρική Πίεση βρίσκεται εντός ορίων

### 6.2.3 Η MySql δεν ενημερώνεται

Αν για κάποιο λόγο η βάση δεδομένων δεν ενημερωθεί με τιμές για συγκεκριμένο χρονικό διάστημα -έχει οριστεί στο 1min- τότε το HTML – javascript αρχείο σταματάει να εμφανίζει τα διαγράμματα και αντί αυτών ο user θα δει ένα μήνυμα, ότι κάτι δεν πηγαίνει καλά στο σύστημα όπως φαίνεται στην Εικόνα 6.9




Εικόνα 6.9: Εμφάνιση Μηνύματος Βλάβης Συστήματος

## 6.3 Backend

Στο Backend ανήκουν τα PHP αρχεία που είναι αποθηκευμένα στον server, όπως αναφέρονται και στο Κεφάλαιο 3 -παράγραφος 3.3- καθώς και η MySql βάση δεδομένων για την αποθήκευση των μετρήσεων. Από τα PHP αρχεία, το esp32-postdata.php αποτελεί και το Endpoint για τα HTTPS POST αιτήματα του ESP32, ενώ το esp32-visualize.php αποτελεί το Endpoint για τα HTTPS GET αιτήματα της Javascript, από το αρχείο esp32-visualize.html, ενώ στο αρχείο auth.php αποθηκεύονται τα απαραίτητα username, password και API keys.

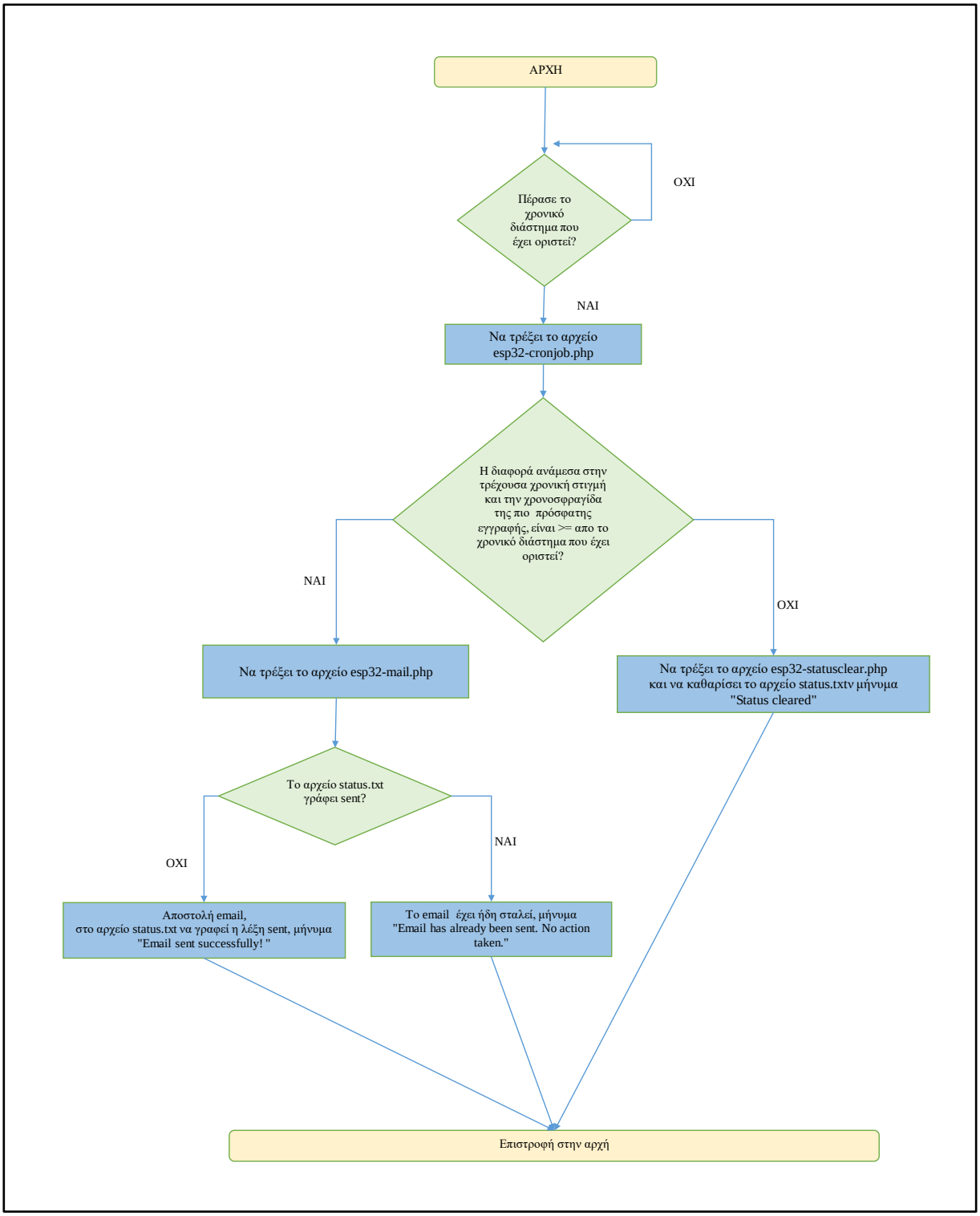
Η χρήση των αρχείων esp32-cronjob.php, esp32-mail.php και esp32-statusclear.php είναι στην λειτουργία αποστολής email στον administrator και περιγράφεται αναλυτικά στην παράγραφο 6.4. Τέλος ένα χρήσιμο αρχείο που υπάρχει στον server είναι το αρχείο my_cron_log.log. Υπάρχει η κατάλληλη ρύθμιση στον server που καθορίζει ότι η έξοδος (output) από την εκτέλεση του esp32-cronjob.php, αλλά και των άλλων δύο αρχείων που καλούνται μέσα από αυτό δηλαδή του esp32-mail.php και του esp32-statusclear.php, θα κατευθύνεται σε ένα αρχείο καταγραφής (log file). Σε αυτήν την περίπτωση, το αρχείο καταγραφής είναι το my_cron_log.log. Επίσης οποιοδήποτε σφάλμα (error) από την εκτέλεση του esp32-cronjob.php θα προωθείται στο ίδιο αρχείο καταγραφής με την έξοδο δηλαδή πάλι στο my_cron_log.log

## 6.4 Αποστολή email

Παράλληλα με την εμφάνιση του μηνύματος ότι για κάποιο λόγο έχει σταματήσει η αποστολή μετρήσεων στον server, ο administrator ενημερώνεται και με την αποστολή ενός email μέσω του αρχείου esp32-mail.php.

### 6.4.1 Διάγραμμα Ροής (Flow Chart) Αποστολής email στον Administrator

Παρακάτω στο Σχήμα 6.10, δίνεται το διάγραμμα ροής για την αποστολή email στον administrator.




Σχήμα 6.10: Διάγραμμα Ροής Αποστολής email στον Administrator

### 6.4.2 Εξήγηση Διαγράμματος Ροής

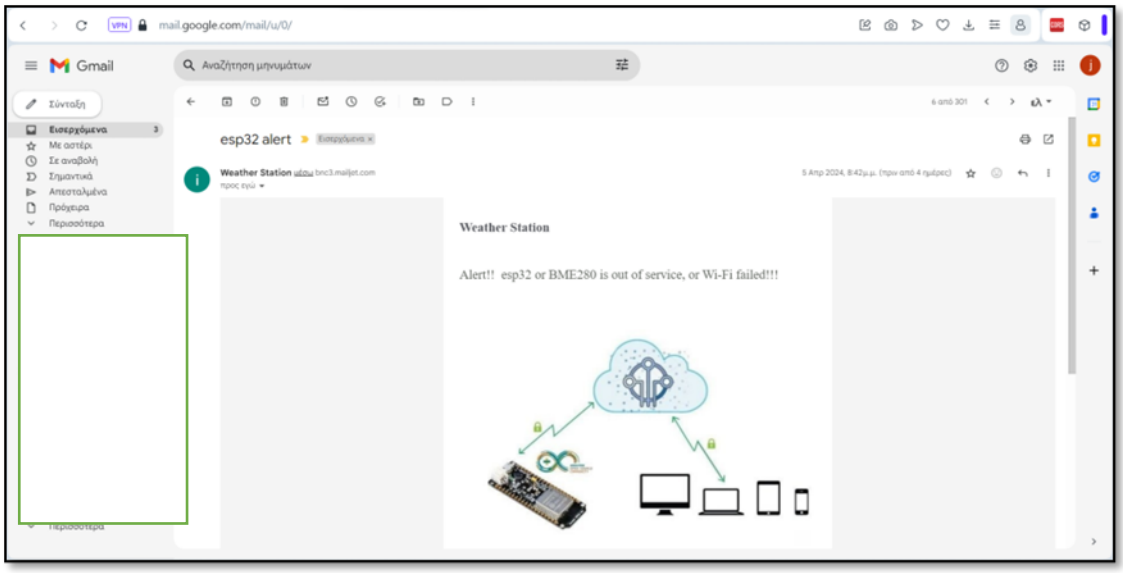
Για την αποστολή του email στον administrator, ακολουθείται η εξής διαδικασία:

Το αρχείο esp32-cronjob.php τρέχει σε τακτά χρονικά διαστήματα που έχει ορίσει ο administrtator

**Σενάριο 1ο**

Η διαφορά της τρέχουσας χρονικής στιγμής από την χρονοσφραγίδα της πιο πρόσφατης εγγραφής στον πίνακα sensor, είναι >= από το χρονικό διάστημα που έχει ορίσει ο administrator. Αυτό σημαίνει, ότι δεν έχουν αποθηκευτεί καινούριες μετρήσεις στη MySql δηλαδή για κάποιο λόγο έχουμε διακοπή λειτουργίας του ESP32_WS. Καλείται το αρχείο esp32-mail.php.

Αυτό το αρχείο είναι υπεύθυνο για την αποστολή του email. Το esp32-mail.php πριν στείλει email διαβάζει το εξωτερικό αρχείο status.txt και αν δεν διαβάσει τίποτα στέλνει το email όπως φαίνεται στην Εικόνα 6.11 και γράφει στο status.txt τη λέξη sent, έτσι ώστε να μην ξαναστείλει email για την ίδια διακοπή λειτουργίας του ESP32_WS. Ταυτόχρονα στέλνει το αντίστοιχο μήνυμα «Email sent successfully!» στο αρχείο καταγραφής my_cron_log.log, ώστε ο administrator να μπορεί να δει τις ενέργειες που έχουν γίνει, όπως φαίνεται στην Εικόνα 6.12




Εικόνα 6.11: email στον Administrator σε περίπτωση Βλάβης Συστήματος

Αν τώρα, δεν έχει αποκατασταθεί η διακοπή λειτουργίας του ESP32_WS και κληθεί ξανά το αρχείο esp32-mail.php, γιατί έχει περάσει το προκαθορισμένο χρονικό διάστημα, τότε στο αρχείο status.txt θα διαβάσει τη λέξη sent, που σημαίνει ότι ήδη έχει στείλει email, δεν θα εκτελέσει κάποια ενέργεια και θα γράψει στο αρχείο καταγραφής my_cron_log.log το αντίστοιχο μήνυμα «Email has already been sent» ώστε ο administrator να μπορεί να δει τις ενέργειες που έχουν γίνει.

Παρακάτω στην Εικόνα 6.11 φαίνεται τμήμα του αρχείου my_cron_log.log, όπου διακρίνουμε τα μηνύματα που έχει καταγράψει κάθε φορά που καλείται το αρχείο esp32-mail.php.

```
Status clearedCron job executed at 2024-05-02 19:17:01
Email sent successfully!Cron job executed at 2024-05-02 19:18:01
Email has already been sent. No action taken.Cron job executed at 2024-05-02 19:19:01
Email has already been sent. No action taken.Cron job executed at 2024-05-02 19:20:01
Email has already been sent. No action taken.Cron job executed at 2024-05-02 19:21:01
Email has already been sent. No action taken.Cron job executed at 2024-05-02 19:22:01
```

Εικόνα 6.12: Μηνύματα από την εκτέλεση του αρχείου esp32-mail.php.

**Σενάριο 2ο**

Η διαφορά της τρέχουσας χρονικής στιγμής από την χρονοσφραγίδα της πιο πρόσφατης εγγραφής στον πίνακα sensor, είναι < από το χρονικό διάστημα που έχει ορίσει ο administrator. Αυτό σημαίνει, ότι αποθηκεύονται κανονικά καινούριες μετρήσεις στην MySql, δηλαδή ο ESP32_WS λειτουργεί

κανονικά. Καλείται το αρχείο esp32-statusclear.php. Αυτό το αρχείο καθαρίζει το αρχείο status.txt, ώστε να σταλεί εκ νέου email στον administrator στην επόμενη διακοπή και στέλνει το αντίστοιχο μήνυμα «Status cleared» στο αρχείο καταγραφής my_cron_log.log, ώστε ο administrator να μπορεί να δει τις ενέργειες που έχουν γίνει.

Παρακάτω στην Εικόνα 6.13 φαίνεται τμήμα του αρχείου my_cron_log.log, όπου διακρίνουμε τα μηνύματα που έχει καταγράψει κάθε φορά που καλείται το αρχείο esp32-statusclear.php.


Εικόνα 6.13: Μηνύματα από την εκτέλεση του αρχείου esp32-statusclear.php

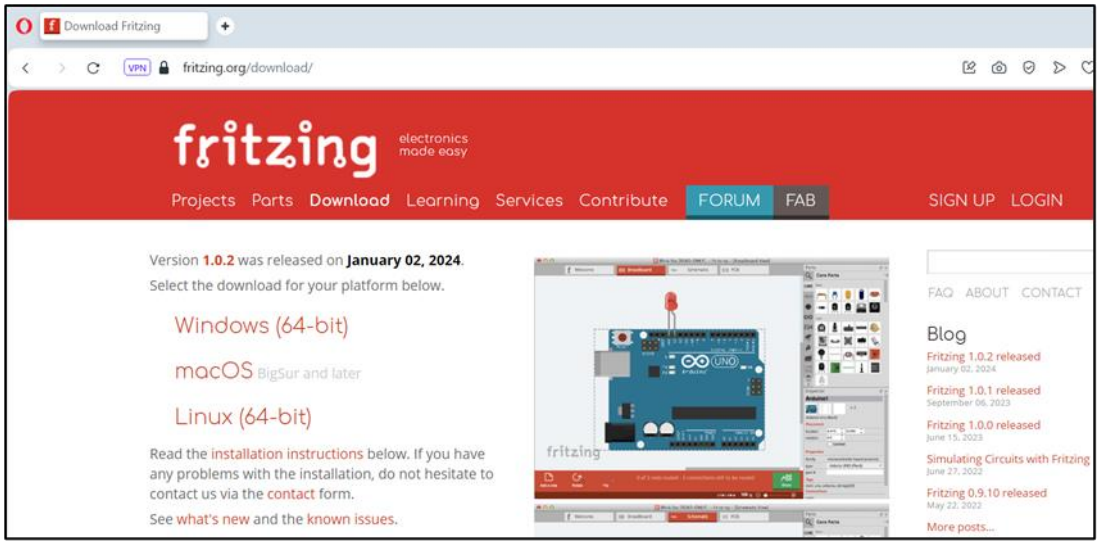
# Κεφάλαιο 7ο: Ηλεκτρονικό Κύκλωμα – Κατασκευή ESP32_WS

## 7.1 Εισαγωγή

Στο κεφάλαιο αυτό θα δούμε το σχεδιασμό του ηλεκτρονικού κυκλώματος αλλά και την κατασκευή του ESP32_WS.

## 7.2 Το fritzing

Για τον σχεδιασμό του ηλεκτρονικού κυκλώματος χρησιμοποιήθηκε το πρόγραμμα fritzing η εγκατάσταση του οποίου μπορεί να γίνει από την ιστοσελίδα https://fritzing.org/download/ όπως φαίνεται και παρακάτω στην Εικόνα 7.1. Για την εκπόνηση της παρούσης εργασίας χρησιμοποιήθηκε η έκδοση 1.0.2 η οποία κυκλοφόρησε στις 2 Ιανουαρίου 2024. Το κόστος αγοράς του προγράμματος ανέρχεται στα 8€.




Εικόνα 7.1: Εγκατάσταση fritzing

Το Fritzing είναι ένα δημοφιλές πρόγραμμα ανοιχτού κώδικα που χρησιμοποιείται κυρίως για τον σχεδιασμό ηλεκτρονικών πλακετών (PCBs) και τη δημιουργία σχηματικών διαγραμμάτων. Έχει σχεδιαστεί για να είναι εύχρηστο και προσιτό σε αρχάριους, καθώς παρέχει μια φιλική γραφική διεπαφή και εργαλεία που είναι εύκολο να κατανοηθούν.

Οι βασικές δυνατότητες του Fritzing περιλαμβάνουν:

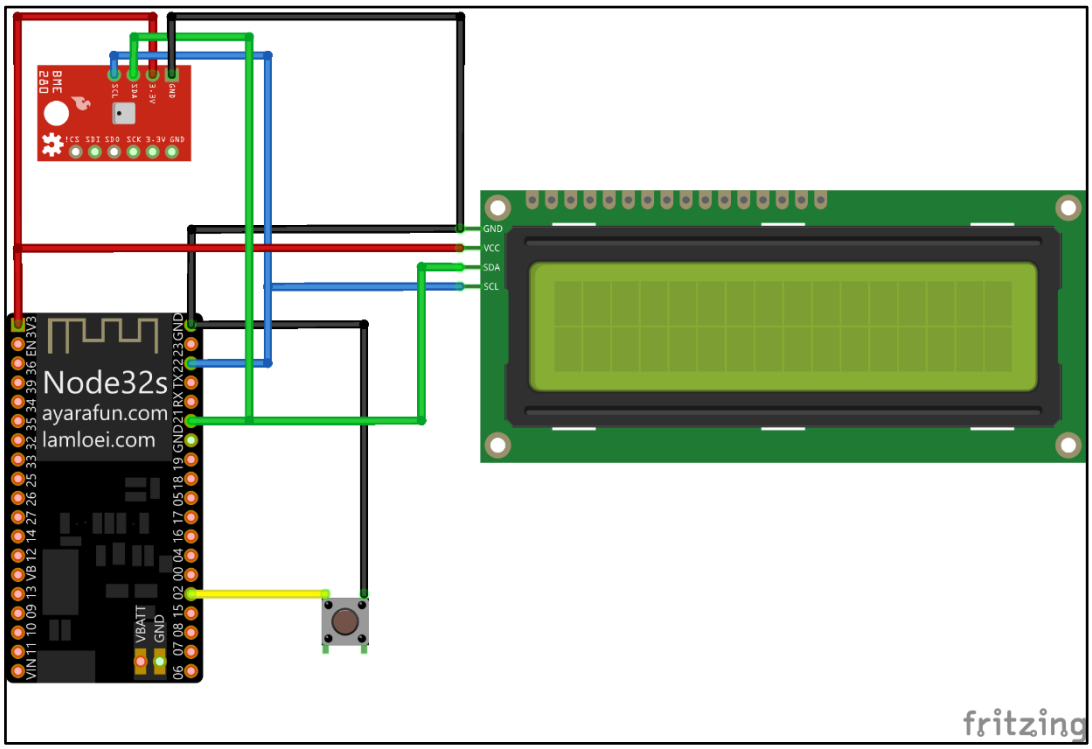
- **Σχεδίαση Πλακέτας Κυκλώματος (PCB Design)**: Το Fritzing επιτρέπει τη δημιουργία και τον σχεδιασμό PCBs με εύκολο και ευέλικτο τρόπο. Οι χρήστες μπορούν να τοποθετήσουν εξαρτήματα και να συνδέσουν ακροδέκτες.
- **Δημιουργία Σχηματικών Διαγραμμάτων:** Το Fritzing παρέχει εργαλεία για τη δημιουργία σχηματικών διαγραμμάτων, τα οποία μπορούν να χρησιμοποιηθούν για την αναπαράσταση των ηλεκτρονικών συνδέσεων και συστατικών του κυκλώματος.
- **Κοινότητα και Ανταλλαγή:** Το Fritzing διαθέτει μια ενεργή κοινότητα χρηστών που μπορούν να ανταλλάξουν ιδέες, πρότυπα και πληροφορίες. Υπάρχει επίσης η δυνατότητα να κατεβάσετε έτοιμα σχήματα και PCBs από την κοινότητα.

- **Δυνατότητα Εξατομίκευσης:** Το Fritzing επιτρέπει την προσθήκη νέων εξαρτημάτων και την προσαρμογή του χρήστη προς τις ανάγκες του.

Συνολικά, το Fritzing είναι ένα ευέλικτο και εύχρηστο εργαλείο για τους ηλεκτρονικούς και τους ερασιτέχνες που επιθυμούν να σχεδιάσουν PCBs και να δημιουργήσουν σχηματικά διαγράμματα [31].

## 7.3 Το Ηλεκτρονικό Κύκλωμα

Χρησιμοποιώντας λοιπόν το fritzing δημιουργούμε το Ηλεκτρονικό Κύκλωμα του ESP32_WS το οποίο φαίνεται παρακάτω στην Εικόνα 7.2




Σχέδιο 7.2: Το Ηλεκτρονικό Κύκλωμα του ESP32_WS

Αναλύοντας το Ηλεκτρονικό Κύκλωμα μπορούμε να σημειώσουμε τα εξής:

- Στις ακίδες 3.3V και GND του ESP32 θα συνδέσουμε την τροφοδοσία -VCC- και την γείωση -GND- αντίστοιχα του BME280 και της LCD 16x2 RGB. Στο σχέδιο σημειώνονται με κόκκινη γραμμή η τροφοδοσία και με μαύρη γραμμή η γείωση.
- Η ακίδα SCL, τόσο του BME280, όσο και της LCD 16x2 RGB συνδέονται στην ακίδα 22 του ESP32. Η ακίδα GPIO22 -στην Εικόνα 5.3 σημειώνεται και ως WIRE_SCL- χρησιμοποιείται για τη γραμμή ρολογιού SCL του πρωτοκόλλου I2C. Με αυτήν τη σύνδεση, ο ESP32 θα μπορεί να επικοινωνεί με τις συσκευές μέσω του πρωτοκόλλου I2C, επιτρέποντας τη μεταφορά δεδομένων μεταξύ τους. Η σύνδεση στο σχέδιο σημειώνεται με την μπλε γραμμή.
- Η ακίδα SDA, τόσο του BME280, όσο και της LCD 16x2 RGB συνδέονται στην ακίδα 21 του ESP32. Η ακίδα GPIO21 -στην Εικόνα 5.3 σημειώνεται και ως WIRE_SDA- χρησιμοποιείται για τη γραμμή δεδομένων SDA του πρωτοκόλλου I2C. Με τη σύνδεση αυτή, ο ESP32 μπορεί να επικοινωνεί με τη συσκευή μέσω του πρωτοκόλλου I2C, ανταλλάσσοντας δεδομένα και ελέγχοντας τη συσκευή. Η σύνδεση στο σχέδιο σημειώνεται με την πράσινη γραμμή.

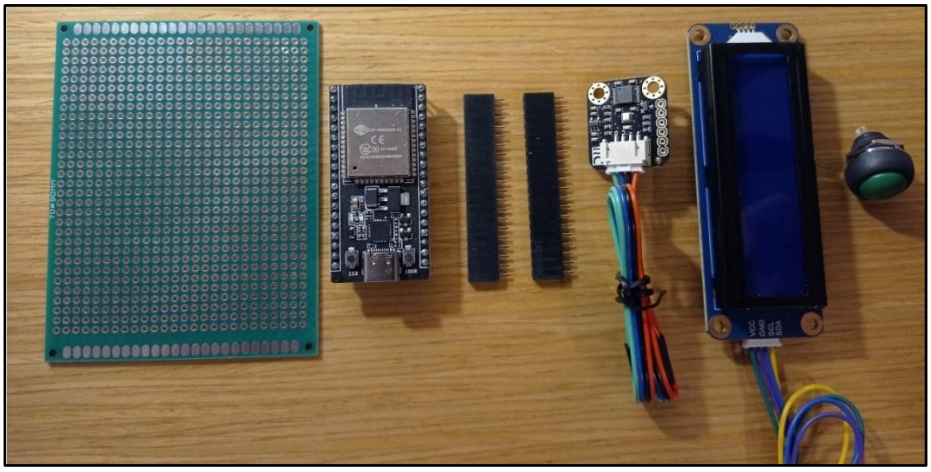
- Το push-button συνδέεται στην ακίδα 2 του ESP32 - γιατί έτσι έχει οριστεί στο πρόγραμμα του ESP32 όπως φαίνεται και στην Εικόνα 7.3 - και στην GND -γείωση-. Η σύνδεση στο σχέδιο σημειώνεται με την κίτρινη γραμμή.

```
const int buttonPin = 2;  // Η ακίδα που είναι συνδεδεμένο το κουμπί
volatile bool buttonPressed = false; // Μεταβλητή για τον έλεγχο του πατήματος του κουμπιού
int displayMode = 0; // Μετρητής για το τι να εμφανίζει η οθόνη
```

Εικόνα 7.3: Δήλωση μεταβλητής buttonPin για την σύνδεση του ESP32 με το κουμπί.

## 7.4 Υλικά Κατασκευής - Κατασκευή

Τα υλικά που χρησιμοποιήθηκαν για την κατασκευή του ESP32_WS φαίνονται παρακάτω στην Εικόνα 7.4

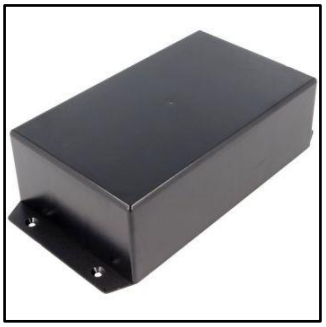


Εικόνα 7.4: Υλικά κατασκευής ESP32_WS

Από αριστερά και προς τα δεξιά έχουμε:

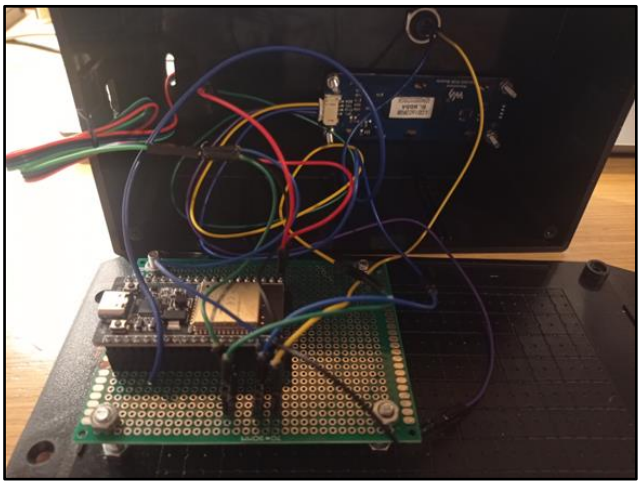
- Πλακέτα Διάτρητη 70x90mm
- ESP32 Development Board - NodeMCU-32S
- Pin Headers 1x20 Female 2.54mm
- Gravity Αισθητήρας Περιβάλλοντος I2C - BME280
- Basic 16x2 Character LCD - RGB 3.3V/5V (I2C Protocol)
- Push Button Momentary - 12mm Green

Για την τοποθέτηση των εξαρτημάτων θα χρησιμοποιηθεί το Κουτί Κατασκευής 197.4x113x63mm Μαύρο - G1025BF που φαίνεται στην Εικόνα 7.5

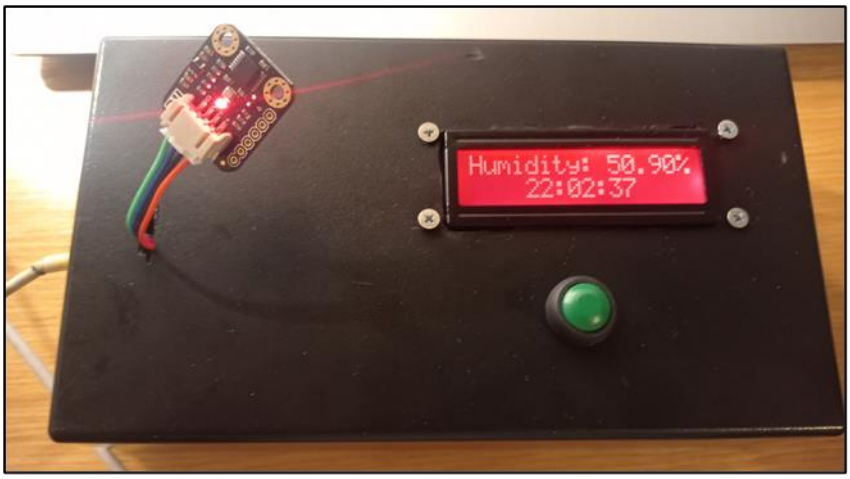


Εικόνα 7.5 Κουτί Κατασκευής

Το συνολικό κόστος της κατασκευής, μαζί με την αγορά της άδειας του fritzing υπολογίζεται περίπου στα 70 €. Παρακάτω στις Εικόνες 7.6 και 7.7 φαίνεται ο ESP32_WS. Το σύνολο των φωτογραφιών από την διαδικασία κατασκευής του ESP32_WS βρίσκεται στο Παράρτημα Β.



Εικόνα 7.6: Το εσωτερικό του ESP32_WS




Εικόνα 7.7: Ο ESP32_WS

# Κεφάλαιο 8ο: Προτάσεις για Τροποποιήσεις - Συμπεράσματα

## 8.1 Προτάσεις για Τροποποιήσεις

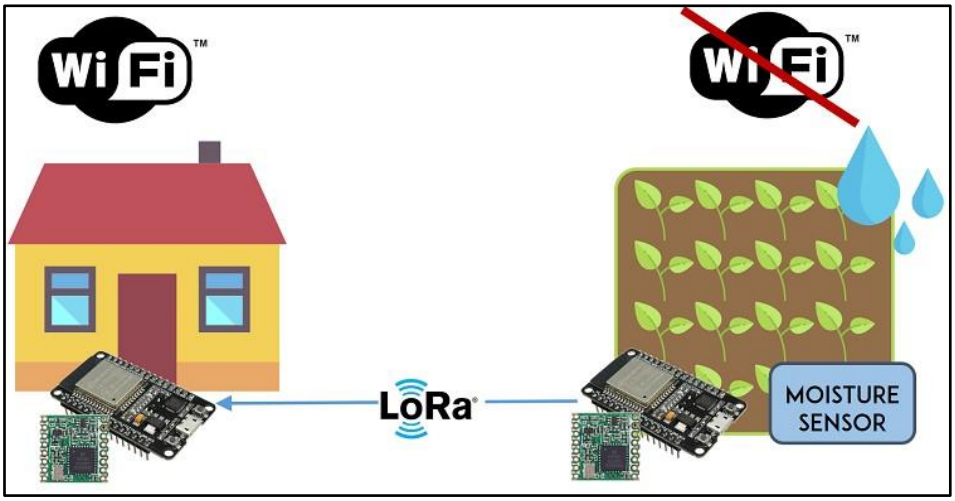
Σίγουρα ο ESP32_WS είναι μια εργασία που μπορεί να δεχτεί πολλές τροποποιήσεις ανάλογα με τη λειτουργικότητα που θέλουμε να προσθέσουμε ή να αφαιρέσουμε. Παρακάτω αναφέρονται κάποια παραδείγματα.

- **Προσθήκη Περισσότερων Αισθητήρων:** Μπορούμε να προσθέσουμε περισσότερους αισθητήρες για να μετρήσουμε περισσότερα δεδομένα, όπως η υγρασία του εδάφους, η ακτινοβολία UV, η πίεση του αέρα κ.λπ. Ανάλογα με τον αισθητήρα που θα επιλέξουμε, η ενσωμάτωση του στον κώδικα μπορεί να είναι αρκετά εύκολη, ειδικά αν χρησιμοποιηθούν βιβλιοθήκες που υποστηρίζουν τον αισθητήρα. Παράλληλα, με τη χρήση του πρωτοκόλλου I2C είναι αρκετά εύκολη και η σύνδεση του υλικού, αφού όπως αναφέρουμε και στην Παράγραφο 4.4 χρειάζονται μόνο τέσσερις (4) ακροδέκτες -VDD, GND, SCL, SDA-
- **Προσθήκη Κάμερας:** Μια αρκετά ενδιαφέρουσα τροποποίηση είναι η προσθήκη κάμερας για να λαμβάνουμε φωτογραφίες ή βίντεο από το περιβάλλον του ESP32_WS. Αυτό μπορεί να προσφέρει επιπλέον πληροφορίες για τις καιρικές συνθήκες ή ακόμα και να μας επιτρέψει να παρακολουθούμε τις συνθήκες σε πραγματικό χρόνο.
- **Αποθήκευση Ενέργειας:** Αν ο ESP32_WS βρίσκεται σε απομακρυσμένη περιοχή και δεν έχει πρόσβαση σε συνεχές ρεύμα, μπορούμε να χρησιμοποιήσουμε φωτοβολταϊκά πάνελ για να τροφοδοτήσουμε το ESP32 σε συνδυασμό με μπαταρίες ή Power Bank. Μπορούμε να χρησιμοποιήσουμε τεχνικές όπως η λειτουργία ύπνου -sleep mode- που αναφέρεται στην παράγραφο 5.3 για να εξοικονομήσουμε ενέργεια και να επεκτείνουμε τη διάρκεια ζωής της μπαταρίας. Επίσης και αφού μιλάμε για λειτουργία σε απομακρυσμένη περιοχή, μπορούμε να αφαιρέσουμε την LCD 16x2 RGB.
- **Επέκταση Τηλεπικοινωνιών:** Αν ο ESP32_WS βρίσκεται σε μια περιοχή με περιορισμένη σύνδεση στο διαδίκτυο, μπορούμε να εξετάσουμε τη χρήση τεχνολογιών όπως τα δίκτυα LoRa για ασύρματη επικοινωνία με άλλους σταθμούς ή σταθμούς βάσης. Με αυτόν τον τρόπο, μπορεί να γίνει μεταφορά τα δεδομένων σε μια πιο κεντρική τοποθεσία για μεταφορά ανάλυση ή αποθήκευση.

  Το πρωτόκολλο LoRa (Long Range) είναι ένα ασύρματο πρωτόκολλο επικοινωνίας που επιτρέπει τη μετάδοση δεδομένων μεγάλων αποστάσεων με χαμηλή κατανάλωση ενέργειας. Αυτό το πρωτόκολλο είναι ιδιαίτερα χρήσιμο για εφαρμογές που απαιτούν μετάδοση δεδομένων σε μεγάλες αποστάσεις, όπως αστική ή αγροτική ασύρματη παρακολούθηση ή αισθητήρες που βρίσκονται σε απομακρυσμένες περιοχές.

  Ο ESP32 είναι συμβατός με το πρωτόκολλο LoRa, αλλά για να χρησιμοποιηθεί η τεχνολογία LoRa με το ESP32, θα πρέπει να χρησιμοποιηθεί ένα επιπλέον πρόσθετο μονάδας LoRa, όπως για παράδειγμα το LoRa module SX1278. Αυτό το module μπορεί να συνδεθεί στο ESP32 μέσω διεπαφών όπως η SPI.

  Παρακάτω στην Εικόνα 8.1 φαίνεται η σύνδεση μεταξύ δύο ESP32 με LORA




Εικόνα 8.1: Σύνδεση μεταξύ δύο ESP32 με LORA [39]

Ο ESP32 στα δεξιά υποστηρίζει έναν ESP32_WS, στέλνει με LORA τις μετρήσεις στον ESP32 στα αριστερά, και από εκεί με Wi-Fi μπορούν να σταλούν για αποθήκευση στη βάση δεδομένων.

## 8.2 Συμπεράσματα

Συμπερασματικά η σχεδίαση, ο προγραμματισμός αλλά και η κατασκευή μιας IoT εφαρμογής, είναι μια διαδικασία που απαιτεί χρόνο, κόπο, συνδυασμό γνώσεων αλλά και κάποιο -συνήθως μικρό- κόστος. Μπορεί να χρειαστεί η επίλυση πλήθους προβλημάτων όπως τα παρακάτω:

- **Επιλογή Κατάλληλων Αισθητήρων και Μικροελεγκτών:** Η επιλογή των κατάλληλων αισθητήρων και μικροελεγκτών για τη συγκεκριμένη εφαρμογή είναι κρίσιμη και απαιτεί γνώσεις για τις λειτουργίες και τις δυνατότητες κάθε συσκευής.
- **Προγραμματισμός Συσκευών:** Ο προγραμματισμός των μικροελεγκτών για την ανάγνωση των αισθητήρων, την επεξεργασία των δεδομένων και την επικοινωνία με άλλες συσκευές ή τον ιστό απαιτεί γνώσεις σε γλώσσες προγραμματισμού και πλατφόρμες ανάπτυξης. Για παράδειγμα αναφέρουμε ότι στην παρούσα εργασία χρειάζονται γνώσεις από Wiring C (C/C++) για τον ESP32, HTML-Javascript για το Frontend, PHP για το Backend και SQL για την Βάση Δεδομένων.
- **Ασφάλεια:** Η διασφάλιση της ασφάλειας των συσκευών και των δεδομένων είναι ένα σημαντικό θέμα που απαιτεί σχεδιασμό και υλοποίηση κατάλληλων μέτρων ασφαλείας, ειδικά σε εφαρμογές με χρηματικές συναλλαγές ή προσωπικά δεδομένα ασθενών.
- **Διαχείριση Ενέργειας:** Η διαχείριση της ενέργειας είναι σημαντική, ιδίως σε ασύρματες συσκευές που λειτουργούν με μπαταρίες ή ενέργεια από φωτοβολταϊκά πάνελ.

Όμως, παρά την εμφάνιση πλήθους προβλημάτων, η εκπόνηση μιας IoT εφαρμογής από την αρχή μέχρι το τέλος, είναι και μια διαδικασία άκρως ενδιαφέρουσα και δημιουργική. Τα κέρδη μπορούν να είναι πολλαπλά και ποικίλα σε τομείς όπως:

- **Επαγγελματική Εξέλιξη:** Η ανάπτυξη μιας IoT εφαρμογής μπορεί να βελτιώσει τις δεξιότητες και την εμπειρία σε τομείς όπως ο προγραμματισμός, η ηλεκτρονική και η διαχείριση έργων, προσφέροντας νέες ευκαιρίες στην αγορά εργασίας.

- **Ηθική Ικανοποίηση:** Η δημιουργία μιας IoT εφαρμογής μπορεί να προσφέρει ικανοποίηση και επιβράβευση λόγω της δημιουργικής διαδικασίας και του τελικού προϊόντος.
- **Κοινωνικό Αντίκτυπο:** Μια επιτυχημένη IoT εφαρμογή μπορεί να έχει θετικό κοινωνικό αντίκτυπο, βοηθώντας στην επίλυση προβλημάτων και τη βελτίωση της ποιότητας ζωής.

Τέλος, είναι σημαντικό να τονίσουμε ότι η επιτυχία δεν είναι εγγυημένη και ότι η διαδικασία αυτή απαιτεί συνήθως σκληρή δουλειά, αφοσίωση και συνεχή μάθηση.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


[1] https://physicsopenlab.org/2020/05/10/datalogging-with-esp32-module/

[2] grobotronics.com

[3] https://webuilddatabases.com/mysql-design/

[4] https://logos-world.net/wifi-logo/

[5] https://www.freepik.com/icon/wifi-router_2876882

[6]https://www.kindpng.com/imgv/TmTwRiw_co2-temperature-relative-humidity-atmospheric-pressure-circle-hd/

[7] Mirjana Maksimović, Marijana Ćosović, "Preservation of Cultural Heritage Sites using IoT", 18th International Symposium INFOTEH-JAHORINA, 20-22 March 2019.

[8] https://www.waveshare.com/NodeMCU-32S.htm

[9] https://randomnerdtutorials.com/visualize-esp32-esp8266-sensor-readings-from-anywhere/

[10] Alexander Maier, Andrew Sharp, Yuriy Vagapov, "Comparative Analysis and Practical Implementation of the ESP32 Microcontroller Module for the Internet of Things"

[11] https://pdf1.alldatasheet.com/datasheet-pdf/download/1148023/ESPRESSIF/ESP32.html

[12] R. Niranjana, Arvind S, Vignesh M, Vishaal S, "Effectual Home Automation using ESP32 NodeMCU"

[13] Francesco Colace, Angelo Lorusso, Cristina Elia, Francesco Marongiu, Caterina Gabriella Guida, Domenico Santaniello, "An IoT-based Framework to Protect Cultural Heritage Buildings", 2021 IEEE International Conference on Smart Computing (SMARTCOMP).

[14] Husam Kareem, Dmitriy Dunaev, "The Working Principles of ESP32 and Analytical Comparison of using Low-Cost Microcontroller Modules in Embedded Systems Design"

[15] https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

[16] https://docs.espressif.com/projects/esp-idf/en/v4.3.3/esp32/api-eference/system/sleep_modes.html

[17]https://wiki.dfrobot.com/Gravity__I2C_BME280_Environmental_Sensor__Temperature,_Humidity,_Barometer__SKU__SEN0236

[18] https://www.waveshare.com/wiki/LCD1602_RGB_Module

[19] www.ti.com/lit/an/sbaa565/sbaa565.pdf

[20] https://www.php.net/manual/en/intro-whatis.php

[21] Pengxiang Gao, Qi Chen ,Xin Xie, Chen Wang "Research on Performance Optimizationof MySQL Database"

[22] https://www.arduino.cc/en/software

[23] Francesco Colace, Angelo Lorusso, Cristina Elia, Francesco Marongiu, Caterina Gabriella Guida, Domenico Santaniello, "An IoT-based Framework to Protect Cultural Heritage Buildings", 2021 IEEE International Conference on Smart Computing (SMARTCOMP).

[24] https://www.researchgate.net/figure/IoT-based-Smart-Environments_fig1_309242296

[25] Md. Milon Islam , Sheikh Nooruddin , Fakhri Karray and Ghulam Muhammad " Internet of Things: Device Capabilities, Architectures, Protocols, and Smart Applications in Healthcare Domain"

[26] Sudeep Pasricha "Embedded Systems Education: Experiences With Application-Driven Pedagogy"

[27] https://www.pantechelearning.com/embedded-system/

[28] https://microcontrollerslab.com/embedded-systems-basics/

[29]https://embeddedhardwaredesign.com/20-characteristics-of-embedded-system/#Real-Time_and_Deterministic_Operation

[30] https://www.prepbytes.com/blog/general/embedded-systems/

[31] https://fritzing.org

[32] https://www.embien.com/blog/embedded-system-design-architecture

[33] https://www.elprocus.com/basics-of-embedded-system-and-applications/

[34] Παναγιώτης Παπάζογλου και Σπύρος – Πολυχρόνης Λιωνής, *Ανάπτυξη Εφαρμογών με το ARDUINO*, Εκδόσεις Τζιόλα 2021

[35] Ninigi DATA SHEET

[36] code.visualstudio.com

[37] https://bigblue.academy/gr/ti-einai-to-github

[38] https://www.aresmush.com/tutorials/code/git.html

[39] https://randomnerdtutorials.com/esp32-lora-rfm95-transceiver-arduino-ide/

# ΠΑΡΑΡΤΗΜΑ Α : Κώδικας

- Wiring C

Αρχείο **esp32_wifi_postdata_LCD.ino**

Είναι αυτό που τρέχει στον ESP32

Στην εργασία παρουσιάζεται τμηματικά, για να μας βοηθήσει στην εξήγηση του Διαγράμματος ροής (Flow Chart) του Προγράμματος του ESP32.

```
//Εισαγωγή των απαραίτητων βιβλιοθηκών
#include <Waveshare_LCD1602_RGB.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <HTTPClient.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <WiFiManager.h>
#include <time.h>

String apiKeyValue = "**********"; // Κλειδί API για την αποστολή δεδομένων μέσω HTTP

Adafruit_BME280 bme; // Δήλωση του αισθητήρα
Waveshare_LCD1602_RGB lcd(16, 2); // Διεύθυνση I2C και διαστάσεις οθόνης

const int buttonPin = 2;  // Η ακίδα που είναι συνδεδεμένο το κουμπί
volatile bool buttonPressed = false; // Μεταβλητή για τον έλεγχο του πατήματος του κουμπιού
int displayMode = 0; // Μετρητής για το τι να εμφανίζει η οθόνη

const char* ntpServer = "pool.ntp.org";  // Ορισμός NTP server από τον οποίο θα λαμβάνεται η ώρα
const long  gmtOffset_sec = 7200;  // Ορισμός απόκλισης ώρας σε δευτερόλεπτα από το GMT, εδώ GMT+2 ώρες για Ανατολική Ευρωπαϊκή Ζώνη
const int   daylightOffset_sec = 3600;  // Ορισμός απόκλισης ώρας για τη θερινή ώρα, προσθέτοντας 3600 δευτερόλεπτα ή μία ώρα
```

```
//Δημιουργία πινάκων που χρησιμοποιούνται για να αποθηκεύσουν τις τιμές των μετρήσεων
//που παίρνονται από τον αισθητήρα BME280 σε μορφή κειμένου, αλλά και την ώρα
char bufferT[6];
char bufferH[6];
char bufferP[10];
char timeString[20];

//Μεταβλητές πάνω και κάτω ορίων για θερμοκρασία - υγρασία - πίεση
float TempH = 25;
float TempL=24.0;
float HumH=45;
float HumL=40;
float PresH=1016.7;
float PresL=1000;

void IRAM_ATTR changeDisplayMode() {
  buttonPressed = true; // Αλλαγή της κατάστασης όταν πατηθεί το κουμπί
}

void setup() {
  Serial.begin(921600);
  pinMode(buttonPin, INPUT_PULLUP); // Ορισμός της ακίδας του κουμπιού ως είσοδο με εσωτερικό pull-up resistor
  attachInterrupt(digitalPinToInterrupt(buttonPin), changeDisplayMode, FALLING); // Ρύθμιση διακοπής για την αλλαγή του display mode  με το πάτημα του κουμπιού
  lcd.init(); // Αρχικοποίηση της οθόνης LCD
  lcd.setRGB(255, 255, 255); // Ενεργοποίηση του φωτισμού σε χρώμα μπλε by default



  bool status = bme.begin(0x77); // Αρχικοποίηση του BME280 στη διεύθυνση (0x77)
  if (!status) {
  Serial.println("Could not find a valid BME280 sensor, check wiring!");
  lcd.clear();
```

```
  lcd.setCursor(0, 0);
  lcd.setRGB(255,255,255);
  lcd.send_string("BME280 ");
  lcd.setCursor(0, 1);
  lcd.send_string("   disconnected!");
  while (1);
  }

WiFi.begin();
  unsigned long startConnection = millis(); // Αποθήκευση της τρέχουσας χρονικής στιγμής

  // Περιμένουμε μέχρι να συνδεθούμε στο WiFi ή να περάσουν 30 δευτερόλεπτα
  while (WiFi.status() != WL_CONNECTED && millis() - startConnection < 30000) {
    delay(500);
    Serial.print(".");
    //lcd.clear();
    lcd.setCursor(0, 0);
    lcd.setRGB(255,255,255);
    lcd.send_string("connection...");
  }
   // Έλεγχος αν η σύνδεση απέτυχε
  if(WiFi.status() != WL_CONNECTED) {
    Serial.println("Απέτυχε η σύνδεση στο WiFi");
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.setRGB(255,255,255);
    lcd.send_string("connection ");
    lcd.setCursor(7, 1);
    lcd.send_string("failed");
    delay(3000);
    checkWiFiConnection(); // Καλούμε την checkWiFiConnection σε περίπτωση αποτυχίας
  } else {
    Serial.println("Επιτυχία Σύνδεσης");
```

```
    Serial.print("IP Address: ");
    Serial.println(WiFi.localIP()); // Εμφάνιση IP διεύθυνσης
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.setRGB(255,255,255);
    lcd.send_string("connection ");
    lcd.setCursor(7, 1);
    lcd.send_string("success!!!");
  }

  // Ρύθμιση NTP
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);

  // Αναμονή για τη λήψη ώρας
  // Αυτός ο βρόχος εκτελείται επαναληπτικά όσο η συνάρτηση time επιστρέφει null, δηλαδή όσο δεν υπάρχει διαθέσιμη ώρα.
  while (!time(nullptr)) {
    Serial.print(".");
    delay(1000);
  }
  Serial.println("Time received!");
}

void loop() {

static unsigned long lastDisplayUpdate = 0; // Τελευταία ενημέρωση οθόνης
static unsigned long lastDataSend = 0; // Τελευταία αποστολή δεδομένων
unsigned long currentMillis = millis(); // Τρέχον χρονικό σημείο

checkWiFiConnection();

if (buttonPressed) { //Ελεγχος εάν το κουμπί έχει πατηθεί
displayMode = (displayMode + 1) % 3; // Κυκλική αλλαγή μεταξύ των λειτουργιών εμφάνισης
```

```
buttonPressed = false; // Επαναφορά της σημαίας για το πάτημα του κουμπιού
delay(100);
displaySensorData(); // Ενημέρωση της οθόνης με τα νέα δεδομένα
}

if (WiFi.status() == WL_CONNECTED && currentMillis - lastDataSend >= 30000) {
   lastDataSend = currentMillis;
   if (!bme.begin(0x77)) {
   Serial.println("BME280 disconnected!");
   }else {
      sendSensorData(); //Αποστολή δεδομένων στον server
   }
}
if (currentMillis - lastDisplayUpdate >= 30000) {
   lastDisplayUpdate = currentMillis;
   if (!bme.begin(0x77)) {
   Serial.println("BME280 disconnected!");
   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.setRGB(255,255,255);
   lcd.send_string("BME280 ");
   lcd.setCursor(0, 1);
   lcd.send_string("   disconnected!");
   }else{
     displaySensorData();  //Εμφάνιση δεδομένων στην οθόνη
   }
   }
}

void checkWiFiConnection() {
  if (WiFi.status() != WL_CONNECTED) {
   lcd.clear();
   lcd.setCursor(2, 0);
```

```
    lcd.setRGB(255,255,255);
    lcd.send_string("ACCESS POINT");
    WiFiManager wifiManager;
    wifiManager.autoConnect("ESP32_AP"); //Δημιουργία access point
  }
}

void sendSensorData(){
  if (WiFi.status() == WL_CONNECTED) { // Αποστολή δεδομένων μόνο εάν το WiFi είναι συνδεδεμένο
  WiFiClientSecure client; // Δημιουργία ενός ασφαλούς WiFiClient
  client.setInsecure();  // Ρύθμιση του πελάτη να μην ελέγχει την εγκυρότητα
  HTTPClient https; //Δημιουργία αντικειμένου για την διαχείριση HTTP POST

  String serverPath = "https://users.iee.ihu.gr/~iee2019243/Diplomatica/esp32-postdata.php"; //Διεύθυνση URL του διακομιστή

  https.begin(client, serverPath); // Αρχίζει μια HTTPS σύνδεση
  https.addHeader("Content-Type", "application/x-www-form-urlencoded"); // Ορισμός του τύπου του περιεχομένου της αίτησης

  String httpRequestData = "api_key=" + apiKeyValue + "&value1=" + String(bme.readTemperature()) + "&value2=" + String(bme.readHumidity()) + "&value3=" + String(bme.readPressure() / 100.0F);
  int httpResponseCode = https.POST(httpRequestData); //Αποστολή των δεδομένων POST

  if (httpResponseCode > 0) {
    Serial.print(httpRequestData);
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
  } else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
```

```
  https.end();  // Λήξη της σύνδεσης με τον διακομιστή
}
}

//Συνάρτηση απεικόνισης τιμών στην οθόνη
void displaySensorData() {

float temperature = bme.readTemperature();
float humidity = bme.readHumidity();
float pressure = bme.readPressure() / 100.0F; // Μετατροπή σε hPa

// Λήψη της τρέχουσας ώρας
  struct tm timeinfo;
  if (!getLocalTime(&timeinfo)) {
    Serial.println("Failed to obtain time");
    return;  // Αποτυχία λήψης ώρας, διακοπή της λειτουργίας
  }

lcd.clear(); // Καθαρισμός της οθόνης πριν την εμφάνιση δεδομένων

switch (displayMode) {
case 0: // Θερμοκρασία
lcd.setCursor(0, 0);
if (temperature >= TempL && temperature <= TempH)
{
  lcd.setRGB(0,255,0);
}
  else if(temperature > TempH)
{
    lcd.setRGB(255,0,0);
}
else lcd.setRGB(255,255,255);
lcd.send_string("Temp: ");
```

```
dtostrf(temperature, 4, 2, bufferT);// Μορφοποίηση της θερμοκρασίας σε string
lcd.send_string(bufferT);
lcd.send_string((" "+String(char(223)) + "C").c_str());
strftime(timeString, sizeof(timeString), "%H:%M:%S", &timeinfo);
lcd.setCursor(4, 1);
lcd.send_string(timeString); // Εμφάνιση της ώρας
break;

case 1: // Υγρασία
lcd.setCursor(0, 0);
if (humidity >= HumL && humidity <= HumH)
{
  lcd.setRGB(0,255,0);
}
  else if(humidity > HumH)
{
    lcd.setRGB(255,0,0);
}
else lcd.setRGB(255,255,255);
lcd.send_string("Humidity: ");
dtostrf(humidity, 4, 2, bufferH);// Μορφοποίηση της υγρασίας σε string
lcd.send_string(bufferH);
lcd.send_string("%");
strftime(timeString, sizeof(timeString), "%H:%M:%S", &timeinfo);
lcd.setCursor(4, 1);
lcd.send_string(timeString); // Εμφάνιση της ώρας
break;

case 2: // Πίεση
lcd.setCursor(0, 0);
if (pressure >= PresL && pressure <= PresH)
{
  lcd.setRGB(0,255,0);
```

```
}
  else if(pressure > PresH)
{
    lcd.setRGB(255,0,0);
}
else lcd.setRGB(255,255,255);
lcd.send_string("Pres: ");
dtostrf(pressure, 6, 2, bufferP); // Μορφοποίηση της πίεσης σε string
lcd.send_string(bufferP);
lcd.send_string(" hPa");
strftime(timeString, sizeof(timeString), "%H:%M:%S", &timeinfo);
lcd.setCursor(4, 1);
lcd.send_string(timeString); // Εμφάνιση της ώρας
break;
}
}
```

- PHP

Αρχεία **esp32-postdata.php**, **esp32-visualize.php**, **esp32-cronjob.php**, **esp32-mail.php**, **esp32-statusclear.php**, όπως αναφέρονται και στην παράγραφο 3.3

Το αρχείο **esp32-postdata.php** δίνεται ως παράδειγμα στην παράγραφο 3.3

**esp32-visualize.php**

```
<?php
require_once 'auth.php';


$servername = "localhost";
$dbname = "esp32";
$username = $user;
$password = $pass;


$statusFile = 'status.txt';
```

```
$conn = new mysqli($servername, $username, $password, $dbname, null, '/home/student/iee/2019/iee2019243/mysql/run/mysql.sock');
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM (SELECT id, value1, value2, value3, reading_time FROM sensor ORDER BY reading_time DESC LIMIT 30) AS subquery ORDER BY reading_time ASC;";
$result = $conn->query($sql);
$sensor_data = [];

while ($data = $result->fetch_assoc()) {
    $sensor_data[] = $data;
}

echo json_encode($sensor_data);
$conn->close();
?>
```

**esp32-cronjob.php**

```
<?php
date_default_timezone_set('Europe/Athens');

$log = '/home/student/iee/2019/iee2019243/public_html/Diplomatica/my_cron_log.log';
file_put_contents($log, "Cron job executed at ".date("Y-m-d H:i:s")."\n", FILE_APPEND);

require_once 'auth.php';
$servername = "localhost";
$dbname = "esp32";
$username = $user;
$password = $pass;

 // Δημιουργία Σύνδεσης
```

```
$conn = new mysqli($servername, $username, $password, $dbname,null,'/home/student/iee/2019/iee2019243/mysql/run/mysql.sock');
// Έλεγχος Σύνδεσης
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Εκτέλεσε το query για να πάρεις την πιο πρόσφατη εγγραφή
$result = $conn->query("SELECT `reading_time` FROM `sensor` ORDER BY `reading_time` DESC LIMIT 1");

if ($result->num_rows > 0) {
   // Πάρε το reading_time
   $row = $result->fetch_assoc();
   $lastReadingTime = new DateTime($row['reading_time']);
   $currentTime = new DateTime();

   // Υπολόγισε τη διαφορά
   $interval = $currentTime->diff($lastReadingTime);
   $minutes = $interval->days * 24 * 60;
   $minutes += $interval->h * 60;
   $minutes += $interval->i;
   //echo "Minutes difference: " . $minutes;
   //echo date_default_timezone_get();

   if ($minutes >= 1 ) {
      // Καλέσε το esp32-mail.php αν η διαφορά είναι >= 1 λεπτό
      include("esp32-mail.php");
   } elseif ($minutes < 1) {
      // Καλέσε το statusClear.php αν η διαφορά είναι < 1 λεπτό
      include("esp32-statusclear.php");
   }
} else {
```

```
    echo "No records found.";
}

$conn->close();
?>
```

**esp32-mail.php**

```
<?php
require 'vendor/autoload.php';
require_once 'auth.php';
use \Mailjet\Resources;

$statusFile = '/home/student/iee/2019/iee2019243/public_html/Diplomatica/status.txt';
$emailSentStatus = file_get_contents($statusFile);

if ($emailSentStatus !== 'sent') {
$mj = new \Mailjet\Client($public_mail, $key_mail, true,['version' => 'v3.1']);

$body = [
    'Messages' => [
        [
            'From' => [
                'Email' => "iee2019243@iee.teithe.gr",
                'Name' => "Weather Station"
            ],
            'To' => [
                [
                    'Email' => "skoul37@gmail.com",
                    'Name' => "john"
                ]
            ],
            'TemplateID' => 5852786, // ID του template
```

```
            'TemplateLanguage' => true, // Χρήση των δυναμικών μεταβλητών του template
            'Subject' => "esp32 alert",
            //'TextPart' => "Hi",
            //'HTMLPart' => "<h3>esp32 or BME280 is out of service or wi-fi failed</h3>"
        ]
    ]
];

$response = $mj->post(Resources::$Email, ['body' => $body]);
if ($response->success()) {
    echo "Email sent successfully!";
    file_put_contents($statusFile, 'sent'); // Update the status
} else {
    echo "Failed to send email. Error: " . $response->getStatus();
}
}else {
    echo "Email has already been sent. No action taken.";
}
?>
```

**esp32-statusclear.php**

```
<?php
$statusFile = '/home/student/iee/2019/iee2019243/public_html/Diplomatica/status.txt';
$result = file_put_contents($statusFile, '');
if ($result === false) {
    echo "Error clearing status file.";
} else {
    echo "Status cleared";
}
?>
```

- HTML – Javascript

Αρχείο **esp32-visualize.html**

Είναι αυτό που εμφανίζει τα διαγράμματα των μεγεθών στην ιστοσελίδα https://users.iee.ihu.gr/~iee2019243/Diplomatica/esp32-visualize.html

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script src="https://code.highcharts.com/highcharts.js"></script>
  <style>
    body {
      min-width: 310px;
      max-width: 1280px;
      height: 500px;
      margin: 0 auto;
    }
    h2 {
      font-family: Arial;
      font-size: 2.5rem;
      text-align: center;
    }
  </style>
</head>
<body>
  <h2>Weather Station with ESP32 - BME280</h2>
  <div id="chart-temperature" class="container"></div>
  <div id="chart-humidity" class="container"></div>
  <div id="chart-pressure" class="container"></div>

  <script>
  //let isSensorDisconnected = false;
  //let isEmailSent = false;

  function fetchData() {
```

```
$.ajax({
  url: 'esp32-visualize.php',
  type: 'GET',
  dataType: 'json',
  success: function(data) {
    let readings_time = data.map(a => a.reading_time);
    let value1 = data.map(a => parseFloat(a.value1));
    let value2 = data.map(a => parseFloat(a.value2));
    let value3 = data.map(a => parseFloat(a.value3));

    if (readings_time.length > 0) {
      let lastReadingTime = new Date(readings_time[readings_time.length - 1]);
      let currentTime = new Date();
      let timeDifference = currentTime - lastReadingTime;

      if (timeDifference > 60000) {
        // Απόκρυψη των διαγραμμάτων και εμφάνιση μηνύματος
        document.getElementById("chart-temperature").style.display = 'none';
        document.getElementById("chart-humidity").style.display = 'none';
        document.getElementById("chart-pressure").style.display = 'none';
        displayCenterMessage("ESP32 or BME280 is out of service, or WiFi failed");

      } else {
        // Εμφάνιση των διαγραμμάτων και ενημέρωση
        clearCenterMessage();
        document.getElementById("chart-temperature").style.display = 'block';
        document.getElementById("chart-humidity").style.display = 'block';
        document.getElementById("chart-pressure").style.display = 'block';
        updateChart(chartT, value1, readings_time);
        updateChart(chartH, value2, readings_time);
        updateChart(chartP, value3, readings_time);
      }
    }
```

```
    },
    error: function(xhr, status, error) {
      console.error("Error fetching data: " + error);
    }
  });
}

function displayCenterMessage(message) {
  var messageDiv = document.getElementById("centerMessage");
  if (!messageDiv) {
    messageDiv = document.createElement('div');
    messageDiv.id = "centerMessage";
    document.body.appendChild(messageDiv);
  }
  messageDiv.innerHTML = message;
  messageDiv.style.cssText = 'position: fixed; top: 50%; left: 50%; transform: translate(-50%, -50%);
font-size: 2em; color: red; text-align: center;';
}

function clearCenterMessage() {
  var messageDiv = document.getElementById("centerMessage");
  if (messageDiv) {
    messageDiv.parentNode.removeChild(messageDiv);
  }
}

    function createChart(container, title, yAxisTitle, zonesConfig, name) {
      return Highcharts.chart(container, {
        chart: { type: 'line' },
        title: { text: title },
        series: [{
          name:name,
          data: [],
```

```
      zones: zonesConfig
    }],
    xAxis: {
      type: 'datetime'
      },
    yAxis: {
      title: { text: yAxisTitle }
    },
    credits: { enabled: false }
  });
}

function updateChart(chart, data, categories) {
  chart.series[0].setData(data);
  chart.xAxis[0].setCategories(categories);
}

let chartT = createChart('chart-temperature', 'BME280 Temperature', 'Temperature (Celsius)', [{
  value: 24,
  color: '#0000ff'
}, {
  value: 25,
  color: '#00ff00'
}, {
  color: '#ff0000'
}], 'Temperature');

let chartH = createChart('chart-humidity', 'BME280 Humidity', 'Humidity (%)', [{
  value: 40,
  color: '#0000ff'
}, {
  value: 45,
  color: '#00ff00'
```

```
    }, {
      color: '#ff0000'
    }], 'Humidity');

    let chartP = createChart('chart-pressure', 'BME280 Pressure', 'Pressure (hPa)', [{
      value: 1000,
      color: '#0000ff'
    }, {
      value: 1016.7,
      color: '#00ff00'
    }, {
      color: '#ff0000'
    }], 'Pressure');

    fetchData();
    setInterval(fetchData, 30000); // Ενημερώνει τα δεδομένα κάθε 30 δευτερόλόλεπτα
</script>

</body>
</html>
```

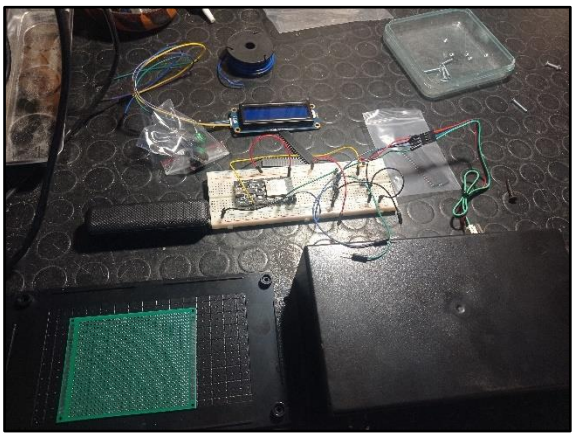
- MySql

Αρχείο **esp32_schema.sql**

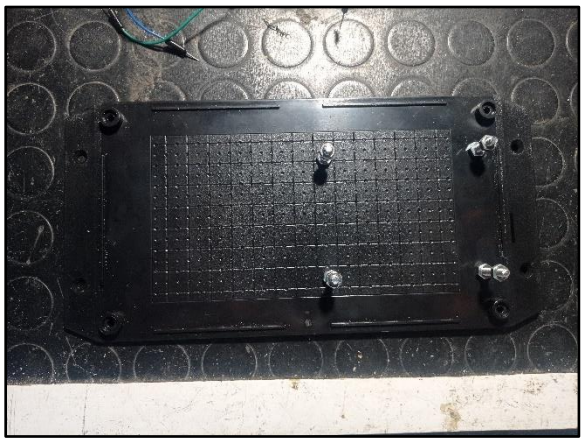
Δίνεται στην παράγραφο 3.3 ως παράδειγμα κώδικα για την δημιουργία της βάσης δεδομένων.

# ΠΑΡΑΡΤΗΜΑ Β : Φωτογραφίες Κατασκευής ESP32_WS

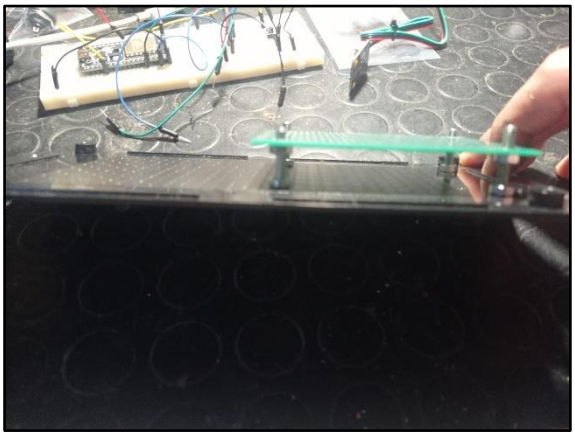
Οι φωτογραφίες του Παραρτήματος Β μας δείχνουν τα στάδια κατασκευής του ESP32_WS



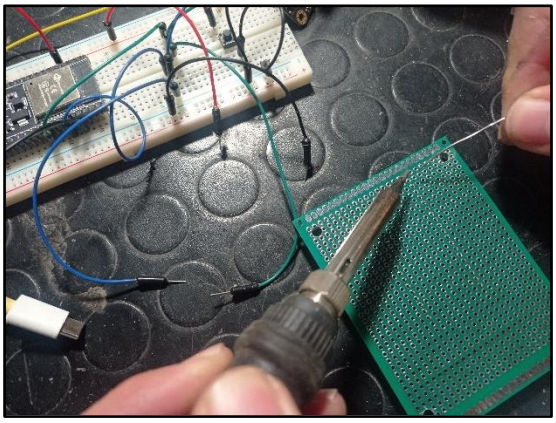
Εικόνα Β1: Συγκέντρωση Υλικών και Εργαλείων



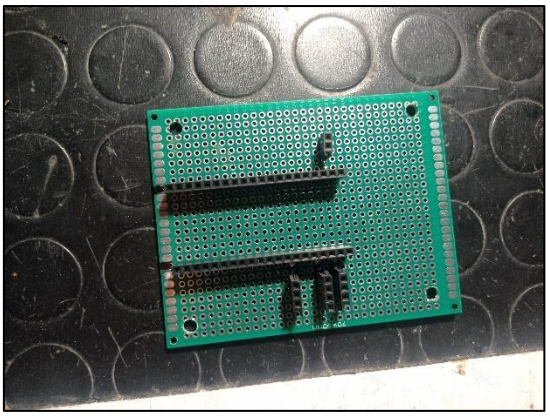
Εικόνα Β2: Τοποθέτηση στηριγμάτων Πλακέτας



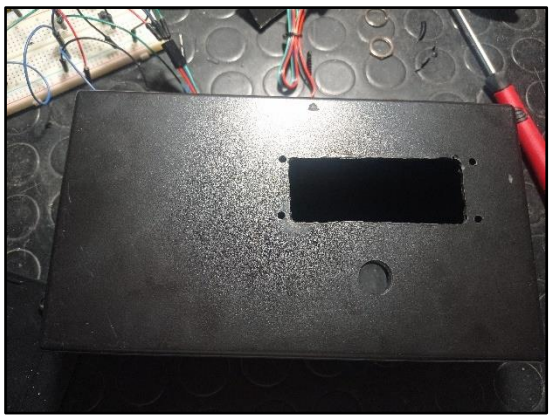
Εικόνα Β3: Τοποθέτηση πλακέτας



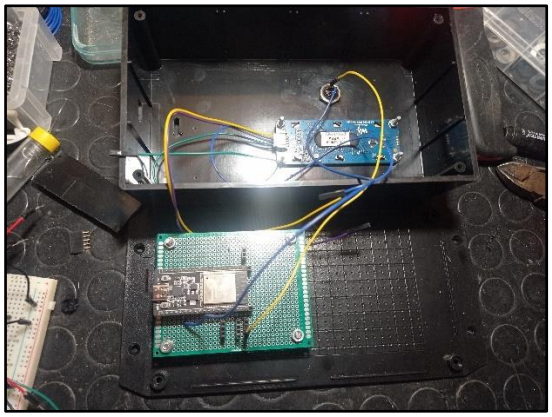
Εικόνα Β4: Συγκόλληση Βάσεων



Εικόνα Β5: Η πλακέτα με τις βάσεις

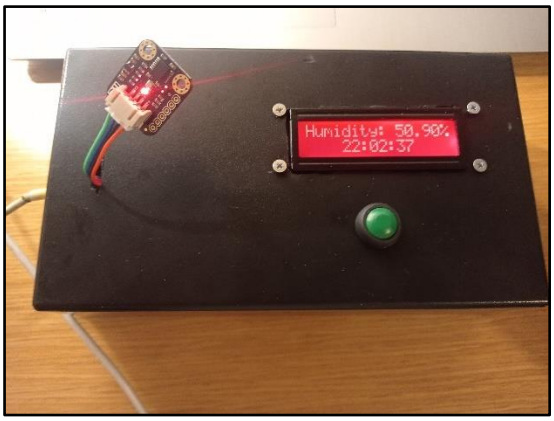


Εικόνα Β6: Τρύπημα κουτιού



Εικόνα Β7: Σύνδεση LCD 16x2 RGB

Εικόνα Β8: Σύνδεση push_button




Εικόνα Β9: Τελική Εμφάνιση και Λειτουργία