



Σχολή Θετικών Επιστημών και Τεχνολογίας

Τμήμα Πληροφορικής

Πτυχιακή Εργασία

Σχεδιασμός και ανάπτυξη «Έξυπνης Κολώνας» δημόσιου φωτισμού σε περιβάλλον μίας έξυπνης πόλης

Αριστοκλής Κυριακίδης

Επιβλέπων καθηγητής: Ευάγγελος Τοπάλης

Πάτρα, Ιούλιος 2024

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Κυριακίδη Αριστοκλή που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης ο συγγραφέας/δημιουργός εκχωρεί στο ΕΑΠ, μη αποκλειστική άδεια χρήσης του δικαιώματος αναπαραγωγής, προσαρμογής, δημόσιου δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσής τους διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος και για όλο το χρόνο διάρκειας των δικαιωμάτων πνευματικής ιδιοκτησίας. Η ανοικτή πρόσβαση στο πλήρες κείμενο για μελέτη και ανάγνωση δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, αποθήκευση, πώληση, εμπορική χρήση, μετάδοση, διανομή, έκδοση, εκτέλεση, «μεταφόρτωση» (downloading), «ανάρτηση» (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού. Ο συγγραφέας/δημιουργός διατηρεί το σύνολο των ηθικών και περιουσιακών του δικαιωμάτων.



Σχεδιασμός και ανάπτυξη «Έξυπνης Κολώνας» δημόσιου φωτισμού σε περιβάλλον μίας έξυπνης πόλης

Αριστοκλής Κυριακίδης

Επιτροπή Επίβλεψης Πτυχιακής Εργασίας

Επιβλέπων Καθηγητής:

Δρ. Ευάγγελος Τοπάλης

Συνεργαζόμενο Εκπαιδευτικό Προσωπικό
(ΣΕΠΙ)

Ελληνικό Ανοιχτό Πανεπιστήμιο (ΕΑΠ)

Συν-Επιβλέπων Καθηγητής:

Δρ. Βασίλειος Φωτόπουλος

Συνεργαζόμενο Εκπαιδευτικό Προσωπικό
(ΣΕΠΙ)

Ελληνικό Ανοιχτό Πανεπιστήμιο (ΕΑΠ)

Συν-Επιβλέπων Καθηγητής:

Δρ. Κωνσταντίνος Γιαννακόπουλος
Συνεργαζόμενο Εκπαιδευτικό Προσωπικό
(ΣΕΠΙ)

Ελληνικό Ανοιχτό Πανεπιστήμιο (ΕΑΠ)

Πάτρα, Ιούλιος 2024



*«Αφιερωμένο στη Χαρά, στο Νικόλα και στο Βασίλη για την άμετρη συμπαράσταση,
βοήθεια και κατανόηση που δείξατε σε όλη τη διάρκεια των σπουδών μου.
Σας ευχαριστώ που με στηρίζετε και πραγματοποιώ τα όνειρα μου.»*

Περίληψη

Η παρούσα πτυχιακή εργασία πραγματεύεται το σχεδιασμό και την ανάπτυξη ενός δικτύου από «έξυπνες κολώνες» δημόσιου φωτισμού σε περιβάλλον έξυπνης πόλης. Η υλοποίηση των συστημάτων βασίζεται στο συνδυασμό των δυνατοτήτων που προσφέρουν οι αναπτυξιακές πλατφόρμες Arduino, ESP32 και Raspberry Pi σε συνδυασμό με τη χρήση αισθητήρων. Η επικοινωνία μεταξύ αυτών βασίζεται στα πρότυπα ενσύρματων και ασύρματων τοπικών δικτύων, καθώς και της χρήσης πρωτοκόλλων διαδικτύου. Η διακίνηση των πληροφοριών (από τη συλλογή των δεδομένων των αισθητήρων) στηρίζεται στο δίκτυο επικοινωνιών που υλοποιείται και η αποθήκευση τους γίνεται με χρήση βάσεων δεδομένων σε κεντρικό εξυπηρετητή καθώς και σε πλατφόρμα απομακρυσμένης πρόσβασης Internet of Things (IoT). Απότερος στόχος είναι η κατασκευή ενός δικτύου από κολώνες φωτισμού, οι οποίες θα συλλέγουν δεδομένα από τον περιβάλλοντα χώρο, θα διαχειρίζονται αυτόνομα σενάρια αλληλεπίδρασης, όπως η διαχείριση του δημόσιου φωτισμού, η συλλογή πληροφοριών του περιβάλλοντος χώρου με χρήση διαφόρων αισθητήρων και θα παρέχουν στους πολίτες της έξυπνης πόλης χρήσιμο πληροφοριακό υλικό μέσω ιστοσελίδων. Στο πλαίσιο της παρούσας εργασίας γίνεται μια προσπάθεια παρουσίασης του συνόλου των δομικών στοιχείων που χρησιμοποιήθηκαν στην υλοποίηση των συστημάτων, καθώς και των τεχνολογιών που μελετήθηκαν και συνδυάστηκαν με σκοπό την αποτελεσματική ανάλυση των δεδομένων συλλογής και της απεικόνισης τους. Αναλύονται τεχνολογίες σχετικά με τις επικοινωνίες που χρησιμοποιήθηκαν, με τις βάσεις δεδομένων, το διαδίκτυο (MySQL, Apache Web Server, PHP, JS) και τα μορφότυπα ανταλλαγής δεδομένων στο διαδίκτυο (XML, JSON). Επίσης γίνεται παρουσίαση της τεχνολογίας τρισδιάστατης εκτύπωσης (3D print) που χρησιμοποιήθηκε για τη δημιουργία μερών των συστημάτων υλοποίησης. Από την πλευρά του λογισμικού γίνεται παρουσίαση των γλωσσών προγραμματισμού που χρησιμοποιήθηκαν στον προγραμματισμό των μικροελεγκτών (C++, microPython), στην υλοποίηση της βάσης δεδομένων (JAVA, PHP, SQL) και στην διασύνδεση με τις ιστοσελίδες (PHP, JavaScript, Html, CSS). Τέλος, γίνεται σχετική αναφορά σε μελέτες πάνω στο αντικείμενο της ανάπτυξης έξυπνου δικτύου δημόσιου φωτισμού.

Λέξεις – Κλειδιά

Διαδικτύο των Πραγμάτων, Έξυπνη πόλη, Δημόσιος Φωτισμός, Αυτοματισμοί, Έξυπνη κολώνα, Έξυπνος Φωτισμός

Abstract

This thesis deals with the design and development of a network of "smart columns" of public lighting in a smart city environment. The implementation of the systems is based on the combination of the capabilities, offered by the Arduino, ESP32 and Raspberry development platforms in combination with the use of sensors. The communication between them is based on the standards of wired and wireless local area networks and the use of internet protocols. The movement of information, from the collection of sensor data, is based on the communication network implemented and its storage is done using databases on a central server as well as an Internet of Things (IoT) remote access platform. The ultimate goal is to build a network of lighting columns, which will collect data from the surrounding space, manage autonomous interaction scenarios such as managing public lighting, collecting information of the surrounding space using various application In this thesis, an attempt is made to present all the building blocks used in the implementation of the systems, as well as the technologies studied and combined in order to effectively analyse the data collection and its visualisation. Technologies related to the communications used, with the databases, the web (MySQL, Apache Web Server, PHP, JS) and the web data exchange formats (XML, JSON) are analysed. A presentation is also made of the 3D printing technology (3D print) used to create parts of the implementation systems. On the software side, the programming languages used in the programming of the microcontrollers (C++, microPython), in the implementation of the database (JAVA, PHP, SQL) and in the interface with the web pages (PHP, JavaScript, Html, CSS) are presented. Finally, reference is made to studies on the development of a smart grid for public lighting.

Keywords

Internet of Things (IoT), Smart City, Public Lighting, Arduino, Raspberry Pi, Smart Lamppost, Smart Lighting System

Περιεχόμενα

Περίληψη	v
Abstract.....	vii
Περιεχόμενα	viii
Κατάλογος Εικόνων / Σχημάτων	x
Κατάλογος Πινάκων	xiii
Συντομογραφίες & Ακρωνύμια	xv
1. Εισαγωγή	1
1.1 Σύστημα «έξυπνου φωτισμού» - Smart Street Light.....	3
1.2 Σύστημα προς υλοποίηση	6
1.3 Οργάνωση πτυχιακής	8
2. Τεχνολογίες Διαδικτυακών Εφαρμογών	10
2.1. Τεχνολογίες Διαδικτύου	10
2.1.1. HTTP (HyperText Transfer Protocol).....	10
2.1.2 URL (Uniform Resource Locator)	12
2.1.3 HTML (HyperText Markup Language)	12
2.1.4 CSS (Cascading Style Sheets).....	13
2.1.5 DOM (Document Object Model)	13
2.1.6 JSON (JavaScript Object Notation)	14
2.1.7 XML (Extensible Markup Language)	15
2.1.8 AJAX (Asynchronous JavaScript and XML).....	16
2.2 Πρωτόκολλα Δικτύωσης και Επικοινωνίας	16
2.2.1 Το πρότυπο IEEE 802.11 (Wi - Fi).....	16
2.2.2 Το πρότυπο IEEE 802.15 (Zigbee)	18
2.2.3 Το πρότυπο IEEE 802.3 (Ethernet).....	18
2.2.4 LoRaWAN (Long Range Wide Area Network).....	19
2.3 Διακομιστές εφαρμογών και βάσεων δεδομένων (Application Server / DataBase Server)	20
2.3.1 JAVA	20
2.3.2 Java Persistence API (JPA).....	21
2.3.4 MySQL	23
2.4 Ανάπτυξη Διαδικτυακών Εφαρμογών	24
2.4.1 Πλευρά Πελάτη (Client Side)	24
2.4.2 Πλευρά Διακομιστή (Server Side)	25
2.4.3 JavaScript.....	25
2.4.4 Web Server	26
2.4.5 PHP	27
2.4.6 IoT Πλατφόρμες	28
3. Πλατφόρμες Ανάπτυξης	29
3.1 Εισαγωγή	29
3.2 Ολοκληρωμένα περιβάλλοντα ανάπτυξης (IDE)	29
3.3 X-CTU Configuration & Test Utility Software	32
3.4 Apache HTTP Server.....	33

3.5 LAMP	34
3.6 Fritzing.....	35
3.7 Λειτουργικό Raspbian.....	36
3.8 Google charts	37
3.9 IOT ThingSpeak	38
4. Πλατφόρμες υλικού ανάπτυξης IoT.....	40
4.1 Πλατφόρμα Arduino	40
4.1.1 Αρχιτεκτονική Arduino.....	40
4.2.2 Εκδόσεις Arduino	41
4.2 Raspberry Pi.....	55
4.2.1 Εκδόσεις Raspberry Pi.....	55
4.2.2 Εκδόσεις Raspberry Pi Pico	60
5. Υλικό Συστήματος.....	62
5.1 Εισαγωγή	62
5.2 Arduino Uno	63
5.3 Raspberry Pi 4 Model B.....	64
5.4 Raspberry Pi Pico W	65
5.5 Xbee S1.....	67
5.6 LoRa Ai Thinker.....	71
5.7 ESP 32 Ai Thinker Cam	74
5.8 Αισθητήρες	75
5.8.1 Αισθητήρας Φωτός (Φωτοαντίσταση LDR)	75
5.8.3 Αισθητήρας Θερμοκρασίας και Υγρασίας.....	77
5.8.4 Αισθητήρας Βαρομετρικής Πίεσης BMP 280.....	78
5.8.5 Αισθητήρας Βροχής.....	81
5.8.6 Αισθητήρας ύπαρξης επικίνδυνων αερίων.....	83
5.8.7 Αισθητήρας Μονοξειδίου του άνθρακα.....	85
5.8.8 Αισθητήρας μέτρησης υπεριώδους ακτινοβολίας.....	86
5.8.9 Αισθητήρας μέτρησης θορύβου.....	89
5.8.10 Αισθητήρας ύπαρξης πυρκαγιάς	91
5.8.11 Αισθητήρας Απόστασης (υπερήχων)	93
5.8.12 Ρολόι πραγματικού χρόνου I2C	95
5.8.13 Αισθητήρας μέτρησης στάθμης υγρού.....	98
5.8.14 Αισθητήρας Ανίχνευσης κίνησης.....	100
6. Υλοποίηση Συστήματος.....	103
6.1 Εισαγωγή	103
6.2 Δίκτυο επικοινωνίας Έξυπνων κολώνων Φωτισμού.....	103
6.2.1 Δίκτυο από «Έξυπνες κολώνες» με επικοινωνία Xbee	104
6.2.2 Δίκτυο από «Έξυπνες κολώνες» με επικοινωνία LoRa	106
6.3 Σενάρια Αυτοματοποίησης Έξυπνων κολώνων Φωτισμού	108
6.3.1 «Έξυπνη κολώνα» Arduino Uno - Xbee S1	108
6.3.2 «Έξυπνη κολώνα» Raspberry Pico W - Xbee S1».....	110
6.3.3 «Έξυπνη κολώνα Arduino Uno – LoRa».....	112
6.3.4 «Έξυπνη κολώνα» Raspberry Pico W – LoRa.....	115
6.4 Συνδεσμολογία πλακετών ανάπτυξης	117



6.5 Κατασκευή Έξυπνων κολώνων Φωτισμού με χρήση τεχνολογίας εκτύπωσης 3D	118
6.6 Συνδεσμολογία πλακετών ανάπτυξης - Φωτογραφίες Υλοποίησης.....	119
7. Υλοποίηση Διασύνδεσης	125
7.1 Εισαγωγή	125
7.2 Βάση δεδομένων.....	125
7.3 Rest Server.....	126
7.4 ThingSpeak API.....	128
7.5 Διαδικτυακή εφαρμογή (PHP, ThingSpeak, Google charts).....	132
8. Συμπεράσματα Προτάσεις	144
Βιβλιογραφία	148
Ελληνόγλωσση	148
Ξενόγλωσση	149
Παράρτημα Α: «Arduino UNO – Xbee module»	155
Παράρτημα Β: «Raspberry Pico – Xbee module»	163
Παράρτημα Γ: «Arduino Uno – LoRa module»	169
Παράρτημα Δ: «Raspberry Pico – LoRa module».....	176
Παράρτημα E: «REST Server».....	185
E.1.1 Η κλάση HttpServer:	185
E.1.2 Η κλάση ServerListenerThread:	186
E.1.3 Η κλάση HttpConnectionWorkerThread:	187
E.1.4 Η κλάση HttpConfigurationException:	189
E.1.5 Η κλάση ConfigurationManager:	189
E.1.6 Η κλάση Configuration:	190
E.1.7 Η κλάση HttpVersion:	191
E.1.8 Η κλάση HttpStatusCode:.....	192
E.1.9 Η κλάση HttpRequest:	192
E.1.10 Η κλάση HttpParsingException:.....	194
E.1.11 Η κλάση HttpMethod:	197
E.1.12 Η κλάση DBquery:	198
E.1.13 Η κλάση ConnectDB:	201
E.1.14 Η κλάση TableMapper:	202
Παράρτημα ΣΤ: «Διαδικτυακή Εφαρμογή - PHP».....	204
ΣΤ.1 Controllers.....	204
ΣΤ.2 Views	205
ΣΤ.3 Providers	226
ΣΤ.4 Partials.....	231
Παράρτημα Ζ: «Διαδικτυακή Κάμερα – ESP32»	236

Κατάλογος Εικόνων / Σχημάτων

Εικόνα 2-1 HTTP Request – HTTP Response.....	11
Εικόνα 2-2 Δομή HTTP Request	11
Εικόνα 2-3 Βασική ροή κόμβων δένδρου HTML.....	14
Εικόνα 2-4 Παράδειγμα κώδικα JSON	15

Εικόνα 2-5 Παράδειγμα κώδικα XML.....	16
Εικόνα 2-6 Τυπική Αρχιτεκτονική WLAN IEEE 802.11	17
Εικόνα 2-7 Εμβέλεια κάλυψης Ασύρματων επικοινωνιών.....	20
Εικόνα 2-8 Επικοινωνία Web Server	27
Εικόνα 3-1 IntelliJ IDEA.....	32
Εικόνα 3-2 Εφαρμογή XCTU	33
Εικόνα 3-3 Δομή πλέγματος LAMP	35
Εικόνα 3-4 Περιβάλλον σχεδίασης Fritzing	36
Εικόνα 3-5 Λειτουργικό Σύστημα Raspbian.....	37
Εικόνα 3-6 Παράδειγμα διαγράμματος Google Charts.....	38
Εικόνα 3-7 Κανάλι ThingSpeak.....	39
Εικόνα 4-1 Αρχιτεκτονική Harvard.....	41
Εικόνα 4-2 Αρχιτεκτονική Arduino	41
Εικόνα 4-3 Arduino Uno R3.....	42
Εικόνα 4-4 Arduino Uno R4 minima.....	43
Εικόνα 4-5 Arduino Uno R4 WiFi	44
Εικόνα 4-6 Arduino Leonardo	44
Εικόνα 4-7 Arduino Uno WiFi Rev2.....	45
Εικόνα 4-8 Arduino Uno Zero	45
Εικόνα 4-9 Arduino Micro	46
Εικόνα 4-10 Arduino Uno Mini Limited Edition	46
Εικόνα 4-11 Arduino Nano 33 IoT	47
Εικόνα 4-12 Arduino Nano RP2040.....	48
Εικόνα 4-13 Arduino ESP32	48
Εικόνα 4-14 Arduino Nano 33 BLE	48
Εικόνα 4-15 Arduino Nano 33 BLE Sense	49
Εικόνα 4-16 Arduino Nano Every	50
Εικόνα 4-17 Arduino Mega 2560 Rev3	50
Εικόνα 4-18 Arduino Due	51
Εικόνα 4-19 Arduino Giga R1 WiFi	51
Εικόνα 4-20 Arduino MKR Zero	52
Εικόνα 4-21 Arduino MKR Vidor 4000	52
Εικόνα 4-22 Arduino MKR NB 1500	53
Εικόνα 4-23 Arduino MKR WAN 1310	53
Εικόνα 4-24 Arduino MKR WAN 1300	54
Εικόνα 4-25 Arduino MKR 1000.....	54
Εικόνα 4-26 Arduino MKR 1010.....	55
Εικόνα 4-27 Raspberry Pi 1 Model B	56
Εικόνα 4-28 Raspberry Pi 2 Model B	57
Εικόνα 4-29 Raspberry Pi 3 Model B	58
Εικόνα 4-30 Raspberry Pi 4 Model B	59
Εικόνα 4-31 Raspberry Pi 5	60
Εικόνα 4-32 Οικογένεια Raspberry Pico (Pico,H,W,WH)	61



Εικόνα 5-1 Raspberry Pi 4 ARM Cortex-A72 processor.....	65
Εικόνα 5-2 Αρχιτεκτονική Pico Pi 5	66
Εικόνα 5-3 Raspberry Pico W Pinout	67
Εικόνα 5-4 Xbee S1 και Xbee Usb Adapter	68
Εικόνα 5-5 Data Flow Diagram in a UART-interface	68
Εικόνα 5-6 Δομή εντολής AT.....	69
Εικόνα 5-7 Δομή API frame.....	69
Εικόνα 5-8 Διασύνδεση Arduino - Xbee S1.....	71
Εικόνα 5-9 LoRa Network star topology	72
Εικόνα 5-10 Διασύνδεση Arduino - LoRa.....	73
Εικόνα 5-11 ESP32 Ai Thinker Cam PinOut	74
Εικόνα 5-12 LDR Pinout	76
Εικόνα 5-13 DHT11 PinOut	78
Εικόνα 5-14 BMP280 PinOut.....	79
Εικόνα 5-15 Αισθήτηρας Βροχής PinOut	82
Εικόνα 5-16 MQ 135 PinOut	84
Εικόνα 5-17 MQ-7 PinOut.....	86
Εικόνα 5-18 UV Sensor PinOut.....	88
Εικόνα 5-19 Αισθητήρας Θορύβου PinOut.....	90
Εικόνα 5-20 Αισθητήρας ύπαρξης Πυρκαγιάς PinOut.....	92
Εικόνα 5-21 Αισθητήρας Απόστασης PinOut.....	94
Εικόνα 5-22 Ρολόι πραγματικού χρόνου PinOut	96
Εικόνα 5-23 Αισθητήρας μέτρησης στάθμης υγρού PinOut.....	99
Εικόνα 5-24 Αισθητήρας Ανίχνευσης κίνησης PinOut	101
 Εικόνα 6-1 Σχεδιάγραμμα δικτύου συστήματος υλοποίησης	104
Εικόνα 6-2 Υποδίκτυο βασισμένο στην επικοινωνία με χρήση Xbee	106
Εικόνα 6-3 Υποδίκτυο βασισμένο στην επικοινωνία με χρήση LoRa	107
Εικόνα 6-4 Συνδεσμολογία σεναρίου Arduino	109
Εικόνα 6-5 Συνδεσμολογία σεναρίου Raspberry Pico	111
Εικόνα 6-6 Συνδεσμολογία σεναρίου Arduino	114
Εικόνα 6-8 Συνδεσμολογία συστήματος υλοποίησης	118
Εικόνα 6-9 Σχέδιο εκτύπωσης με τεχνολογία 3D.....	119
Εικόνα 6-10 Έξυπνες κολώνες υλοποίησης	120
Εικόνα 6-11 Αισθητήρας βροχής στο πάνω μέρος της κολώνας	120
Εικόνα 6-12 Εσωτερικό τμήμα της έξυπνης κολώνας.....	121
Εικόνα 6-13 Οι έξυπνες κολώνες με επικοινωνία LoRa	121
Εικόνα 6-14 Συνολική υλοποίηση της έξυπνης κολώνας.....	122
Εικόνα 6- 15 Έξυπνη κολώνα με Αισθητήρας κίνησης.....	123
Εικόνα 6- 16 Μακέτα Δικτύου με έξυπνες Κολώνες	124
 Εικόνα 7-1 Διάγραμμα REST API	125
Εικόνα 7- 2 Σχήμα βάσης Δεδομένων.....	126
Εικόνα 7-3 Διάγραμμα εξαρτήσεων Rest Server	128



Εικόνα 7-4 Δημιουργία καναλιού ThingSpeak.....	130
Εικόνα 7-5 Ρυθμίσεις πεδίων	130
Εικόνα 7- 6 Δημιουργία - Ρύθμιση Διαγράμματος	131
Εικόνα 7-7 Διάγραμμα Θερμοκρασίας	131
Εικόνα 7-8 Δημιουργία κλειδιών API.....	131
Εικόνα 7-9 Κώδικας HTML - iFrame	132
Εικόνα 7-10 Δομή φακέλου Διαδικτυακής Εφαρμογής.....	132
Εικόνα 7-11 Αρχική σελίδα εφαρμογής.....	133
Εικόνα 7-12 Σελίδα εισόδου εφαρμογής.....	134
Εικόνα 7- 13 Σελίδα εγγραφής νέου χρήστη	134
Εικόνα 7-14 Αρχική σελίδα εφαρμογής.....	135
Εικόνα 7-15 Σελίδα χάρτη επιλογών.....	136
Εικόνα 7-16 Εμφάνιση Διεύθυνσης στο Χάρτη	136
Εικόνα 7-17 Σελίδα λεπτομέρειών κολώνας	137
Εικόνα 7-18 Ενεργοποίηση Αισθητήρα	138
Εικόνα 7-19 Επιλογές στατιστικών	138
Εικόνα 7-20 Σελίδα στατιστικών κολώνας.....	139
Εικόνα 7-21 Σελίδα Συγκριτικών στατιστικών	139
Εικόνα 7-22 Σελίδα Camera	140
Εικόνα 7-23 Προβολή πλέγματος με ζωντανή μετάδοση	141
Εικόνα 7-24 Σελίδα ρυθμίσεων κάμερας.....	142
Εικόνα 7-25 Σελίδα IoT - ThingSpeak	143

Κατάλογος Πινάκων

Πίνακας 4-1 Χαρακτηριστικά Pi 1 Model B.....	56
Πίνακας 4-2 Χαρακτηριστικά Pi 2 Model B.....	57
Πίνακας 4-3 Χαρακτηριστικά Pi 3 Model B.....	58
Πίνακας 4-4 Χαρακτηριστικά Pi 4 Model B.....	58
Πίνακας 4-5 Χαρακτηριστικά Pi 5	59
Πίνακας 4-6 Χαρακτηριστικά Raspberry Pi Pico	61
Πίνακας 4-7 Χαρακτηριστικά Raspberry Pi Pico W	61
Πίνακας 5-1 Arduino Uno - Pin Out	64
Πίνακας 5-2 Ακροδέκτες Xbee S1	70
Πίνακας 5-3 ESP32 Ai Thinker Cam.....	75
Πίνακας 5-4 Αισθητήρας LDR	76
Πίνακας 5-5 Ενδεικτικός Κώδικας LDR	76
Πίνακας 5-6 Αισθητήρας DHT11.....	77
Πίνακας 5-7 Ενδεικτικός Κώδικας DHT11	78
Πίνακας 5-8 Ο Αισθητήρας BMP280	79
Πίνακας 5-9 Ενδεικτικός Κώδικας BMP280	80
Πίνακας 5-10 Αισθητήρας Βροχής.....	81
Πίνακας 5-11 Ενδεικτικός Κώδικας Αισθητήρα Βροχής	83



Πίνακας 5-12 Αισθητήρας Επικίνδυνων Αερίων.....	83
Πίνακας 5-13 Ενδεικτικός Κώδικας MQ 135	84
Πίνακας 5-14 Αισθητήρας MQ-7	85
Πίνακας 5-15 Ενδεικτικός Κώδικας MQ-7	86
Πίνακας 5-16 Αισθητήρας Υπεριώδους Ακτινοβολίας	87
Πίνακας 5-17 Ενδεικτικός Κώδικας αισθητήρα UV	89
Πίνακας 5-18 Αισθητήρας μέτρησης θορύβου	90
Πίνακας 5-19 Ενδεικτικός Κώδικας Αισθητήρα Θορύβου	91
Πίνακας 5-20 Αισθητήρας ύπαρξης Πυρκαγιάς	92
Πίνακας 5-21 Ενδεικτικός Κώδικας Αισθητήρα ύπαρξης Πυρκαγιάς	93
Πίνακας 5-22 Αισθητήρας Απόστασης	94
Πίνακας 5-23 Ενδεικτικός Κώδικας Αισθητήρα Απόστασης	95
Πίνακας 5-24 Ρολόι πραγματικού χρόνου	95
Πίνακας 5-25 Ενδεικτικός Κώδικας Ρολογιού πραγματικού χρόνου	97
Πίνακας 5-26 Αισθητήρας μέτρησης στάθμης υγρού	98
Πίνακας 5-27 Ενδεικτικός Κώδικας Αισθητήρα μέτρησης στάθμης υγρού	99
Πίνακας 5-28 Αισθητήρας Ανίχνευσης κίνησης.....	100
Πίνακας 5-29 Ενδεικτικός Κώδικας Αισθητήρας Ανίχνευσης κίνησης.....	102
Πίνακας 6-1 Επεξηγήσεις σχημάτων εικόνας 6-1	104

Συντομογραφίες & Ακρωνύμια

3G	Third Generation (of Wireless Mobile Telecommunications Technology)
ΔΕ	Διπλωματική Εργασία
ΕΑΠ	Ελληνικό Ανοικτό Πανεπιστήμιο
ΘΕ	Θεματική Ενότητα
NMEA	National Marine Electronics Association
ΠΕ	Πτυχιακή Εργασία
ΠΣ	Πρόγραμμα Σπουδών
ΣΥΝ	Συντονιστής
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
Arduino IDE	Arduino Integrated Development Environment
AT-Commands	ATtention-Commands
AVR	Automatic Voltage Regulator
BLE	Bluetooth
BMP	Bitmap Image file
CAN	Controller Area Network
CLI	Command-Line Interface
COBRA	Common Object Request Broker Architecture
CPHA	Clock Phase
CPOL	Clock Polarity
CSMA/CA	Carrier-sense multiple access with collision detection
CSS	Cascading Style Sheets
DAC	Digital-to-Analog Converter
DBMS	Database Management Systems
DC	Direct Current
DMP	Digital Motion Processor
DOM	Document Object Model
EDGE	Enhanced Data for Global Evolution



FDCAN	Flexible Data-Rate Controller Area Network
FDD	Frequency Division Duplex
FIFO	First In First Out
GNSS	Global Navigation Satellite System
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
I2C	Inter-Integrated Circuit
ICSP	In-Circuit Serial Programming
IDE	Integrated Development Environment
IMD	International Institute for Management Development
IoT	Internet of Things
IP	Internet Protocol
IR	Infrared Radiation
JPA	JAVA Persistence API
JSON	JavaScript Object Notation
JVM	JAVA Virtual Machine
LAMP	Linux Apache MySQL PHP
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LoRaWAN	Long Range Wide Area Network
LTE	Long Term Evolution
MISO	Master In Slave Out
MIPI	Mobile Industry Processor Interface
MOSI	Master Out Slave In
MVC	Model View Controller
MySQL	My Structured Query Language
OP AMP	OPerational AMPlifier
OPP	Object-Oriented Programming



ORM	Object Relational Mapping
PCB	Printed Circuit Board
PHP	Hypertext Preprocessor
PIR	Passive Infra-Red
PWM	Pulse Width Modulation
RDBMS	Relational database management systems
REST	REpresentational State Transfer
RFC	Request for Comments
RFID	Radio-Frequency Identification
ROA	Resource Oriented Architecture
RPC	Remote Procedure Call
SCK	Serial Clock
SCL	Serial Clock Pin
SCO	Smart City Observatory
SDA	Serial Data Pin
SGML	Standard Generalized Markup Language
SIM	Subscriber Identity Module
SMS	Short Message Service
SOAP	Simple Object Access Protocol
SPI	Serial Peripheral Interface
SQL	Structured Query Language
SS	Slave Select
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TFT	Thin Film Transistor
TSL	Transport Layer Security
UART	Universal Asynchronous Receiver/Transmitter
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
W3C	World Wide Web Consortium



WCC	World Competitiveness Center
WSDL	Web services Description Language
WWW	World Wide Web
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XSLT	eXtensible Stylesheet Language

1. Εισαγωγή

Από τα πρώτα χρόνια της ύπαρξης τους οι άνθρωποι ζούσαν σε ομάδες. Στην αρχή μετακινούνταν, όταν εξαντλούνταν οι διαθέσιμοι πόροι του περιβάλλοντος τους, στη συνέχεια, όμως, ανακάλυψαν τη γεωργία και άλλαξαν σταδιακά τη νομαδική συμπεριφορά τους, υιοθετώντας πρότυπα ζωής που περιελάμβαναν την εγκατάσταση τους σε μόνιμους οικισμούς. Ήταν η πρώτη επανάσταση που άλλαξε δραματικά την ανθρώπινη εξέλιξη, κι έγινε η βάση για να αναπτυχθούν οι πρώτοι οικισμοί, το εμπόριο και οι στοιχειώδεις πολιτικές λειτουργίες. Ακολούθησε η φάση της αστικοποίησης, όπου οι άνθρωποι προσπάθησαν να αντιμετωπίσουν ορθολογικά τις ανάγκες τους. Με το πέρασμα των χρόνων οι πόλεις εξελίχθηκαν και έγιναν ένα σύστημα με συνεχή ροή, ευμετάβλητες στο πέρασμα του χρόνου, δημιουργώντας νέες συνθήκες διαβίωσης, στις οποίες ο άνθρωπος κλήθηκε να προσαρμοστεί. (Μεταξίδης, 2019)

Η διαδικασία αυτή έγινε ιδιαίτερα αισθητή κατά τα τελευταία πενήντα χρόνια, καθώς η ραγδαία τεχνολογική εξέλιξη και η αυξανόμενη ανάγκη του ανθρώπου σε όλα τα είδη πληροφορίας ανά πάσα στιγμή, καθιστά αναγκαίο τον μετασχηματισμό των πόλεων σε «έξυπνες». Σύμφωνα με τον επίσημο ιστότοπο της Ευρωπαϊκής Επιτροπής μια «έξυπνη πόλη» θα μπορούσε να οριστεί ως ο τόπος, όπου η χρήση ψηφιακών λύσεων μετατρέπει τα παραδοσιακά δίκτυα και τις υπηρεσίες σε πιο αποδοτικές υποδομές, προς όφελος των κατοίκων και των επιχειρήσεών της. Μια έξυπνη πόλη χρησιμοποιεί την τεχνολογία με σκοπό την καλύτερη αξιοποίηση των πόρων της σε συνδυασμό με λιγότερες εκπομπές βλαβερών αερίων για την προστασία του περιβάλλοντος. Σε μια έξυπνη πόλη έχουμε πιο έξυπνα δίκτυα αστικών μεταφορών, αναβαθμισμένες εγκαταστάσεις (ύδρευση / διάθεση αποβλήτων) και πιο αποδοτικούς τρόπους φωτισμού και θέρμανσης των κτιρίων. (Ευρωπαϊκή Επιτροπή, Επίσημος Ιστότοπος, 2023)

Στις πόλεις, ο δημόσιος φωτισμός αποτελεί μια από τις βασικές υπηρεσίες που παρέχεται στους πολίτες, γιατί καθιστά ασφαλέστερες τις μετακινήσεις. Ωστόσο, υπάρχουν πολλά περισσότερα που μπορούν να γίνουν για να διασφαλιστεί ότι ο δημόσιος φωτισμός μπορεί να εξυπηρετήσει καλύτερα τους πολίτες. Οι «απλές» κολώνες φωτισμού, μπορούν να μετατραπούν σε «έξυπνες» αφού δεν θα εκπληρώνουν μόνο τη λειτουργία του φωτισμού, αλλά θα προσφέρουν πρόσθετες υπηρεσίες για τους πολίτες. Οι υπηρεσίες αυτές μπορούν να περιλαμβάνουν (χωρίς να περιορίζονται σε αυτά), την παρακολούθηση του

περιβάλλοντος (θερμοκρασία, υγρασία, θόρυβος, ποιότητα του αέρα κ.α), τη βελτίωση του δημόσιου Wi-Fi και τη διαχείριση της λειτουργίας φωτισμού με ενεργειακά αποδοτικό τρόπο. Στο σύνολό της Ευρωπαϊκής Ένωσης (ΕΕ) υπάρχουν πάνω από 60 εκατομμύρια κολώνες δημόσιου φωτισμού, με το μεγαλύτερο μέρος από αυτούς (περίπου το 75%) να είναι ηλικίας μεγαλύτερης των 25 ετών. Μόνο ένα πολύ μικρό ποσοστό τους χρησιμοποιεί λαμπτήρες ενεργειακής απόδοσης. Υπάρχουν εκτιμήσεις που αναφέρουν ότι κάθε εβδομάδα σπαταλούνται 200 εκατομμύρια ευρώ σε ενέργεια, τα οποία θα μπορούσαν να εξοικονομηθούν με τη χρήση σύγχρονων φωτιστικών σωμάτων. (Medium, 2018)

Η χρήση ενός έξυπνου δικτύου δημόσιου φωτισμού αποτελεί βασικό συστατικό στο μετασχηματισμό των πόλεων σε έξυπνες. Η Ευρωπαϊκή Επιτροπή έχει αναπτύξει δράσεις για την ενημέρωση προς αυτή την κατεύθυνση, όπως το «Humble street Lamppost» το 2019 και το «Smart Cities Marketplaces» (το οποίο αποτελεί την τρέχουσα δράση της). Η δράση «Smart Cities Marketplaces» έχει ως στόχο να χρησιμεύσει ως κόμβος πρακτικών γνώσεων για την ανάπτυξη ικανοτήτων καθώς επίσης να διευκολύνει την χρηματοδότηση σε διάφορους τομείς, μεταξύ των οποίων είναι η βιώσιμη αστική κινητικότητα, οι ολοκληρωμένες υποδομές στην ενέργεια, τις τεχνολογίες πληροφοριών και επικοινωνιών και τις μεταφορές που έχουν ως στόχο τη βελτίωση της ποιότητας ζωής των πολιτών και των επιχειρήσεων της ΕΕ αλλά και στην επίτευξη των στόχων της ΕΕ σε τομείς όπως η ενέργεια και η κλιματική αλλαγή. (Energy and Smart Cities, 2023)

Από το 2019, το Παρατηρητήριο Έξυπνων Πόλεων (SCO), τμήμα του Παγκόσμιου Κέντρου Ανταγωνιστικότητας (WCC) του IMD (International Institute for Management Development), συνδυάζει στοιχεία από έρευνες για να δείξει το βαθμό στον οποίο η τεχνολογία επιτρέπει στις πόλεις να αντιμετωπίσουν τις προκλήσεις που αντιμετωπίζουν για να επιτύχουν υψηλότερη ποιότητα ζωής για τους κατοίκους τους. Σύμφωνα με στοιχεία του IMD για το έτος 2023 οι 7 κορυφαίες έξυπνες πόλεις στον κόσμο με τους υψηλότερους δείκτες είναι η Ζυρίχη (Ελβετία), το Όσλο (Νορβηγία), η Καμπέρα (Αυστραλία), η Κοπεγχάγη (Δανία), η Λοζάνη (Ελβετία), το Λονδίνο (Ηνωμένων Βασίλειο) και η Σιγκαπούρη η οποία βρίσκεται στην κορυφή των περισσοτέρων καταλόγων όταν πρόκειται για έξυπνες πόλεις. (Madureira, 2023)

1.1 Σύστημα «έξυπνου φωτισμού» - Smart Street Light

Ο «έξυπνος» έλεγχος του δημόσιου φωτισμού, σε συνδυασμό με την αξιοποίηση πληροφοριών από τη συλλογή δεδομένων καιρού και κυκλοφορίας σε πραγματικό χρόνο είναι μια αναδυόμενη υλοποίηση του IoT(Internet of Things) στις έξυπνες πόλεις. Οι κολώνες δημόσιου φωτισμού είναι από τις πιο κοινές υποδομές στις πόλεις και ο συνδυασμός τους με μικροελεγκτές και αισθητήρες μπορεί να δημιουργήσουν ένα δίκτυο συλλογής δεδομένων, των οποίων η ανάλυση και η περαιτέρω αξιοποίηση αποτελεί στοιχείο μίας πόλης που στοχεύει στην αποτελεσματικότερη χρήση των ψηφιακών λύσεων προς όφελος των κατοίκων και των υποδομών της. Επιπλέον, οι στόχοι βιωσιμότητας των σύγχρονων έξυπνων πόλεων απαιτούν αποτελεσματικές στρατηγικές για την επίτευξη ενεργειακά αποδοτικής λειτουργίας. Η ενσωμάτωση τεχνολογιών IoT μπορεί να συμβάλει αποτελεσματικά στη συγκεκριμένη κατεύθυνση.

Οι Hettiarachchi και Naz Islam (2022) ερεύνησαν και ανέπτυξαν ένα πλαίσιο έξυπνου δημόσιου φωτισμού με βάση τη συλλογή δεδομένων καιρού και δεδομένων κυκλοφοριακής κίνησης. Το πλαίσιο που προτείνουν βελτιστοποιεί την κατανάλωση ενέργειας σε μια συγκεκριμένη περιοχή με χρήση απενεργοποίησης του φωτισμού ή την λειτουργία του σε χαμηλότερη ένταση, χωρίς όμως να υποβαθμίζει την ποιότητα του παρεχόμενου φωτισμού. Ανέπτυξαν έναν αλγόριθμο που λαμβάνει υπόψη την κάλυψη νέφους στον ουρανό και την ορατότητα, ώστε να οριστούν τα κατώτατα όρια προτεραιότητας για μεμονωμένους δρόμους με βάση τα δεδομένα πυκνότητας κυκλοφορίας.

Οι Babu, Nisha, Dhasan, Venkatesan και Karthikyan (2021) εξέτασαν τη σημασία του έξυπνου φωτισμού στις σύγχρονες πόλεις και πρότειναν λύσεις για την εξοικονόμηση ενέργειας, τη μείωση των εκπομπών διοξειδίου του άνθρακα (CO₂), τον περιορισμό της φωτορύπανσης και την ελαχιστοποίηση του κόστους συντήρησης, προχωρώντας στο σχεδιασμό μίας έξυπνης κολώνας φωτισμού, δίνοντας έμφαση στην ενεργειακά αποδοτική διαχείριση του φωτισμού και στις υπηρεσίες έκτακτης ανάγκης. Τα χαρακτηριστικά του σχεδιασμού τους περιλαμβάνουν, την παρακολούθηση των καιρικών συνθηκών, την ανίχνευση βροχής και πλημμύρας καθώς και τη δυνατότητα παροχής εμπορικών εφαρμογών όπως η διαφήμιση επιχειρήσεων.

Οι Merlino, Bruneo, Distefano, Longo, Puliafito και Al-Anbuky (2015) παρουσίασαν μια εφαρμογή έξυπνου φωτισμού σε ένα δημόσιο πάρκο, κατά το οποίο οι έξυπνες κολώνες



φωτισμού είναι εξοπλισμένες με αισθητήρες, με σκοπό την ενίσχυση της δημόσιας ασφάλειας και της μείωσης του κόστους λειτουργίας, παρέχοντας παρακολούθηση σε πραγματικό χρόνο. Η αρχιτεκτονική τους βασίζεται σε συσκευές IoT (π.χ. πλακέτες Arduino) και διασύνδεση τους στο cloud. Η έρευνά τους ανέδειξε την ευελιξία του συστήματος υποστηρίζοντας διάφορους τύπους συσκευών IoT, όπως αισθητήρες φωτός, αισθητήρες ήχου καθώς και δεδομένα γεωγραφικού εντοπισμού με τη χρήση του Google Maps.

Οι Lau, Merrett, Weddell και White από το πανεπιστήμιο του Σαουθάμπτον (2015) σχεδίασαν ένα σύστημα έξυπνου φωτισμού, το TALiSMaN (Traffic-Aware Lighting Scheme Management Network), το οποίο λαμβάνει τα δεδομένα κυκλοφορίας και ρυθμίζει αυτόνομα τα επίπεδα φωτισμού με βάση την κίνηση που ανιχνεύει από τους πεζούς και τα αυτοκίνητα. Οι προσομοιώσεις από τη χρήση του συστήματος δείχνουν ότι μειώνει σημαντικά την κατανάλωση ενέργειας, διατηρώντας παράλληλα την καλή χρησιμότητα του οδικού φωτισμού.

Οι Gouthami, Santosh, Pavan, Karthik και Ramya (2016) υλοποίησαν ένα σύστημα που αντιμετωπίζει το ζήτημα του αναποτελεσματικού ελέγχου του δημόσιου φωτισμού λόγω της ανεπαρκούς αυτοματοποίησης. Πρότειναν ένα σύστημα αυτόματου ελέγχου με τη χρήση μιας φωτό-εξαρτώμενης αντίστασης (LDR) και ενός μικροελεγκτή (ATmega8). Η LDR ανιχνεύει την ένταση του φωτός και ο μικροελεγκτής επεξεργάζεται αυτά τα δεδομένα μαζί με ρυθμίσεις πραγματικού χρόνου και χρόνου ενεργοποίησης/απενεργοποίησης για τον έλεγχο του φωτισμού. Το σύστημα ενεργοποιεί και απενεργοποιεί αποτελεσματικά τα φώτα με βάση το διαθέσιμο φως της ημέρας, εξοικονομώντας ενέργεια και μειώνοντας τη σπατάλη. Είναι οικονομικά αποδοτικό, εξαιρετικά αξιόπιστο και εύκολο στη συντήρηση, καθιστώντας το μια πρακτική λύση για τη βελτίωση του δημόσιου φωτισμού.

Μια ακόμη προσέγγιση είναι η υλοποίηση των Yang, Lee, Chen, Yang, Huang και Hou (2016), η οποία βασίστηκε στη χρήση της πλατφόρμας Arduino σε συνδυασμό με διαφόρους αισθητήρες καθώς και IP (Internet Protocol) καμερών. Η αρχιτεκτονική τους αποτελείται από έναν διακομιστή νέφους και end points. Στην υλοποίηση τους εστίασαν στη χρήση μέτρων ασφαλείας (κρυπτογράφηση και SSH (Secure Shell) tunnels) με σκοπό την προστασία των επικοινωνιών μεταξύ νέφους και των end-points.



Οι Shweta, Shruthi, Shreya, Shwetha, Sujay και Chaitanya (2019) υλοποίησαν ένα σύστημα που αποσκοπεί στην αποτελεσματική διαχείριση του δημόσιου φωτισμού, μέσω ασύρματου δικτύου αισθητήρων. Με την εργασία τους επεδίωξαν να μειώσουν την κατανάλωση ηλεκτρικής ενέργειας, ελέγχοντας εξ αποστάσεως τις κολώνες φωτισμού με τη χρήση αισθητήρων, όπως αισθητήρες IR (Infrared Radiation) και LDRs. Έκαναν χρήση της πλατφόρμας Arduino και παρουσίασαν τα αποτελέσματα της πειραματικής διάταξης τους υλοποιώντας ένα σύστημα που προσφέρει μια οικονομικά αποδοτική προσέγγιση για την εξοικονόμηση ενέργειας και τη βελτίωση της αποδοτικότητας του δημόσιου φωτισμού.

Οι Sravani, Malarvezhi και Dayana (2018) προχώρησαν στην υλοποίηση ενός έξυπνου συστήματος φωτισμού με τη χρήση της τεχνολογίας IoT συνδυάζοντας τις πλατφόρμες Arduino και Raspberry. Το σύστημα αποτελείται από το Raspberry Pi ως κύριο κόμβο και το Arduino ως δευτερεύοντα κόμβο, που επικοινωνούν μεταξύ τους. Το σύστημα χρησιμοποιεί αισθητήρες LDR και αισθητήρες παθητικής υπέρυθρης ακτινοβολίας (PIR) για την προσαρμογή της έντασης του φωτός με βάση την παρουσία σκοταδιού ή και αντικειμένων. Χρησιμοποιεί διαμόρφωση πλάτους παλμού (PWM) για την προσαρμογή της έντασης του φωτός σύμφωνα με προκαθορισμένα χρονοδιαγράμματα και τις περιβαλλοντικές συνθήκες, εξοικονομώντας ενέργεια. Το σύστημα επιτρέπει την απομακρυσμένη παρακολούθηση και έλεγχο του φωτισμού, ενώ παρέχει δεδομένα στο cloud για περαιτέρω ανάλυση.

Οι Daely, Reda, Satrya, Kim και Shin (2017) ανέπτυξαν ένα έξυπνο σύστημα φωτισμού με διόδους εκπομπής φωτός (LED) που ενσωματώνει αισθητήρες και ασύρματες μονάδες αισθητήρων με βάση το ZigBee δίκτυο επικοινωνίας. Η υλοποίηση του συστήματος τους περιλαμβάνει έναν κεντρικό διακομιστή ιστού που συγκεντρώνει τα δεδομένα των αισθητήρων σε πραγματικό χρόνο και πληροφορίες για τις καιρικές συνθήκες από τις κολώνες φωτισμού.

Οι Aiswarya, Basudeb and Mazumdar (2022) ανέπτυξαν ένα σύστημα έξυπνου φωτισμού ελεγχόμενο από μικροελεγκτή Arduino που ρυθμίζει αυτόματα την ένταση του φωτισμού με βάση τον φωτισμό του περιβάλλοντος και την ανίχνευση κίνησης, εξοικονομώντας ενέργεια κατά τις περιόδους χαμηλής δραστηριότητας. Οι χρήστες μπορούν να ελέγχουν χειροκίνητα τα φώτα του δρόμου μέσω μηνυμάτων κειμένου. Στο σύστημα τους ενσωματώνονται διάφοροι αισθητήρες, συμπεριλαμβανομένου ενός αισθητήρα φωτισμού

και ενός αισθητήρα παθητικής υπέρυθρης ακτινοβολίας (PIR) για την ανίχνευση κίνησης, μονάδα GSM (Global System for Mobile Communications) για τηλεχειρισμό, ρολόι πραγματικού χρόνου (RTC) για τον υπολογισμό της ώρας της ημέρας, κύκλωμα διαμόρφωσης πλάτους παλμού (PWM) που επιτρέπει τη μεταβλητή ένταση με σκοπό τη μείωση κατανάλωσης ενέργειας, καθώς και σύστημα που καταγράφει τιμές φωτισμού περιβάλλοντος με σφραγίδες ώρας και ημερομηνίας σε αρχείο κειμένου.

Οι Gehlot, Alshamrani, Singh, Rashid, Akram, AlGhamdi και Albogamy (2021) σε μία προσέγγιση που στοχεύει στη χρήση διάφορων τεχνολογιών ασύρματης επικοινωνίας, όπως LoRa, Zigbee και Bluetooth (BLE) για τη συνδεσιμότητα IoT, υλοποίησαν σύστημα έξυπνου φωτισμού στο οποίο μπορούν να ρυθμίζουν το φωτισμό, να προσφέρουν Wi-Fi και να παρακολουθούν τις περιβαλλοντικές συνθήκες στις έξυπνες πόλεις. Τα βασικά εξαρτήματα που χρησιμοποιήθηκαν περιλαμβάνουν διάφορους αισθητήρες, μικροελεγκτές (Arduino Uno), μια μονάδα ESP32 με κάμερα, διακομιστή cloud και τρισδιάστατη εκτύπωση για δημιουργία διαφόρων εξαρτημάτων (αισθητήρες εικόνας).

Οι Kiran, Nityanand, Ozair Khan και Tiwari (2020) προχώρησαν σε εφαρμογή ενός συστήματος έξυπνου δημόσιου φωτισμού που προσαρμόζεται στην κίνηση πεζών, ποδηλατών και αυτοκινήτων. Τα εξαρτήματα που χρησιμοποιήθηκαν στο σύστημα περιλαμβάνουν έναν μικροελεγκτή Arduino Uno, αισθητήρες υπέρυθρης ακτινοβολίας, μια φωτοεξαρτώμενη αντίσταση (LDR) και τεχνολογία φωτισμού LED. Το σύστημα τους λειτουργεί ενεργοποιώντας αυτόματα το φωτισμό όταν περνάει ένα όχημα, χρησιμοποιώντας αισθητήρες IR για την ανίχνευση της κίνησης και LDR για την ανίχνευση των επιπέδων φωτισμού.

1.2 Σύστημα προς υλοποίηση

Στην παρούσα πτυχιακή θα υλοποιηθεί ένα δίκτυο από έξυπνες κολώνες φωτισμού σε περιβάλλον έξυπνης πόλης, βασισμένο σε διάφορες πλατφόρμες IoT. Το σύστημα που θα υλοποιηθεί θα προσομοιώσει τη λειτουργία ενός δικτύου φωτισμού σε περιβάλλον έξυπνης πόλης, το οποίο θα λειτουργεί αυτόνομα με σκοπό την εξοικονόμηση ενέργειας αλλά και την παροχή πληροφοριών προς τους πολίτες της. Συγκεκριμένα το σύστημα θα συλλέγει πληροφορίες από τον περιβάλλοντα χώρο για τις καιρικές συνθήκες, όπως θερμοκρασία και

υγρασία, την ύπαρξη επικίνδυνων αερίων στην ατμόσφαιρα, το επίπεδο διοξειδίου του άνθρακα, την ύπαρξη πυρκαγιάς, την ύπαρξη βροχόπτωσης, τα επίπεδα υπεριώδους ακτινοβολίας, τα επίπεδα στάθμης νερού στο δημόσιο οδόστρωμα για την αποφυγή πλημμυρών. Επίσης το σύστημα θα έχει τη δυνατότητα να ελέγχει το ποσοστό φωτισμού της έξυπνης κολώνας με βάση τη χρονική στιγμή σε συνδυασμό με τα επίπεδα του φυσικού φωτός στον περιβάλλοντα χώρο, αλλά και το επίπεδο κινητικότητας από διερχόμενους πεζούς ή αυτοκίνητα. Το σύστημα θα υλοποιηθεί βασισμένο στις πλατφόρμες Arduino, ESP32 και Raspberry Pi. Η επικοινωνία μεταξύ των υποσυστημάτων θα επιτευχθεί με χρήση ασυρμάτων επικοινωνιών βασισμένο στις μονάδες ασύρματης συνδεσιμότητας Xbee και LoRa. Επίσης το σύστημα θα έχει τη δυνατότητα να αποθηκεύει ενέργεια που θα παράγεται από την ηλιακή ακτινοβολία και θα αποθηκεύεται σε συλλέκτες (μπαταρίες), ώστε να παρέχει υπηρεσία ανέπαφης φόρτισης συσκευών, όπως κινητά τηλέφωνα. Το σύστημα θα παρέχει δυνατότητα διαχείρισης και ελέγχου, της διαθεσιμότητας των θέσεων στάθμευσης αυτοκινήτων. Για την κατασκευή του ιστότοπου θα χρησιμοποιηθεί Apache Web Server, MariaDB, PHP, JavaScript και για την γραφική ανάλυση των δεδομένων Google Charts ενώ θα παρέχεται και η δυνατότητα διασύνδεσης με νέφος (cloud) μέσω των υπηρεσιών που παρέχονται από την πλατφόρμα ThingSpeak, για την παρακολούθηση και την ανάλυση δεδομένων. Για την κατασκευή του μοντέλου παρουσίασης (κολώνες φωτισμού) θα γίνει χρήση της τεχνολογίας τρισδιάστατης εκτύπωσης. Σκοπός του συστήματος θα είναι να παρέχει ένα πλαίσιο ολοκληρωμένων υπηρεσιών προς τους πολίτες της έξυπνης πόλης μέσα από τη βελτίωση και την αναβάθμιση των υποδομών της, με χρήση σύγχρονων τεχνολογιών του δικτύου των πραγμάτων (IoT). Μέσα από το σύνολο των υλοποιήσεων που θα πραγματεύεται η παρούσα πτυχιακή, στόχος είναι να αναδειχτούν τα οφέλη της ενσωμάτωσής αυτού του είδους των τεχνολογιών σε ένα περιβάλλον έξυπνης πόλης.

1.3 Οργάνωση πτυχιακής

Η οργάνωση της πτυχιακής εργασίας θα έχει την ακόλουθη δομή:

Στο δεύτερο κεφάλαιο θα παρουσιαστούν οι τεχνολογίες του διαδικτύου και συγκεκριμένα στην αρχιτεκτονική των διαδικτυακών εφαρμογών με μία σύντομη ανάλυση των συστατικών μερών της. Θα γίνει αναφορά στα πρωτόκολλα δικτύωσης και επικοινωνίας με έμφαση σε εκείνα που χρησιμοποιήθηκαν στην υλοποίηση του συστήματος. Στη συνέχεια θα γίνει σύντομη αναφορά στους διακομιστές εφαρμογών και των βάσεων δεδομένων καθώς και στην ανάπτυξη των διαδικτυακών εφαρμογών τόσο από την πλευρά του πελάτη όσο από την πλευρά του διακομιστή.

Στο τρίτο κεφάλαιο θα γίνει αναφορά στις πλατφόρμες και τα περιβάλλοντα ανάπτυξης που συνδυάστηκαν με σκοπό την υλοποίηση του συστήματος της παρούσας πτυχιακής εργασίας. Επίσης, θα παρουσιαστούν τα προγράμματα καθώς και οι διαδικτυακές πλατφόρμες που αξιοποιήθηκαν για την καλύτερη οπτική αναπαράσταση των δεδομένων του συστήματος που υλοποιήθηκε.

Στο τέταρτο κεφάλαιο θα παρουσιαστούν τα υλικά του συστήματος που αναπτύχθηκε στο πλαίσιο της παρούσας πτυχιακής εργασίας, καθώς επίσης και σύντομες αναφορές στην αρχιτεκτονική και τις εκδόσεις των πλατφόρμων υλικού ανάπτυξης IoT. Στη συνέχεια θα παρουσιαστούν τα υλικά που χρησιμοποιήθηκαν στην υλοποίηση των ασύρματων επικοινωνιών, καθώς επίσης και οι αισθητήρες και ο τρόπος λειτουργίας τους.

Στο πέμπτο κεφάλαιο θα αναλυθεί η υλοποίηση του συστήματος που περιλαμβάνει τα σενάρια αυτοματοποίησης, τη συνδεσμολογία των επιμέρους μερών που αποτελούν το σύστημα, όπως αισθητήρες και εξαρτήματα. Εν συνεχεία θα γίνει αναφορά στις βασικές βιβλιοθήκες που χρησιμοποιήθηκαν και θα παρουσιαστεί αναλυτικά ο κώδικας υλοποίησης κάθε σεναρίου.

Στο έκτο κεφάλαιο, θα παρουσιαστεί η υλοποίηση της διασύνδεσης. Αρχικά θα αναλυθεί η διασύνδεση του Raspberry Pi Pico με το διαδίκτυο μέσω WiFi καθώς και του Raspberry Pi μέσω του Ethernet. Εν συνεχεία, θα γίνει αναφορά των λειτουργιών του Raspberry Pi4 μέσω του Raspbian. Επίσης, θα αναλυθεί η διασύνδεση της βάσης δεδομένων, των γραφημάτων με την ιστοσελίδα και θα παρουσιαστεί η διασύνδεση της IOT πλατφόρμας (ThingSpeak) και η λειτουργία της διαδικτυακής εφαρμογής.



Τέλος, στο έβδομο και τελευταίο κεφάλαιο θα γίνει παρουσίαση των αποτελεσμάτων και συμπερασμάτων απόρροια της θεωρητικής ανάλυσης σε συνδυασμό με την υλοποίηση του συστήματος. Θα παρατεθούν τα σημαντικότερα προβλήματα κατά τη φάση της υλοποίησης του συστήματος αλλά και οι τρόποι αντιμετώπισης αυτών. Ολοκληρώνοντας θα υποβληθούν προτάσεις εξέλιξης αλλά και μελλοντικής βελτίωσης του συστήματος.

Τέλος, στα παραρτήματα θα παρουσιαστεί ο κώδικας που αναπτύχθηκε για τον προγραμματισμό του Arduino, ESP32, του Raspberry Pi, για την υλοποίηση της διαδικτυακής εφαρμογής και της βάσης δεδομένων.

2. Τεχνολογίες Διαδικτυακών Εφαρμογών

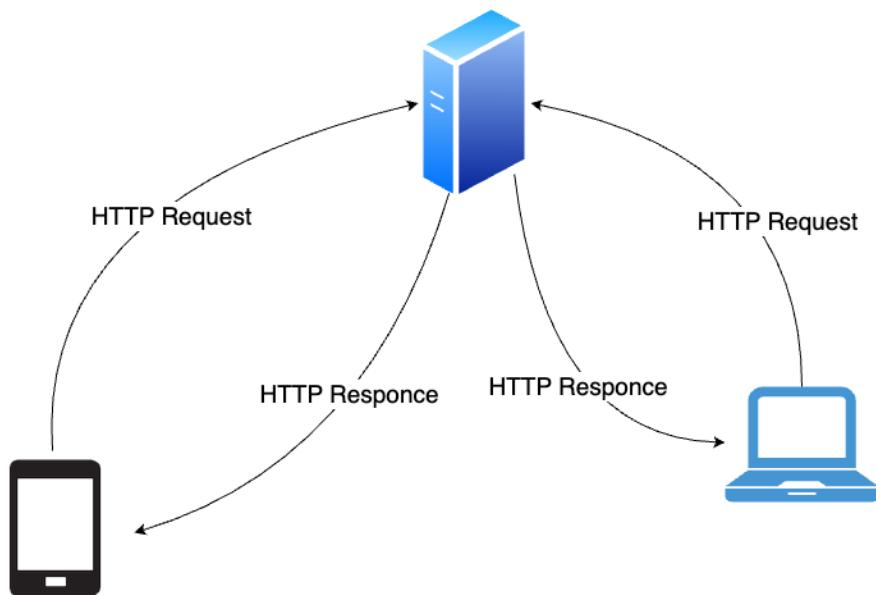
2.1. Τεχνολογίες Διαδικτύου

2.1.1. HTTP (HyperText Transfer Protocol)

Το Πρωτόκολλο Μεταφοράς Υπερκειμένου (Hypertext Transfer Protocol, HTTP), ένα θεμελιώδες πρωτόκολλο για τον Παγκόσμιο Ιστό, αποτελεί το κέντρο της διαδικτυακής επικοινωνίας. Περιγράφεται επίσημα στα πρότυπα RFC 1945 και RFC 2616 και λειτουργεί μέσω προγραμμάτων πελάτη και διακομιστή που ανταλλάσσουν μηνύματα HTTP. Η ιστοσελίδα, που συχνά αναφέρεται ως έγγραφο, αποτελείται από αντικείμενα, όπως αρχεία Hyper Text Markup Language (HTML), εικόνες ή Java applets, καθένα από τα οποία μπορεί να διευθυνθεί μοναδικά μέσω ενός Uniform Resource Locator (URL). Ειδικότερα, οι φυλλομετρητές ιστού, που λειτουργούν ως πελάτες, και οι διακομιστές ιστού (π.χ. Apache) διευκολύνουν από κοινό αυτή την αλληλεπίδραση.

Ο ρόλος του HTTP είναι καθοριστικός για τον τρόπο με τον οποίο οι πελάτες ιστού ζητούν σελίδες από τους διακομιστές και πώς οι διακομιστές απαντούν μεταφέροντας αυτές τις σελίδες πίσω στους πελάτες. Το πρωτόκολλο μεταφοράς είναι το Transmission Control Protocol (TCP), το οποίο εξασφαλίζει αξιόπιστη μεταφορά δεδομένων. Η διαδικασία αρχίζει με την έναρξη μιας σύνδεσης TCP από τον πελάτη με τον διακομιστή, ακολουθούμενη από την ανταλλαγή μηνυμάτων αίτησης (request) και απάντησης (response) HTTP. Η πολυεπίπεδη αρχιτεκτονική επιτρέπει στο HTTP να βασίζεται στο TCP για την ακεραιότητα των δεδομένων, χωρίς να ανησυχεί για τις λεπτομέρειες χαμηλότερου επιπέδου. (Kurose & Ross, 2017).

Τα μηνύματα HTTP διευκολύνουν την ανταλλαγή δεδομένων μεταξύ διακομιστών και πελατών, περιλαμβάνοντας δύο κύριους τύπους: αιτήματα που προέρχονται από τον πελάτη και προκαλούν ενέργειες του διακομιστή και απαντήσεις που δημιουργούνται από τον διακομιστή. Τόσο τα αιτήματα όσο και οι απαντήσεις HTTP έχουν παρόμοια δομή, αποτελούμενη από μια γραμμή έναρξης που υποδεικνύει την ενέργεια ή την κατάσταση (HTTP Method), ακολουθούμενη από προαιρετικές επικεφαλίδες (HTTP Headers) που παρέχουν πρόσθετες πληροφορίες.



Εικόνα 2-1 HTTP Request – HTTP Response.

Μια κενή γραμμή σηματοδοτεί την ολοκλήρωση της μετάδοσης των μετα-πληροφοριών. Προαιρετικά, ένα σώμα (HTTP body) μπορεί να περιέχει δεδομένα που σχετίζονται με την αίτηση ή το έγγραφο που συνδέεται με μια απάντηση. Η συλλογική γραμμή έναρξης και οι επικεφαλίδες αναφέρονται ως κεφαλή, ενώ το ωφέλιμο φορτίο αποτελεί το σώμα του μηνύματος HTTP (HTTP request methods - HTTP | MDN, n.d.). Στην εικόνα 2- 2 παρουσιάζεται ένα αίτημα HTTP (Post Request).

```
Method : POST
URL : /
Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests : 1
Connection : keep-alive
User-Agent : Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
Sec-Fetch-Site : cross-site
Host : localhost:3360
Accept-Encoding : gzip, deflate, br
Accept-Language : el-GR,el;q=0.9,en;q=0.8,it;q=0.7
Content-Length : 13
Sec-Fetch-Mode : navigate
=====
```

Εικόνα 2-2 Δομή HTTP Request

2.1.2 URL (Uniform Resource Locator)

Ο Ενιαίος Εντοπιστής Πόρων, URL (Uniform Resource Locator), χρησιμεύει ως μια τυποποιημένη διεύθυνση που προσδιορίζει τη θέση ενός πόρου στο διαδίκτυο. Συνήθως αποτελείται από διάφορα στοιχεία, συμπεριλαμβανομένου του πρωτοκόλλου (π.χ. "http" ή "https"), του ονόματος τομέα ή της διεύθυνσης IP του διακομιστή και της διαδρομής προς τον συγκεκριμένο πόρο στον εν λόγω διακομιστή. Οι διευθύνσεις URL είναι θεμελιώδους σημασίας για την πλοιήγηση στο διαδίκτυο, επιτρέποντας στους χρήστες να έχουν πρόσβαση σε ιστότοπους, έγγραφα ή περιεχόμενο πολυμέσων.

Στην ακόλουθη διεύθυνση URL, <http://www.example.org:56789/a/b/c.txt?t=win&s=chess#para5> η αρχή της, που βρίσκεται μετά το ":" και πριν την επόμενη κάθετο ("/"), περιλαμβάνει το πλήρως αναγνωρισμένο όνομα τομέα και έναν προαιρετικό αριθμό θύρας. Η διαδρομή ακολουθεί την αρχή και φτάνει μέχρι το ερωτηματικό (?) ή το τέλος της διεύθυνσης URL, που αντιπροσωπεύει τη θέση του αρχείου ή του πόρου. Το τμήμα μεταξύ του ερωτηματικού και του αριθμητικού σημείου χρησιμοποιείται παραδοσιακά για τη διαβίβαση όρων αναζήτησης σε έναν διακομιστή ιστού. Το τελευταίο προαιρετικό τμήμα, μετά το σύμβολο του αριθμού, χρησιμεύει ως αναγνωριστικό τμήματος για τους φυλλομετρητές για την τροποποίηση της εμφάνισης του εγγράφου (Jackson, 2007).

2.1.3 HTML (HyperText Markup Language)

Η Γλώσσα Σήμανσης Υπερκειμένου, HTML (HyperText Markup Language), αποτελεί εφεύρεση του Sir Timothy John Berners-Lee, ο οποίος είναι επίσης ο εφευρέτης του Παγκόσμιου Ιστού, του συστήματος URL καθώς και του πρωτοκόλλου HTTP. Η γλώσσα HTML αποτελεί τον πυρήνα της ανάπτυξης ιστοσελίδων, παρέχοντας μια τυποποιημένη γλώσσα σήμανσης απαραίτητη για τη δημιουργία εγγράφων στον Παγκόσμιο Ιστό. Η HTML λειτουργεί με τη χρήση ενός συνόλου ετικετών και χαρακτηριστικών για τη διάρθρωση και τη μορφοποίηση διαφόρων στοιχείων περιεχομένου, όπως κείμενο, εικόνες και πολυμέσα σε ιστοσελίδες. Η HTML διευκολύνει την παρουσίαση των πληροφοριών στις ιστοσελίδες.

Στην γλώσσα HTML οι οδηγίες μορφοποίησης διαχωρίζονται από το υπόλοιπο κείμενο με τη χρήση ειδικών χαρακτήρων. Συγκεκριμένα περικλείονται από τους ειδικούς χαρακτήρες "<" και ">". Η δομή ενός εγγράφου HTML αποτελείται από τα εξής τρία βασικά τμήματα:

1. Μία γραμμή που περιέχει την έκδοση και τον τύπο της HTML που ακολουθήθηκε κατά την συγγραφή του εγγράφου,
2. Την επικεφαλίδα (οριοθετείται με διπλή ετικέτα HEAD),
3. Το κυρίως σώμα (οριοθετείται με διπλή ετικέτα BODY), που περιέχει το κείμενο του εγγράφου. (Φουύσκας, 2003)

Για ιστορικούς λόγους αξίζει να αναφέρουμε ότι οι σημαντικότερες εκδόσεις της HTML ήταν οι HTML 2.0 (1996), HTML 3.2 (1997), HTML 4.0 (1997) και HTML 5.0 (2014).

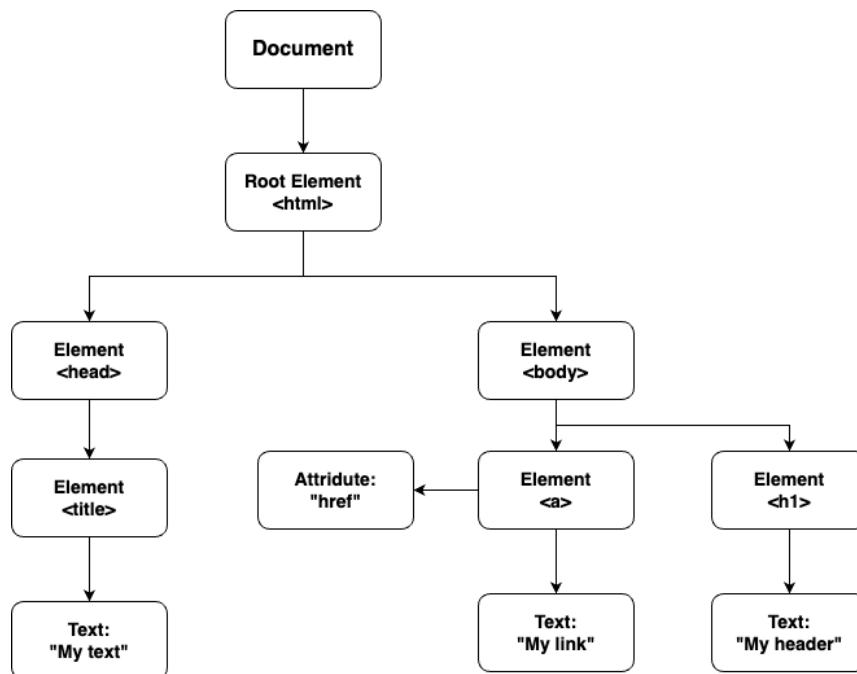
2.1.4 CSS (Cascading Style Sheets)

Το CSS (Cascading Style Sheets), είναι γλώσσα προγραμματισμού η οποία χρησιμοποιείται στην ανάπτυξη ιστοσελίδων και καθορίζει την παρουσίαση καθώς και τη διάταξη των εγγράφων HTML ή XML (Extensible Markup Language). Το CSS επιτρέπει στους προγραμματιστές να εφαρμόζουν στυλ, όπως γραμματοσειρές, χρώματα, διαστήματα και τοποθετήσεις, στα διάφορα στοιχεία μιας ιστοσελίδας. Διαχωρίζοντας το περιεχόμενο (HTML) από την παρουσίασή του (CSS), οι προγραμματιστές μπορούν να επιτύχουν μια καθαρή και αρθρωτή δομή, διευκολύνοντας τη διαχείριση και την ενημέρωση της εμφάνισης ενός ιστότοπου. Το CSS λειτουργεί μέσω ενός συνόλου κανόνων, όπου οι επιλογείς (selectors) στοχεύουν σε στοιχεία HTML και οι αντίστοιχες δηλώσεις (statements) τους καθορίζουν τις επιθυμητές ιδιότητες μορφοποίησης των συγκεκριμένων στοιχείων HTML. Η φύση του CSS αναφέρεται στην ιεραρχημένη εφαρμογή, επιτρέποντας στους προγραμματιστές να ορίζουν στυλ σε διαφορετικά επίπεδα, με κανόνες που κλιμακώνονται από το γενικότερο προς το ειδικότερο. Αυτός ο διαχωρισμός των προβλημάτων, μαζί με την ευελιξία που παρέχει, καθιστά το CSS αναπόσπαστο μέρος της δημιουργίας οπτικά ελκυστικών και ευέλικτων ιστοσελίδων (CSS: Cascading style sheets | MDN, n.d.).

2.1.5 DOM (Document Object Model)

Το Document Object Model (DOM) είναι μια προγραμματιστική διεπαφή (Application Programming Interface, API) που εξυπηρετεί την αναπαράσταση δομημένων εγγράφων, όπως η HTML και η XML, σε μια δενδροειδή δομή. Περιλαμβάνει αντικείμενα, τα οποία

συνδέονται με μία ιεραρχική δομή μεταξύ τους και αντιστοιχούν στα στοιχεία ενός εγγράφου, όπως επίσης και στις ιδιότητες και μεθόδους οι οποίες καθορίζουν το χειρισμό τους. Στο πλαίσιο της ανάπτυξης ιστοσελίδων, το DOM παρέχει ένα δυναμικό, ιεραρχικό μοντέλο μιας ιστοσελίδας, επιτρέποντας σε σενάρια (συνήθως γραμμένα σε JavaScript) να αλληλεπιδρούν και να χειρίζονται τη δομή, το περιεχόμενο και το στυλ του εγγράφου. Κάθε στοιχείο του εγγράφου, όπως παράγραφοι, επικεφαλίδες ή εικόνες, αναπαρίσταται ως κόμβος στο δέντρο DOM. Οι προγραμματιστές μπορούν να διασχίζουν αυτό το δέντρο, να έχουν πρόσβαση σε στοιχεία, να τροποποιούν τα χαρακτηριστικά τους και να ενημερώνουν το περιεχόμενο δυναμικά. Το DOM χρησιμεύει ουσιαστικά ως μεσάζων μεταξύ της δομής του εγγράφου και της γλώσσας σεναρίων, επιτρέποντας τη δημιουργία διαδραστικών εφαρμογών, διευκολύνοντας τον χειρισμό των στοιχείων της σελίδας σε πραγματικό χρόνο χωρίς να απαιτείται ανανέωση της σελίδας. Στην εικόνα 2-3 παρουσιάζεται σε δενδροειδής μορφή κόμβων ένα τυπικό DOM.



Εικόνα 2-3 Βασική ροή κόμβων δένδρου HTML

2.1.6 JSON (JavaScript Object Notation)

Το JSON προκύπτει από τα αρχικά των λέξεων JavaScript Object Notation. Αποτελεί μια ελαφριά μορφή ανταλλαγής δεδομένων που έχει γίνει πρότυπο για την αναπαράσταση και την ανταλλαγή δεδομένων στην ανάπτυξη ιστοσελίδων. Είναι μορφότυπο βασισμένο σε

κείμενο και χρησιμεύει ως μια αναγνώσιμη από τον άνθρωπο μορφή, καθιστώντας εύκολη τόσο τη δημιουργία όσο και την ανάλυση δεδομένων. Το JSON είναι γλωσσικά αδιάφορο, πράγμα που σημαίνει ότι μπορεί να χρησιμοποιηθεί με διάφορες γλώσσες προγραμματισμού. Αποτελείται κυρίως από ζεύγη κλειδιών-τιμών και υποστηρίζει πίνακες και εμφωλευμένες δομές, επιτρέποντας την ευέλικτη και ιεραρχική αναπαράσταση δεδομένων. Το JSON χρησιμοποιείται συνήθως σε διαδικτυακά API για τη μετάδοση δεδομένων μεταξύ διακομιστών και πελατών και έχει αποκτήσει ευρεία αποδοχή λόγω της απλότητας και της αποτελεσματικότητάς του. Η σύνταξή του προέρχεται από τη JavaScript, αλλά υποστηρίζεται ευρέως και σε άλλες γλώσσες προγραμματισμού. Η αναγνωσιμότητα και η ευελιξία του JSON το καθιστούν δημοφιλή επιλογή για τη διαμόρφωση ρυθμίσεων, την αποθήκευση και τη μετάδοση δομημένων δεδομένων και τη διευκόλυνση της απρόσκοπτης επικοινωνίας μεταξύ διαφορετικών στοιχείων εφαρμογών ιστού (Crockford, 2024). Στην εικόνα 2-4 παρουσιάζεται ένα αντικείμενο τύπου Json.

```
{
    "mq7": 43,
    "mq135": 7.573481,
    "uv": 0.02,
    "temperature": 20.6,
    "humidity": 60,
    "ldr": 624,
    "flame": 0,
    "timestamp": "04/01/2024 16:07",
    "gas": 0,
    "poleId": "arduinoXbee"
}
```

Εικόνα 2-4 Παράδειγμα κώδικα JSON

2.1.7 XML (Extensible Markup Language)

Η Επεκτάσιμη Γλώσσα Σήμανσης XML (Extensible Markup Language) είναι μια ευέλικτη γλώσσα σήμανσης που έχει σχεδιαστεί για την αποθήκευση και τη μεταφορά δεδομένων σε μορφή που είναι τόσο αναγνώσιμη από τον άνθρωπο όσο και από τη μηχανή. Η XML χρησιμοποιεί ετικέτες για τον ορισμό στοιχείων και χαρακτηριστικών, επιτρέποντας στους χρήστες να δημιουργούν προσαρμοσμένες δομές δεδομένων για την κάλυψη των αναγκών τους. Χρησιμοποιείται ευρέως για την αναπαράσταση δομημένων δεδομένων με τυποποιημένο και ανεξάρτητο από την πλατφόρμα τρόπο, καθιστώντας την θεμελιώδη



τεχνολογία για την ανταλλαγή πληροφοριών και την αποθήκευση δεδομένων (W3, n.d.).

Στην εικόνα 2-5 παρουσιάζεται παράδειγμα αρχείου XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                               modelVersion">4.0.0</modelVersion>

<groupId>org.example</groupId>
<artifactId>httpServer</artifactId>
<version>1.0-SNAPSHOT</version>
```

Εικόνα 2-5 Παράδειγμα κώδικα XML

2.1.8 AJAX (Asynchronous JavaScript and XML)

Το AJAX είναι το ακρωνύμιο από τα αρχικά των λέξεων Asynchronous JavaScript and XML. Αποτελεί μια τεχνική ανάπτυξης ιστοσελίδων που επιτρέπει τη δημιουργία δυναμικών και ευέλικτων διεπαφών χρήστη. Συγκεκριμένα επιτρέπει στις ιστοσελίδες να στέλνουν και να ανακτούν δεδομένα ασύγχρονα από έναν διακομιστή ιστού, χωρίς να χρειάζεται να επαναφορτώνεται ολόκληρη η σελίδα. Αυτή η ασύγχρονη επικοινωνία διευκολύνεται μέσω ενός συνδυασμού JavaScript, HTML και CSS. Το AJAX συμβάλλει καθοριστικά στη βελτίωση της εμπειρίας του χρήστη, επιτρέποντας την απρόσκοπτη ενημέρωση συγκεκριμένων τμημάτων μιας ιστοσελίδας, με αποτέλεσμα ταχύτερες και πιο διαδραστικές εφαρμογές. Ενώ ο όρος υποδηλώνει τη χρήση XML, οι σύγχρονες υλοποιήσεις AJAX χρησιμοποιούν συχνά JSON για την ανταλλαγή δεδομένων λόγω της απλότητας και της ευκολίας χρήσης του. Το AJAX χρησιμοποιείται ευρέως στην ανάπτυξη διαδραστικών εφαρμογών ιστού, επιτρέποντας ενημερώσεις σε πραγματικό χρόνο, υποβολές φορμών και δυναμική φόρτωση περιεχομένου, συμβάλλοντας τελικά σε μια πιο ελκυστική και αποτελεσματική εμπειρία χρήστη (Zakas, 2012).

2.2 Πρωτόκολλα Δικτύωσης και Επικοινωνίας

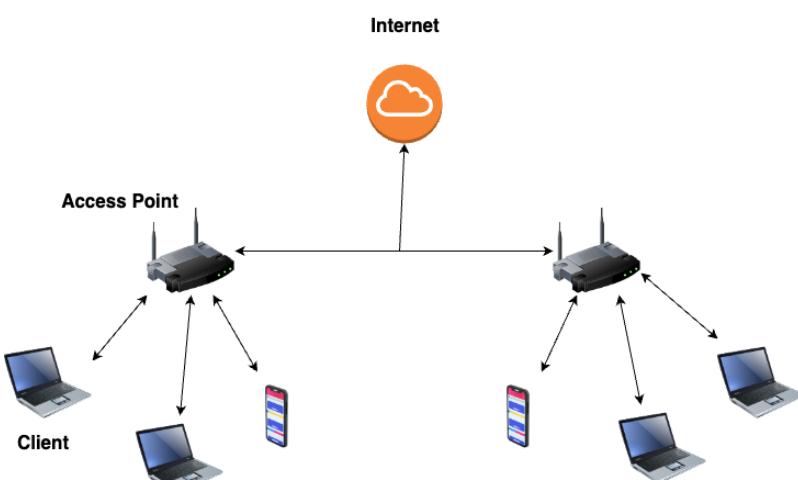
2.2.1 Το πρότυπο IEEE 802.11 (Wi - Fi)

Τα ασύρματα τοπικά δίκτυα LAN (Local Area Network), όπως το Wi-Fi, που προέρχεται από την οικογένεια προτύπων IEEE 802.11, αποτελεί ένα ευρέως αποδεκτό σύνολο

πρωτοκόλλων ασύρματου δικτύου που διευκολύνει τη δικτύωση τοπικά και την πρόσβαση στο διαδίκτυο. Τα δίκτυα Wi-Fi, έχουν γίνει ζωτικής σημασίας τεχνολογία δικτύου καθώς επιτρέπουν σε πληθώρα ψηφιακών συσκευών να ανταλλάσσουν δεδομένα μέσω ραδιοκυμάτων, σε διάφορα περιβάλλοντα, όπως χώρους εργασίας, σπίτια, εκπαιδευτικά ιδρύματα, καφετέριες, αεροδρόμια και δημόσιους χώρους.

Μεταξύ των διαφόρων τεχνολογιών που αναπτύχθηκαν τη δεκαετία του 1990, το IEEE 802.11, κοινώς γνωστό ως WiFi, έχει αναδειχθεί ως η κυρίαρχη κατηγορία στα ασύρματα δίκτυα. Η οικογένεια IEEE 802.11 περιλαμβάνει διάφορα πρότυπα, όπως το 802.11g και το 802.11ac, τα οποία μοιράζονται κοινά χαρακτηριστικά, όπως η χρήση CSMA/CA (Carrier-sense multiple access with collision detection), η συνεπής δομή πλαισίου, η προσαρμοστικότητα στους ρυθμούς μετάδοσης για εκτεταμένη εμβέλεια και, κυρίως, η προς τα πίσω συμβατότητα μεταξύ των διαφόρων προτύπων, εξασφαλίζοντας τη διαλειτουργικότητα μεταξύ παλαιότερων και νεότερων προϊόντων 802.11 (Kurose & Ross, 2017).

Τα διάφορα πρότυπα IEEE 802.11 καθορίζουν τις εκδόσεις Wi-Fi, καθένα από τα οποία καθορίζει τις ραδιοτεχνολογίες, τις ζώνες, τις εμβέλειες και τις ταχύτητες. Συνήθως χρησιμοποιούνται οι ραδιοζώνες 2,4 GHz και 5 GHz. Η κάλυψη των σημείων πρόσβασης κυμαίνεται από περίπου 20 μέτρα σε εσωτερικούς χώρους έως δυνητικά 150 μέτρα σε εξωτερικούς χώρους και οι ταχύτητες έχουν εξελιχθεί, φτάνοντας σε ορισμένες περιπτώσεις έως και τα 9,6 Gbit/s (Wikipedia, 2023). Στην εικόνα 2-6 παρουσιάζεται μια τυπική αρχιτεκτονική WLAN IEEE 802.11.



Εικόνα 2-6 Τυπική Αρχιτεκτονική WLAN IEEE 802.11

2.2.2 Το πρότυπο IEEE 802.15 (Zigbee)

Το Zigbee, βασισμένο στο πρότυπο IEEE 802.15.4, είναι ένα προσωπικό δίκτυο περιοχής σχεδιασμένο για εφαρμογές χαμηλότερης ισχύος και χαμηλότερου ρυθμού δεδομένων σε σύγκριση με τεχνολογίες υψηλής ταχύτητας όπως το Bluetooth. Το Zigbee είναι κατάλληλο για απλές συσκευές χαμηλής κατανάλωσης ενέργειας, όπως οικιακοί αισθητήρες, συσκευές ασφαλείας και διακόπτες. Λειτουργεί σε ρυθμούς καναλιού 20, 40, 100 και 250 Kbps, προσφέροντας ευελιξία ανάλογα με τις απαιτήσεις της εφαρμογής. Τα δίκτυα Zigbee αποτελούνται από δύο τύπους κόμβων: «συσκευές μειωμένης λειτουργίας» (slaves) και «συσκευές πλήρους λειτουργίας» (masters) που ελέγχουν πολλαπλούς slaves ή σχηματίζουν δίκτυα πλέγματος (mesh network). Τα δίκτυα Zigbee μπορούν να διαμορφωθούν με διάφορους τρόπους, με μια δομή που χωρίζει το χρόνο σε ενεργές και ανενεργές περιόδους, επιτρέποντας στις συσκευές να εξοικονομούν ενέργεια (Kurose & Ross, 2017).

2.2.3 Το πρότυπο IEEE 802.3 (Ethernet)

Το Ethernet έχει γίνει η κυρίαρχη τεχνολογία ενσύρματων τοπικών δικτύων, από την εφεύρεσή του στα μέσα της δεκαετίας του 1970, μέχρι και σήμερα. Παρά την αντιμετώπιση του ανταγωνισμού από άλλες τεχνολογίες LAN στη δεκαετία του 1980 και στις αρχές της δεκαετίας του 1990, η απλότητα και η αποδοτικότητα του Ethernet συνέβαλαν στην ευρεία υιοθέτησή του. Η συνεχής εξέλιξή του καθώς και η εισαγωγή του switched Ethernet εδραίωσαν περαιτέρω τη θέση του.

Το αρχικό τοπικό δίκτυο Ethernet, που εφευρέθηκε από τους Bob Metcalfe και David Boggs στα μέσα της δεκαετίας του 1970, χρησιμοποιούσε τοπολογία ομοαξονικού διαύλου. Με την πάροδο του χρόνου, οι τοπολογίες διαύλου παρέμειναν, αλλά τελικά έδωσαν τη θέση τους σε τοπολογίες αστέρα με κόμβους στα τέλη της δεκαετίας του 1990. Στις αρχές της δεκαετίας του 2000, το switched Ethernet με τοπολογία αστέρα και μεταγωγείς αντικατέστησε τους διανομείς, προσφέροντας δυνατότητες μεταγωγής πακέτων χωρίς συγκρούσεις. Η δομή του πλαισίου Ethernet περιλαμβάνει βασικά στοιχεία όπως το πεδίο δεδομένων, τις διευθύνσεις προορισμού και πηγής, το πεδίο τύπου και το CRC (Cyclic Redundancy Check) για την ανίχνευση σφαλμάτων.

Οι τεχνολογίες Ethernet έχουν εξελιχθεί με διάφορα πρότυπα, που συμβολίζονται με ακρωνύμια όπως 10BASE-T, 100BASE-T και 1000BASE-LX, τα οποία έχουν τυποποιηθεί

από την ομάδα εργασίας IEEE 802.3 CSMA/CD. Αυτά τα πρότυπα αντιπροσωπεύουν διαφορετικές ταχύτητες, χαρακτηριστικά βασικής ζώνης και φυσικά μέσα. Το Gigabit Ethernet, που εισήχθη ως επέκταση του 10/100 Mbps Ethernet, προσφέρει υψηλές ταχύτητες διατηρώντας την συμβατότητα με τις παλαιότερες. Παρά τις σημαντικές προόδους και αλλαγές, η μόνιμη σταθερά του Ethernet είναι η μορφή πλαισίου του, αναδεικνύοντας την προσαρμοστικότητα και τη συνεχή σημασία του επί 30 χρόνια (Kurose & Ross, 2017).

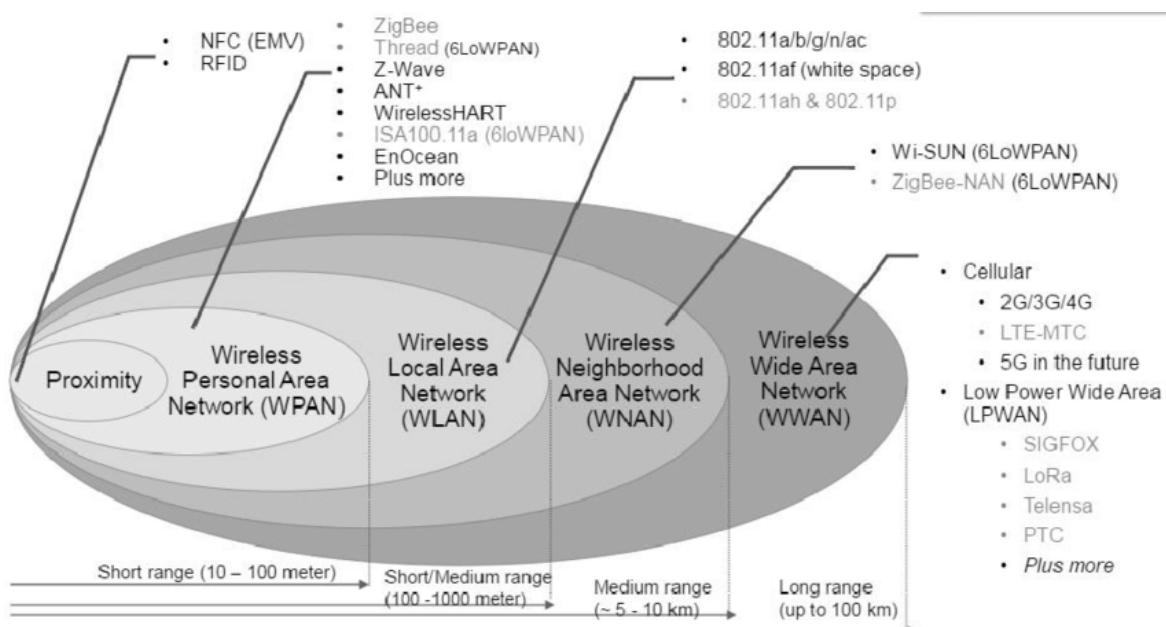
2.2.4 LoRaWAN (Long Range Wide Area Network)

Το LoRaWAN, ή Long Range Wide Area Network, είναι μια τεχνολογία ασύρματης επικοινωνίας που έχει σχεδιαστεί για επικοινωνία μεγάλης εμβέλειας με χαμηλή κατανάλωση ενέργειας, καθιστώντας την κατάλληλη για εφαρμογές IoT. Η τεχνολογία βασίζεται στο φυσικό επίπεδο LoRa (Long Range), το οποίο λειτουργεί στις ζώνες συχνοτήτων κάτω των gigahertz, χρησιμοποιώντας συνήθως μη αδειοδοτημένο φάσμα, γεγονός που επιτρέπει την οικονομικά αποδοτική και ευρεία ανάπτυξη.

Ένα από τα βασικά χαρακτηριστικά του LoRaWAN είναι η ικανότητά του να παρέχει εκτεταμένη κάλυψη, που φτάνει έως και αρκετά χιλιόμετρα σε αστικά περιβάλλοντα και ακόμη περισσότερο σε αγροτικές περιοχές. Αυτή η εκτεταμένη εμβέλεια καθιστά το LoRaWAN κατάλληλο για εφαρμογές που απαιτούν συνδεσιμότητα σε μεγάλες αποστάσεις, όπως η έξυπνη γεωργία, οι έξυπνες πόλεις και το βιομηχανικό IoT. Η τεχνολογία επιτυγχάνει αυτή την εμβέλεια με τη χρήση χαμηλών ρυθμών δεδομένων και βελτιστοποιημένων ραδιοπαραμέτρων, εξασφαλίζοντας αποτελεσματική χρήση του διαθέσιμου φάσματος.

Τα δίκτυα LoRaWAN αναπτύσσονται συνήθως σε τοπολογία αστέρα-αστέρα, όπου οι τελικές συσκευές επικοινωνούν με πύλες που με τη σειρά τους προωθούν τα δεδομένα σε έναν κεντρικό διακομιστή δικτύου. Αυτή η αποκεντρωμένη αρχιτεκτονική επιτρέπει την επεκτασιμότητα, την εύκολη ανάπτυξη και την αποτελεσματική διαχείριση των συσκευών. Επιπλέον, το LoRaWAN υποστηρίζει αμφίδρομη επικοινωνία, επιτρέποντας όχι μόνο τη μετάδοση δεδομένων αισθητήρων από τις συσκευές στο δίκτυο, αλλά και τη δυνατότητα αποστολής εντολών ή ενημερώσεων από το δίκτυο στις συσκευές.

Όπως συμβαίνει με κάθε τεχνολογία, είναι σημαντικό να λαμβάνονται υπόψη διάφοροι παράγοντες, όπως η αρχιτεκτονική του δικτύου, οι μηχανισμοί ασφαλείας και η επεκτασιμότητα. Η υλοποίηση του LoRaWAN μπορεί να περιλαμβάνει τη διαμόρφωση και διαχείριση συσκευών, πυλών και διακομιστών δικτύου, καθώς και τη διασφάλιση ασφαλούς επικοινωνίας μεταξύ τους (LoRa, n.d.). Στην εικόνα παρουσιάζονται οι αποστάσεις που καλύπτουν τα ασύρματα πρωτόκολλα επικοινωνία που αναλύθηκαν παραπάνω.



Εικόνα 2-7 Εμβέλεια κάλυψης Ασύρματων επικοινωνιών
(Wireless technologies for IoT | Researchgate, 2024)

2.3 Διακομιστές εφαρμογών και βάσεων δεδομένων (Application Server / DataBase Server)

2.3.1 JAVA

Η JAVA, μια ευέλικτη και ισχυρή γλώσσα προγραμματισμού, έχει διαδραματίσει καθοριστικό ρόλο στην εξέλιξη της ανάπτυξης λογισμικού από την εισαγωγή της στα μέσα της δεκαετίας του 1990. Η JAVA αναπτύχθηκε αρχικά από τη Sun Microsystems και σχεδιάστηκε με στόχο την παροχή μιας γλώσσας ανεξάρτητης από την πλατφόρμα που θα μπορούσε να τρέχει σε διάφορες συσκευές. Αυτή η αρχή "Write Once, Run Anywhere" έγινε ένα από τα καθοριστικά χαρακτηριστικά της JAVA, καθιστώντας την μια δημοφιλή επιλογή για την ανάπτυξη σε πολλαπλές πλατφόρμες.

Ιστορικά, η JAVA προέκυψε ως απάντηση στις προκλήσεις που έθεταν οι παραδοσιακές γλώσσες προγραμματισμού. Με την άνοδο του διαδικτύου, υπήρχε μια αυξανόμενη ανάγκη για μια γλώσσα που θα μπορούσε να διευκολύνει την ανάπτυξη δυναμικού και διαδραστικού περιεχομένου. Η JAVA αντιμετώπισε αυτή την ανάγκη εισάγοντας ένα μοντέλο μεταγλώττισης bytecode, επιτρέποντας τη δημιουργία applets που θα μπορούσαν να εκτελεστούν μέσα σε φυλλομετρητές ιστού. Αυτό σηματοδότησε μια σημαντική αλλαγή στον τρόπο ανάπτυξης λογισμικού για το διαδίκτυο.

Ένα από τα βασικά πλεονεκτήματα της JAVA είναι η ανεξαρτησία της από την πλατφόρμα, η οποία επιτυγχάνεται μέσω της JAVA Virtual Machine (JVM). Ο κώδικας που γράφεται σε JAVA μεταγλωττίζεται σε bytecode, ο οποίος μπορεί να εκτελεστεί σε οποιαδήποτε συσκευή με συμβατή JVM, ανεξάρτητα από το υποκείμενο υλικό και το λειτουργικό σύστημα.

Επιπλέον, η JAVA διαθέτει ένα πλούσιο οικοσύστημα βιβλιοθηκών, πλαισίων και εργαλείων που ενισχύουν την παραγωγικότητα των προγραμματιστών. Η μεγάλη έμφαση της γλώσσας στον αντικειμενοστραφή προγραμματισμό και η ολοκληρωμένη βιβλιοθήκη της συμβάλλουν στη αποτελεσματική δημιουργία κώδικα. Επιπλέον, η αυτόματη διαχείριση μνήμης της JAVA μέσω της συλλογής απορριμμάτων (java garbage collector), η οποία λειτουργεί στο υπόβαθρο (background) κατά την εκτέλεση ενός προγράμματος, απλοποιεί το χειρισμό της, προσφέροντας στους προγραμματιστές μεγαλύτερη ευελιξία.

Ωστόσο, καμία τεχνολογία δεν παραμένει εκτός προκλήσεων. Η JAVA έχει αντιμετωπίσει κριτική για τη πολυλογία της, με ορισμένους προγραμματιστές να θεωρούν τη σύνταξη πιο πολυλεκτική σε σύγκριση με άλλες σύγχρονες γλώσσες. Επιπλέον, ο χρόνος εκκίνησης των εφαρμογών JAVA αποτελούσε ιστορικά πρόβλημα, ιδίως για εφαρμογές μικρότερης κλίμακας ή περιβάλλοντα όπου η ταχεία εκτέλεση είναι κρίσιμη.

Η JAVA έχει επικρατήσει σε ένα ευρύ φάσμα συσκευών. Η ευελιξία της, της επιτρέπει, την ανάπτυξή της, από εταιρικούς διακομιστές έως κινητές συσκευές και έχει συμβάλει στην εκτεταμένη υιοθέτησή της (Deitel, 2015).

2.3.2 Java Persistence API (JPA)

Το JPA (JAVA Persistence API) είναι βασικό χαρακτηριστικό της πλατφόρμας JAVA EE (Enterprise Edition). Παρουσιάζει ένα τυποποιημένο πλαίσιο (framework) για την χρήση

ORM (Object Relational Mapping). Εισήγθη ως μέρος των προδιαγραφών της JAVA EE 5, με σκοπό να απλοποιήσει τον τρόπο παραμονής των αντικειμένων της JAVA σε σχεσιακές βάσεις δεδομένων καθώς και την ανάκτησή τους. Αυτή η προσέγγιση διευκολύνει τους προγραμματιστές στη χρήση ερωτημάτων SQL όπως επίσης και την αλληλεπίδραση με τις βάσεις δεδομένων, προσφέροντας μια πιο αντικειμενοστραφή προσέγγιση στη διαχείριση τους.

Το JPA επιτρέπει την απρόσκοπτη επικοινωνία μεταξύ της εφαρμογής και της βάσης δεδομένων. Επιτρέπει στους προγραμματιστές να επικεντρωθούν στη λειτουργικότητα της εφαρμογής τους αντί να διαχειρίζονται δύσκολες εντολές SQL, όπως επίσης να μπορούν να δημιουργήσουν μια σαφή και εύκολη αντιστοίχιση μεταξύ των αντικειμένων JAVA και των πινάκων της βάσης δεδομένων.

Ένα από τα πλεονεκτήματα του JPA η ενσωμάτωση του Entity Manager, μιας διεπαφής που διαχειρίζεται τον κύκλο ζωής των οντοτήτων και τις αλληλεπιδράσεις τους με τη βάση δεδομένων. Επίσης το JPA αντιμετωπίζει την «αναντιστοιχία» μεταξύ της αντικειμενοστρέφειας της JAVA και της σχεσιακής φύσης των βάσεων δεδομένων. Αυτό έχει ως αποτέλεσμα καθαρότερο, πιο συντηρήσιμο κώδικα και μειώνει την ανάγκη για κώδικα SQL.

Συνοψίζοντας, το JPA χρησιμεύει ως ισχυρό εργαλείο για την απλοποίηση των αλληλεπιδράσεων με τη βάση δεδομένων στις εφαρμογές JAVA. Παρουσιάζει μια τυποποιημένη τεχνική για την αντιστοίχιση αντικειμένων-σχεσιακών στοιχείων που στοχεύει στην αφαίρεση της πολυπλοκότητας των λειτουργιών της βάσης δεδομένων, δίνοντας στους προγραμματιστές τη δυνατότητα να εστιάσουν την προσοχή τους στην ανάπτυξη ανθεκτικών, αποδοτικών και συντηρήσιμων πακέτων (Deitel, 2015).

2.3.3 Συστήματα Διαχείρισης Βάσεων Δεδομένων (DBMS)

Ένα Σύστημα Διαχείρισης Βάσεων Δεδομένων (DBMS - Database Management System) είναι μια συνίτια λογισμικού που διευκολύνει τη δημιουργία, οργάνωση, ανάκτηση και διαχείριση δεδομένων σε μια βάση δεδομένων. Ο θεμελιώδης σκοπός ενός DBMS είναι να παρέχει έναν αποτελεσματικό και δομημένο μηχανισμό για τους χρήστες και τις εφαρμογές ώστε να αλληλεπιδρούν με μεγάλο όγκο δεδομένων με συστηματικό τρόπο. Λειτουργεί ως

ενδιάμεσος κρίκος μεταξύ της βάσης δεδομένων και των τελικών χρηστών ή εφαρμογών, εξασφαλίζοντας απρόσκοπη και ασφαλή πρόσβαση στις πληροφορίες.

Το DBMS αποτελεί βασικό συστατικό των σύγχρονων συστημάτων πληροφοριών, παρέχοντας μια συγκεντρωτική και δομημένη προσέγγιση για την αποθήκευση και ανάκτηση δεδομένων. Προσφέρει χαρακτηριστικά όπως η ακεραιότητα των δεδομένων, ο έλεγχος ταυτόχρονης χρήσης και τα μέτρα ασφαλείας για να διασφαλίζει τη συνέπεια και την αξιοπιστία των αποθηκευμένων πληροφοριών.

Ένα βασικό χαρακτηριστικό των DBMS είναι η δυνατότητα υποστήριξης πολλαπλών ταυτόχρονων χρηστών και εφαρμογών, επιτρέποντάς τους να έχουν πρόσβαση και να τροποποιούν δεδομένα ταυτόχρονα, διασφαλίζοντας παράλληλα τη συνέπεια των δεδομένων. Υπάρχουν διάφοροι τύποι DBMS, συμπεριλαμβανομένων των σχεσιακών, των αντικειμενοστραφών και των NoSQL βάσεων δεδομένων (όπως MongoDB), καθένας από τους οποίους είναι προσαρμοσμένος σε διαφορετικές απαιτήσεις αποθήκευσης και ανάκτησης δεδομένων.

Τα Συστήματα Διαχείρισης Βάσεων Δεδομένων διαδραματίζουν καθοριστικό ρόλο στη σύγχρονη διαχείριση πληροφοριών, προσφέροντας μια δομημένη και αποτελεσματική προσέγγιση στην οργάνωση, ανάκτηση και συντήρηση δεδομένων (Connolly, 2014).

2.3.4 MySQL

Η MySQL και συγκεκριμένα η MariaDB είναι συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS – Relational Database Management Systems) ανοικτού κώδικα που έχουν αποκτήσει ευρεία δημοτικότητα λόγω των επιδόσεων αλλά και της ευκολίας στην χρήση τους. Η MySQL, η οποία αναπτύχθηκε αρχικά από μια σουηδική εταιρεία, αποτελεί βασικό στοιχείο της κοινότητας των βάσεων δεδομένων εδώ και δεκαετίες. Η MariaDB αποτελεί μια παραλαγή της MySQL που δημιουργήθηκε από τους αρχικούς προγραμματιστές της MySQL πριν την εξαγορά της MySQL από την Oracle Corporation.

Η MariaDB μπορεί να χρησιμοποιηθεί σε διάφορα έργα και εφαρμογές προσφέροντας μια κλιμακούμενη και ευέλικτη αρχιτεκτονική η οποία έχει τη δυνατότητα να χειριστεί έργα τόσο μικρής όσο και μεγάλης κλίμακας. Η γλώσσα ερωτημάτων της βασίζεται στην SQL παρέχοντας εξαιρετικές επιδόσεις και αποτελεσματικό indexing, διασφαλίζοντας ότι οι

λειτουργίες ανάκτησης και επεξεργασίας δεδομένων βελτιστοποιούνται τόσο σε ταχύτητα όσο και σε αποδοτικότητα. Επιπλέον, εξασφαλίζει την ακεραιότητα των δεδομένων ακόμη και σε περίπτωση βλάβης του συστήματος. Η φιλική προς το χρήστη φύση της, την καθιστά προσιτή σε προγραμματιστές όλων των επιπέδων. Στην παρούσα εργασία θα γίνει χρήση του RDBMS της MariaDB που αποτελεί ένα ισχυρό σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων ανοιχτού κώδικα (MySQL, n.d.).

2.4 Ανάπτυξη Διαδικτυακών Εφαρμογών

2.4.1 Πλευρά Πελάτη (Client Side)

Η πλευρά του πελάτη και η πλευρά του διακομιστή αντιπροσωπεύουν διακριτά στοιχεία σε μια αρχιτεκτονική πελάτη-εξυπηρετητή (client - server), όπου οι υπολογιστικές εργασίες κατανέμονται μεταξύ του υπολογιστή του χρήστη (client) και ενός κεντρικού διακομιστή (server).

Η πλευρά πελάτη αναφέρεται στο μηχάνημα ή τη συσκευή του χρήστη, συνήθως έναν προσωπικό υπολογιστή, ένα tablet ή ένα smartphone, όπου εκτελείται η client εφαρμογή. Αυτή η εφαρμογή είναι υπεύθυνη για την παρουσίαση της διεπαφής χρήστη, τη συλλογή των δεδομένων που εισάγει ο χρήστης και την αποστολή αιτημάτων στον διακομιστή. Οι εφαρμογές στην πλευρά του πελάτη αλληλεπιδρούν απευθείας με τον χρήστη και ξεκινούν την επικοινωνία με τον διακομιστή για την ανάκτηση ή την αποστολή δεδομένων. Η επεξεργασία από την πλευρά του πελάτη δίνει έμφαση στην εμπειρία του χρήστη, στη γραφική διεπαφή και στις τοπικές αλληλεπιδράσεις (Deitel, 2015).

Στο διαδίκτυο, η πλευρά πελάτη συχνά συνδέεται με αυτό που οι χρήστες βιώνουν άμεσα από το πρόγραμμα περιήγησης στο διαδίκτυο. Τα προγράμματα, όπως το Chrome, το Firefox ή το Safari, λειτουργούν ως πελάτες, ερμηνεύοντας και αποδίδοντας HTML, CSS και JavaScript για να παρουσιάσουν μια φιλική προς το χρήστη εμπειρία. Η πλευρά πελάτη είναι υπεύθυνη για τις αλληλεπιδράσεις του χρήστη, τις επικυρώσεις φορμών και κάποιο βαθμό επεξεργασίας δεδομένων πριν από την επικοινωνία με τον διακομιστή.



2.4.2 Πλευρά Διακομιστή (Server Side)

Η πλευρά διακομιστή είναι το κεντρικό στοιχείο που διαχειρίζεται και επεξεργάζεται τα αιτήματα από πολλούς πελάτες. Πρόκειται για ένα μηχάνημα ή ένα κατανεμημένο σύστημα (cluster) που είναι υπεύθυνο για την διαχείριση των αιτημάτων από τους πελάτες, την πρόσβαση σε βάσεις δεδομένων και τη διαχείριση των πόρων. Η επεξεργασία από την πλευρά του διακομιστή επικεντρώνεται στον κεντρικό έλεγχο, την ασφάλεια και την αποτελεσματική διαχείριση των πόρων.

Η κατανόηση της αρχιτεκτονικής πελάτη-εξυπηρετητή είναι απαραίτητη για το σχεδιασμό και την υλοποίηση κατανεμημένων συστημάτων όπου ο πελάτης και ο εξυπηρετητής συνεργάζονται για την απρόσκοπτη και αποτελεσματική παροχή εφαρμογών (Deitel, 2015).

Η πλευρά του διακομιστή στην τεχνολογία ιστού είναι εκεί όπου βρίσκονται η λογική της εφαρμογής, οι αλληλεπιδράσεις με τη βάση δεδομένων και οι επιχειρηματικές διαδικασίες (business procedures). Από την πλευρά του διακομιστή, διαχειρίζονται όλα τα αιτήματα που αποστέλλονται από τον πελάτη. Εκτελείται επεξεργασία δεδομένων, εκτελείται επιχειρησιακή λογική (business logic) και αλληλεπίδραση με βάσεις δεδομένων για την ανάκτηση ή την αποθήκευση πληροφοριών. Η ανάπτυξη από την πλευρά του διακομιστή διασφαλίζει την ακεραιότητα των δεδομένων, την ασφάλεια και την εκτέλεση κρίσιμων εργασιών.

Στον παγκόσμιο ιστό, ο πελάτης και ο διακομιστής επικοινωνούν μέσω του HTTP ή της ασφαλούς παραλλαγής του, του HTTPS (Hypertext Transfer Protocol Secure). Ο πελάτης στέλνει αιτήματα στον διακομιστή και ο διακομιστής απαντά με τα ζητούμενα δεδομένα ή εκτελεί τις απαραίτητες ενέργειες. Αυτή η αλληλεπίδραση αποτελεί τη ραχοκοκαλιά των δυναμικών και διαδραστικών εφαρμογών ιστού, επιτρέποντας ενημερώσεις σε πραγματικό χρόνο, απρόσκοπτη εμπειρία χρήστη και αποτελεσματική επεξεργασία δεδομένων.

2.4.3 JavaScript

Η JavaScript είναι μια ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού που παίζει καθοριστικό ρόλο στην ανάπτυξη ιστοσελίδων. Αρχικά δημιουργήθηκε από τον Brendan Eich το 1995 και σε σύντομο χρονικό διάστημα εξελίχθηκε σε θεμελιώδες συστατικό των σύγχρονων διαδικτυακών εφαρμογών. Πρόκειται για μια υψηλού επιπέδου γλώσσα που

επιτρέπει στους προγραμματιστές να δημιουργούν διαδραστικό και δυναμικό περιεχόμενο εντός των φυλλομετρητών ιστού (web browser).

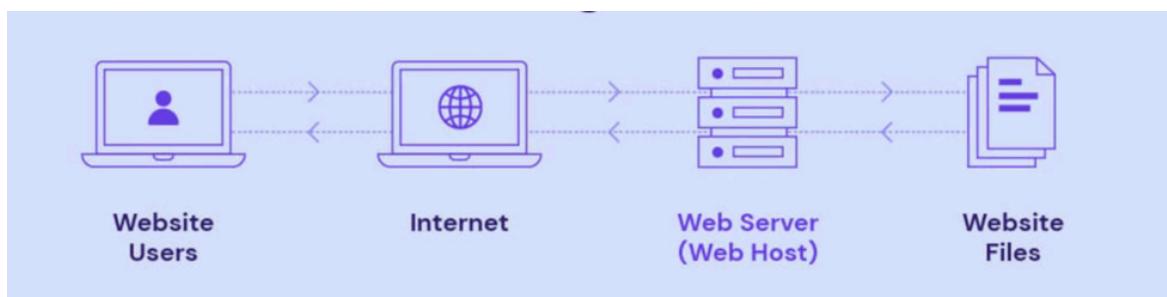
Ένα από τα βασικά πλεονεκτήματα της JavaScript είναι η ικανότητά της να εκτελείται σχεδόν σε οποιαδήποτε συσκευή με πρόγραμμα περιήγησης ιστού. Χρησιμοποιείται κυρίως για την ανάπτυξη front-end με σκοπό την βελτίωση της εμπειρίας του χρήστη, επιτρέποντας δυναμικές ενημερώσεις περιεχομένου χωρίς να απαιτείται επαναφόρτωση (refresh) της σελίδας. Η JavaScript αποτελεί αναπόσπαστο μέρος της τριάδας των τεχνολογιών ιστού, μαζί με την HTML για τη δομή και την CSS για την παρουσίαση (JS: JavaScript | MDN, n.d.).

2.4.4 Web Server

Ένας διακομιστής ιστού (Web Server) μπορεί να αναφέρεται τόσο σε στοιχεία του υλικού όσο και σε στοιχεία λογισμικού που συνεργάζονται μεταξύ τους. Από την πλευρά του υλικού, πρόκειται για έναν υπολογιστή που αποθηκεύει το λογισμικό με ρόλο διακομιστή ιστού και τα αρχεία ενός δικτυακού τόπου και συνδέεται με το διαδίκτυο για την ανταλλαγή δεδομένων. Από την πλευρά του λογισμικού, περιλαμβάνει έναν διακομιστή HTTP που κατανοεί τις διευθύνσεις URL και παραδίδει το περιεχόμενο στη συσκευή του χρήστη.

Για την ανάκτηση μιας ιστοσελίδας, ένα πρόγραμμα περιήγησης στέλνει ένα αίτημα στον διακομιστή ιστού, ο οποίος αναζητά το ζητούμενο αρχείο, το διαβάζει, το επεξεργάζεται και το στέλνει πίσω. Οι διακομιστές ιστού αποθηκεύουν αρχεία και υποστηρίζουν το πρωτόκολλο HTTP για την επικοινωνία μεταξύ υπολογιστών. Οι διακομιστές HTTP επεξεργάζονται και απαντούν στα αιτήματα, ελέγχοντας αν οι διευθύνσεις URL αντιστοιχούν σε υπάρχοντα αρχεία ή αλλιώς δημιουργούν περιεχόμενο δυναμικά.

Οι διακομιστές ιστού μπορούν να εξυπηρετούν στατικό ή δυναμικό περιεχόμενο. Το στατικό περιεχόμενο σερβίρεται ως έχει, ενώ το δυναμικό περιεχόμενο περιλαμβάνει την επεξεργασία ή τη δημιουργία του από μια βάση δεδομένων. Οι προγραμματιστές μπορούν να χρησιμοποιήσουν έτοιμους διακομιστές εφαρμογών ή να δημιουργήσουν τους δικούς τους, αξιοποιώντας υπάρχοντα κώδικα και βιβλιοθήκες (What is a Web Server? | MDN, n.d.). Στην εικόνα 2-8 παρουσιάζεται η επικοινωνία μεταξύ διακομιστή και περιηγητή ιστού.



Εικόνα 2-8 Επικοινωνία Web Server
(*What Is a Web Server? | Hostinger tutorials, 2024*)

2.4.5 PHP

Η PHP (Hypertext Preprocessor) είναι μια γλώσσα σεναρίων από την πλευρά του διακομιστή που έχει σχεδιαστεί για την ανάπτυξη ιστοσελίδων. Δημιουργήθηκε αρχικά από τον Rasmus Lerdorf το 1994 και έκτοτε έχει εξελιχθεί σε μια ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού.

Είναι γλώσσα ανοιχτού κώδικα, το οποίο σημαίνει ότι ο πηγαίος κώδικας της είναι ελεύθερα διαθέσιμος για τροποποίηση και αναδιανομή. Αυτό έχει συμβάλει στην ευρεία υιοθέτησή της και στην ανάπτυξη μιας μεγάλης και ενεργής κοινότητας προγραμματιστών. Ενσωματώνεται στον κώδικα HTML και εκτελείται στον διακομιστή, παράγοντας δυναμικές ιστοσελίδες που μπορούν να αλληλεπιδρούν με βάσεις δεδομένων και να εκτελούν διάφορες εργασίες. Όταν ένας χρήστης ζητά μια ιστοσελίδα, ο κώδικας PHP εκτελείται στον διακομιστή και η παράγωγη HTML αποστέλλεται στο πρόγραμμα περιήγησης του χρήστη. Επίσης είναι συμβατή με διάφορα λειτουργικά συστήματα, καθιστώντας την μια ευέλικτη επιλογή για την ανάπτυξη ιστοσελίδων σε διάφορες πλατφόρμες. Μπορεί να συνδεθεί με διάφορα συστήματα διαχείρισης βάσεων δεδομένων, με τη MySQL να αποτελεί μια ευρέως χρησιμοποιούμενη επιλογή. Διαθέτει μια τεράστια συλλογή ενσωματωμένων συναρτήσεων και βιβλιοθηκών, απλοποιώντας τις εργασίες ανάπτυξης ιστοσελίδων. Η PHP χαρακτηρίζεται από την απλότητα και την ευκολία εκμάθησής της, καθιστώντας την προσιτή στους αρχάριους. Ωστόσο, προσφέρει επίσης προηγμένα χαρακτηριστικά για έμπειρους προγραμματιστές, συμβάλλοντας στην ευρεία χρήση της σε όλο το φάσμα των δεξιοτήτων (Welling & Thomson, 2017).

2.4.6 IoT Πλατφόρμες

Το Διαδίκτυο των Πραγμάτων (IoT) αναφέρεται στο διασυνδεδεμένο δίκτυο φυσικών συσκευών με αισθητήρες και λογισμικό που τους επιτρέπει να συλλέγουν και να ανταλλάσσουν δεδομένα. Οι συσκευές αυτές μπορεί να είναι είτε οικιακές συσκευές είτε βιομηχανικά μηχανήματα και υποδομές έξυπνων πόλεων. Στόχος του IoT είναι η δημιουργία ενός απρόσκοπτου και ευφυούς περιβάλλοντος, επιτρέποντας στις συσκευές αυτές να επικοινωνούν και να συνεργάζονται.

Όταν αναφερόμαστε στην έννοια της πλατφόρμας IoT, αναφερόμαστε σε μια συλλογή τεχνολογιών που παρέχουν τα βασικά στοιχεία για την ανάπτυξη ευφυούς περιβάλλοντος στο οποίο θα λειτουργούν οι διασυνδεμένες συσκευές. Η βασική επιδίωξη μιας πλατφόρμας IoT είναι να παρέχει ολοκληρωμένη λειτουργικότητα για την εφαρμογή, επιτρέποντας στον προγραμματιστή να επικεντρωθεί στη λογική και τις μοναδικές ιδιότητες που παρέχει η τελική υλοποίηση, η οποία περιλαμβάνει τόσο το υλικό όσο και το λογισμικό.

Το IoT αποσκοπεί στη σύνδεση συσκευών από απόσταση για απρόσκοπη λειτουργία και ευκολία στη χρήση. Μια πλατφόρμα IoT επιδιώκει να συνδυάσει τα δεδομένα των αισθητήρων από μεμονωμένες συσκευές σε ένα δικτύων δεδομένων. Επιπλέον, αποτελεί τη βάση που επιτρέπει στους προγραμματιστές να κατανέμουν εφαρμογές, να συλλέγουν δεδομένα από απόσταση, να διασφαλίζουν τη συνδεσιμότητα, και να διαχειρίζονται τους αισθητήρες (Buyya & Dastjerdi, 2016)

3. Πλατφόρμες Ανάπτυξης

3.1 Εισαγωγή

Τα τελευταία χρόνια το τοπίο της ανάπτυξης εφαρμογών έχει μεταμορφωθεί σημαντικά λόγω της χρήσης εξελιγμένων εργαλείων γνωστών ως πλατφόρμων ανάπτυξης εφαρμογών. Αυτό το δυναμικό και ολοκληρωμένο σύνολο πόρων παίζει καθοριστικό ρόλο στην σχεδίαση, κατασκευή και ανάπτυξη εφαρμογών λογισμικού με αποτελεσματικότητα και ακρίβεια.

Μια πλατφόρμα ανάπτυξης εφαρμογών χρησιμεύει ως ένα ολοκληρωμένο πλαίσιο που περιλαμβάνει όχι μόνο τα απαραίτητα εργαλεία για την κωδικοποίηση, αλλά και μια σειρά από χαρακτηριστικά που είναι ιδιαίτερης σημασίας για ολόκληρο τον κύκλο ζωής της ανάπτυξης λογισμικού. Αυτές οι πλατφόρμες έχουν σχεδιαστεί για να παρέχουν ένα ενιαίο περιβάλλον για την κωδικοποίηση, τη δοκιμή και την ανάπτυξη. Επιπλέον, μια πλατφόρμα ανάπτυξης εφαρμογών ενσωματώνει υποστήριξη για μια ποικιλία γλωσσών προγραμματισμού και πλαισίων, προσφέροντας ευελιξία και προσαρμοστικότητα στις ποικίλες ανάγκες των έργων λογισμικού.

Με ολοκληρωμένες διεπαφές προγραμματισμού εφαρμογών (API), οι προγραμματιστές μπορούν να ενσωματώνουν χωρίς κόπο λειτουργικότητες, όπως η πιστοποίηση ταυτότητας, η πρόσβαση σε δεδομένα και υπηρεσίες, συμβάλλοντας τόσο στην αποτελεσματικότητα, όσο και στην επεκτασιμότητα.

Καθώς το ψηφιακό τοπίο συνεχίζει να εξελίσσεται, η σημασία των πλατφόρμων ανάπτυξης εφαρμογών γίνεται όλο και πιο έντονη, παρέχοντας ένα δομημένο και συνεκτικό περιβάλλον, με τις πλατφόρμες ανάπτυξης να δίνουν τη δυνατότητα στους προγραμματιστές να ανταπεξέλθουν στις πολυπλοκότητες της σύγχρονης μηχανικής λογισμικού.

3.2 Ολοκληρωμένα περιβάλλοντα ανάπτυξης (IDE)

Ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment, IDE) είναι ένα εργαλείο λογισμικού που παρέχει ένα ενοποιημένο και φιλικό προς το χρήστη περιβάλλον για την ανάπτυξη λογισμικού. Συνδυάζει διάφορα χαρακτηριστικά σε μια ενιαία πλατφόρμα, βελτιώνοντας ολόκληρη τη διαδικασία ανάπτυξης και ενισχύοντας την

παραγωγικότητα των προγραμματιστών. Τα IDE έχουν σχεδιαστεί για να προσφέρουν μια λύση "όλα σε ένα", καλύπτοντας τις ανάγκες των προγραμματιστών κατά την κωδικοποίηση, την αποσφαλμάτωση και την ανάπτυξη.

Τα IDE συνήθως διαθέτουν έναν εξελιγμένο επεξεργαστή κώδικα που υποστηρίζει διάφορες γλώσσες προγραμματισμού. Αυτός ο συντάκτης συχνά περιλαμβάνει χαρακτηριστικά όπως επισήμανση συντακτικού, συμπλήρωση κώδικα και έξυπνες προτάσεις κώδικα, διευκολύνοντας τους προγραμματιστές να γράφουν καθαρό και χωρίς λάθη κώδικα. Επίσης ενσωματώνουν μεταγλωττιστές ή διερμηνείς, επιτρέποντας στους προγραμματιστές να εκτελούν και να δοκιμάζουν τον κώδικα τους απευθείας μέσα στο περιβάλλον ανάπτυξης. Αυτό βοηθά στην έγκαιρη ανίχνευση σφαλμάτων κατά τη διαδικασία ανάπτυξης. Ακόμη διαθέτουν εργαλεία εντοπισμού σφαλμάτων εντός του IDE τα οποία επιτρέπουν την εκτέλεση του κώδικα βήμα προς βήμα, τον έλεγχο των τιμών των μεταβλητών σε πραγματικό χρόνο, ενισχύοντας τη διαδικασία εντοπισμού σφαλμάτων. Πολλά IDE υποστηρίζουν συστήματα ελέγχου εκδόσεων (version control), όπως το Git, επιτρέποντας στους προγραμματιστές να διαχειρίζονται τις αλλαγές στον κώδικα τους, να συνεργάζονται με τα μέλη της ομάδας και να παρακολουθούν το ιστορικό του έργου.

Η χρήση ενός IDE προσφέρει πολλά πλεονεκτήματα σε σχέση με έναν απλό επεξεργαστή κειμένου διότι μειώνει την ανάγκη εναλλαγής μεταξύ διαφορετικών εργαλείων, εξοικονομώντας χρόνο και βελτιώνοντας την αποτελεσματικότητα της ροής εργασίας.

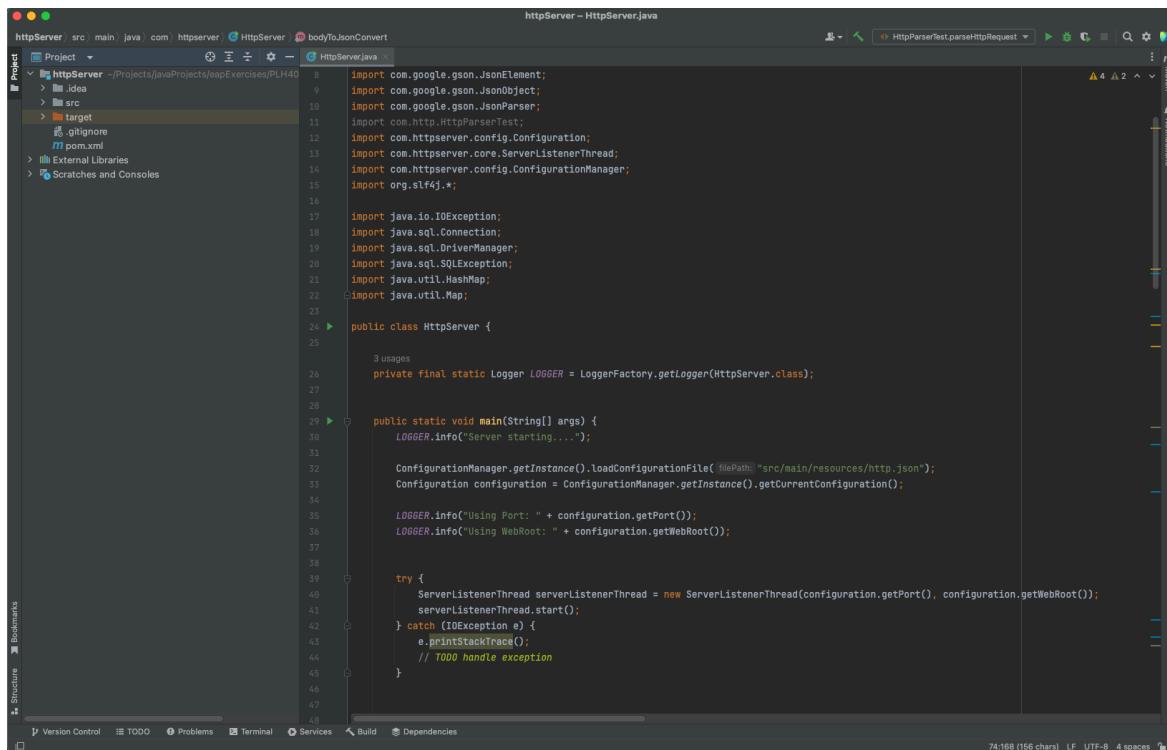
Τα παρεχόμενα εργαλεία ανάλυσης κώδικα και ελέγχου σφαλμάτων, συμβάλλουν στην αποδοτικότερη εργασία των προγραμματιστών.

Μερικά από τα πιο διαδεδομένα IDE σε διάφορες γλώσσες προγραμματισμού είναι τα κάτωθι, τα οποία θα χρησιμοποιηθούν κατά την ανάπτυξη κώδικα για τις υλοποιήσεις του λογισμικού της παρούσας εργασίας:

1. Visual Studio Code, το οποίο αναπτύχθηκε από τη Microsoft και υποστηρίζει γλώσσες όπως η C#, η C++ και η Python.
2. IntelliJ IDEA, το οποίο χρησιμοποιείται για την ανάπτυξη JAVA και Kotlin, παρέχοντας ισχυρά εργαλεία ανάλυσης και αναδιαμόρφωσης κώδικα.

3. Arduino IDE, το οποίο απλοποιεί τη διαδικασία προγραμματισμού των πλακετών Arduino. Διαθέτει χαρακτηριστικά όπως επεξεργαστή κώδικα, μεταγλωττιστή και μια απλοποιημένη διεπαφή για τη μεταφόρτωση κώδικα σε συσκευές Arduino.
4. PhpStorm, είναι IDE το οποίο αναπτύχθηκε από την JetBrains όπως και το IntelliJ. Το PhpStorm είναι ένα δημοφιλές ολοκληρωμένο περιβάλλον ανάπτυξης για την PHP. Προσφέρει προηγμένα εργαλεία ανάλυσης κώδικα, αποσφαλμάτωσης και ελέγχου, καθιστώντας το μια προτιμώμενη επιλογή για τους προγραμματιστές PHP.
5. Thonny (microPython IDE), το οποίο αποτελεί ένα φιλικό προς το χρήστη IDE σχεδιασμένο για προγραμματισμό σε MicroPython, ένα υποσύνολο της γλώσσας Python βελτιστοποιημένο για μικροελεγκτές. Το Thonny παρέχει ένα ολοκληρωμένο περιβάλλον για τη συγγραφή, τον έλεγχο και την ανάπτυξη κώδικα MicroPython σε μικροελεγκτές.

Ένα IDE βελτιώνει σημαντικά την εμπειρία ανάπτυξης παρέχοντας ένα ολοκληρωμένο περιβάλλον που περιλαμβάνει εργαλεία κωδικοποίησης, αποσφαλμάτωσης και διαχείρισης έργου. Τα ολοκληρωμένα χαρακτηριστικά του συμβάλλουν στην αύξηση της παραγωγικότητας, της ποιότητας του κώδικα και της συνεργασίας μεταξύ των προγραμματιστών. Στην εικόνα 3-1 παρουσιάζεται το περιβάλλον ανάπτυξης IntelliJ IDEA το οποίο χρησιμοποιείται για την ανάπτυξη κώδικα σε JAVA.



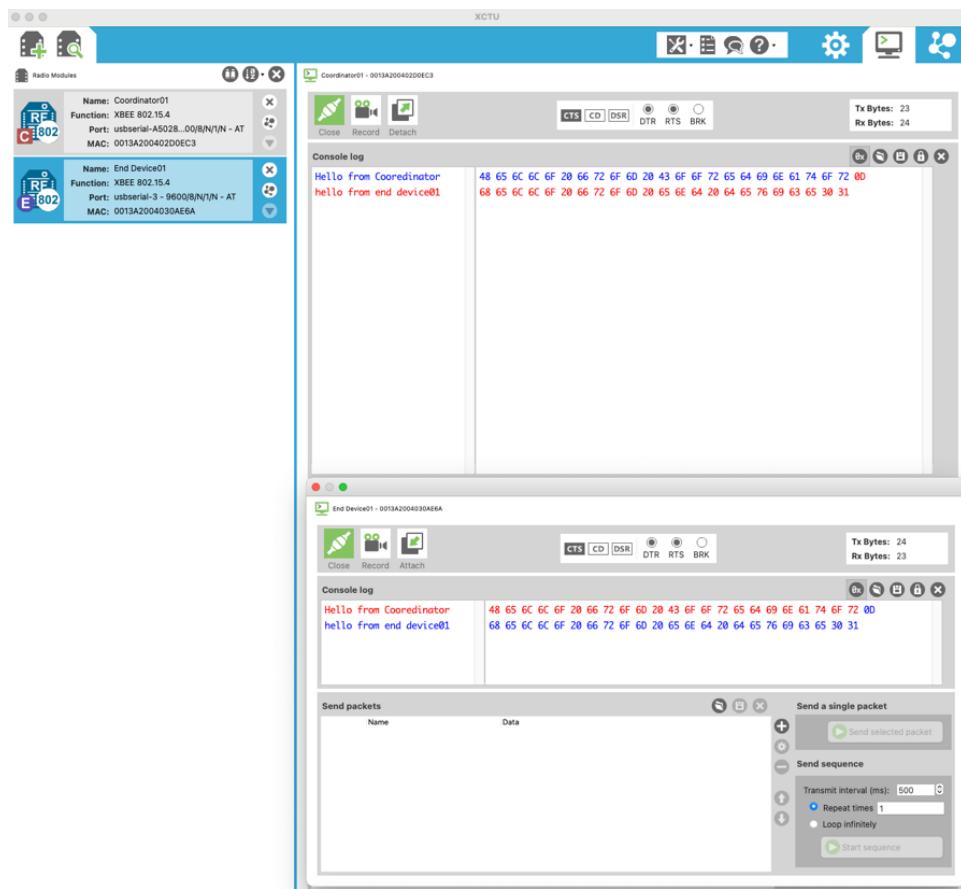
Εικόνα 3-1 IntelliJ IDEA

3.3 X-CTU Configuration & Test Utility Software

Το XCTU είναι λογισμικό της Digi XCTU. Είναι μια δωρεάν εφαρμογή πολλαπλών πλατφόρμων που έχει σχεδιαστεί για να διευκολύνει την αλληλεπίδραση μεταξύ των προγραμματιστών και των μονάδων Digi RF (Radio Frequency) μέσω μιας γραφικής διεπαφής (Graphical User Interface, GUI). Επικεντρωμένο κυρίως στις μονάδες RF XBee, το XCTU απλοποιεί τις διαδικασίες εγκατάστασης, διαμόρφωσης και δοκιμής.

Τα βασικά χαρακτηριστικά του XCTU περιλαμβάνουν μια γραφική αναπαράσταση του δικτύου XBee και της ισχύος του σήματος, και το XBee API (Application Programming Interface) frame builder, που βοηθά στη δημιουργία και ερμηνεία των πλαισίων API για τις μονάδες που λειτουργούν σε λειτουργία API. Πρόσθετα σημαντικά σημεία περιλαμβάνουν τη δυνατότητα διαχείρισης και διαμόρφωσης πολλαπλών συσκευών RF, ακόμη και εξ αποστάσεως, ενημερώσεις υλικολογισμικού με αυτόματο χειρισμό των ρυθμίσεων, ειδικές κονσόλες API και AT, αποθήκευση και φόρτωση, ενσωματωμένα εργαλεία για τη δημιουργία και ερμηνεία πλαισίων, αποκατάσταση μονάδων με κατεστραμμένο υλικολογισμικό, δοκιμή εμβέλειας και εξερεύνηση υλικολογισμικού.

Το XCTU προσφέρει επίσης μια διαδικασία ενημέρωσης τόσο για τη βιβλιοθήκη υλικού και λογισμικού της εφαρμογής όσο και για τη βιβλιοθήκη υλικού και λογισμικού του module χωρίς να απαιτούνται πρόσθετες λήψεις. Η εφαρμογή περιλαμβάνει ολοκληρωμένη τεκμηρίωση, η οποία είναι προσβάσιμη ανά πάσα στιγμή, συμβάλλοντας στον φιλικό προς τον χρήστη χαρακτήρα της και καθιστώντας την ανάπτυξη στην πλατφόρμα XBee πιο προσιτή και αποτελεσματική (XCTU, n.d.). Στην εικόνα 3-2 παρουσιάζεται η εφαρμογή XCTU.



Εικόνα 3-2 Εφαρμογή XCTU

3.4 Apache HTTP Server

Ο Apache HTTP Server είναι ένας από τους πιο διαδεδομένους διακομιστές ιστού παγκοσμίως. Είναι λογισμικό ανοικτού κώδικα το οποίο έχει καθοριστικό ρόλο στη φιλοξενία και την εξυπηρέτηση περιεχομένου ιστού, παρέχοντας μια αξιόπιστη και προσαρμόσιμη πλατφόρμα για ένα ευρύ φάσμα εφαρμογών.

Ο διακομιστής υποστηρίζει πλήθος γλωσσών προγραμματισμού, συμπεριλαμβανομένων των Perl, PHP και Python, καθιστώντας τον ευέλικτο για τη φιλοξενία δυναμικών και διαδραστικών εφαρμογών. Επιπλέον, η υποστήριξη του Apache για τα πρωτόκολλα Secure Sockets Layer (SSL) και Transport Layer Security (TLS) διασφαλίζει την ασφαλή μετάδοση δεδομένων μέσω του διαδικτύου.

Η διαμόρφωση του Apache ελέγχεται κυρίως μέσω ενός αρχείου ρυθμίσεων που βασίζεται σε κείμενο, παρέχοντας στους διαχειριστές ένα απλό και διαφανές μέσο προσαρμογής της συμπεριφοράς του διακομιστή. Αυτή η απλότητα, δίνει τη δυνατότητα στους χρήστες να ρυθμίζουν λεπτομερώς τον διακομιστή ώστε να ανταποκρίνεται στις μοναδικές απαιτήσεις των εφαρμογών τους.

Ως έργο ανοικτού κώδικα, ο Apache επωφελείται από μια ενεργή κοινότητα προγραμματιστών, διαχειριστών και χρηστών. Αυτό το περιβάλλον ευνοεί την καινοτομία, τη συνεχή βελτίωση και την ταχεία επίλυση των προβλημάτων μέσω των φόρουμ υποστήριξης της κοινότητας (Apache Software Foundation, n.d.)

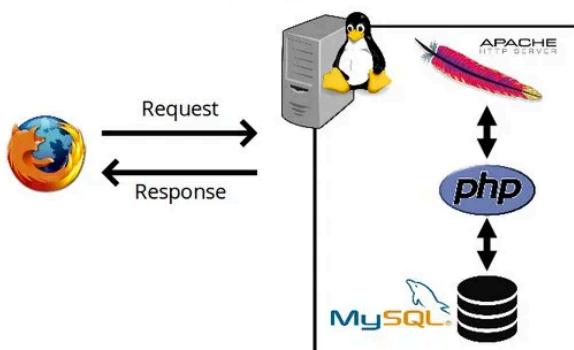
3.5 LAMP

Το LAMP είναι το ακρωνύμιο των λέξεων Linux, Apache, MySQL και PHP. Αποτελεί ένα πλέγμα (stack) διακομιστών το οποία περικλείει έναν ισχυρό συνδυασμό τεχνολογιών που αποτελεί τη βάση για δυναμικές και διαδραστικές διαδικτυακές εφαρμογές. Είναι λογισμικό ανοικτού κώδικα το οποίο παρέχει ένα ισχυρό περιβάλλον για τη φιλοξενία ιστοσελίδων, τη διαχείριση βάσεων δεδομένων και τη δημιουργία σεναρίων από την πλευρά του διακομιστή.

Στον πυρήνα της στοίβας LAMP βρίσκεται το λειτουργικό σύστημα Linux, το οποίο είναι μια ευέλικτη και σταθερή πλατφόρμα ανοικτού κώδικα. Το Linux είναι το θεμέλιο για ολόκληρη τη στοίβα, προσφέροντας ένα ασφαλές και προσαρμόσιμο περιβάλλον. Ο Apache HTTP Server, λειτουργεί ως διακομιστής ιστού που είναι υπεύθυνος για το χειρισμό και την εξυπηρέτηση των αιτημάτων. Το σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS) ανοικτού κώδικα MySQL, χρησιμοποιείται για τη διαχείριση και οργάνωση δομημένων δεδομένων. Η PHP επιτρέπει τη δυναμική δημιουργία περιεχομένου ιστού μέσω της ενσωμάτωσης δυναμικού κώδικα σε σελίδες HTML.

Το LAMP αποτελεί μια ολοκληρωμένη και αποτελεσματική λύση για την ανάπτυξη ιστοσελίδων, λόγω του συνδυασμού των Linux, Apache, MySQL και PHP σε ένα ενιαίο πλαίσιο. Αυτός ο συνδυασμός παρέχει ένα σταθερό και ασφαλές περιβάλλον για τη φιλοξενία και την ανάπτυξη δυναμικών διαδικτυακών εφαρμογών (LAMP, n.d.). Στην εικόνα 3-3 φαίνεται η αρχιτεκτονική δομή στοίβας LAMP.

LAMP architecture



Εικόνα 3-3 Δομή πλέγματος LAMP
(Build and Deploy LAMP Server on AWS | Medium, 2024)

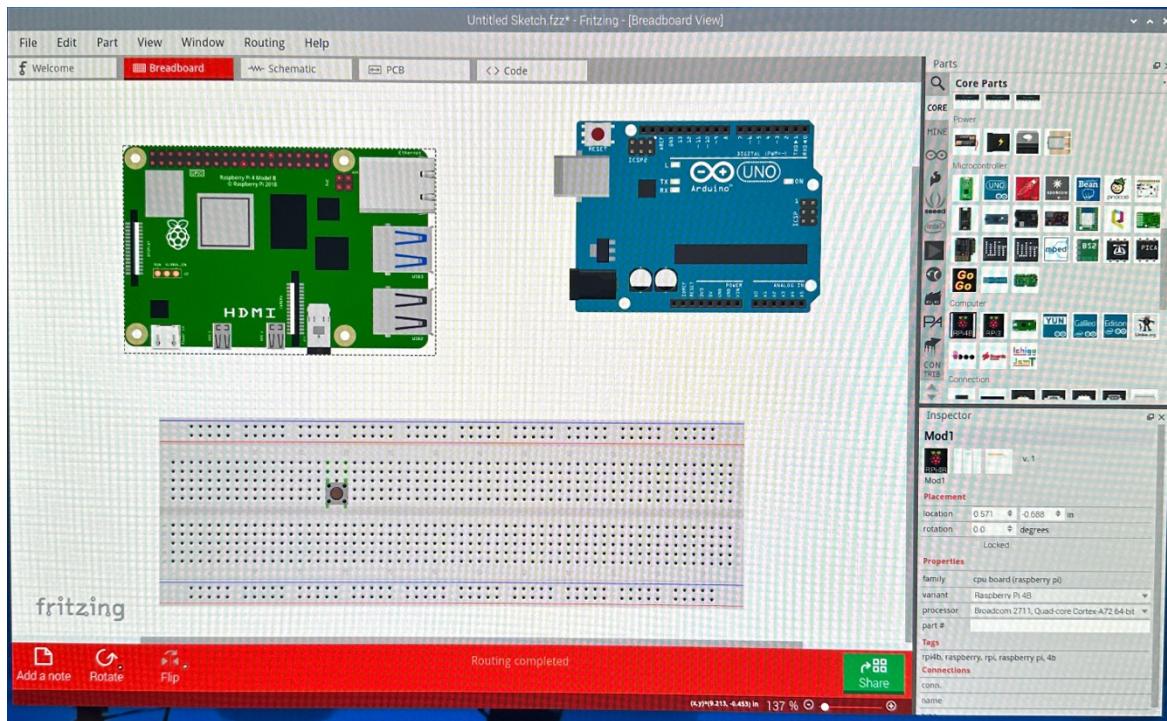
3.6 Fritzing

Το Fritzing είναι ένα λογισμικό ανοικτού κώδικα που έχει σχεδιαστεί για να διευκολύνει τη δημιουργία σχεδίων ηλεκτρονικών κυκλωμάτων και πρωτότυπων, απευθυνόμενο κυρίως σε χρήστες με διαφορετικά επίπεδα εμπειρίας.

Η σχεδίαση με τη χρήση του Fritzing επιτρέπει στους χρήστες να σέρνουν και να αποθέτουν εξαρτήματα σε μια εικονική πλακέτα breadboard, επιτρέποντας την εύκολη συναρμολόγηση κυκλωμάτων. Το λογισμικό υποστηρίζει επίσης τις προβολές σχηματικών και PCB (Printed Circuit Board).

Ένα από τα πλεονεκτήματα του Fritzing είναι ότι χρησιμοποιείται ως εκπαιδευτικό εργαλείο, καθιστώντας το ιδανική επιλογή για εκπαιδευτικούς και φοιτητές. Επίση διαθέτει μια ενεργή και κοινότητα χρηστών που μοιράζονται έργα, στοιχεία και τεχνογνωσία. Αυτό το περιβάλλον συμβάλλει στη συνεχή βελτίωση του λογισμικού και επεκτείνει τη βιβλιοθήκη των εξαρτημάτων του. Στην παρούσα εργασία θα γίνει χρήση του Fritzing για

την παρουσίαση όλων των συνδεσμολογιών του υλικού (Fritzing, n.d.). Στην εικόνα 3-4 φαίνεται το περιβάλλον σχεδίασης της εφαρμογής Fritzing.



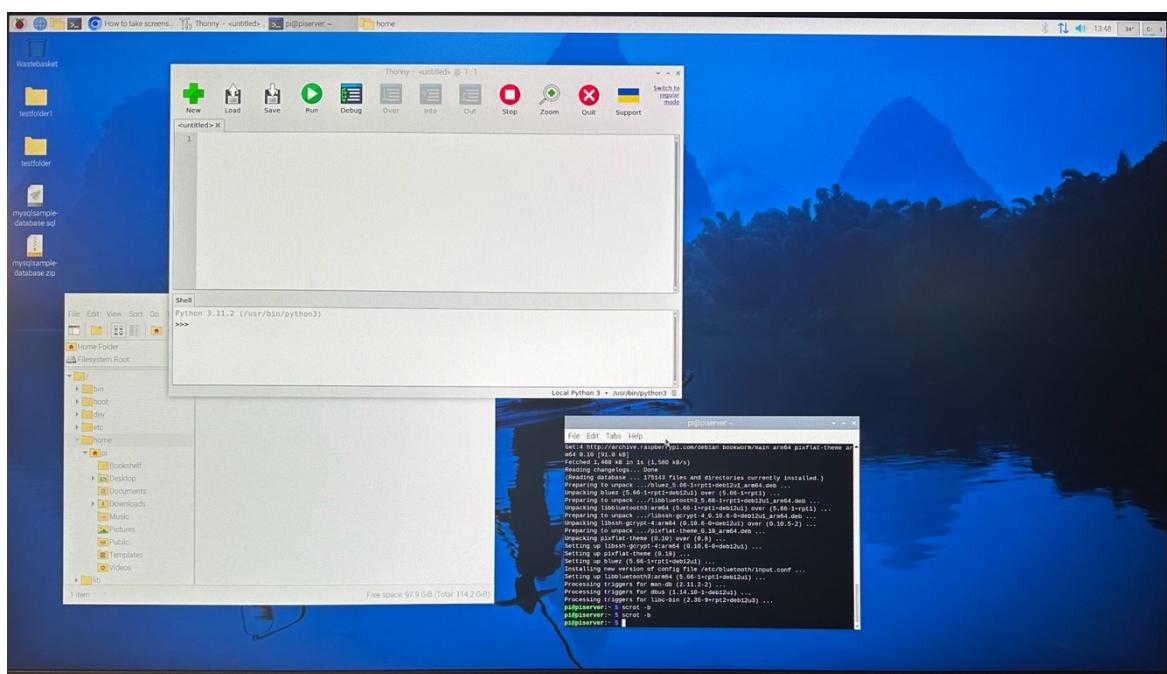
Εικόνα 3-4 Περιβάλλον σχεδίασης Fritzing

3.7 Λειτουργικό Raspbian

Το Raspbian OS, το επίσημο λειτουργικό σύστημα για τους υπολογιστές Raspberry Pi, είναι μια εξειδικευμένη διανομή Linux. Το Raspbian χρησιμεύει ως μια φιλική προς το χρήστη πλατφόρμα που δίνει τη δυνατότητα σε απλούς χρήστες, εκπαιδευτικούς και προγραμματιστές να εξερευνήσουν τον κόσμο της πληροφορικής και της ψηφιακής καινοτομίας.

Το Raspbian είναι προσαρμοσμένο για να υποστηρίζει την αρχιτεκτονική του Raspberry Pi, παρέχοντας απρόσκοπτη συμβατότητα και βέλτιστη απόδοση. Ένας από τους πρωταρχικούς στόχους του Raspbian είναι να χρησιμεύσει ως εκπαιδευτικό εργαλείο, εισάγοντας τους χρήστες στον προγραμματισμό, τα ηλεκτρονικά και τις έννοιες της πληροφορικής. Στο λειτουργικό σύστημα είναι προεγκατεστημένο το Scratch, η Python και το Sonic Pi, διευκολύνοντας μια πρακτική και διαδραστική εμπειρία μάθησης. To Raspbian επωφελείται από μια ενεργή κοινότητα χρηστών που συμβάλλουν στην ανάπτυξή του και

μοιράζονται έργα, σεμινάρια και λύσεις. Το λειτουργικό σύστημα Raspbian OS αποτελεί βασικό συστατικό του οικοσυστήματος Raspberry Pi, παρέχοντας ένα λειτουργικό σύστημα που έχει κατασκευαστεί για συγκεκριμένο σκοπό και δίνει τη δυνατότητα στους χρήστες να εξερευνούν, να μαθαίνουν και να δημιουργούν (Raspberry Pi Foundation, n.d.). Στην παρούσα εργασία το Raspbian θα φιλοξενήσει μέσω του Raspberry Pi4, τους διακομιστές LAMP και τον application server (Java - REST API).



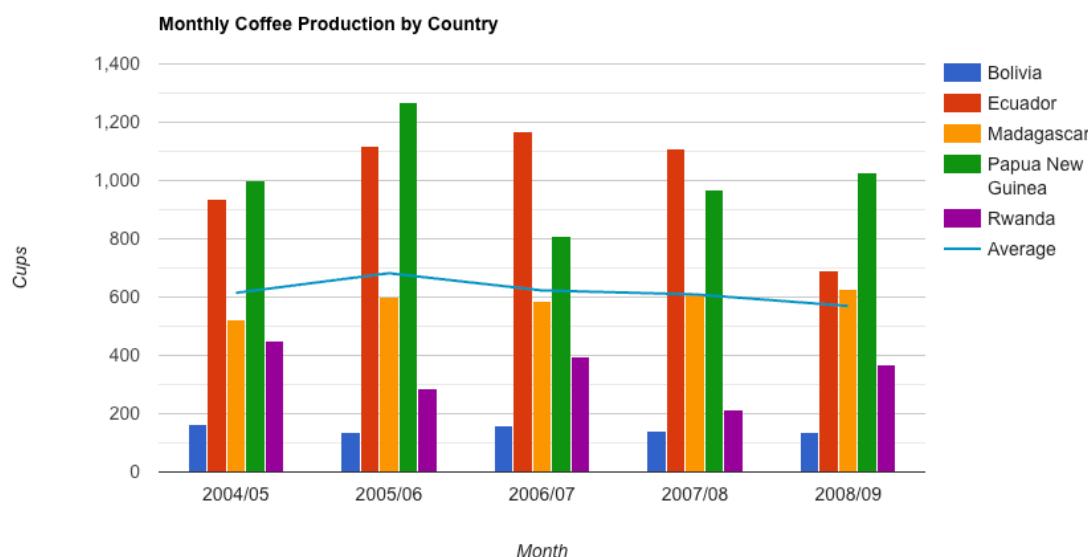
Εικόνα 3-5 Λειτουργικό Σύστημα Raspbian

3.8 Google charts

Τα διαγράμματα Google (Google charts), μια ευέλικτη βιβλιοθήκη οπτικοποίησης δεδομένων που αναπτύχθηκε από την Google, δίνει τη δυνατότητα στους χρήστες να μετατρέπουν τα ακατέργαστα δεδομένα σε οπτικές αναπαραστάσεις.

Το Google Charts απλοποιεί τη δημιουργία διαγραμμάτων και γραφικών παραστάσεων. Τα διαγράμματα Google μπορούν να ενσωματωθούν σε διάφορες τεχνολογίες ιστού. Χρησιμοποιώντας JavaScript και HTML5, οι χρήστες μπορούν εύκολα να ενσωματώσουν διαδραστικά διαγράμματα απευθείας σε ιστοσελίδες, εφαρμογές ή πίνακες ελέγχου. Η συμβατότητα της βιβλιοθήκης με διάφορες πηγές δεδομένων, συμπεριλαμβανομένων των JSON, CSV και Google Sheets, το καθιστά εύκολο, επιτρέποντας στους χρήστες να

απεικονίζουν δεδομένα από διαφορετικές προελεύσεις. Επίσης μέσω μιας απλής διαδικασίας διαμόρφωσης, οι χρήστες μπορούν να προσαρμόζουν τα χρώματα, τις γραμματοσειρές, τις ετικέτες και άλλα οπτικά στοιχεία. Στην παρούσα εργασία θα γίνει χρήση των Google Charts για την απεικόνιση των δεδομένων σε γραφήματα είτε για την παρουσίαση ιστορικών δεδομένων είτε για την παρουσίαση των τρέχοντων δεδομένων στην ιστοσελίδα (Google Charts, n.d.). Στην εικόνα 3-6 παρουσιάζεται ένα παράδειγμα διαγράμματος με χρήση Google Charts.



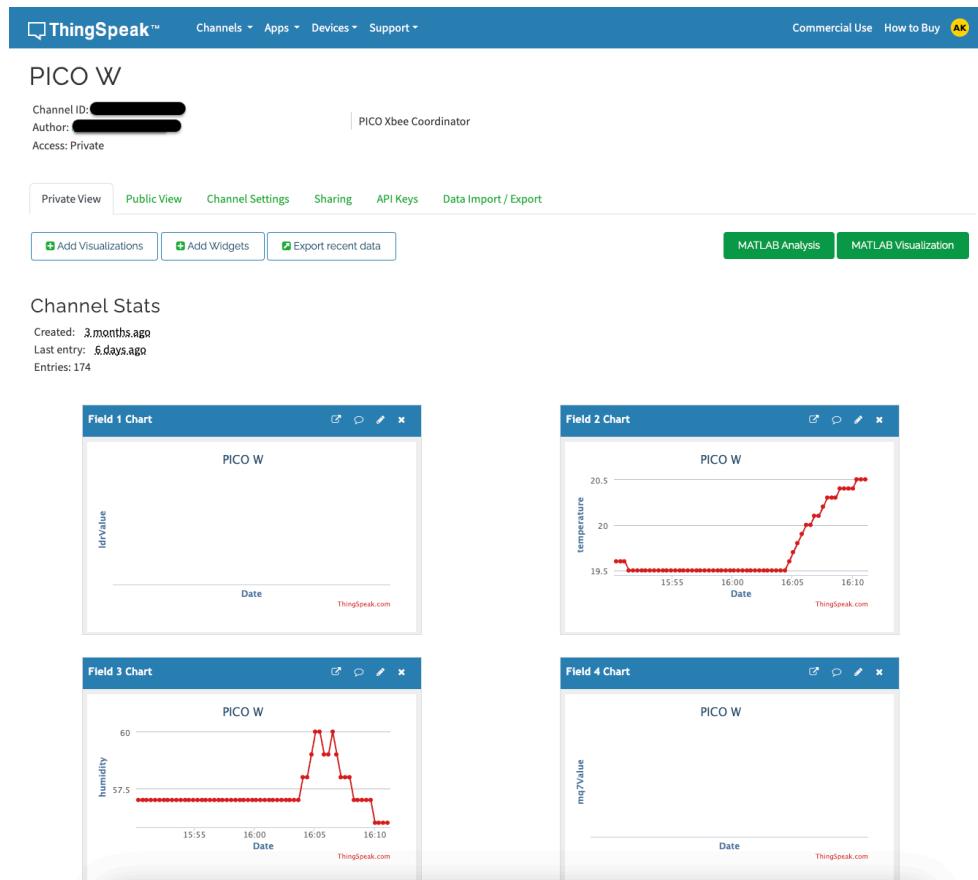
Εικόνα 3-6 Παράδειγμα διαγράμματος Google Charts

3.9 IOT ThingSpeak

To ThingSpeak, μια πλατφόρμα IoT που παρέχει στους χρήστες της, τη δυνατότητα να συλλέγουν, να αναλύουν και να απεικονίζουν δεδομένα από συνδεδεμένες συσκευές. To ThingSpeak παρέχει τη δυνατότητα οπτικοποίησης δεδομένων σε πραγματικό χρόνο με αποτέλεσμα οι χρήστες να μπορούν να δημιουργήσουν απεικονίσεις, όπως διαγράμματα και χάρτες, για να παρουσιάσουν αποτελεσματικά τα δεδομένα των IoT συσκευών τους.

Οι χρήστες μπορούν να έχουν πρόσβαση σε πληθώρα πόρων, συμπεριλαμβανομένων των φόρουμ της κοινότητας, προωθώντας την ανταλλαγή γνώσεων που σχετίζονται με τη διαχείριση και την ανάλυση δεδομένων IoT (ThingSpeak for IoT Projects | ThingSpeak, n.d.). Στην εικόνα 3-7 παρουσιάζεται το κανάλι συλλογής και αναπαράστασης των

δεδομένων των αισθητήρες που χρησιμοποιήθηκαν στις υλοποιήσεις της παρούσας εργασίας.



Εικόνα 3-7 Κανάλι ThingSpeak

4. Πλατφόρμες υλικού ανάπτυξης IoT

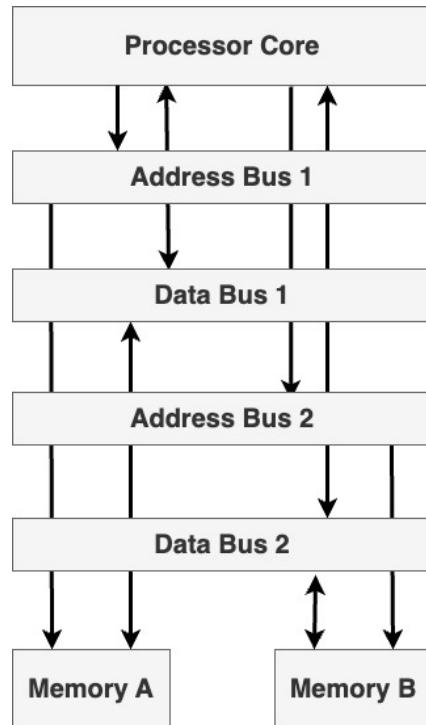
4.1 Πλατφόρμα Arduino

Το Arduino είναι μια ηλεκτρονική πλατφόρμα ανοικτού κώδικα, γνωστή για το φιλικό προς το χρήστη υλικό και λογισμικό της. Παρέχει τη δυνατότητα στους χρήστες να διαβάζουν τις διάφορες τιμές που λαμβάνουν στις εισόδους του, όπως είναι ένας διακόπτης ή οι ενδείξεις αισθητήρων και να το μετατρέπουν σε σήματα εξόδου, όπως είναι η ενεργοποίηση κινητήρων ή φωτισμού LED.

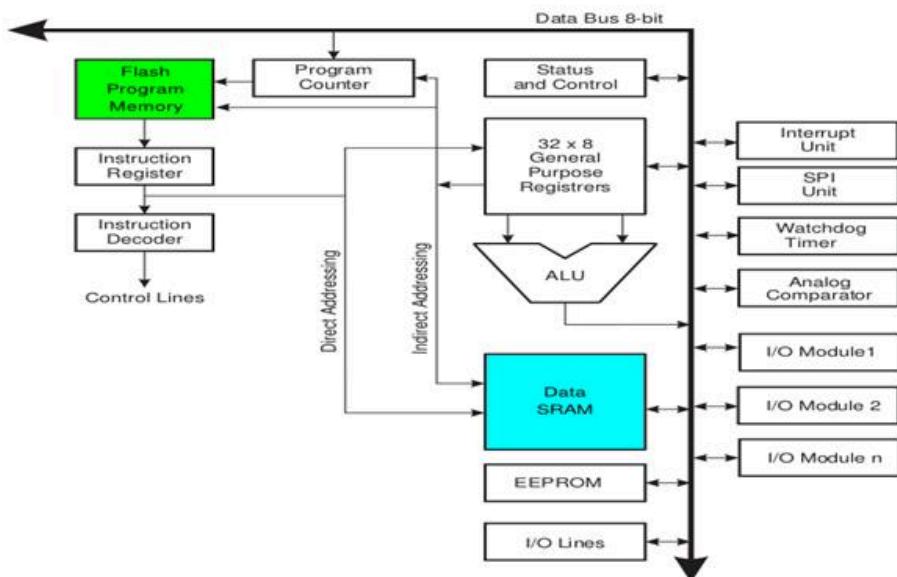
Το Arduino, αρχικά σχεδιάστηκε ως εργαλείο ταχείας κατασκευής πρωτοτύπων για μαθητές που δεν είχαν υπόβαθρο στα ηλεκτρονικά και τον προγραμματισμό και γρήγορα εξελίχθηκε σε μια ευέλικτη πλατφόρμα που εξυπηρετεί ένα ευρύ φάσμα έργων. Η ελκυστικότητα του Arduino έγκειται στην απλότητα και την προσαρμοστικότητά του, καθώς και στην προσιτή του τιμή, που το καθιστούν κατάλληλο για πολλών ειδών έργα και εφαρμογές.

4.1.1 Αρχιτεκτονική Arduino

Το Arduino χρησιμοποιεί την αρχιτεκτονική Harvard (εικόνα 4-1), όπου το κύριο χαρακτηριστικό της είναι ότι ο επεξεργαστής είναι συνδεδεμένος με δύο ανεξάρτητες μνήμες μέσω δύο ξεχωριστών σετ διαύλων. Ο κώδικας προγράμματος αποθηκεύεται στη μνήμη flash, ενώ τα δεδομένα αποθηκεύονται στην SRAM. Στον πυρήνα του Arduino Uno βρίσκεται ο μικροελεγκτής Atmega328P, ο οποίος διαθέτει μια αρχιτεκτονική 8-bit χρονισμένη στα 16 MHz (εικόνα 4-2). Ο μικροελεγκτής είναι εξοπλισμένος με 32KB μνήμης flash για την αποθήκευση του κώδικα του χρήστη και 2KB SRAM για την αποθήκευση μεταβλητών. Επιπλέον, προσφέρει ένα ευρύ φάσμα ενσωματωμένων περιφερειακών, συμπεριλαμβανομένων ψηφιακών ακροδεκτών εισόδου/εξόδου, αναλογικών/ψηφιακών μετατροπέων (ADC), εξόδων διαμόρφωσης εύρους παλμών (PWM) και διεπαφών επικοινωνίας UART (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-1 Αρχιτεκτονική Harvard
(*Modern Component Families and Circuit Block Design*, 2000)



Εικόνα 4-2 Αρχιτεκτονική Arduino
(*Balas*, 2019)

4.2.2 Εκδόσεις Arduino

Το Arduino, έχει εξελιχθεί με την πάροδο των ετών, οδηγώντας στη δημιουργία διαφόρων οικογενειών Arduino προσαρμοσμένων σε συγκεκριμένες ανάγκες και εφαρμογές. Κάθε



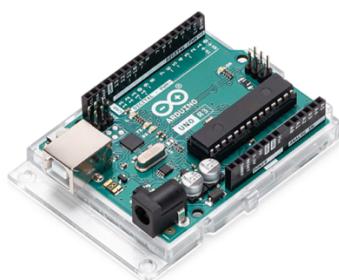
οικογένεια προσφέρει μοναδικά χαρακτηριστικά και δυνατότητες, καλύπτοντας ένα ευρύ φάσμα έργων και απαιτήσεων.

1. Οικογένεια Arduino Classic:

Η οικογένεια Arduino Classic (εικόνες 4-3 έως 4-10) περιλαμβάνει μια σειρά από πλακέτες που έχουν θέσει τα θεμέλια της πλατφόρμας Arduino, χρησιμεύοντας ως δομικά στοιχεία για αμέτρητα έργα στον κόσμο των ηλεκτρονικών. Οι πλακέτες αυτές, χαρακτηρίζονται από την απλότητα, την αξιοπιστία και την ευκολία χρήσης τους. Από την απλότητα του Uno έως τη δύναμη των Mega και Due, αυτές οι πλακέτες έχουν δώσει τη δυνατότητα σε κατασκευαστές, εκπαιδευτικούς και επαγγελματίες να μετατρέψουν τις ιδέες τους σε πραγματικότητα, προωθώντας την καινοτομία και τη δημιουργικότητα σε τομείς που κυμαίνονται από την τέχνη και το σχεδιασμό έως τη μηχανική και την επιστήμη. Καθώς η πλατφόρμα Arduino συνεχίζει να εξελίσσεται, η οικογένεια Classic παραμένει ένα διαχρονικό θεμέλιο για εξερεύνηση, μάθηση και ανακάλυψη.

- Uno R3

To Arduino Uno ονομάστηκε έτσι επειδή ήταν το πρώτο της σειράς Arduino. Τροφοδοτούμενο από τον ATmega328P, προσφέρει 14 ψηφιακές ακίδες εισόδου/εξόδου (με 6 εξόδους PWM) και 6 αναλογικές εισόδους, μαζί με έναν ταλαντωτή (crystal oscillator) 16 MHz για σταθερή λειτουργία. Με σύνδεση USB, υποδοχή τροφοδοσίας, κεφαλίδα ICSP και κουμπί επαναφοράς, το Uno παρέχει απρόσκοπτη συνδεσιμότητα και ευκολία χρήσης. Υποστηρίζει τάση λειτουργίας 5V, με συνιστώμενη τάση εισόδου που κυμαίνεται από 7-12V. Διαθέτει 32 KB μνήμης flash, 2 KB SRAM και 1 KB EEPROM (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-3 Arduino Uno R3
(Arduino Docs | Arduino Documentation, n.d.)

- Uno R4 Minima

To Arduino UNO R4 Minima, εξοπλισμένο με τον μικροελεγκτή 32-bit Renesas RA4M1, αποτελεί σημαντική βελτίωση σε σχέση με τους προκατόχους του. Έχει τάση λειτουργίας 5 V και προσφέρει διευρυμένη μνήμη, αυξημένη επεξεργαστική ισχύ και μια σειρά νέων ενσωματωμένων περιφερειακών. Οι προσθήκες περιλαμβάνουν ένα 12-bit DAC, CAN BUS και OP AMP (Operational Amplifier), παρέχοντας αυξημένη ευελιξία στα σχέδια έργων. Περιλαμβάνει 14 ψηφιακές ακίδες I/O, 6 αναλογικές ακίδες εισόδου, 6 ακίδες PWM και διεπαφές επικοινωνίας, συμπεριλαμβανομένων των UART, I2C και SPI (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-4 Arduino Uno R4 minima
(*Arduino Docs | Arduino Documentation, n.d.*)

- Uno R4 Wifi

To Arduino UNO R4 WiFi είναι τελευταία προσθήκη στη σειρά Arduino, συνδυάζοντας την ισχυρή επεξεργαστική ισχύ του μικροελεγκτή RA4M1 της Renesas με την ασύρματη συνδεσιμότητα της μονάδας ESP32-S3 της Espressif. Έχει τάση λειτουργίας 5 V, με διευρυμένη μνήμη, μεγαλύτερες ταχύτητες ρολογιού και μια σειρά από νέα ενσωματωμένα περιφερειακά, όπως πίνακας LED 12x8 και μια υποδοχή Qwiic (είδος I2C connector). Η ενσωμάτωση της υποστήριξης HID, της συνδεσιμότητας Wi-Fi και Bluetooth επιτρέπει την απομακρυσμένη παρακολούθηση και τον έλεγχο μέσω του Arduino IoT Cloud (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-5 Arduino Uno R4 WiFi
(*Arduino Docs | Arduino Documentation, n.d.*)

- Arduino Leonardo

To Arduino Leonardo τροφοδοτείται από τον ATmega32u4. Η πλατφόρμα προσφέρει 20 ψηφιακές ακίδες εισόδου/εξόδου, συμπεριλαμβανομένων 7 εξόδων PWM και 12 αναλογικών εισόδων. Εξοπλισμένη με ταλαντωτή κρυστάλλου (crystal oscillator) 16 MHz, σύνδεση micro USB, υποδοχή τροφοδοσίας, κεφαλή ICSP και κουμπί επαναφοράς, παρέχει απρόσκοπτη συνδεσιμότητα και ευκολία χρήσης (Arduino Docs | Arduino Documentation, n.d.)



Εικόνα 4-6 Arduino Leonardo
(*Arduino Docs | Arduino Documentation, n.d.*)

- Arduino Uno WiFi Rev2

To Arduino UNO WiFi Rev.2 διαθέτει τα τυπικά χαρακτηριστικά της σειράς Uno σε συνδυασμό με δυνατότητες WiFi και Bluetooth. Τροφοδοτείται από τον μικροελεγκτή ATmega 4809 της Microchip και διαθέτει 14 ψηφιακές ακίδες I/O (5 PWM), 6 αναλογικές εισόδους και πλήθος ενσωματωμένων χαρακτηριστικών, όπως σύνδεση USB, υποδοχή τροφοδοσίας, κεφαλίδα ICSP και κουμπί επαναφοράς. Η ταχύτητα του ρολογιού είναι 16 MHz, ενώ περιλαμβάνει 48 KB μνήμης flash, 6.144

bytes SRAM και 256 bytes EEPROM (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-7 Arduino Uno WiFi Rev2
(Arduino Docs | Arduino Documentation, n.d.)

- Arduino Zero

To Arduino Zero αντιπροσωπεύει μια σημαντική εξέλιξη στο οικοσύστημα Arduino, προσφέροντας μια απλή αλλά ισχυρή επέκταση 32-bit στην πλατφόρμα UNO. Με την MCU SAMD21 της Atmel, η οποία διαθέτει πυρήνα ARM Cortex M0+ 32-bit, το Zero παρέχει αυξημένη απόδοση για ένα ευρύ φάσμα εφαρμογών, από έξυπνες συσκευές IoT έως ρομποτική. Το Zero λειτουργεί στα 3,3V, παρέχοντας 20 ψηφιακές ακίδες I/O, 6 αναλογικές εισόδους και 3 ακίδες PWM, παράλληλα με την υποστήριξη UART και τα κανάλια ADC 12-bit. Διαθέτει 256 KB μνήμης flash, 32 KB SRAM και ταχύτητα ρολογιού στα 48 MHz (Arduino Docs | Arduino Documentation, n.d.).



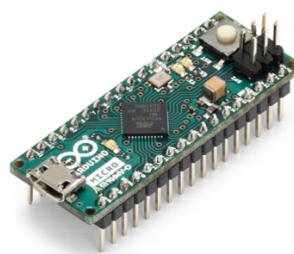
Εικόνα 4-8 Arduino Uno Zero
(Arduino Docs | Arduino Documentation, n.d.)

- Arduino Micro

To Arduino Micro είναι μια ευέλικτη πλακέτα μικροελεγκτή, που αναπτύχθηκε σε συνεργασία με την Adafruit και βασίζεται στον ATmega32U4. Διαθέτει 20 ψηφιακές ακίδες I/O (7 εξόδους PWM και 12 αναλογικές εισόδους), ταλαντωτή 16



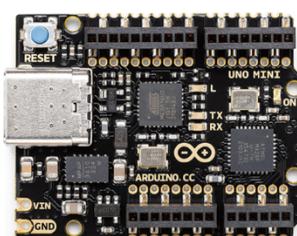
MHz, σύνδεση micro USB, κεφαλίδα ICSP και κουμπί επαναφοράς. Όπως και με το Arduino Leonardo, η ενσωματωμένη επικοινωνία USB του ATmega32U4 επιτρέπει στο Micro να εξομοιώνει ένα ποντίκι και ένα πληκτρολόγιο, μαζί με μια εικονική σειριακή θύρα/COM όταν είναι συνδεδεμένο σε έναν υπολογιστή. Η τάση λειτουργίας είναι 5V, και διαθέτει 32 KB μνήμης flash, 2,5 KB SRAM και 1 KB EEPROM (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-9 Arduino Micro
(*Arduino Docs | Arduino Documentation, n.d.*)

- **Arduino Uno Mini Limited Edition**

To Arduino UNO Mini Limited Edition προσφέρει μια μικροσκοπική και κομψή έκδοση για τους λάτρεις της τεχνολογίας. Κάθε μονάδα διακρίνεται από έναν μοναδικό σειριακό αριθμό στην πλακέτα. Διατηρεί τη λειτουργικότητα του κλασικού UNO, διαθέτει τον ίδιο μικροελεγκτή ATmega328P με 14 ψηφιακές ακίδες I/O (6 PWM), 6 αναλογικές εισόδους και ταλαντωτή 16MHz. Εξοπλισμένο με υποδοχή USB-C και κουμπί επαναφοράς, παρέχει απρόσκοπτη συνδεσιμότητα και ευκολία στη χρήση. Η τάση λειτουργίας είναι 5V και διαθέτει διάφορες διεπαφές επικοινωνίας, όπως UART, I2C και SPI (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-10 Arduino Uno Mini Limited Edition
(*Arduino Docs | Arduino Documentation, n.d.*)

2. Οικογένεια Arduino Nano:

Η οικογένεια Arduino Nano περιλαμβάνει μια σειρά ευέλικτες πλακέτες σχεδιασμένες για έργα όπου ο χώρος και η φορητότητα είναι βασικά κριτήρια. Παρά το μικρό τους μέγεθος, οι πλακέτες Nano προσφέρουν ένα πλούσιο σύνολο λειτουργιών και συμβατότητα με το οικοσύστημα Arduino, καθιστώντας τις ιδανικές για ένα ευρύ φάσμα εφαρμογών.

- Arduino Nano 33 IoT

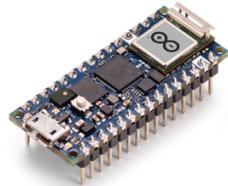
To Arduino Nano 33 IoT προσφέρει τη δυνατότητα ενσωμάτωσης συσκευών σε δίκτυα IoT. Διαθέτει επεξεργαστή Arm Cortex-M0 32-bit SAMD21 χαμηλής κατανάλωσης και παρέχει συνδεσιμότητα WiFi και Bluetooth. Η τάση λειτουργίας είναι 3,3V και διαθέτει ψηφιακές ακίδες I/O, ακίδες PWM, διεπαφές UART, SPI και I2C, καθώς και αναλογικές ακίδες εισόδου και εξόδου (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4- 11 Arduino Nano 33 IoT
(*Arduino Docs | Arduino Documentation, n.d.*)

- Arduino Nano RP2040

To Arduino Nano RP2040 συνδυάζει τη δύναμη του RP2040 του Raspberry Pi με την ευελιξία της σειράς Nano του Arduino. Λειτουργεί με έναν διπύρηνο επεξεργαστή ARM Cortex-M0+ χρονισμένο στα 133MHz, με 264KB SRAM και 16MB εξωτερικής μνήμης flash. Προσφέρει συνδεσιμότητα WiFi 802.11b/g/n, Bluetooth και Bluetooth Low Energy v4.2, επιτρέποντας την απρόσκοπτη ενσωμάτωση IoT και τη συμβατότητα με το Arduino IoT Cloud. Διαθέτει 22 ψηφιακές, 20 PWM και 8 αναλογικές ακίδες I/O (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-12 Arduino Nano RP2040
(*Arduino Docs | Arduino Documentation, n.d.*)

- Arduino Nano ESP32

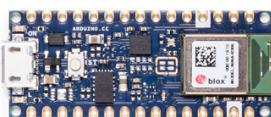
To Nano ESP32 ενσωματώνει τον μικροελεγκτή ESP32-S3 στο οικοσύστημα Arduino καθώς και δυνατότητες Wi-Fi, Bluetooth, παρέχοντας απρόσκοπτες επιλογές συνδεσιμότητας. To Nano ESP32 είναι ένα ισχυρό εργαλείο για την εξερεύνηση του κόσμου του IoT και της MicroPython (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-13 Arduino ESP32
(*Arduino Docs | Arduino Documentation, n.d.*)

- Arduino Nano 33 BLE

To Arduino Nano 33 BLE λειτουργεί με τον μικροελεγκτή nRF52840 της Nordic Semiconductors και προσφέρει βελτιωμένες δυνατότητες επεξεργασίας σε σύγκριση με το παραδοσιακό Arduino Nano. Διαθέτει μνήμη flash 1MB και μνήμη RAM 256KB, καθώς και συνδεσιμότητα Bluetooth. Λειτουργεί στα 3,3V και διαθέτει υποδοχή micro-USB (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-14 Arduino Nano 33 BLE
(*Arduino Docs | Arduino Documentation, n.d.*)

- Arduino Nano 33 BLE Sense

Το Arduino Nano 33 BLE Sense έχει σχεδιαστεί για έργα με δυνατότητα AI στον τομέα του IoT. Διαθέτει ένα ευρύ φάσμα ενσωματωμένων αισθητήρων, όπως αισθητήρα υγρασίας και θερμοκρασίας, βαρομετρικό αισθητήρα, μικρόφωνο και αισθητήρα χειρονομίας, εγγύτητας, χρώματος φωτός και έντασης φωτός. Με τον μικροελεγκτή nRF52840 της Nordic Semiconductors, το Nano 33 BLE Sense προσφέρει βελτιωμένες δυνατότητες επεξεργασίας σε σύγκριση με το παραδοσιακό Arduino Nano. Έχει μνήμη flash 1MB και μνήμη RAM 256KB και το κύριο χαρακτηριστικό του είναι η δυνατότητά του να εκτελεί εφαρμογές Edge Computing (AI) χρησιμοποιώντας το TinyML. Υποστηρίζει μοντέλα μηχανικής μάθησης που δημιουργούνται με τη χρήση του TensorFlow Lite, επιτρέποντας στους χρήστες να αναπτύσσουν αλγορίθμους AI απευθείας στην πλακέτα. Λειτουργεί στα 3,3V και διαθέτει υποδοχή micro-USB (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-15 Arduino Nano 33 BLE Sense
(Arduino Docs | Arduino Documentation, n.d.)

- Arduino Nano Every

Το Nano Every είναι ο διάδοχος του παραδοσιακού Arduino Nano. Διαθέτει έναν πιο ισχυρό επεξεργαστή, τον ATMega4809, προσφέροντας αυξημένη μνήμη προγράμματος και RAM για το χειρισμό μεγαλύτερων έργων και μεγαλύτερου αριθμού μεταβλητών. Η τάση λειτουργίας είναι στα 5V και παρέχει διάφορες επιλογές συνδεσιμότητας, όπως UART, SPI και I2C (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-16 Arduino Nano Every
(*Arduino Docs | Arduino Documentation, n.d.*)

3. Οικογένεια Arduino Mega

Η οικογένεια Arduino Mega αντιπροσωπεύει μια σειρά από ισχυρές πλακέτες μικροελεγκτών σχεδιασμένες για να χειρίζονται σύνθετα έργα που απαιτούν πλήθος εισόδων και εξόδων. Οι πλακέτες της οικογένειας Mega είναι εξοπλισμένες με επεξεργαστές, όπως ο ATmega2560, παρέχοντας άφθονη μνήμη προγράμματος και RAM για την υλοποίηση έργων μεγάλης κλίμακας.

- Arduino Mega 2560 Rev3

Το Arduino Mega 2560 διαθέτει τον μικροελεγκτή ATmega2560 καθώς και 54 ψηφιακές ακίδες I/O, 16 αναλογικές εισόδους και 4 UART. Είναι εξοπλισμένο με ταλαντωτή 16 MHz, σύνδεση USB, υποδοχή τροφοδοσίας και κουμπί επαναφοράς (*Arduino Docs | Arduino Documentation, n.d.*).



Εικόνα 4-17 Arduino Mega 2560 Rev3
(*Arduino Docs | Arduino Documentation, n.d.*)

- Arduino Due

Το Arduino Due διαθέτει CPU Atmel SAM3X8E ARM Cortex-M3, με πυρήνα ARM 32-bit. Διαθέτει 54 ψηφιακές ακίδες I/O, 12 αναλογικές εισόδους και 4 UART, προσφέρει εκτεταμένες επιλογές συνδεσιμότητας για ποικίλα έργα. Με ταχύτητα ρολογιού 84 MHz και 512 KB μνήμης flash, το Due παρέχει άφθονη



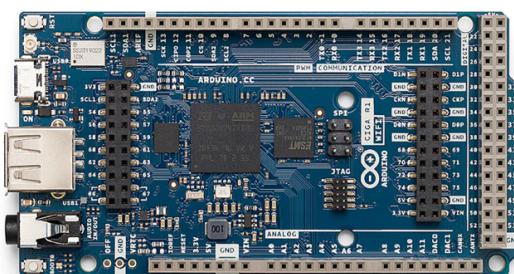
επεξεργαστική ισχύ και χωρητικότητα αποθήκευσης για απαιτητικές εφαρμογές
(Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-18 Arduino Due
(*Arduino Docs | Arduino Documentation, n.d.*)

- **Arduino Giga R1 WiFi**

To Arduino GIGA R1 WiFi διαθέτει διπύρηνο μικροελεγκτή 32-bit STM32H747XI, με CPUs Cortex®-M7 και Cortex®-M4, προσφέροντας δυνατότητες επεξεργασίας στα 480 MHz και 240 MHz αντίστοιχα. Εξοπλισμένο με WiFi και Bluetooth καθώς και μια σειρά από θύρες όπως 4 UART, 3 I2C, 2 SPI και FDCAN (Flexible Data-Rate Controller Area Network), παρέχει εκτεταμένες επιλογές διασύνδεσης. Προσφέρει 76 ακίδες GPIO, υποδοχές USB-A και USB-C®, υποδοχή κάμερας Arducam 20 ακίδων κ.α (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-19 Arduino Giga R1 WiFi
(*Arduino Docs | Arduino Documentation, n.d.*)

4. Οικογένεια Arduino MKR

Η οικογένεια Arduino MKR αντιπροσωπεύει μια σειρά από ισχυρές και ενεργειακά αποδοτικές πλακέτες προσαρμοσμένες για εφαρμογές IoT (Internet of Things). Είναι κατά κύριο λόγο φορητές συσκευές και χρησιμοποιούνται σε έργα χαμηλής κατανάλωσης ενέργειας. Διαθέτουν προηγμένες επιλογές συνδεσιμότητας όπως π.χ. το Wi-Fi – GSM και στα ισχυρά χαρακτηριστικά τους εντάσεται το χαμηλό κόστος και η ενεργειακή απόδοση.

- Arduino MKR Zero

To MKR ZERO διαθέτει MCU SAMD21 της Atmel με πυρήνα 32-bit ARM Cortex M0 καθώς επίσης ενσωματωμένη υποδοχή SD και διεπαφές SPI διευκολύνοντας τους χρήστες να εξερευνήσουν μουσικά αρχεία χωρίς πρόσθετο υλικό. Λειτουργεί στα 3,3V, και διαθέτει 22 ψηφιακούς ακροδέκτες I/O, 12 ακροδέκτες PWM και 7 ακροδέκτες αναλογικής εισόδου (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-20 Arduino MKR Zero
(Arduino Docs | Arduino Documentation, n.d.)

- Arduino MKR Vidor 4000

To Arduino MKR Vidor 4000 διαθέτει τον πανίσχυρο Intel Cyclone 10CL016 FPGA και 504 KB ενσωματωμένης μνήμης RAM. Κάθε ακροδέκτης μπορεί να εναλλάσσεται σε πάνω από 150 MHz και μπορεί να διαμορφωθεί για διάφορες λειτουργίες όπως UART, PWM, I2C και άλλες. Επίσης διαθέτει μια σειρά από υποδοχές, όπως Micro HDMI για έξοδο βίντεο, υποδοχή κάμερας MIPI (Mobile Industry Processor Interface) για είσοδο βίντεο και μια θύρα Mini PCI Express με 25 προγραμματιζόμενες από τον χρήστη ακίδες. Τέλος παρέχει συνδεσιμότητα WiFi/Bluetooth (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-21 Arduino MKR Vidor 4000
(Arduino Docs | Arduino Documentation, n.d.)



- Arduino MKR NB 1500

Το Arduino MKR NB 1500 διαθέτει μικροελεγκτή Arm® Cortex-M0 SAMD21 χαμηλής κατανάλωσης ενέργειας και χρησιμοποιεί τη μονάδα u-blox SARA-R410M-02B για Narrowband επικοινωνία, εξασφαλίζοντας αποτελεσματική συνδεσιμότητα στο εύρος κυψελωτών δικτύων IoT LTE. Παρέχει 8 ψηφιακές ακίδες I/O, 13 ακίδες PWM και διάφορες διεπαφές επικοινωνίας, το MKR NB 1500 προσφέρει ευελιξία για ένα ευρύ φάσμα εφαρμογών IoT (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-22 Arduino MKR NB 1500
(*Arduino Docs | Arduino Documentation, n.d.*)

- Arduino MKR WAN 1300

Το Arduino MKR WAN 1300 είναι η αναβάθμιση του MKR WAN 1300. Διαθέτει Microchip SAMD21 χαμηλής κατανάλωσης. Διαθέτει συνδεσιμότητα LoRa και τσιπ κρυπτογράφησης ECC508. Επίσης παρέχει 2MByte SPI Flash για ενσωματωμένη αποθήκευση και μπορεί να λειτουργεί με ή χωρίς μπαταρίες, ενώ η θύρα USB μπορεί να χρησιμοποιηθεί για την παροχή ρεύματος (5V). Η πλακέτα είναι συμβατή με το Arduino IoT Cloud, διευκολύνοντας την ασφαλή επικοινωνία για συνδεδεμένα έργα (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-23 Arduino MKR WAN 1310
(*Arduino Docs | Arduino Documentation, n.d.*)

- Arduino MKR WAN 1300

Το Arduino MKR WAN 1300 έχει σχεδιαστεί για να παρέχει μια προσιτή λύση για κατασκευαστές που επιθυμούν να ενσωματώσουν τη συνδεσιμότητα LoRa στα έργα



τους με ελάχιστη εμπειρία δικτύωσης. Διαθέτει υπολογιστική ισχύ 32 bit παρόμοια με την πλακέτα MKR ZERO και πλούσιο σύνολο διεπαφών I/O, συμπεριλαμβανομένης της επικοινωνίας LoRa χαμηλής κατανάλωσης. Το MKR WAN 1300 προσφέρει 8 ψηφιακές ακίδες I/O, 12 ακίδες PWM, 1 UART, 1 SPI, 1 I2C και 7 ακίδες αναλογικής εισόδου. Διαθέτει 256 KB μνήμης flash και 32 KB SRAM, με ταχύτητα ρολογιού 32,768 kHz (RTC) και 48 MHz (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-24 Arduino MKR WAN 1300
(*Arduino Docs | Arduino Documentation, n.d.*)

- Arduino MKR 1000

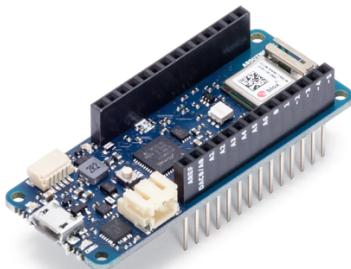
Το Arduino MKR 1000 αποτελεί προσιτή λύση για την ενσωμάτωση της συνδεσιμότητας Wi-Fi σε έργα με ελάχιστη εμπειρία δικτύωσης. Λειτουργεί με Cortex-M0+ 32-bit χαμηλής ισχύος, μια μονάδα Wi-Fi WINC1500 χαμηλής ισχύος 2,4GHz IEEE 802.11 b/g/n και ένα τσιπ ECC508 CryptoAuthentication για ασφαλή επικοινωνία. Η πλακέτα λειτουργεί στα 3,3V και παρέχει 8 ψηφιακές ακίδες I/O, 12 ακίδες PWM, διεπαφές UART, SPI, I2C, 7 ακίδες αναλογικής εισόδου και 1 ακίδα αναλογικής εξόδου (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-25 Arduino MKR 1000
(*Arduino Docs | Arduino Documentation, n.d.*)

- Arduino MKR 1010

To Arduino MKR WiFi 1010 διαθέτει μικροελεγκτή Arm Cortex-M0 32-bit SAMD21 χαμηλής κατανάλωσης και συνδεσιμότητα u-blox NINA-W10 Wi-Fi και Bluetooth. To MKR WiFi 1010 υποστηρίζει διάφορες υπηρεσίες cloud και λειτουργίες Bluetooth μέσω της βιβλιοθήκης ArduinoBLE. Λειτουργεί με μπαταρία Li-Po ή εξωτερική πηγή 5V (Arduino Docs | Arduino Documentation, n.d.).



Εικόνα 4-26 Arduino MKR 1010
(Arduino Docs | Arduino Documentation, n.d.)

4.2 Raspberry Pi

4.2.1 Εκδόσεις Raspberry Pi

Η οικογένεια του Raspberry Pi περιλαμβάνει αρκετές εκδόσεις του διάσημου υπολογιστή μικρού μεγέθους. Ο αριθμός που ακολουθεί το όνομα Raspberry Pi καθορίζει την σειρά με την οποία αναπτύχθηκε η κάθε πλακέτα με πρώτη να είναι το Pi 1 και τελευταία το Pi 5, ενώ το μοντέλο A και B καθορίζει την έκδοση της κάθε υλοποίησης, με την πρώτη να είναι πιο ελαφριά σε σχέση με τη δεύτερη.

- Raspberry Pi 1 Model B

To Raspberry Pi 1 Model B κυκλοφόρησε το 2012 και ήταν το πρώτο μοντέλο της σειράς Raspberry Pi. Τροφοδοτούμενο από ένα σύστημα-σε-τσιπ (SoC) Broadcom BCM2835 με επεξεργαστή ARM11, το Pi 1 Model B προσέφερε μια προσιτή πλατφόρμα για την εκμάθηση προγραμματισμού, τον πειραματισμό με ηλεκτρονικά και την εκτέλεση βασικών εφαρμογών (Raspberry Pi Foundation | Raspberry Documentation, n.d.).

Βασικά χαρακτηριστικά	Raspberry Pi 1 Model B
CPU	Broadcom BCM2835 (ARM11)



RAM	512MB
Θήρες USB	2
Εξόδοι	HDMI, Composite Video
Είσοδοι	26 GPIO pins, Camera, Display
Διεπαφές	Ήχος, Ethernet, Micro SD

Πίνακας 4-1 Χαρακτηριστικά Pi 1 Model B
(Raspberry Pi Foundation | Raspberry Documentation, n.d.)



Εικόνα 4-27 Raspberry Pi 1 Model B
(Raspberry Pi Foundation | Raspberry Documentation, n.d.)

- Raspberry Pi 2 Model B

To Raspberry Pi 2 Model B κυκλοφόρησε το 2015 και παρουσίασε σημαντικές αναβαθμίσεις σε σχέση με τον προκάτοχό του, διαθέτοντας τετραπύρηνο επεξεργαστή ARM Cortex-A7 και 1GB RAM. Αυτές οι βελτιώσεις είχαν ως αποτέλεσμα βελτιωμένες επιδόσεις και διευρυμένες δυνατότητες, καθιστώντας το Pi 2 Model B κατάλληλο για πιο απαιτητικές εργασίες, όπως η αναπαραγωγή πολυμέσων και η εξομοίωση παιχνιδιών (Raspberry Pi Foundation | Raspberry Documentation, n.d.).

Βασικά χαρακτηριστικά	Raspberry Pi 2 Model B
CPU	Broadcom BCM2836 (Quad-core ARM Cortex-A7)
RAM	1GB



Θήρες USB	4
Έξοδοι	HDMI, Composite Video
Είσοδοι	40 GPIO pins, Camera, Display
Διεπαφές	Ήχος, Ethernet, Micro SD

Πίνακας 4-2 Χαρακτηριστικά Pi 2 Model B
(Raspberry Pi Foundation | Raspberry Documentation, n.d.)



Εικόνα 4-28 Raspberry Pi 2 Model B
(Raspberry Pi Foundation | Raspberry Documentation, n.d.)

- Raspberry Pi 3 Model B

To Raspberry Pi 3 Model B κυκλοφόρησε το 2016 και συνέχισε την τάση βελτίωσης των επιδόσεων με τετραπύρηνο επεξεργαστή ARM Cortex-A53 και ενσωματωμένη συνδεσιμότητα Wi-Fi και Bluetooth. Αυτές οι βελτιώσεις κατέστησαν το Pi 3 Model B ιδανική επιλογή για έργα IoT, εφαρμογές δικτύωσης και ροή πολυμέσων (Raspberry Pi Foundation | Raspberry Documentation, n.d.).

Βασικά χαρακτηριστικά	Raspberry Pi 3 Model B
CPU	Broadcom BCM2837 (Quad-core ARM Cortex-A53)
RAM	1GB
Θήρες USB	4
Έξοδοι	HDMI, Composite Video
Είσοδοι	40 GPIO pins, Camera, Display



Διεπαφές	Ηχος, Ethernet, Micro SD
----------	--------------------------

Πίνακας 4-3 Χαρακτηριστικά Pi 3 Model B
(Raspberry Pi Foundation | Raspberry Documentation, n.d.)



Εικόνα 4-29 Raspberry Pi 3 Model B
(Raspberry Pi Foundation | Raspberry Documentation, n.d.)

- Raspberry Pi 4 Model B

To Raspberry Pi 4 Model B, που κυκλοφόρησε το 2019, αποτέλεσε ένα σημαντικό άλμα προς τα εμπρός σε επιδόσεις και χαρακτηριστικά. Με επιλογές για 2GB, 4GB και 8GB μνήμης RAM, τετραπύρηνο επεξεργαστή Cortex-A72 και υποστήριξη για δύο οθόνες 4K, το Pi 4 Model B έγινε μια ευέλικτη πλατφόρμα για επιτραπέζιους υπολογιστές, πολυμέσα και εφαρμογές IoT (Raspberry Pi Foundation | Raspberry Documentation, n.d.).

Βασικά χαρακτηριστικά	Raspberry Pi 4 Model B
CPU	Broadcom BCM2711 (Quad-core ARM Cortex-A72)
RAM	2GB, 4GB, 8GB
Θήρες USB	2 USB 2.0, 2 USB 3.0
Έξοδοι	2 HDMI (4K), 3.5mm Audio Jack
Είσοδοι	40 GPIO pins, Camera, Display
Διεπαφές	Ηχος, Gigabit Ethernet, Micro SD

Πίνακας 4-4 Χαρακτηριστικά Pi 4 Model B
(Raspberry Pi Foundation | Raspberry Documentation, n.d.)



Εικόνα 4-30 Raspberry Pi 4 Model B

(Raspberry Pi Foundation | Raspberry Documentation, n.d.)

- Raspberry Pi 5

Τον Οκτώβριο του 2023 έγινε η επίσημη κυκλοφορία του Raspberry Pi 5 το οποίο διαθέτει έναν τετραπύρηνο επεξεργαστή 64-bit Arm Cortex-A76 που λειτουργεί στα 2,4 GHz. Σε σύγκριση με τον προκάτοχο του, το Raspberry Pi 5 προσφέρει 2-3 φορές μεγαλύτερη απόδοση στην CPU και σημαντική αύξηση επιδόσεων στα γραφικά λόγω της GPU η οποία είναι χρονισμένη στα 800MHz και προσφέρει διπλή έξοδο οθόνης με 4Kp60 μέσω HDMI (Raspberry Pi Foundation | Raspberry Documentation, n.d.).

Βασικά χαρακτηριστικά	Raspberry Pi 5
CPU	Broadcom BCM2712 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU
GPU	VideoCore VII GPU, supporting OpenGL ES 3.1
Video Output	Dual 4Kp60 HDMI® display output with HDR support
Video Decoder	4Kp60 HEVC decoder
RAM	LPDDR4X-4267 SDRAM 4GB and 8GB
Wireless Connectivity	Dual-band 802.11ac Wi-Fi®, Bluetooth 5.0
Storage	microSD card slot

Πίνακας 4-5 Χαρακτηριστικά Pi 5

(Raspberry Pi Foundation | Raspberry Documentation, n.d.)



Εικόνα 4-31 Raspberry Pi 5
(*Raspberry Pi Foundation | Raspberry Documentation, n.d.*)

4.2.2 Εκδόσεις Raspberry Pi Pico

Οι εκδόσεις του Raspberry Pi Pico (πίνακες 4-6 και 4-7) προσφέρουν επιλογές προσαρμογής στις ανάγκες του χρήστη, διατηρώντας παράλληλα την ίδια αρχιτεκτονική και τις βασικές λειτουργίες. Η αρχική έκδοση του Raspberry Pi Pico προσφέρει έναν οικονομικό και ευέλικτο μικροελεγκτή, ιδανικό για DIY έργα και προγραμματιστική μάθηση. Η έκδοση Pico H προσφέρει την ίδια λειτουργικότητα με τον αρχικό Pico, με την προσθήκη προ-συγκολλημένων ακίδων. Από την άλλη, το Raspberry Pi Pico W προσφέρει ενσωματωμένη ασύρματη συνδεσιμότητα Wi-Fi, επιτρέποντας στους χρήστες να δημιουργήσουν εφαρμογές IoT που απαιτούν ασύρματη επικοινωνία. Τέλος, η έκδοση Pico WH προσφέρει τις ίδιες λειτουργίες με το Pico W, αλλά με προ-συγκολλημένες ακίδες, προσφέροντας έτσι μια λύση έτοιμη προς χρήση. Με τις παραπάνω εκδόσεις, οι χρήστες έχουν τη δυνατότητα να επιλέξουν το κατάλληλο μοντέλο που εναρμονίζεται με τις απαιτήσεις και τις προδιαγραφές τους, διατηρώντας την ίδια ποιότητα και απόδοση του πρωτότυπου Raspberry Pi Pico. Στην εικόνα 4-32 παρουσιάζονται οι πλακέτες της οικογένειας Raspberry Pi Pico (*Raspberry Pi Foundation | Raspberry Documentation, n.d.*).

Raspberry Pi Pico και Raspberry Pi Pico H:

Εκδόσεις	Raspberry Pi Pico
Αρχιτεκτονική	Dual-core Arm Cortex-M0+ @ 133MHz
Μνήμη RAM	264KB
Ενσωματωμένη Αποθήκευση	2MB
Αριθμός GPIO Pins	26
Υποστήριξη Αναλογικών Εισόδων	Ναι



Τροφοδοσία

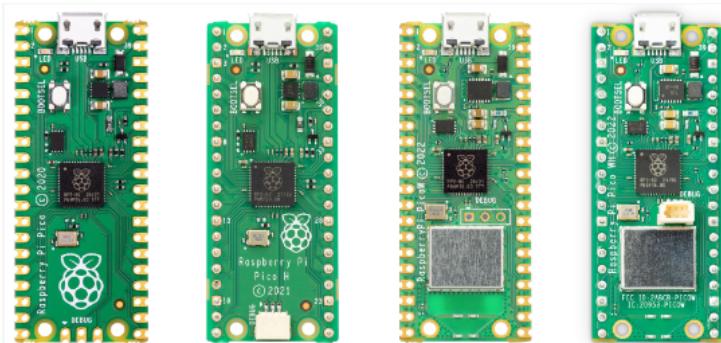
1.8–5.5V DC

Πίνακας 4-6 Χαρακτηριστικά Raspberry Pi Pico
(Raspberry Pi Foundation | Raspberry Documentation, n.d.)

Raspberry Pi Pico W και Raspberry Pi Pico WH:

Εκδόσεις	Raspberry Pi Pico W
Αρχιτεκτονική	Dual-core Arm Cortex-M0+ @ 133MHz
Μνήμη RAM	264KB
Ενσωματωμένη Αποθήκευση	2MB
Αριθμός GPIO Pins	26
Υποστήριξη Αναλογικών Εισόδων	Ναι
Τροφοδοσία	1.8–5.5V DC
Ασύρματη Συνδεσιμότητα	Ναι

Πίνακας 4-7 Χαρακτηριστικά Raspberry Pi Pico W
(Raspberry Pi Foundation | Raspberry Documentation, n.d.)



Εικόνα 4-32 Οικογένεια Raspberry Pico (Pico,H,W,WH)
(Raspberry Pi Foundation | Raspberry Documentation, n.d.)

5. Υλικό Συστήματος

5.1 Εισαγωγή

Στο κεφάλαιο αυτό γίνεται παρουσίαση του υλικού που θα χρησιμοποιηθεί κατά την υλοποίηση του συστήματος «Έξυπνης Κολώνας» δημόσιου φωτισμού, το οποίο σε συνδυασμό με τις τεχνολογίες που παρουσιάστηκαν στα κεφάλαια 2 έως 4, θα αποτελέσουν ένα ολοκληρωμένο σύστημα δημόσιου φωτισμού σε περιβάλλον μιας έξυπνης πόλης.

Το ολοκληρωμένο αυτό σύστημα θα αποτελεί ένα δίκτυο από «Έξυπνες Κολώνες», οι οποίες θα έχουν τη δυνατότητα να προσαρμόζουν τη φωτεινότητα τους με βάση της επικρατούσες συνθήκες του περιβάλλοντος χώρου με σκοπό την εξοικονόμηση ενέργειας χωρίς όμως να επηρεάζεται η ποιότητα των παρεχόμενων υπηρεσιών. Θα έχουν τη δυνατότητα να λαμβάνουν λογικές αποφάσεις για τη λειτουργία τους αξιολογώντας τις τιμές από τους συνδεδεμένους σε αυτές αισθητήρες. Παράλληλα θα έχουν τη δυνατότητα λήψης καιρικών δεδομένων. Επιπλέον το σύστημα θα έχει τη δυνατότητα να αναγνωρίζει κινδύνους για τη δημόσια ασφάλεια, όπως η αύξηση των επιπέδων επικίνδυνων αερίων στην ατμόσφαιρα, η αύξηση των επιπέδων των όμβριων υδάτων, η ύπαρξη πυρκαγιάς, η ύπαρξη μονοξειδίου του άνθρακα καθώς και η αύξηση των επιπέδων υπεριώδους ακτινοβολίας. Επίσης οι «έξυπνες κολώνες» του συστήματος θα έχουν τη δυνατότητα ασύρματης επικοινωνίας μεταξύ τους, καθώς και με τον κεντρικό εξυπηρετητή, αξιοποιώντας διαφορετικές τεχνολογίες ασυρμάτων επικοινωνιών παρουσιάζοντας με αυτό τον τρόπο τις ιδιαιτερότητες και τα οφέλη της κάθε υλοποίησης.

Στην πλευρά της αλληλεπίδρασης με τους χρήστες το ολοκληρωμένο σύστημα θα παρέχει την δυνατότητα απομακρυσμένης παρακολούθησης μέσω ιστοσελίδας. Συγκεκριμένα, η ιστοσελίδα θα παρουσιάζει τα δεδομένα που συλλέγονται από τους αισθητήρες σε πραγματικό χρόνο για την κατάσταση του περιβάλλοντος χώρου της κάθε έξυπνης κολώνας. Τα δεδομένα αυτά θα καταγράφονται σε κεντρικό εξυπηρετητή από όπου θα αξιοποιούνται μέσω γραφημάτων για την παρουσίαση ή την επεξεργασία τους. Επίσης, θα παρέχεται η δυνατότητα αποστολής των δεδομένων στην IOT πλατφόρμα ThingSpeak. Τέλος, το ολοκληρωμένο σύστημα θα παρέχει τη δυνατότητα έγκαιρης αναγνώρισης κρίσιμου για τη δημόσια ασφάλεια κινδύνου, μέσω της προβολής του κίνδυνου στο περιβάλλον διαχείρισης του συστήματος και μέσω κατάλληλων ειδοποιήσεων στον

διαχειριστή του συστήματος μέσω email, καθώς και μέσω της δυνατότητας απομακρυσμένης εποπτείας με τη χρήση κάμερας ασύρματης επικοινωνίας.

5.2 Arduino Uno

Η πλατφόρμα Arduino Uno αποτελεί μια ανοικτού κώδικα ηλεκτρονική πλατφόρμα που χαρακτηρίζεται από το φιλικό προς τον χρήστη λογισμικό και υλικό της. Οι πλατφόρμες Arduino μπορούν να διαβάζουν εισόδους, όπως διακόπτες ή αισθητήρες φωτός, και να τις μετατρέπουν σε εξόδους, όπως η ενεργοποίηση ενός LED ή η εκκίνηση ενός κινητήρα. Μέσω του περιβάλλοντος ανάπτυξης Arduino IDE πραγματοποιείται η επικοινωνία της πλατφόρμας με τον μικροελεγκτή, μέσω της αποστολής ενός συνόλου εντολών (instruction set). Το σύνολο αυτών των εντολών είναι αποτέλεσμα της δημιουργίας και της επεξεργασίας του κώδικα που αναπτύσσει ο προγραμματιστής στο Arduino IDE.

Στην παρούσα εργασία για την υλοποίηση της έξυπνης κολώνας επιλέχτηκε το Arduino Uno (Πίνακας 5-1), το οποίο αναλαμβάνει το ρόλο της συλλογής δεδομένων από τους συνδεδεμένους σε αυτό αισθητήρες, την επεξεργασία των δεδομένων αυτών και τη λήψη λογικών αποφάσεων με βάση τα συγκεκριμένα δεδομένα. Οι αποφάσεις του μικροεπεξεργαστή του Arduino Uno έχουν να κάνουν με τη λειτουργία του φωτισμού ανάλογα με τα δεδομένα του περιβάλλοντος χώρου. Επίσης μέσω των συνδεμένων module επικοινωνίας (Xbee / LoRa) τα δεδομένα των αισθητήρων αποστέλλονται στο Server για την περαιτέρω επεξεργασία και αποθήκευση τους στη βάση δεδομένων.

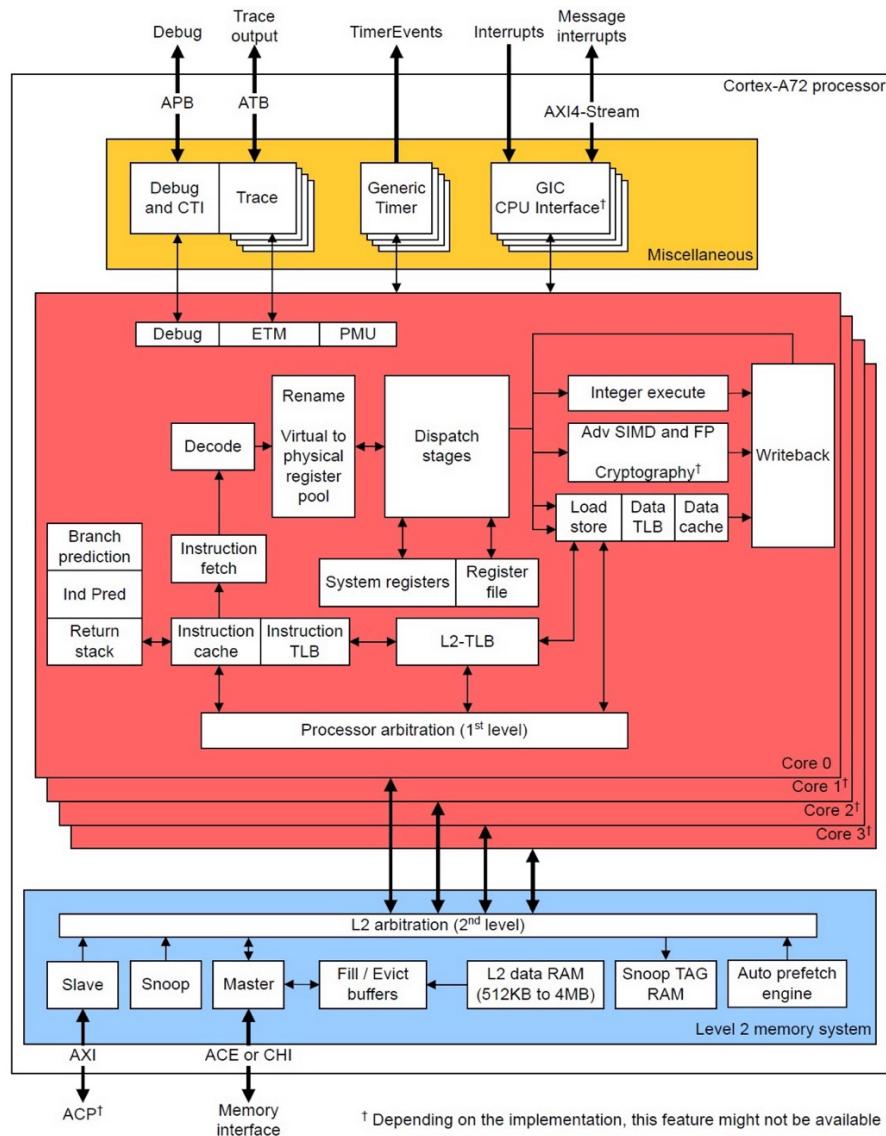
Πλακέτα	Pin Out

Πίνακας 5-1 Arduino Uno - Pin Out
(*Arduino Docs | Arduino Documentation, n.d.*)

5.3 Raspberry Pi 4 Model B

Στην υλοποίηση της παρούσας πτυχιακής εργασίας δημιουργήθηκε η ανάγκη φιλοξενίας των εξυπηρετητών ιστοσελίδας (webserver), εφαρμογής (application server) καθώς και της βάσης δεδομένων του συστήματος. Για την κάλυψη αυτών των αναγκών επιλέχτηκε το Raspberry Pi 4 Model B με 8GB RAM, λόγω των δυνατοτήτων του και των υψηλών του επιδόσεων καθώς και του μικρού του μεγέθους. Η συγκεκριμένη έκδοση του Raspberry pi 4 προσφέρει αυξημένη χωρητικότητα μνήμης, παρέχοντας στους χρήστες βελτιωμένες επιδόσεις και δυνατότητες multitasking για ένα ευρύ φάσμα εφαρμογών. Με τον ισχυρό επεξεργαστή, την άφθονη μνήμη και τις βελτιωμένες δυνατότητες εισόδου/εξόδου, το Raspberry Pi 4 Model B 8GB RAM είναι κατάλληλο για ένα ευρύ φάσμα έργων.

To Raspberry Pi 4 Model B 8GB RAM τροφοδοτείται από έναν τετραπύρηνο επεξεργαστή Broadcom BCM2711 Cortex-A72 (ARMv8-A) με χρονισμό 1,5 GHz (εικόνα 5-1), καθιστώντας το μέχρι την έκδοση του Raspberry Pi 5 (Οκτώβριος 2023) την πιο ισχυρή πλακέτα Raspberry Pi. Διαθέτει γραφικά VideoCore VI, ικανά να υποστηρίζουν δύο οθόνες 4K μέσω θυρών micro HDMI. Η μνήμη 8GB LPDDR4 SDRAM εξασφαλίζει ομαλότερη πολυεργασία (multi-tasking) και βελτιωμένη απόδοση σε εργασίες που απαιτούν αρκετή μνήμη σε σύγκριση με τις προηγούμενες εκδόσεις (Raspberry Pi Foundation | Raspberry Documentation, n.d.)..

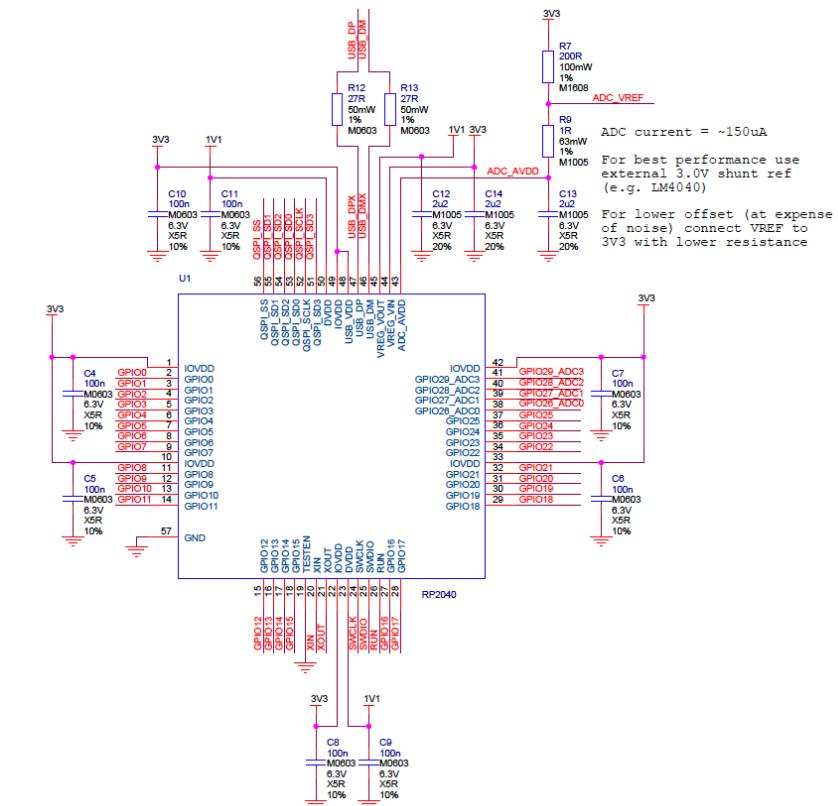


Εικόνα 5-1 Raspberry Pi 4 ARM Cortex-A72 processor
(Sandsoftwaresound n.d.)

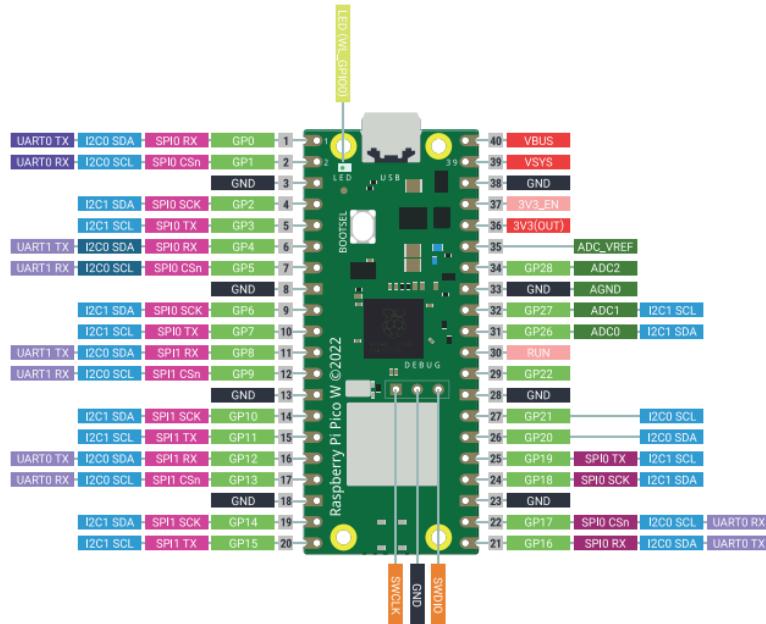
5.4 Raspberry Pi Pico W

To Raspberry Pi Pico W είναι ένας μικροελεγκτής που βασίζεται στον πυρήνα RP2040 της Raspberry Pi Foundation. Ο επεξεργαστής RP2040 (εικόνα 5-2) είναι ένας διπλού πυρήνα (dual-core) ARM Cortex-M0+ επεξεργαστής, με χρονισμό ρολογιού στα 133MHz, ο οποίος προσφέρει υψηλές επιδόσεις σε συνδυασμό με χαμηλή κατανάλωση ενέργειας. Επίσης διαθέτει 40 ακροδέκτες, μεταξύ των οποίων κάποιοι είναι UART, GPIO, PWM, I2C και SPI (εικόνα 5-3). Σε μερικές από τις υλοποιήσεις τις παρούσας πτυχιακής επιλέχτηκε το Raspberry Pi Pico λόγω των παραπάνω δυνατοτήτων που έχει έναντι του Arduion Uno, τόσο σε επίπεδο επεξεργαστή, μνήμης και συνδεσιμότητας, όσο και ότι παρέχει τη

δυνατότητα προγραμματισμού σε δύο γλώσσες (C++, MicroPython). Επίσης η επιλογή του συγκεκριμένου μικροελεγκτή (έκδοση W) έγινε λόγω του ότι διαθέτει δυνατότητα σύνδεσης σε δίκτυο WiFi με αποτέλεσμα να είναι ιδανικός για εφαρμογές IoT.



Εικόνα 5-2 Αρχιτεκτονική Pico Pi 5
(Raspberry Pi Foundation | Raspberry Documentation, n.d.)

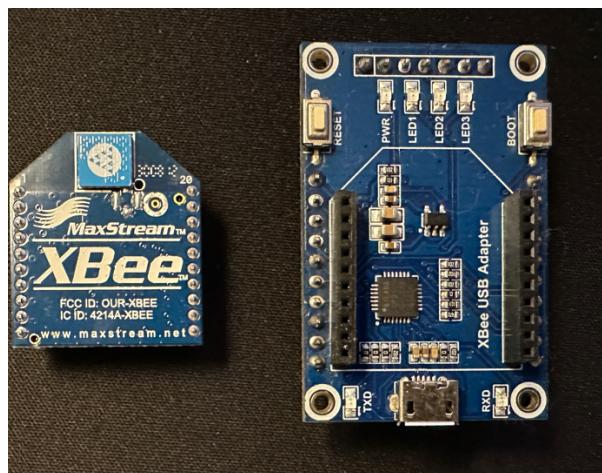


Εικόνα 5-3 Raspberry Pi Pico W Pinout

(Raspberry Pi Foundation | Raspberry Documentation, n.d.)

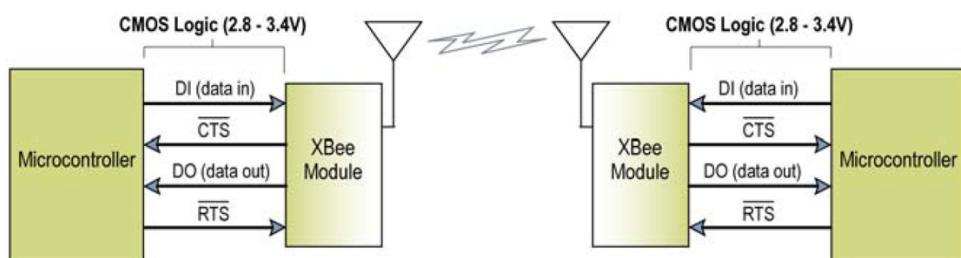
5.5 XBee S1

To XBee S1 (εικόνα 5-4) είναι μια ευέλικτη και αξιόπιστη μονάδα ασύρματης επικοινωνίας που χρησιμοποιείται ευρέως σε διάφορες εφαρμογές. Επικοινωνεί ασύρματα με άλλες συσκευές μέσω του πρωτοκόλλου 802.15.4, προσφέροντας μια αξιόπιστη μέθοδο για τη δημιουργία συνδέσεων μεταξύ διαφόρων συστημάτων ή μικροελεγκτών σε ένα δίκτυο, παρέχοντας απρόσκοπτη συνδεσιμότητα στη ζώνη συχνοτήτων 2,4 GHz με ρυθμό μετάδοσης δεδομένων έως 250 kbps. Η ισχύς εκπομπής φτάνει μέχρι τα 63 mW ενώ η εμβέλεια επικοινωνίας φτάνει έως τα 100 μέτρα σε εσωτερικούς χώρους και σχεδόν 1 χλμ σε εξωτερικούς χώρους, όταν δεν παρεμβάλλονται εμπόδια ενδιάμεσα. Η διασύνδεση του γίνεται μέσω UART (εικόνα 5-5) και η τροφοδοσία του S1 είναι στα 3.3V (SparkFun Electronics, n.d.).



Εικόνα 5-4 Xbee S1 και Xbee Usb Adapter

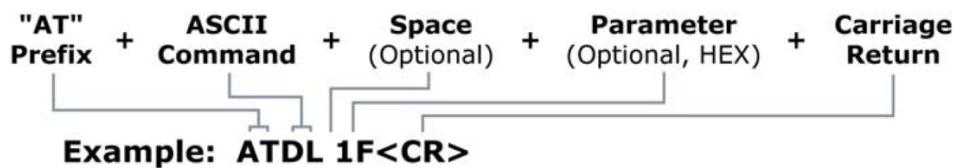
System Data Flow Diagram in a UART-interfaced environment
(Low-asserted signals distinguished with horizontal line over signal name.)



Εικόνα 5-5 Data Flow Diagram in a UART-interface
(SparkFun Electronics | SparkFun Electronics datasheets, n.d.)

Η διασύνδεση, η διαμόρφωση και ο χειρισμός του Xbee S1 γίνεται με χρήση του λογισμικού XCTU της Digi (κεφάλαιο 3.3), το οποίο παρέχει ένα φιλικό προς το χρήστη περιβάλλον για τη ρύθμιση παραμέτρων όπως ο ρυθμός baud, το PAN ID και ο τρόπος λειτουργίας. Επίσης, οι μονάδες Xbee υποστηρίζουν τη δυνατότητα διαμόρφωσης με χρήση εντολών εντολών AT, για την προσαρμογή ρυθμίσεων όπως ρυθμός baud, ισοτιμία, bit εκκίνησης, bit διακοπής και bit δεδομένων. Στην εικόνα 5-6 παρουσιάζεται η δομή που ακολουθείται για την αποστολή εντολών AT (SparkFun Electronics, n.d.).

Syntax for sending AT Commands



Εικόνα 5-6 Δομή εντολής AT
(SparkFun Electronics | SparkFun Electronics datasheets, n.d.)

Οι τρόποι επικοινωνίας μεταξύ των μονάδων XBee S1 για την ανταλλαγή δεδομένων γίνεται με δύο τρόπους, οι οποίοι καθορίζονται κατά τη διαμόρφωση του δικτύου και είναι οι εξής:

- Λειτουργία API (API mode):

Σε αυτή τη λειτουργία, τα δεδομένα αποστέλλονται σε πακέτα (frames) με συγκεκριμένες κεφαλίδες (frame headers), επιτρέποντας μεγαλύτερο έλεγχο της διαδικασίας επικοινωνίας (εικόνα 5-7).

- Διαφανής λειτουργία (Transparent Mode):

Αυτή η λειτουργία αντιμετωπίζει τα δεδομένα ως συνεχή ροή χωρίς πρόσθετη πλαισίωση, απλοποιώντας τη διαδικασία επικοινωνίας (SparkFun Electronics, n.d.).

UART Data Frame Structure:



MSB = Most Significant Byte, LSB = Least Significant Byte

Εικόνα 5-7 Δομή API frame
(SparkFun Electronics | SparkFun Electronics datasheets, n.d.)

Η χρήση των μονάδων XBee S1 με το Arduino UNO και το Raspberry Pi W στις υλοποιήσεις της παρούσας πτυχιακής εργασίας επεκτείνει τις δυνατότητες επικοινωνίας πέραν της χρήσης των δικτύων IEEE 802.3 (Ethernet) και IEEE 802.11 (WiFi). Για τη δημιουργία ενός δικτύου, μέσω τις εφαρμογής XCTU, γίνεται διαμόρφωση των μονάδων XBee, με ορισμό των κατάλληλων παραμέτρων, όπως το κανάλι (CH), το αναγνωριστικό PAN (ID) και οι ρυθμίσεις συντονιστή (coordinator) και τελικής συσκευής (endpoint).

Η μονάδα XBee S1 διαθέτει μηχανισμούς διαχείρισης σφαλμάτων κατά τη μετάδοση δεδομένων ώστε να διασφαλίσει αξιόπιστη επικοινωνία. Όταν δεν λαμβάνει ή δεν μεταδίδει δεδομένα, η μονάδα βρίσκεται σε κατάσταση αδράνειας και μεταβαίνει σε διάφορες καταστάσεις ανάλογα με συγκεκριμένες συνθήκες, όπως κατάσταση εκπομπής, κατάσταση λήψης, κατάσταση αναστολής λειτουργίας ή κατάσταση εντολών. Κατά τη μετάδοση των δεδομένων υπάρχει δυνατότητα επιλογής για άμεση μετάδοση όπου τα δεδομένα



μεταδίδονται αμέσως στη διεύθυνση προορισμού ή για έμμεση μετάδοση όπου τα δεδομένα διατηρούνται για ένα χρονικό διάστημα και μεταδίδονται μόνο αφού τα ζητήσει η συσκευή προορισμού. Αυτή η μέθοδος είναι χρήσιμη για τη διασφάλιση της παράδοσης πακέτων σε έναν κόμβο που είναι σε κατάσταση αδράνειας (sleep mode).

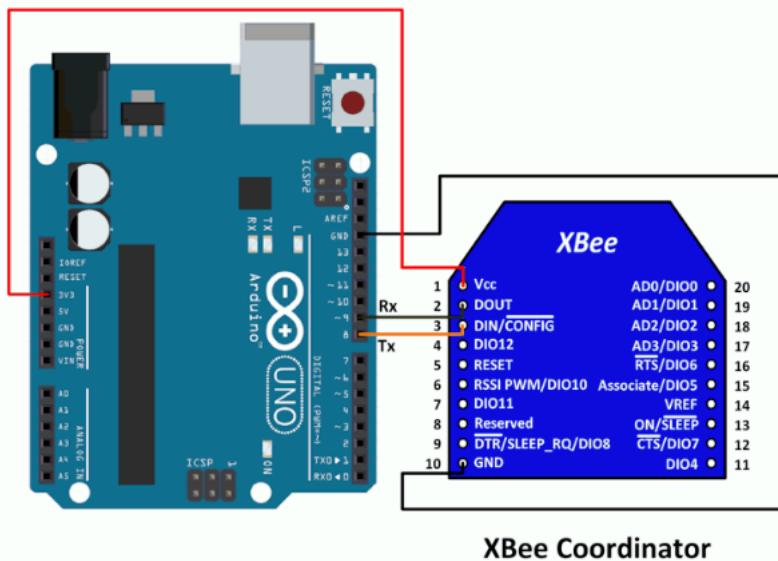
Οι μονάδες XBee λειτουργούν εξ ορισμού σε διαφανή λειτουργία (Transparent Mode), λειτουργώντας ως προέκταση της σειριακής γραμμής μετάδοσης δεδομένων, όπου όλα τα δεδομένα UART που λαμβάνονται μέσω του ακροδέκτη DI μπαίνουν σε ουρά για μετάδοση RF. Όταν λαμβάνονται δεδομένα RF, αποστέλλονται μέσω του ακροδέκτη DOUT. Με τη χρήση της επιλογής API mode οι μονάδες αποθηκεύουν τα δεδομένα, μέχρι ορισμένες συνθήκες ενεργοποιήσουν την πακετοποίηση και τη μετάδοση, όπως είναι η μη λήψη σειριακών χαρακτήρων για ένα καθορισμένο χρονικό διάστημα ή η λήψη μιας συγκεκριμένης ακολουθίας λειτουργίας εντολών. Στον πίνακα 5-2 παρουσιάζονται η κατανομή και οι χρήσεις του κάθε ακροδέκτη της μονάδας XBee S1 (SparkFun Electronics, n.d.).

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / CONFIG	Input	UART Data In
4	DO8*	Output	Digital Output 8
5	RESET	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	DTR / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	CTS / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / SLEEP	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	RTS / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

Πίνακας 5-2 Ακροδέκτες XBee S1
(SparkFun Electronics | SparkFun Electronics datasheets, n.d.)

Η διασύνδεση των μονάδων XBee με το Arduino (εικόνα 5-8) προσφέρει τη δυνατότητα της ασύρματης επικοινωνίας και ενισχύει την ευελιξία του συστήματος που θα υλοποιηθεί. Η διασύνδεση αυτή επιτυγχάνεται αξιοποιώντας την επικοινωνία UART, με χρήση των

ακροδεκτών RX/TX του Arduino. Η ελάχιστη διαμόρφωση που απαιτείται καθιστά την διασύνδεση των μονάδων XBee με τις πλακέτες Arduino εύκολη, επιτρέποντας στους χρήστες να επικεντρωθούν στη λογική της εφαρμογής και όχι στις περίπλοκες διαδικασίες ρύθμισης.



Εικόνα 5-8 Διασύνδεση Arduino - Xbee S1
(*Xbee Interfacing With Arduino UNO | Arduino, n.d.*)

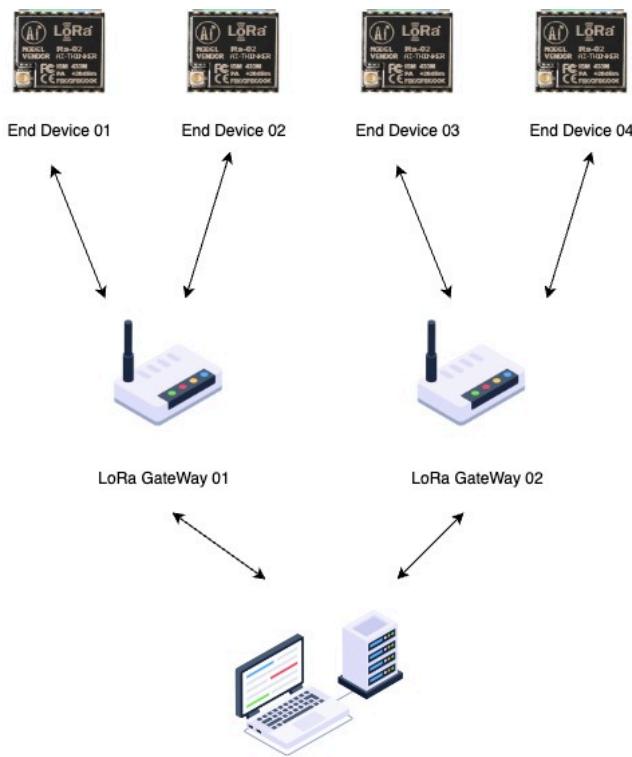
5.6 LoRa Ai Thinker

Η μονάδα LoRa Ai Thinker προσφέρει λύσεις στον τομέα της ασύρματης επικοινωνίας μεγάλης εμβέλειας, παρέχοντας ευελιξία και αποτελεσματικότητα για διάφορες εφαρμογές IoT. Στην παρούσα πτυχιακή εργασία γίνεται χρήση της συγκεκριμένης τεχνολογίας μέσα από υλοποιήσεις που έχουν ως στόχο τη διαμόρφωση του δικτύου των «έξυπνων κολώνων» δημόσιου φωτισμού. Η χρήση του συγκεκριμένου τρόπου επικοινωνίας καθώς και του τρόπου επικοινωνίας με τις μονάδες XBee που παρουσιάστηκαν στην προηγούμενη ενότητα, θα αποτελέσουν τη βάση σύγκρισης με την οποία θα καθοριστούν τα συμπεράσματα στον τομέα των επικοινωνιών του συστήματος όπως θα παρουσιαστούν στο τελευταίο κεφάλαιο της παρούσας πτυχιακής εργασίας.

Η μονάδα LoRa Ai Thinker λειτουργεί στις ζώνες συχνοτήτων 433MHz ή 868/915MHz, με εμβέλεια έως και αρκετά χιλιόμετρα σε αστικό περιβάλλον και ακόμη πιο μακριά σε αγροτικές περιοχές. Υποστηρίζει ρυθμούς δεδομένων που κυμαίνονται από 0,3 kbps έως 50

kbps, καλύπτοντας ποικίλες απαιτήσεις εφαρμογών. Η μονάδα έχει εξαιρετικά χαμηλή κατανάλωση ενέργειας εξασφαλίζοντας με αυτό τον τρόπο βέλτιστη απόδοση σε σενάρια λειτουργίας με μπαταρία (LoRa Alliance, n.d.).

Για να καταστεί δυνατή η επικοινωνία μεταξύ των μονάδων LoRa Ai Thinker, πρέπει να διαμορφωθεί μια υποδομή LoRaWAN (Long Range Wide Area Network). Το LoRaWAN είναι ένα πρωτόκολλο δικτύωσης χαμηλής ισχύος, ευρείας περιοχής, το οποίο έχει σχεδιαστεί για επικοινωνία μεγάλης εμβέλειας μεταξύ συσκευών IoT. Χρησιμοποιεί τοπολογία αστέρα (εικόνα 5-9), όπου οι τελικές συσκευές (endpoint) επικοινωνούν με κοντινές πύλες (gateway), οι οποίες με τη σειρά τους αναμεταδίδουν δεδομένα σε έναν κεντρικό διακομιστή δικτύου (LoRa Nerwork Server).

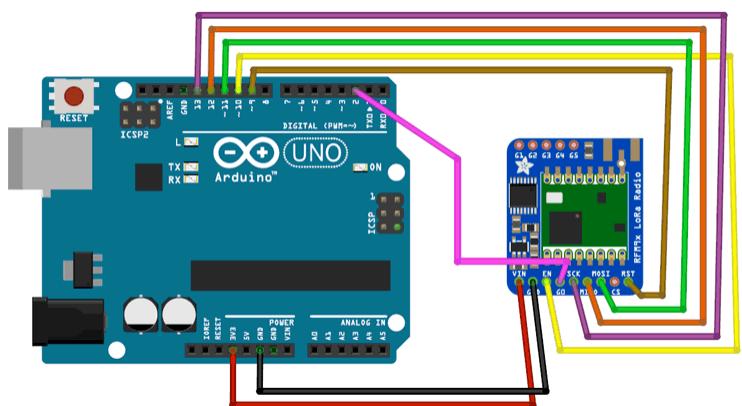


Εικόνα 5-9 LoRa Network star topology

Η μονάδα LoRa Ai Thinker προσφέρει διάφορους τρόπους επικοινωνίας, συμπεριλαμβανομένων των λειτουργιών Transparent και API. Στην παρούσα πτυχιακή εργασία, έχει γίνει χρήση της λειτουργίας API που επιτρέπει πιο προηγμένες λειτουργίες, όπως απομακρυσμένη διαμόρφωση, ενημερώσεις υλικολογισμικού και προσαρμοσμένο

χειρισμό πακέτων, παρέχοντας μεγαλύτερη ευελιξία και έλεγχο της διαδικασίας επικοινωνίας (LoRa Alliance, n.d.).

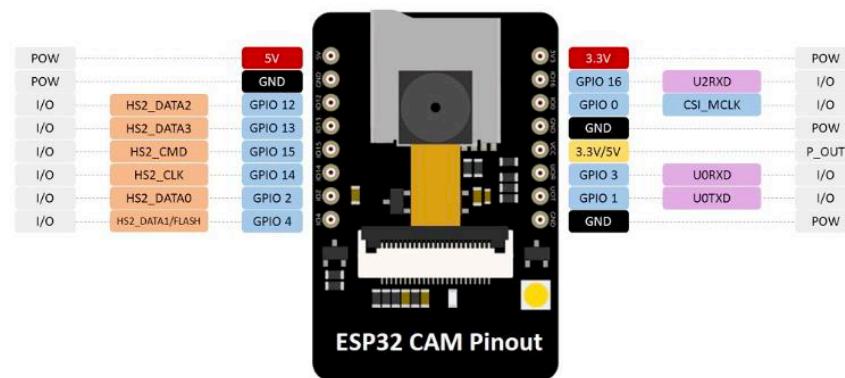
Η διασύνδεση της μονάδας LoRa Ai Thinker με το Arduino UNO (εικόνα 5-10) παρουσιάζει διάφορα πλεονεκτήματα. Το Arduino UNO παρέχει μια οικεία και προσβάσιμη πλατφόρμα για την κατασκευή πρωτοτύπων και την ανάπτυξη, με ένα τεράστιο οικοσύστημα βιβλιοθηκών και πόρων για την ενσωμάτωση του LoRa. Ωστόσο, υπάρχουν περιορισμοί, ιδίως όσον αφορά την επεξεργαστική ισχύ και τη μνήμη. Το Arduino UNO μπορεί να δυσκολεύεται να χειριστεί σύνθετες εργασίες επεξεργασίας δεδομένων ή να διαχειριστεί ταυτόχρονα πολλαπλές μονάδες LoRa. Επιπλέον, η έλλειψη ενσωματωμένων δυνατοτήτων δικτύωσης ενδέχεται να απαιτεί πρόσθετες λύσεις υλικού ή λογισμικού για την αποτελεσματική διασύνδεση με πύλες LoRaWAN. Η διασύνδεση της μονάδας LoRa Ai Thinker με το Raspberry Pi προσφέρει σαφή πλεονεκτήματα, ιδίως όσον αφορά την υπολογιστική ισχύ και τις δυνατότητες δικτύωσης. Το Raspberry Pi παρέχει σημαντικά μεγαλύτερη επεξεργαστική ισχύ και μνήμη σε σύγκριση με το Arduino UNO, επιτρέποντας την υλοποίηση πιο εξελιγμένων εφαρμογών και τη διαχείριση μεγαλύτερων όγκων δεδομένων. Ωστόσο, η διασύνδεση με το Raspberry Pi μπορεί να επιφέρει πολυπλοκότητα όσον αφορά την ανάπτυξη λογισμικού και τη διαμόρφωση του συστήματος, ιδίως για τους αρχάριους χρήστες. Επιπλέον, η υψηλότερη κατανάλωση ενέργειας του Raspberry Pi σε σύγκριση με το Arduino UNO μπορεί να αποτελέσει πρόβλημα σε εφαρμογές που λειτουργούν με μπαταρία (Hackster.io, 2023).



Εικόνα 5-10 Διασύνδεση Arduino - LoRa
(Hackster.io, 2023)

5.7 ESP 32 Ai Thinker Cam

Η μονάδα ESP32 Ai Thinker Cam συνδυάζει τα οφέλη του μικροελεγκτή ESP32 με μια μονάδα κάμερας ώστε να επιτρέπει την ενσωμάτωση δυνατότητας λήψης βίντεο σε έργα IoT. Στον πυρήνα του, το ESP32 Ai Thinker Cam διαθέτει τον μικροελεγκτή ESP32, με επεξεργαστική ισχύ διπλού πυρήνα, ενσωματωμένη συνδεσιμότητα Wi-Fi και Bluetooth και μια μονάδα κάμερας υψηλής ανάλυσης ικανή να καταγράφει εικόνες και βίντεο. Στην εικόνα 5-10 παρουσιάζεται το διάγραμμα των ακροδεκτών της μονάδας ESP32 Ai Thinker Cam.



Εικόνα 5-11 ESP32 Ai Thinker Cam PinOut (Lab, 2024)

Το ESP32 Ai Thinker Cam λειτουργεί με δύο τρόπους, ως αυτόνομο σύστημα και ως σύστημα σε δίκτυο. Στην αυτόνομη λειτουργία, λειτουργεί ως αυτόνομη συσκευή, καταγράφει εικόνες ή βίντεο και τα αποθηκεύει τοπικά στην κάρτα microSD. Αντίθετα, στη δικτυακή λειτουργία, αξιοποιεί τη συνδεσιμότητά της μέσω Wi-Fi για τη μετάδοση δεδομένων σε απομακρυσμένους διακομιστές ή την αλληλεπίδραση με άλλες συσκευές IoT, ανοίγοντας δυνατότητες για παρακολούθηση, επιτήρηση και ροή πολυμέσων σε πραγματικό χρόνο.

Η διασύνδεση του ESP32 Ai Thinker Cam τόσο με το Arduino UNO όσο και με το Raspberry Pi είναι εύκολη διότι υπάρχει πληθώρα παραδειγμάτων υλοποίησης στο διαδίκτυο καθώς επίσης και έτοιμες βιβλιοθήκες που διευκολύνουν την ανάπτυξη έργων.

Με την ενσωμάτωση του ESP32 Ai Thinker Cam στις πλατφόρμες Arduino και Raspberry αυξάνοντες οι δυνατότητές τους λόγω των προηγμένων λειτουργίων καταγραφής βίντεο. Επίσης η ενσωματωμένη συνδεσιμότητα Wi-Fi και Bluetooth επιτρέπει την ασύρματη



επικοινωνία και τον απομακρυσμένο έλεγχο, ενώ η μονάδα κάμερας διευκολύνει την απόκτηση και την επεξεργασία οπτικών δεδομένων. Στον πίνακα 5-3 παρουσιάζεται η μονάδα ESP32 Ai Thinker Cam καθώς και τα βασικά χαρακτηριστικά του (Lab, 2024).

ESP32 Ai Thinker Cam		CPU: ESP32-D0WD Μνήμη Flash: 32Mbit Μνήμη RAM: εσωτερική 512KB, εξωτερική 4M PSRAM Wi-Fi IEEE 802.11 b/g/n/e/i Bluetooth 4.2 BR/EDR and BLE Διεπαφές: UART/SPI/I2C/PWM
----------------------	---	---

Πίνακας 5-3 ESP32 Ai Thinker Cam

(NodeMCU-32S Core Development Board | Ai-Thinker, n.d.)

5.8 Αισθητήρες

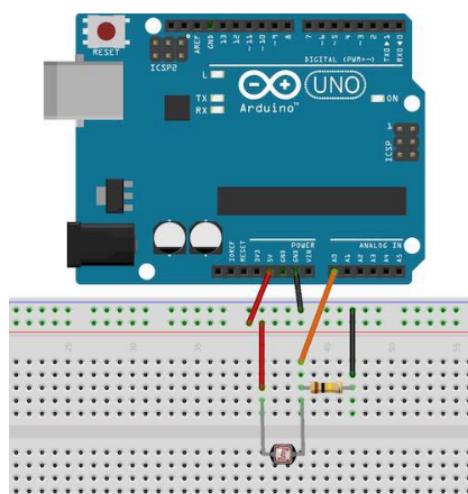
5.8.1 Αισθητήρας Φωτός (Φωτοαντίσταση LDR)

Οι φωτοεξαρτώμενες αντιστάσεις (LDR), χρησιμεύουν ως θεμελιώδη στοιχεία σε διάφορα ηλεκτρονικά κυκλώματα, επιτρέποντας στις συσκευές να αντιλαμβάνονται και να ανταποκρίνονται στις αλλαγές της έντασης του φωτός. Τα LDR θα χρησιμοποιηθούν για τον έλεγχο την ύπαρξης ηλιακής ακτινοβολίας στον περιβάλλοντα χώρο. Συγκεκριμένα μέσω της αντίστασης θα γίνεται μέτρηση του επιπέδου φωτός και ανάλογα με τα όρια που έχουν καθορισθεί θα λαμβάνονται αποφάσεις από το σύστημα για τον καθορισμό του επιπέδου φωτεινότητας του συστήματος φωτισμού. Στον πίνακα 5-4 παρουσιάζονται οι αισθητήρες και τα χαρακτηριστικά του, στην εικόνα 5-11 η συνδεσμολογία του και στον πίνακα 5-5 ενδεικτικός κώδικας για την επικοινωνία με Arduino.

LDR	Χαρακτηριστικά
-----	----------------

	<ul style="list-style-type: none"> • Αντίσταση (Φως): 0-20KOhm • Αντίσταση (Σκοτάδι): 1MOhm (Min) • Max voltage: 150V • Max power: 100mW
--	--

Πίνακας 5-4 Αισθητήρας LDR
(Grobtronics, n.d.)



Εικόνα 5-12 LDR Pinout

```
/**  
 * LDR example  
 */  
int LDR = A0; // LDR input at A0 pin.  
int LDRReading = 0; // to store input value of LDR  
  
void setup(){  
    Serial.begin(9600); // initializing serial  
communication.  
}  
void loop(){  
    LDRReading = analogRead(LDR); // Reading LDR Input.  
    Serial.println(LDRReading); // Printing LDR input  
value.  
    delay(300); // delay to make output  
// readable // on serial monitor.  
}
```

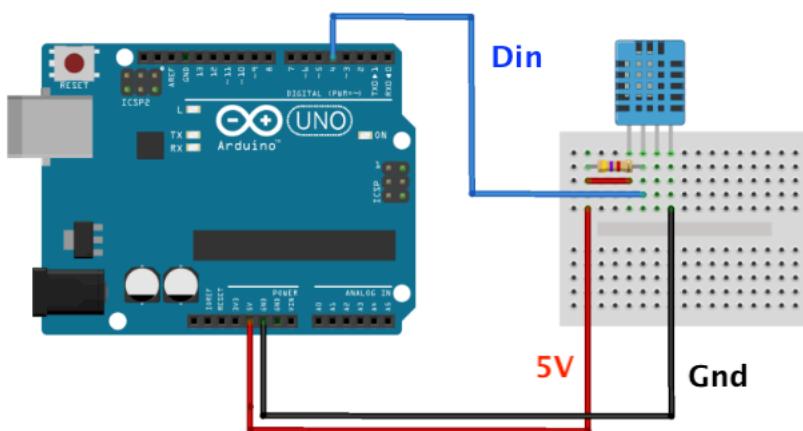
Πίνακας 5-5 Ενδεικτικός Κώδικας LDR
(Arduino Docs | Arduino Built in Examples, n.d.)

5.8.3 Αισθητήρας Θερμοκρασίας και Υγρασίας

Ο αισθητήρας DHT11 λειτουργεί ανιχνεύοντας τις μεταβολές της υγρασίας μετρώντας την ηλεκτρική αγωγιμότητα ενός φιλμ πολυμερούς ευαίσθητου στην υγρασία, ενώ τις μεταβολές της θερμοκρασίας με τη χρήση ενός θερμίστορ (τύπος αντίστασης, η τιμή της οποίας επηρεάζεται από τη θερμοκρασία). Με βάση τις συγκεκριμένες μεταβολές παράγει τα αντίστοιχα ψηφιακά σήματα στην έξοδο του. Στο σύστημα που θα υλοποιηθεί, οι μετρήσεις θερμοκρασίας και υγρασίας του περιβάλλοντος χώρου θα πραγματοποιηθούν με τη χρήση του αισθητήρα DHT11. Συγκεκριμένα, η ενσωμάτωση του αισθητήρα στο σύστημα θα επιτρέψει τη συλλογή δεδομένων θερμοκρασίας και υγρασίας της περιοχής γύρω από την κολώνα έξυπνου φωτισμού. Μέσω τις συλλογής των πληροφοριών από όλες τις περιοχές του δικτύου φωτισμού θα υπάρχει δυνατότητα αξιοποίησης των δεδομένων για περαιτέρω επεξεργασία καθώς και δημιουργία στατιστικών μετρήσεων για την κάθε περιοχή. Στον πίνακα 5-6 παρουσιάζονται ο αισθητήρας και τα χαρακτηριστικά του, στην εικόνα 5-12 η συνδεσμολογία του και στον πίνακα 5-7 ενδεικτικός κώδικας για την επικοινωνία με Arduino.

DHT 11	Χαρακτηριστικά
	<p>Απόκλιση (Θερμοκρασία): $\pm 2^{\circ}\text{C}$, Εύρος: $0^{\circ}\text{C} \sim 50^{\circ}\text{C}$</p> <p>Απόκλιση (Υγρασία): $\pm 5\%\text{RH}$ (0~50°C)</p> <p>Εύρος : $20\%\text{RH} \sim 90\%\text{RH}$ (25°C)</p> <p>Τάση λειτουργίας : $3.3\text{V} \sim 5.5\text{ V}$</p>

Πίνακας 5-6 Αισθητήρας DHT11
(Grobtronics, n.d.)



Εικόνα 5-13 DHT11 PinOut
(*Arduino Project Hub, n.d.*)

```
/*
 *  DHT11 example
 */

#define DHT11_PIN 10
void setup() {
    Serial.begin(115200);
}
void loop() {
    DHT.read(DHT11_PIN);
    Serial.print("temp:");
    Serial.print(DHT.temperature);
    Serial.print("  humi:");
    Serial.println(DHT.humidity);
    delay(1000);
}
```

Πίνακας 5-7 Ενδεικτικός Κώδικας DHT11
(*Arduino Docs | Arduino Built in Examples, n.d.*)

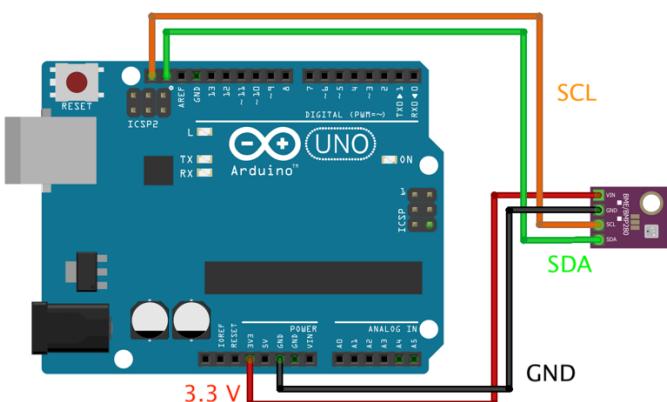
5.8.4 Αισθητήρας Βαρομετρικής Πίεσης BMP 280

Η λειτουργία του αισθητήρα BMP280 βασίζεται σε έναν αισθητήρα πίεσης πυριτίου για την ανίχνευση των αλλαγών της ατμοσφαιρικής πίεσης, μετρώντας τις αλλαγές της αντίστασης σε μια μεμβράνη πυριτίου. Επιπλέον, ενσωματώνει έναν αισθητήρα θερμοκρασίας για τη μέτρηση της θερμοκρασίας περιβάλλοντος. Ο αισθητήρας BMP280 προσφέρει μετρήσεις υψηλής ανάλυσης τόσο της βαρομετρικής πίεσης όσο και της θερμοκρασίας, επιτρέποντας την ακριβή εκτίμηση του υψομέτρου και των μετεωρολογικών δεδομένων. Η εξαιρετικά χαμηλή κατανάλωση ενέργειας και το ευρύ φάσμα τάσης λειτουργίας του τον καθιστούν

κατάλληλο για συσκευές που λειτουργούν με μπαταρία και φορητές εφαρμογές. Με τις διεπαφές I2C και SPI, ο αισθητήρας BMP280 μπορεί να ενσωματωθεί σε μια πληθώρα έργων που βασίζονται σε μικροελεγκτές, από μετεωρολογικούς σταθμούς και υψομετρητές έως drones και smartwatches. Στην παρούσα εργασία θα γίνει χρήση του αισθητήρα στην κολώνα έξυπνου φωτισμού, με σκοπό τη λήψη δεδομένων σχετικά με την βαρομετρική πίεση και τη θερμοκρασία του περιβάλλοντος χώρου του συστήματος. Στον πίνακα 5-8 παρουσιάζονται ο αισθητήρας και τα χαρακτηριστικά του, στην εικόνα 5-13 η συνδεσμολογία του και στον πίνακα 5-9 ενδεικτικός κώδικας για την επικοινωνία με Arduino.

BMP280	Χαρακτηριστικά
	Τροφοδοσία: 3.3 V Εύρος τιμών μέτρησης: from 300 hPa to 1100 hPa Ακρίβεια Μετρήσεων: 1 hPa Διεπαφή Επικοινωνίας: I2C or SPI

Πίνακας 5-8 Ο Αισθητήρας BMP280
(Grobtronics, n.d.)



Εικόνα 5-14 BMP280 PinOut
(Arduino Project Hub, n.d.)

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#define BME_SCK 13
#define BME_MISO 12
#define BME_MOSI 11
#define BME_CS 10

#define SEALEVELPRESSURE_HPA (1013.25)
Adafruit_BME280 bme; // I2C
unsigned long delayTime;

void setup() {
    Serial.begin(9600);
    while(!Serial); // time to get serial running
    Serial.println(F("BME280 test"));
    unsigned status;
    // default settings
    status = bme.begin();
}
Serial.println("-- Default Test --");
delayTime = 1000;
Serial.println();
}

void loop() {
    printValues();
    delay(delayTime);
}

void printValues() {
    Serial.print("Temperature = ");
    Serial.print(bme.readTemperature());
    Serial.println(" °C");
    Serial.print("Pressure = ");
    Serial.print(bme.readPressure() / 100.0F);
    Serial.println(" hPa");
    Serial.print("Approx. Altitude = ");
    Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
    Serial.println(" m");
    Serial.print("Humidity = ");
    Serial.print(bme.readHumidity());
    Serial.println(" %");
}
```

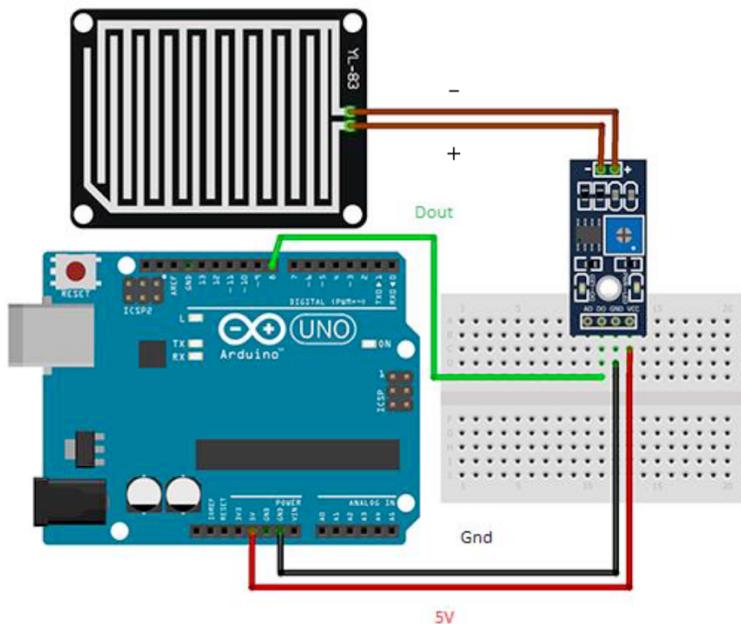
Πίνακας 5-9 Ενδεικτικός Κώδικας BMP280
(*Arduino Docs | Arduino Built in Examples, n.d.*)

5.8.5 Αισθητήρας Βροχής

Ο αισθητήρας βροχής LM393 βασίζει τη λειτουργία του στη αγωγιμότητα του νερού. Λειτουργεί σαν διακόπτης, ο οποίος παραμένει ανοιχτός για όσο δεν υπάρχει επαφή με το νερό, όταν όμως έρθει σε επαφή με το νερό τότε το κύκλωμα κλείνει και δίνεται σήμα ενεργοποίησης, το οποίο παράγεται από την έξοδο του (Ain/Din). Ο αισθητήρας βροχής θα χρησιμοποιηθεί για τον έλεγχο βροχόπτωσης. Συγκεκριμένα το σύστημα μέσω των τιμών του αισθητήρα θα ανιχνεύει βροχόπτωση και θα καταγράφει την έναρξη και τη λήξη του φαινομένου της βροχής. Τα δεδομένα θα συλλέγονται και θα αποστέλλονται στον κεντρικό διακομιστή. Η συγκεκριμένη έκδοση του αισθητήρα χρησιμοποιεί την πλακέτα ελέγχου LM393. Στον πίνακα 5-10 παρουσιάζονται ο αισθητήρας και τα χαρακτηριστικά του, στην εικόνα 5-14 η συνδεσμολογία του και στον πίνακα 5-11 ενδεικτικός κώδικας για την επικοινωνία με Arduino.

Αισθητήρας Βροχής	Χαρακτηριστικά
	<p>Τάση Λειτουργίας: 3.3VDC ~ 5VDC Έξοδοι: Αναλογικοί / Ψηφιακοί Ελεγχόμενη ευαισθησία με ποτενσιόμετρο.</p>

Πίνακας 5-10 Αισθητήρας Βροχής
(Grobotronics, n.d.)



Εικόνα 5-15 Αισθήτηρας Βροχής PinOut
(*Arduino Project Hub, n.d.*)

```
#define POWER_PIN 3 // The pin that provides the power to the rain
sensor
#define DO_PIN 4 // DO pin of the rain sensor

void setup() {
    // initialize serial communication
    Serial.begin(9600);
    // initialize the Arduino's pin as an input
    pinMode(POWER_PIN, OUTPUT); // configure the power pin pin as an
OUTPUT
    pinMode(DO_PIN, INPUT);
}

void loop() {
    digitalWrite(POWER_PIN, HIGH); // turn the rain sensor's power ON
    delay(10); // wait 10 milliseconds
    int rain_state = digitalRead(DO_PIN);
    digitalWrite(POWER_PIN, LOW); // turn the rain sensor's power OFF
    if (rain_state == HIGH)
        Serial.println("The rain is NOT detected");
    else
        Serial.println("The rain is detected");
    delay(1000); // pause for 1 sec to avoid reading sensors
                  //frequently to prolong the sensor lifetime
}
```



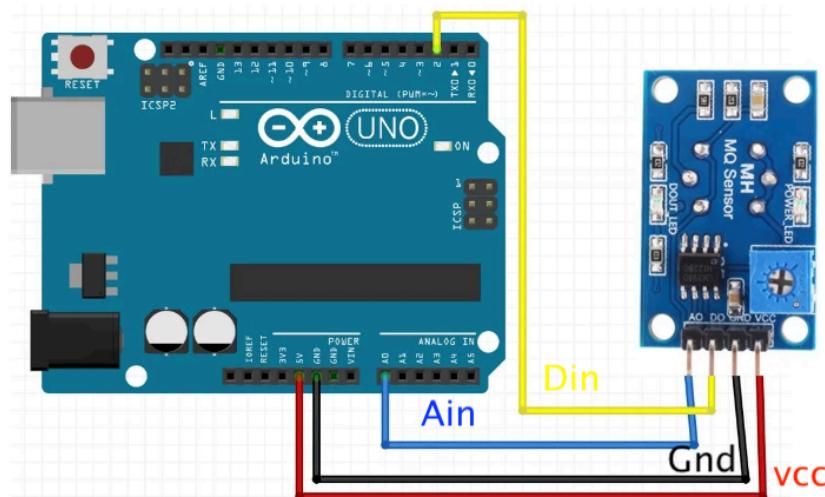
Πίνακας 5-11 Ενδεικτικός Κώδικας Αισθητήρα Βροχής
(*Arduino Docs | Arduino Built in Examples, n.d.*)

5.8.6 Αισθητήρας ύπαρξης επικίνδυνων αερίων

Ο αισθητήρας MQ135 λειτουργεί χρησιμοποιώντας έναν ημιαγωγό διοξειδίου του κασσιτέρου. Μετρά τις μεταβολές της αγωγιμότητας παρουσία διαφόρων αερίων, ανιχνεύοντας ιδιαίτερα ρύπους όπως διοξείδιο του άνθρακα, αμμωνία και βενζόλιο, καθιστώντας τον χρήσιμο για εφαρμογές παρακολούθησης της ποιότητας του αέρα. Ο αισθητήρας ύπαρξης αερίων MQ-135 θα χρησιμοποιηθεί στις υλοποιήσεις του συστήματος για την παρακολούθηση της ποιότητας του αέρα, προσφέροντας τη δυνατότητα ανίχνευσης ενός ευρέος φάσματος επιβλαβών αερίων, συμπεριλαμβανομένου του βενζολίου, της αμμωνίας και του καπνού. Ο αισθητήρας MQ-135 παρέχει αξιόπιστες και ακριβείς μετρήσεις των πτητικών οργανικών ενώσεων στο περιβάλλον. Σκοπός της ενσωμάτωσης στο σχεδιασμό και την υλοποίηση του συστήματος, είναι να χρησιμεύσει ως εργαλείο για τη διασφάλιση της ανθρώπινης υγείας και τον περιορισμό των κινδύνων που σχετίζονται με την έκθεση σε επικίνδυνα αέρια. Στον πίνακα 5-12 παρουσιάζονται ο αισθητήρας και τα χαρακτηριστικά του, στην εικόνα 5-15 η συνδεσμολογία του και στον πίνακα 5-13 ενδεικτικός κώδικας για την επικοινωνία με Arduino.

Αισθητήρας Επικίνδυνων Αερίων (MQ-135)	Χαρακτηριστικά
 A photograph of the MQ-135 gas sensor module. It is a blue printed circuit board (PCB) with a metal mesh cylindrical sensor component attached. The PCB has several pins and labels: 'DOUT', 'GND', 'VCC', 'IN1', 'IN2', and 'PWR'. The text 'Gas Sensor' is printed on the PCB.	<ul style="list-style-type: none">Τάση Λειτουργίας: +5V.Ανίχνευση ύπαρξης: NH₃, alcohol, NO_x, Βενζίνης, CO₂, καπνού etc.Προθέρμανση αισθητήρα: 20 sec.Έξοδοι Αναλογικοί / Ψηφιακοί.Ελεγχόμενη ευαισθησία με ποτενσιόμετρο.

Πίνακας 5-12 Αισθητήρας Επικίνδυνων Αερίων
(*Grobotronics, n.d.*)



Εικόνα 5-16 MQ 135 PinOut
(Microcontrollers lab. n.d.)

```

int sensorValue;
int digitalValue;

void setup()
{
    Serial.begin(9600); // sets the serial port to 9600
    pinMode(13, OUTPUT);
    pinMode(2, INPUT);
}

void loop()
{
    sensorValue = analogRead(0); // read analog input pin 0
    digitalValue = digitalRead(2);
    if (sensorValue > 400)
    {
        digitalWrite(13, HIGH);
    }
    else
        digitalWrite(13, LOW);
    Serial.println(sensorValue, DEC); // prints the value read
    Serial.println(digitalValue, DEC);
    delay(1000); // wait 100ms for next reading
}

```

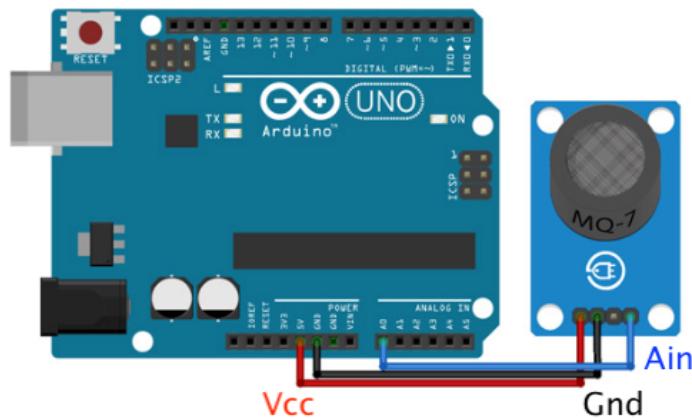
Πίνακας 5-13 Ενδεικτικός Κώδικας MQ 135
(Microcontrollers lab. n.d.)

5.8.7 Αισθητήρας Μονοξειδίου του άνθρακα

Ο αισθητήρας Μονοξειδίου του άνθρακα (MQ-7) λειτουργεί χρησιμοποιώντας έναν θερμαινόμενο ημιαγωγό οξειδίου του μετάλλου. Μετρά τις αλλαγές στην αγωγιμότητα όταν εκτίθεται σε καύσιμα αέρια, όπως το μονοξείδιο του άνθρακα. Ο αισθητήρας MQ7 θα χρησιμοποιηθεί για την παρακολούθηση και ανίχνευση των επιπέδων μονοξειδίου του άνθρακα (CO) στο περιβάλλον. Ο αισθητήρας MQ-7 προσφέρει υψηλή ευαισθησία στο αέριο CO, επιτρέποντας ακριβείς μετρήσεις με ελάχιστη κατανάλωση ενέργειας. Με τη χρήση του αισθητήρα θα δίνεται η δυνατότητα στο σύστημα της έγκαιρης προειδοποίησης για την παρουσία δυνητικά επιβλαβών επιπέδων CO, με σκοπό τη διασφάλιση της ανθρώπινης υγείας και ευημερίας, μειώνοντας τους κινδύνους που σχετίζονται με την έκθεση σε μονοξείδιο του άνθρακα. Στον πίνακα 5-14 παρουσιάζονται ο αισθητήρας και τα χαρακτηριστικά του, στην εικόνα 5-16 η συνδεσμολογία του και στον πίνακα 5-15 ενδεικτικός κώδικας για την επικοινωνία με Arduino.

Αισθητήρας Μονοξειδίου του άνθρακα (MQ-7)	Χαρακτηριστικά
	Τάση Λειτουργίας (v) DC 5 Characteristic gas 100 ppm CO Αναισθησία Αισθητήρα: $\geq 3\%$. Χρόνος Απόκρισης: $\leq 30 \text{ sec}$ Αντίσταση: $\pm 31 \Omega$

Πίνακας 5-14 Αισθητήρας MQ-7
(Grobtronics, n.d.)



Εικόνα 5-17 MQ-7 PinOut

(Damirchi, M., 2021)

```
#include "MQ7.h"
#define A_PIN 2
#define VOLTAGE 5

// init MQ7 device
MQ7 mq7(A_PIN, VOLTAGE);

void setup() {
    Serial.begin(9600);
    while (!Serial) {
        ; // wait for serial connection
    }
    Serial.println(""); // blank new line
    Serial.println("Calibrating MQ7");
    mq7.calibrate(); // calculates R0
    Serial.println("Calibration done!");
}

void loop() {
    Serial.print("PPM = "); Serial.println(mq7.readPpm());

    Serial.println(""); // blank new line
    delay(1000);
}
```

Πίνακας 5-15 Ενδεικτικός Κώδικας MQ-7
(Microcontrollers lab. n.d.)

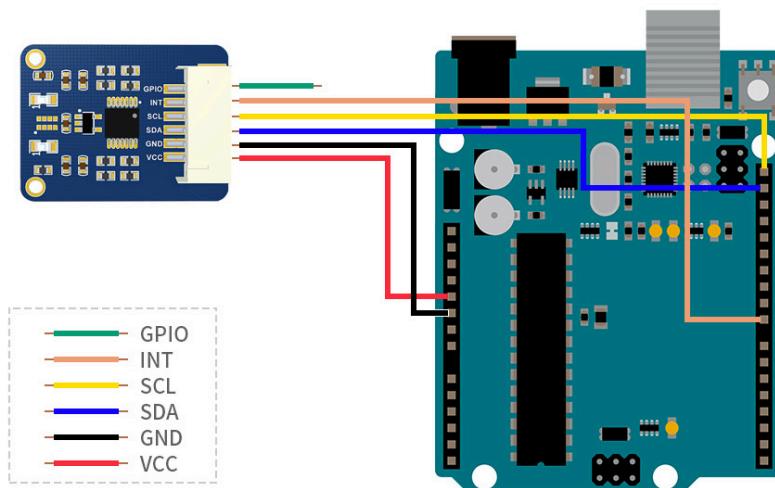
5.8.8 Αισθητήρας μέτρησης υπεριώδους ακτινοβολίας

Ο αισθητήρας υπεριώδους ακτινοβολίας (UV) θα χρησιμοποιηθεί για τη μέτρηση καθώς και την παρακολούθηση των επιπέδων υπεριώδους ακτινοβολίας στο περιβάλλον. Ο

αισθητήρας μετρά με ακρίβεια την ένταση της υπεριώδους ακτινοβολίας, παρέχοντας πολύτιμες πληροφορίες σχετικά με την έκθεση στην υπεριώδη ακτινοβολία για τον περιβάλλοντα χώρο από την έξυπνη κολώνα φωτισμού, παρέχοντας τη δυνατότητα στο σύστημα να συλλέγει δεδομένα με σκοπό τη χρήση τους τόσο σε στατιστικές μετρήσεις όσο και στην έγκαιρη προειδοποίηση σε περίπτωση που οι τιμές ξεπερνούν τα επιτρεπτά όρια. Το σύστημα θα έχει τη δυνατότητα να παρέχει δεδομένα που θα μπορούν να αξιοποιηθούν, ώστε να μπορούν να ληφθούν ενημερωμένες αποφάσεις σχετικά με την ασφάλεια έκθεσης στον ήλιο και να ελαχιστοποιήθουν οι κινδύνοι ηλιακών εγκαυμάτων, δερματικών βλαβών και μακροπρόθεσμων επιπτώσεων στην υγεία που σχετίζονται με την έκθεση στην υπεριώδη ακτινοβολία. Στον πίνακα 5-16 παρουσιάζονται ο αισθητήρας και τα χαρακτηριστικά του, στην εικόνα 5-17 η συνδεσμολογία του και στον πίνακα 5-17 η ενδεικτικός κώδικας για την επικοινωνία με Arduino.

Αισθητήρας Υπεριώδους Ακτινοβολίας (UV-B)	Χαρακτηριστικά												
<p>The image shows a blue printed circuit board (PCB) with various electronic components. A white ribbon cable is attached to the right side, showing six colored wires: green (GPIO), orange (INT), yellow (SCL), blue (SDA), black (GND), and red (VCC). Below the PCB is a legend mapping colors to pin names:</p> <table border="1"> <tr> <td>green</td> <td>GPIO</td> </tr> <tr> <td>orange</td> <td>INT</td> </tr> <tr> <td>yellow</td> <td>SCL</td> </tr> <tr> <td>blue</td> <td>SDA</td> </tr> <tr> <td>black</td> <td>GND</td> </tr> <tr> <td>red</td> <td>VCC</td> </tr> </table>	green	GPIO	orange	INT	yellow	SCL	blue	SDA	black	GND	red	VCC	<p>chip Si1145 με ενσωματωμένο ADC, ανιχνεύει την υπεριώδη ακτινοβολία, αλλά και την ένταση του φωτός του περιβάλλοντος. Διεπαφή I2C Συμβατός με τάσεις λειτουργίας 3,3V/5V</p>
green	GPIO												
orange	INT												
yellow	SCL												
blue	SDA												
black	GND												
red	VCC												

Πίνακας 5-16 Αισθητήρας Υπεριώδους Ακτινοβολίας
(UV Sensor (B) - Waveshare Wiki, n.d.)



Εικόνα 5-18 UV Sensor PinOut
(UV Sensor (B) - Waveshare Wiki, n.d.)

```
/*
This is a library for the Si1145 UV/IR/Visible Light Sensor

Designed specifically to work with the Si1145 sensor in the
adafruit shop
----> https://www.adafruit.com/products/1777

These sensors use I2C to communicate, 2 pins are required to
interface
Adafruit invests time and resources providing this open source code,
please support Adafruit and open-source hardware by purchasing
products from Adafruit!
Written by Limor Fried/Ladyada for Adafruit Industries.
BSD license, all text above must be included in any redistribution
*****
```

```
#include <Wire.h>
#include "Adafruit_SI1145.h"

Adafruit_SI1145 uv = Adafruit_SI1145();
void setup() {
  Serial.begin(9600);
  Serial.println("Adafruit SI1145 test");
  if (! uv.begin()) {
    Serial.println("Didn't find Si1145");
    while (1);
  }
  Serial.println("OK!");
```

```

}

void loop() {
    Serial.println("=====");
    Serial.print("Vis: "); Serial.println(uv.readVisible());
    Serial.print("IR: "); Serial.println(uv.readIR());
    float UVindex = uv.readUV();
    // the index is multiplied by 100 so to get the
    // integer index, divide by 100!
    UVindex /= 100.0;
    Serial.print("UV: "); Serial.println(UVindex);
    delay(1000);
}

```

Πίνακας 5-17 Ενδεικτικός Κώδικας αισθητήρα UV
(*Adafruit, n.d.*)

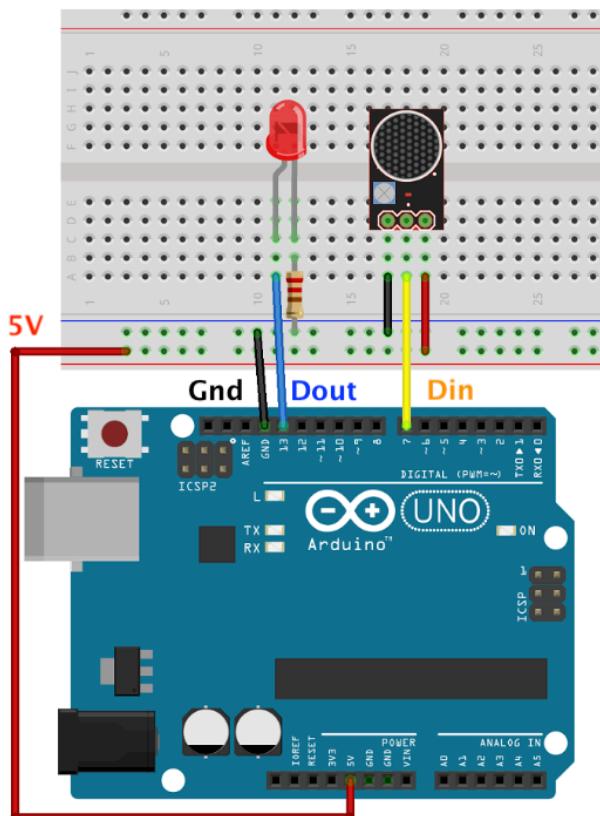
5.8.9 Αισθητήρας μέτρησης θορύβου

Ο αισθητήρας μέτρησης θορύβου, θα χρησιμοποιηθεί για την ανίχνευση και ανάλυση της έντασης του ήχου στον περιβάλλοντα χώρο από τις κολώνες έξυπνου φωτισμού. Ο συγκεκριμένος αισθητήρας διαθέτει ένα μικροφόνου και ένα ολοκληρωμένο κύκλωμα ενισχυτή. Ο αισθητήρας μπορεί να ανιχνεύσει ένα ευρύ φάσμα ηχητικών συχνοτήτων και πλατών, επιτρέποντας την παρακολούθηση των επιπέδων ήχου σε πραγματικό χρόνο, δίνοντας τη δυνατότητα ανάπτυξης εφαρμογών όπως συστήματα παρακολούθησης θορύβου. Με τη χρήση του συγκεκριμένου αισθητήρα, στο σύστημα που θα υλοποιηθεί θα υπάρχει η δυνατότητα καταγραφής των επιπέδων θορύβου και η αποθήκευση των δεδομένων αυτών με σκοπό τόσο την ενημέρωση όσο και την εξαγωγή στατιστικών αποτελεσμάτων. Στον πίνακα 5-18 παρουσιάζονται ο αισθητήρας και τα χαρακτηριστικά του, στην εικόνα 5-18 η συνδεσμολογία του και στον πίνακα 5-19 ενδεικτικός κώδικας για την επικοινωνία με Arduino.

Aισθητήρας Μέτρησης θορύβου	Χαρακτηριστικά
-----------------------------	----------------

	<p>Τάση λειτουργίας 3.3V-5V Ρεύμα λειτουργίας ($V_{cc}=5V$): 4-8mA Ευαισθησία μικροφώνου (1Khz): 52-48dB Αντίσταση μικροφώνου: 2,2KΩ Συχνότητα μικροφώνου: 16-20Khz Αναλογία S/N μικροφώνου: 54dB</p>
--	--

Πίνακας 5-18 Αισθητήρας μέτρησης Θορύβου
(Grobotronics, n.d.)



Εικόνα 5-19 Αισθητήρας Θορύβου PinOut
(Santos & Santos, 2019)

```
/*
 * Sound Sensor Example
 */

int ledPin=13;
```

```

int sensorPin=7;
boolean val =0;

void setup () {
    pinMode(ledPin, OUTPUT);
    pinMode(sensorPin, INPUT);
    Serial.begin (9600);
}

void loop () {
    val =digitalRead(sensorPin);
    Serial.println (val);
    // when the sensor detects a signal above the threshold value, LED
flashes
    if (val==HIGH) {
        digitalWrite(ledPin, HIGH);
    }
    else {
        digitalWrite(ledPin, LOW);
    }
}

```

Πίνακας 5-19 Ενδεικτικός Κώδικας Αισθητήρα Θορύβου
(Santos & Santos, 2019)

5.8.10 Αισθητήρας ύπαρξης πυρκαγιάς

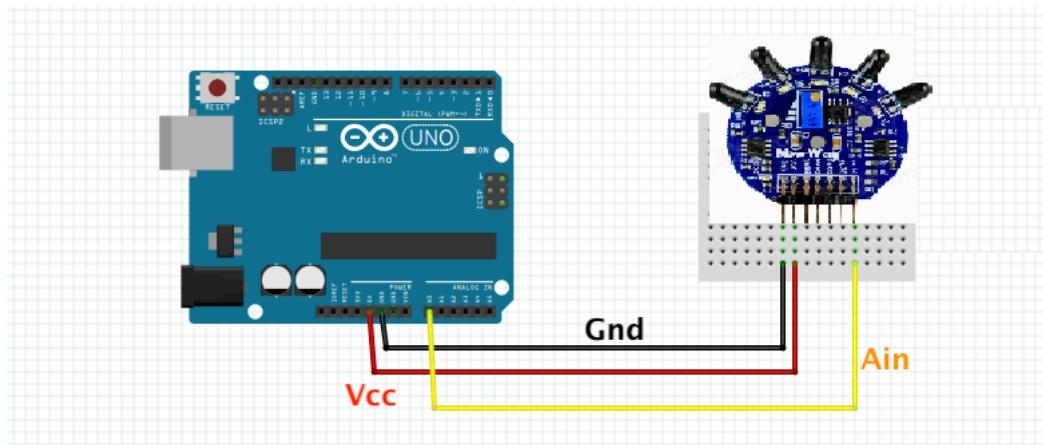
Ο αισθητήρας ύπαρξης πυρκαγιάς είναι ένας αισθητήρας που μπορεί να ανιχνεύσει την ύπαρξη φλόγας. Διαθέτει 5 κανάλια υπερύθρων, με αποτέλεσμα να μπορεί να ανιχνεύσει φλόγες σε ένα ευρύ φάσμα (> 120 μοίρες), από μικρές φλόγες έως έντονες πυρκαγιές. Κάθε κανάλι του αισθητήρα παρέχει ανεξάρτητη ικανότητα ανίχνευσης, επιτρέποντας τον ακριβή εντοπισμό και χαρακτηρισμό των φλογών στο περιβάλλον. Με την ενσωμάτωση του συγκεκριμένου αισθητήρα στο σύστημα των έξυπνων κολώνων φωτισμού θα παρέχεται η δυνατότητα ανίχνευσης πυρκαγιάς για σπίτια, βιομηχανικές εγκαταστάσεις και δημόσιους χώρους και θα δημιουργηθεί ένα συστήμα συναγερμού πυρκαγιάς ώστε να μπορεί έγκαιρα να ενεργοποιηθεί ο μηχανισμός καταστολής πυρκαγιάς στο περιβάλλον της έξυπνης πόλης. Στον πίνακα 5-20 παρουσιάζονται ο αισθητήρας και τα χαρακτηριστικά του, στην εικόνα 5-19 η συνδεσμολογία του και στον πίνακα 5-21 ενδεικτικός κώδικας για την επικοινωνία με Arduino.

Αισθητήρας ύπαρξης πυρκαγιάς	Χαρακτηριστικά
------------------------------	----------------



	<p>Λειτουργεί με τάση: 3.3 - 9V Εξόδους: Αναλογικές & Ψηφιακές Ενσωματωμένο ποτενσιόμετρο και ενδεικτικές λυχνίες LED Αντιστάσεις για να καταστεί η ενότητα αυτή πιο αξιόπιστη και ακριβής Εύρος ανίχνευσης:> 120 μοίρες</p>
---	---

Πίνακας 5-20 Αισθητήρας ύπαρξης Πυρκαγιάς
(Grobotronics, n.d.)



Εικόνα 5-20 Αισθητήρας ύπαρξης Πυρκαγιάς PinOut
(How to Use the Flame Sensor With Arduino, n.d.)

```
/*
 * Flame Sensor Example
 */

const int sensorMin = 0;      // sensor minimum
const int sensorMax = 1024;    // sensor maximum

void setup() {
  Serial.begin(9600);
}

void loop() {
  // read the sensor on analog A0:
  int sensorReading = analogRead(A0);
  int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

  // range value:
  switch (range) {
```

```

case 0:      //No fire detected
    Serial.println("** No Fire **");
    break;
case 1:      // A fire between 1-3 feet away.
    Serial.println("** Fire **");

}
delay(1000);
}

```

Πίνακας 5-21 Ενδεικτικός Κώδικας Αισθητήρα ύπαρξης Πυρκαγιάς
(*How to Use the Flame Sensor With Arduino, n.d.*)

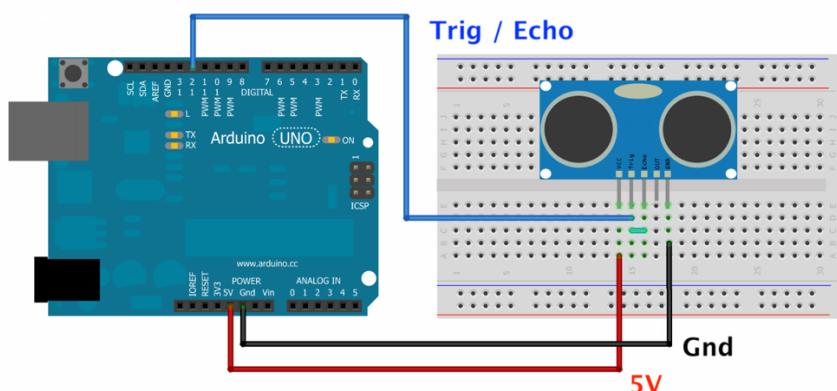
5.8.11 Αισθητήρας Απόστασης (Υπερήχων)

Ο Αισθητήρας Απόστασης (Υπερήχων) HY-SRF05 θα χρησιμοποιηθεί για τον υπολογισμό απόστασης. Ο αισθητήρας λειτουργεί με χρήση υπερήχων, τους οποίους αναπαράγει μέσω του pin trig και τους παραλαμβάνει μέσω του pin echo, μετρώντας το χρόνο που κάνει ο υπέρηχος να επιστρέψει, υπολογίζει την απόσταση. Η ενσωμάτωση του αισθητήρα από το σύστημα θα γίνει με σκοπό τη μέτρηση της απόστασης των διερχόμενων αυτοκινήτων από την έξυπνη κολώνα φωτισμού. Ο συνδυασμός των μετρήσεων από τα δεδομένα που θα συλλέγονται θα καθορίζουν την ένταση του φωτισμού στις κολώνες φωτισμού του οδών ταχείας κυκλοφορίας. Όταν δεν διέρχονται αυτοκίνητα σε συγκεκριμένη απόσταση από τις κολώνες φωτισμού, θα μειώνεται η ένταση φωτισμού, ενώ όταν το σύστημα αντιλαμβάνεται ότι κάποιο όχημα πλησιάζει θα αναπροσαρμόζει το φωτισμό του στα κανονικά επίπεδα. Στον πίνακα 5-22 παρουσιάζονται ο αισθητήρας και τα χαρακτηριστικά του, στην εικόνα 5-20 η συνδεσμολογία του και στον πίνακα 5-23 ενδεικτικός κώδικας για την επικοινωνία με Arduino.

Αισθητήρας Απόστασης	Χαρακτηριστικά
----------------------	----------------



Πίνακας 5-22 Αισθητήρας Απόστασης
(Grobtronics, n.d.)



Εικόνα 5-21 Αισθητήρας Απόστασης PinOut
(Grobtronics, n.d.)

```
/*
 * Distance Sensor Example
 */
const unsigned int TRIG_PIN=13;
const unsigned int ECHO_PIN=12;
const unsigned int BAUD_RATE=9600;

void setup() {
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    Serial.begin(BAUD_RATE);
}

void loop() {
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
```

```

delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);
const unsigned long duration= pulseIn(ECHO_PIN, HIGH);
int distance= duration/29/2;
if(duration==0) {
    Serial.println("Warning: no pulse from sensor");
}
else{
    Serial.print("distance to nearest object:");
    Serial.println(distance);
    Serial.println(" cm");
}
delay(100);
}

```

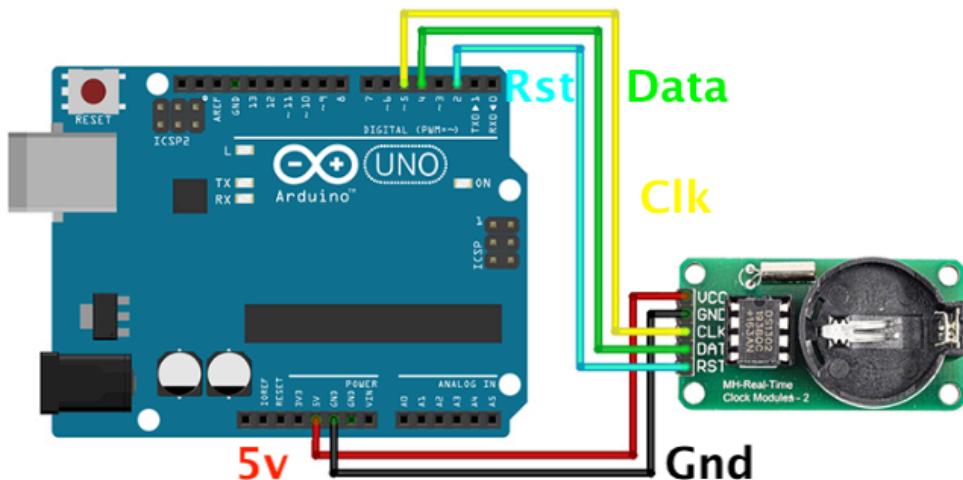
Πίνακας 5-23 Ενδεικτικός Κώδικας Αισθητήρα Απόστασης
(*Arduino Project Hub, n.d.*)

5.8.12 Ρολόι πραγματικού χρόνου I2C

Το ρολόι πραγματικού χρόνου θα χρησιμοποιηθεί για την αποτύπωση της χρονοσφραγίδας στις μετρήσεις που θα λαμβάνονται από τους αισθητήρες που θα συνδεθούν στις υλοποιήσεις των έξυπνων κολώνων με μικροελεγκτή Arduino Uno. Στις υλοποιήσεις με Raspberry δεν θα χρειαστεί διότι έχει ενσωματωμένο ρολόι. Στον πίνακα 5-24 παρουσιάζονται ο αισθητήρας και τα χαρακτηριστικά του, στην εικόνα 5-21 η συνδεσμολογία του και στον πίνακα 5-25 ενδεικτικός κώδικας για την επικοινωνία με Arduino.

Αισθητήρας Απόστασης	Χαρακτηριστικά
	Απαιτήσεις τάσης: 3,3-5 VDC. Απαιτήσεις ρεύματος: 5 VDC. Επικοινωνία: SPI

Πίνακας 5-24 Ρολόι πραγματικού χρόνου
(*Maleki, 2023*)



Εικόνα 5-22 Ρολό πραγματικού χρόνου PinOut
(Maleki, 2023)

```
/*
 * Real clock Sensor Example
 */

#include <ThreeWire.h>
#include <RtcDS1302.h>
ThreeWire myWire(4,5,2); // IO, SCLK, CE
RtcDS1302<ThreeWire> Rtc(myWire);
void setup () {
    Serial.begin(9600);
    Serial.print("compiled: ");
    Serial.print(__DATE__);
    Serial.println(__TIME__);
    Rtc.Begin();
    RtcDateTime compiled = RtcDateTime(__DATE__, __TIME__);
    printDateTime(compiled);
    Serial.println();
    if (!Rtc.IsDateTimeValid()) {
        // Common Causes:
        // 1) first time you ran and the device wasn't running yet
        // 2) the battery on the device is low or even missing
        Serial.println("RTC lost confidence in the DateTime!");
        Rtc.SetDateTime(compiled);
    }
    if (Rtc.GetIsWriteProtected()) {
        Serial.println("RTC was write protected, enabling writing now");
        Rtc.SetIsWriteProtected(false);
    }
}
```

```

if (!Rtc.GetIsRunning()) {
    Serial.println("RTC was not actively running, starting now");
    Rtc.SetIsRunning(true);
}

RtcDateTime now = Rtc.GetDateTime();
if (now < compiled) {
    Serial.println("RTC is older than compile time! (Updating
DateTime)");
    Rtc.SetDateTime(compiled);
} else if (now > compiled) {
    Serial.println("RTC is newer than compile time. (this is
expected)");
} else if (now == compiled) {
    Serial.println("RTC is the same as compile time! (not expected
but all is fine)");
}
}

void loop () {
    RtcDateTime now = Rtc.GetDateTime();
    printDateTime(now);
    Serial.println();
    if (!now.IsValid()) {
        Serial.println("RTC lost confidence in the DateTime!");
    }
    delay(5000); // five seconds
}

#define countof(a) (sizeof(a) / sizeof(a[0]))
void printDateTime(const RtcDateTime& dt) {
    char datestring[20];
    snprintf_P(datestring,
               countof(datestring),
               PSTR("%02u/%02u/%04u %02u:%02u:%02u"),
               dt.Month(),
               dt.Day(),
               dt.Year(),
               dt.Hour(),
               dt.Minute(),
               dt.Second());
    Serial.print(datestring);
}

```

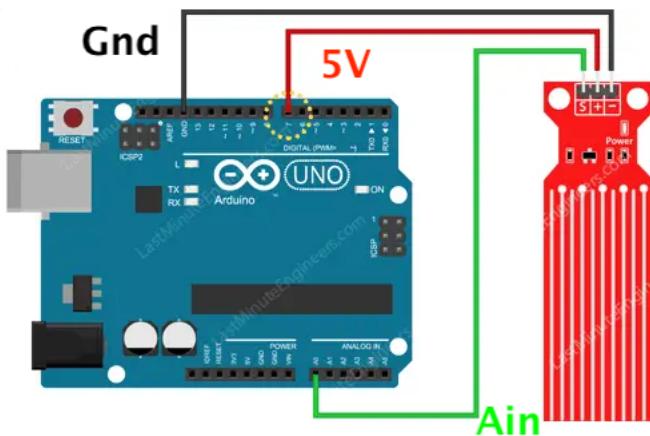
Πίνακας 5-25 Ενδεικτικός Κώδικας Ρολογιού πραγματικού χρόνου
(Maleki, 2023)

5.8.13 Αισθητήρας μέτρησης στάθμης υγρού

Ο αισθητήρας μέτρησης στάθμης υγρού βασίζει τη λειτουργία του στη αγωγιμότητα του νερού. Συγκρίνει σήματα τάσης εισόδου και εξάγει ένα σήμα με βάση τα σχετικά μεγέθη τους, το οποίο παράγεται από την έξοδο του (Ain). Ο αισθητήρας μέτρησης στάθμης υγρού θα χρησιμοποιηθεί για την ανίχνευση υπέρβασης συγκεκριμένου ορίου στις αποχετεύσεις όμβριων υδάτων, με σκοπό την πρόληψη πλημυρών στους δημόσιους δρόμους. Οι έξυπνες κολώνες θα διαθέτουν τους συγκεκριμένους αισθητήρες και θα παρέχουν τη δυνατότητα σε περίπτωση υπέρβασης του ορίου να ενημερώνεται ο κεντρικός διακομιστής. Στον πίνακα 5-26 παρουσιάζονται ο αισθητήρας και τα χαρακτηριστικά του, στην εικόνα 5-22 η συνδεσμολογία του και στον πίνακα 5-27 ενδεικτικός κώδικας για την επικοινωνία με Arduino.

Αισθητήρας μέτρησης στάθμης υγρού	Χαρακτηριστικά
	<p>Τάση λειτουργίας: 3-5V Ρεύμα εργασίας: <20mA Περιοχή ανίχνευσης: 40 mm x 16 mm Θερμοκρασία εργασίας: 16,5 mm: °C έως 30 °C Υγρασία εργασίας: 10% έως 90% χωρίς συμπύκνωση Μέγεθος: 65 mm x 20 mm x 8 mm</p>

Πίνακας 5-26 Αισθητήρας μέτρησης στάθμης υγρού
(Water Level Sensor for Arduino - Devobox, n.d.)



Εικόνα 5-23 Αισθητήρας μέτρησης στάθμης υγρού PinOut
(LME Editorial Staff, 2022)

```
/*
 * liquid level Sensor Example
 */

// Sensor pins
#define sensorPower 7
#define sensorPin A0

// Value for storing water level
int val = 0;

void setup() {
    // Set D7 as an OUTPUT
    pinMode(sensorPower, OUTPUT);

    // Set to LOW so no power flows through the sensor
    digitalWrite(sensorPower, LOW);

    Serial.begin(9600);
}

void loop() {
    //get the reading from the function below and print it
    int level = readSensor();
    Serial.print("Water level: ");
    Serial.println(level);
    delay(1000);
}
```

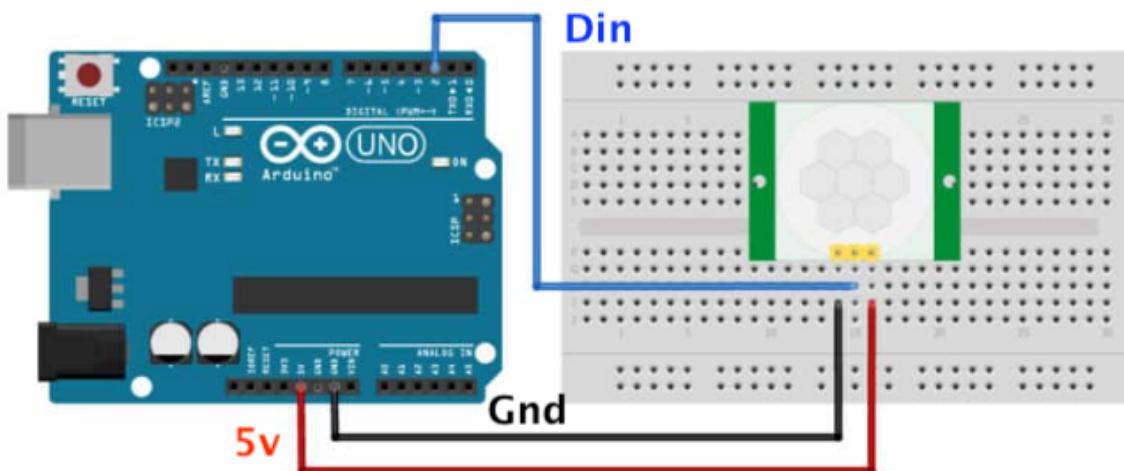
Πίνακας 5-27 Ενδεικτικός Κώδικας Αισθητήρα μέτρησης στάθμης υγρού
(LME Editorial Staff, 2022)

5.8.14 Αισθητήρας Ανίχνευσης κίνησης

Ο Αισθητήρας Ανίχνευσης Κίνησης (PIR - HC-SR501) ανιχνεύει την κίνηση μέσω των αλλαγών στην υπέρυθρη ακτινοβολία που εκπέμπεται από τα αντικείμενα στην περιοχή ανίχνευσής του. Αποτελείται από έναν αισθητήρα που παράγει ηλεκτρικό σήμα όταν εκτίθεται σε υπέρυθρη ακτινοβολία, ενεργοποιώντας την έξοδο του αισθητήρα. Ο PIR - HC-SR501, θα χρησιμοποιηθεί στην ενίσχυση της λειτουργικότητας των έξυπνων κολώνων φωτισμού, ανιχνεύοντας την κίνηση και ρυθμίζοντας ανάλογα το επίπεδο φωτισμού. Με τη δυνατότητα ανίχνευσης κινούμενων αντικείμενων εντός του οπτικού του πεδίου, η έξυπνη κολώνα φωτισμού θα μπορεί να ρυθμίσει δυναμικά το επίπεδο φωτισμού με βάση την ανιχνευόμενη κίνηση, παρέχοντας επαρκή φωτισμό, ενώ παράλληλα θα εξοικονομεί ενέργεια σε περιόδους αδράνειας. Αυτή η έξυπνη λειτουργία θα προωθήσει την ενεργειακή απόδοση και τη βιωσιμότητα, ελαχιστοποιώντας τον περιττό φωτισμό στο περιβάλλον της έξυπνης πόλης. Στον πίνακα 5-28 παρουσιάζονται ο αισθητήρας και τα χαρακτηριστικά του, στην εικόνα 5-23 η συνδεσμολογία του και στον πίνακα 5-29 ενδεικτικός κώδικας για την επικοινωνία με Arduino.

Αισθητήρας Ανίχνευσης κίνησης	Χαρακτηριστικά
	<p>Εύρος τάσης λειτουργίας: 20V Τάση εξόδου: Τάση εξόδου: 3.3V TTL εξόδου Απόσταση ανίχνευσης: (μπορεί να ρυθμιστεί) Εύρος ανίχνευσης: <140° Χρόνος καθυστέρησης: 5-200S (μπορεί να ρυθμιστεί, προεπιλογή 5s +3%)</p>

Πίνακας 5-28 Αισθητήρας Ανίχνευσης κίνησης
(Grobtronics, n.d.)



Εικόνα 5-24 Αισθητήρας Ανίχνευσης κίνησης PinOut
(De Bakker, 2024)

```
// Example code for HC-SR501 PIR motion sensor with Arduino.

// Define connection pins:
#define pirPin 2
#define ledPin 13

// Create variables:
int val = 0;
bool motionState = false; // We start with no motion detected.

void setup() {
    // Configure the pins as input or output:
    pinMode(ledPin, OUTPUT);
    pinMode(pirPin, INPUT);

    // Begin serial communication at a baud rate of 9600:
    Serial.begin(9600);
}

void loop() {
    // Read out the pirPin and store as val:
    val = digitalRead(pirPin);

    // If motion is detected (pirPin = HIGH), do the following:
    if (val == HIGH) {
        digitalWrite(ledPin, HIGH); // Turn on the on-board LED.

        // Change the motion state to true (motion detected):
        if (motionState == false) {
            Serial.println("Motion detected!");
            motionState = true;
        }
    }
}
```

```
}
```

```
// If no motion is detected (pirPin = LOW), do the following:  
else {  
    digitalWrite(ledPin, LOW); // Turn off the on-board LED.  
  
    // Change the motion state to false (no motion):  
    if (motionState == true) {  
        Serial.println("Motion ended!");  
        motionState = false;  
    }  
}  
}
```

Πίνακας 5-29 Ενδεικτικός Κώδικας Αισθητήρας Ανίχνευσης κίνησης
(De Bakker, 2024)

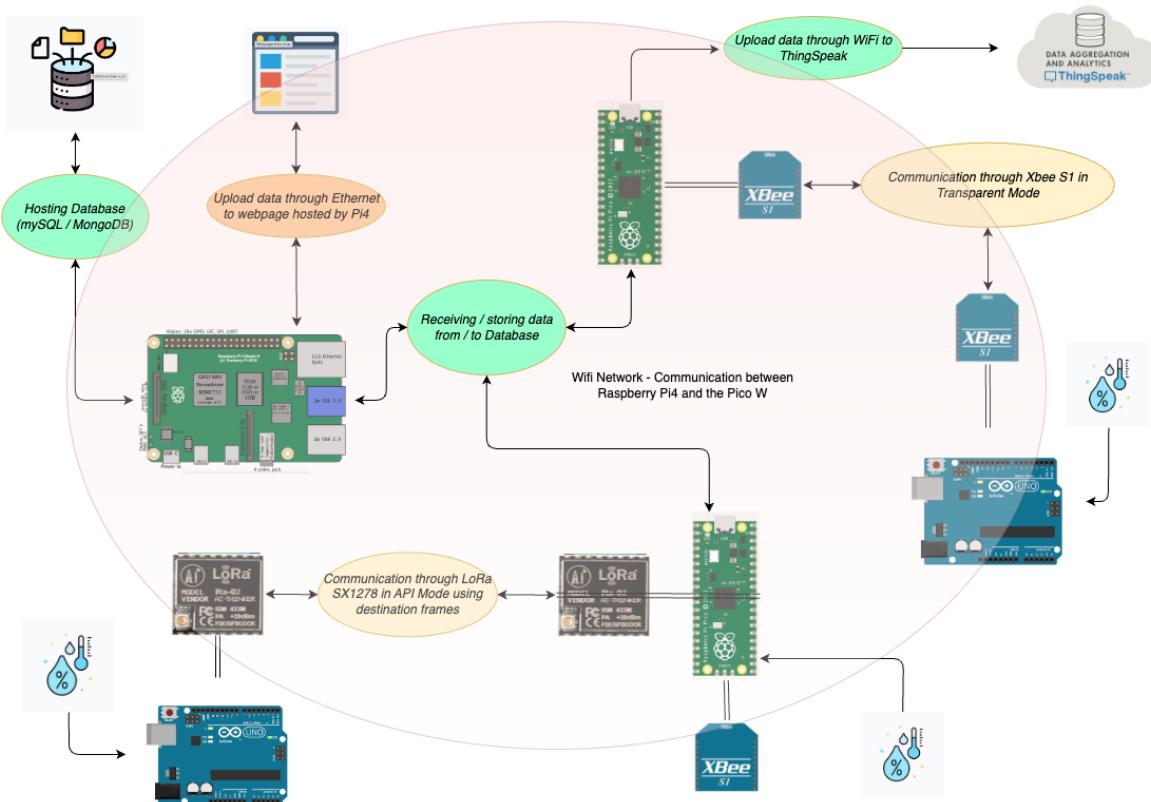
6. Υλοποίηση Συστήματος

6.1 Εισαγωγή

Στο παρόν κεφάλαιο θα γίνει αναλυτική παρουσίαση της υλοποίησης του συστήματος των «έξυπνων κολώνων φωτισμού» σε περιβάλλον έξυπνης πόλης. Συγκεκριμένα, θα γίνει αναλυτική περιγραφή των σεναρίων αυτοματοποίησης του συστήματος, καθώς επίσης και της συνδεσμολογίας των πλακετών ανάπτυξής και των αισθητήρων. Θα γίνει αναφορά στους τρόπους επικοινωνίας των κολώνων στο δίκτυο της έξυπνης πόλης, αξιοποιώντας τις διάφορες τεχνολογίες που παρουσιάστηκαν στα προηγούμενα κεφάλαια. Στο τέλος του κεφαλαίου, θα παρουσιαστεί ο προγραμματισμός των Arduino UNO, των Raspberry Pi Pico W, ESP32 Ai Thinker Cam, XBEE S1 και SX1278 (LoRa).

6.2 Δίκτυο επικοινωνίας Έξυπνων κολώνων Φωτισμού

Στο συγκεκριμένο σενάριο θα γίνει υλοποίηση ενός δικτύου από τέσσερις «έξυπνες κολώνες» οι οποίες για την μεταξύ τους επικοινωνίας χρησιμοποιούν τις τεχνολογίες Zigbee, LoRa και WiFi. Συγκεκριμένα θα γίνει υλοποίηση ενός δικτύου το οποίο θα περιλαμβάνει, ένα υποδίκτυο από δύο «έξυπνες κολώνες» οι οποίες για τη μεταξύ τους επικοινωνία θα χρησιμοποιούν τα Xbee modules (Zigbee), ένα υποδίκτυο από δύο «έξυπνες κολώνες» οι οποίες για τη μεταξύ τους επικοινωνία θα χρησιμοποιούν τα SX1278 (LoRa) modules και τέλος ένα ασύρματο υποδίκτυο με σύνδεση στο internet το οποίο οι κεντρικοί κόμβοι των ανωτέρω υποδικτύων θα το χρησιμοποιούν για την επικοινωνία με τον REST Server και την πλατφόρμα ThingSpeak. Στην εικόνα 6-1 παρουσιάζεται το η απεικόνιση του συνολικού δικτύου που θα υλοποιηθεί στο οποίο περιλαμβάνονται οι τρόποι επικοινωνίας, οι controllers και οι αισθητήρες κάθε «έξυπνης κολώνας». Στον πίνακα 6-1 παρουσιάζεται η επεξήγηση των εικονιδίων της εικόνας 6-1.



Εικόνα 6-1 Σχεδιάγραμμα δικτύου συστήματος υλοποίησης



Πίνακας 6-1 Επεξηγήσεις σχημάτων εικόνας 6-1

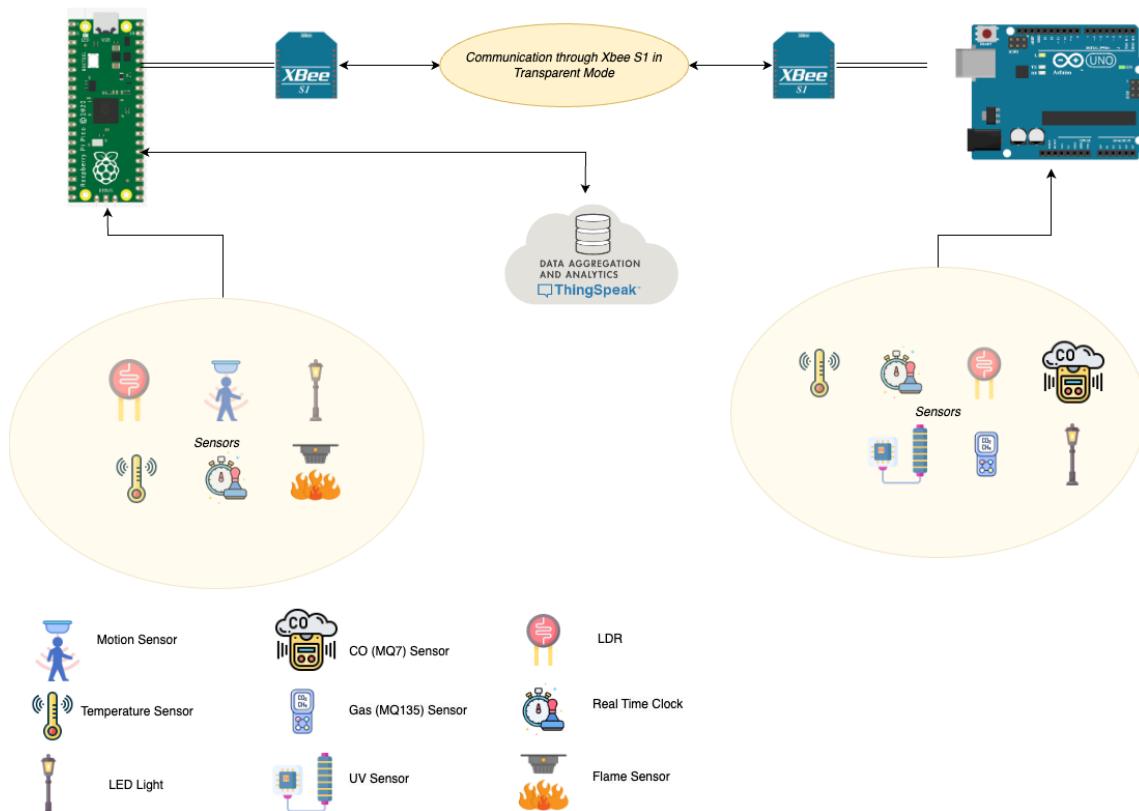
6.2.1 Δίκτυο από «Έξυπνες κολώνες» με επικοινωνία Xbee

Στο συγκεκριμένο σενάριο θα γίνει υλοποίηση ενός υποδικτύου του συνολικού δικτύου που αναφέρθηκε στην προηγούμενη ενότητα, το οποίο θα αποτελείται από δύο «έξυπνες

κολώνες», οι οποίες για την μεταξύ τους επικοινωνίας χρησιμοποιούν τις τεχνολογίες Zigbee και WiFi. Συγκεκριμένα, η μία από τις δύο «έξυπνες κολώνες» θα έχει το ρόλο του κεντρικού κόμβου του υποδικτύου και η άλλη το ρόλο του πελάτη (client). Ο client θα λειτουργεί αυτόνομα και θα συλλέγει τις μετρήσεις από τους συνδεδεμένους σε αυτόν αισθητήρες, τις οποίες θα αποστέλλει στον κεντρικό κόμβο χρησιμοποιώντας το δίκτυο Zigbee. Ο κεντρικός κόμβος θα λειτουργεί αυτόνομα και θα συλλέγει τόσο τις μετρήσεις από τους συνδεδεμένους σε αυτόν αισθητήρες όσο και τις μετρήσεις που λαμβάνει από τους συνδεδεμένους clients. Η επικοινωνία με το κεντρικό σύστημα υλοποιείται χρησιμοποιώντας δύο XBee S1, το πρώτο είναι συνδεμένο στο Arduino UNO, το οποίο είναι ο client του υποδικτύου και το δεύτερο στο Raspberry Pi W το οποίο είναι ο κεντρικός κόμβος του υποδικτύου. Το Raspberry Pi W με την σειρά του λαμβάνει τα δεδομένα σε μορφή JSON από το XBee S1 του client και τα διαβιβάζει στον REST Server μέσω WiFi. Ο τρόπος λειτουργίας του REST Server θα παρουσιαστεί στο επόμενο κεφάλαιο. Η υλοποίηση του παραπάνω σεναρίου απαιτεί:

- Arduino Uno
- Raspberry Pi W
- 2x XBee S1
- Raspberry Pi 4
- Internet

Στην εικόνα 6-2 παρουσιάζεται το υποδίκτυο που θα υλοποιηθεί στο οποίο περιλαμβάνονται οι τρόποι επικοινωνίας, οι μικροελεγκτές και οι αισθητήρες κάθε «έξυπνης κολώνας».



Εικόνα 6-2 Υποδίκτυο βασισμένο στην επικοινωνία με χρήση XBee

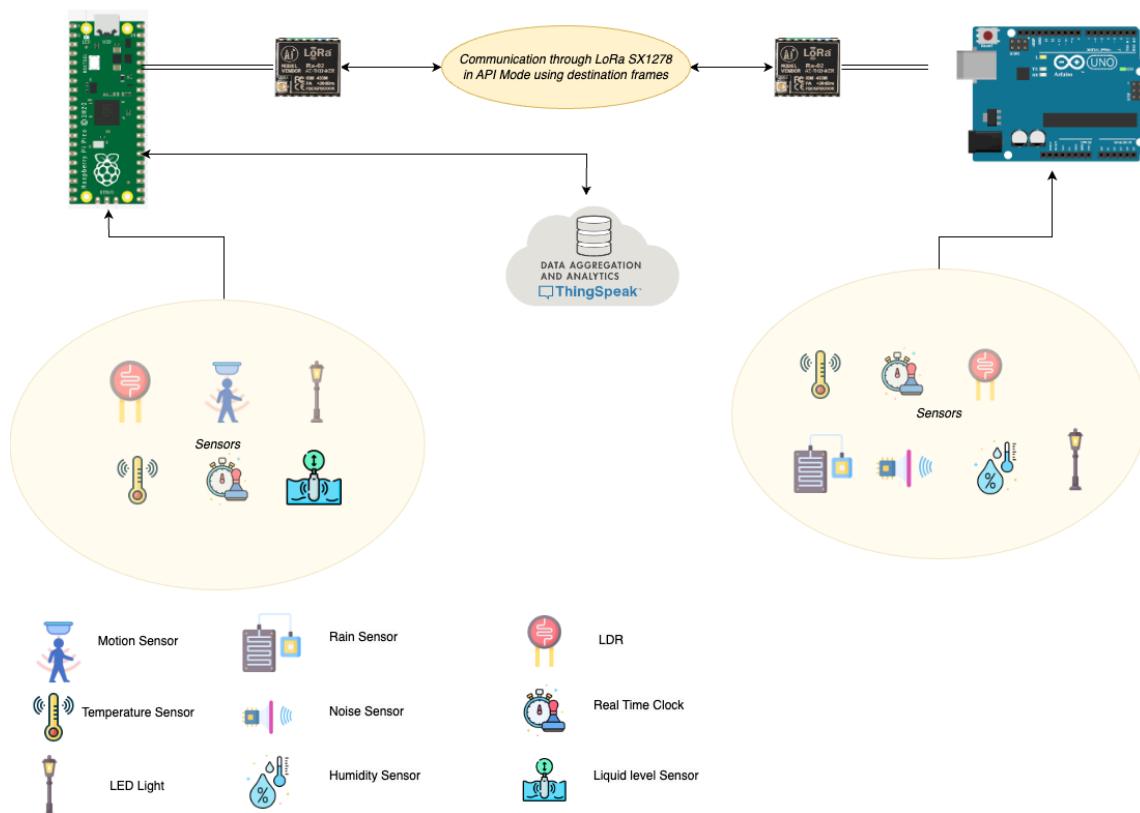
6.2.2 Δίκτυο από «Έξυπνες κολώνες» με επικοινωνία LoRa

Στο συγκεκριμένο σενάριο θα γίνει υλοποίηση ενός υποδικτύου του συνολικού δικτύου που αναφέρθηκε στην προηγούμενη ενότητα, το οποίο θα αποτελείται από δύο «έξυπνες κολώνες», οι οποίες για την μεταξύ τους επικοινωνίας χρησιμοποιούν τις τεχνολογίες LoRa και WiFi. Συγκεκριμένα, η μία από τις δύο «έξυπνες κολώνες» θα έχει το ρόλο του κεντρικού κόμβου του υποδικτύου και η άλλη το ρόλο του πελάτη (client). Ο client θα λειτουργεί αυτόνομα και θα συλλέγει τις μετρήσεις από τους συνδεδεμένους σε αυτόν αισθητήρες, τις οποίες θα αποστέλλει στον κεντρικό κόμβο χρησιμοποιώντας το δίκτυο LoRa στα 868 Mhz. Ο κεντρικός κόμβος θα λειτουργεί αυτόνομα και θα συλλέγει τόσο τις μετρήσεις από τους συνδεδεμένους σε αυτόν αισθητήρες όσο και τις μετρήσεις που λαμβάνει από τους συνδεδεμένους clients. Η επικοινωνία με το κεντρικό σύστημα υλοποιείται χρησιμοποιώντας δύο SX1278, το πρώτο είναι συνδεμένο στο Arduino UNO, το οποίο είναι ο client του υποδικτύου και το δεύτερο στο Raspberry Pico W το οποίο είναι ο κεντρικός κόμβος του υποδικτύου. Το Raspberry Pico W με την σειρά του λαμβάνει τα

δεδομένα σε μορφή JSON από το SX1278 του client και τα διαβιβάζει στον REST Server μέσω Wi-Fi. Η υλοποίηση του παραπάνω σεναρίου απαιτεί:

- Arduino Uno
- Raspberry Pico W
- 2x SX1278
- Raspberry Pi 4
- Internet

Στην εικόνα 6-3 παρουσιάζεται το υποδίκτυο που θα υλοποιηθεί στο οποίο περιλαμβάνονται οι τρόποι επικοινωνίας, οι controllers και οι αισθητήρες κάθε «έξυπνης κολώνας».



Εικόνα 6-3 Υποδίκτυο βασισμένο στην επικοινωνία με χρήση LoRa

6.3 Σενάρια Αυτοματοποίησης Έξυπνων κολώνων Φωτισμού

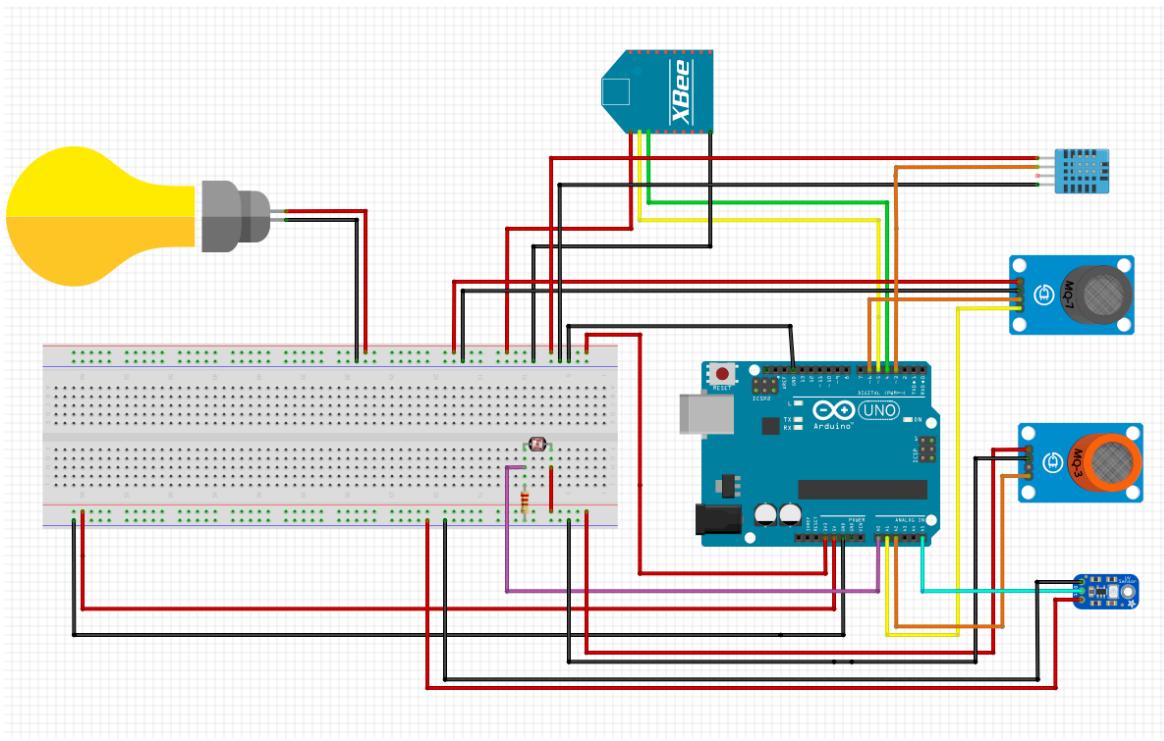
6.3.1 «Έξυπνη κολώνα» Arduino Uno - Xbee S1

Στο συγκεκριμένο σενάριο η «έξυπνη κολώνα» λειτουργεί αυτόνομα ως προς τις λειτουργίες του φωτισμού με βάση τα δεδομένα εξωτερικού περιβάλλοντος. Συλλέγει συνεχώς πληροφορίες για το επίπεδο έντασης της ηλιακής ακτινοβολίας με χρήση φωτοαντίστασης (LDR) και ανάλογα με τις συγκεκριμένες τιμές το Arduino επιλέγει αν θα ανάψει το φωτισμό LED και προσαρμόζει την ένταση λειτουργίας του, ώστε να πετύχει συνδυασμό ποιοτικού φωτισμού, αλλά και εξοικονόμηση ενέργειας. Η συγκεκριμένη «έξυπνη κολώνα» διαθέτει αισθητήρα μέτρησης θερμοκρασίας, μέτρησης υγρασίας, μέτρησης των επιπέδων μονοξειδίου του άνθρακα, αισθητήρα μέτρησης επικίνδυνων αερίων και αισθητήρα μέτρησης υπεριώδους ακτινοβολίας. Τα δεδομένα από τις μετρήσεις που λαμβάνουν οι αισθητήρες μετασχηματίζονται σε κατάλληλη μορφή JSON και αποστέλλονται στον κεντρικό κόμβο κάθε δέκα λεπτά για τα δεδομένα των αισθητήρων και συνεχώς όταν ενεργοποιούνται τα alarm. Στη δομή του JSON περιέχονται τα δεδομένα σε μορφή ζευγαριών κλειδιού, τιμής. Κατά τη δημιουργία του μηνύματος στα δεδομένα περιέχονται το όνομα της κολώνας και τα ονόματα των αισθητήρων με τις αντίστοιχες τιμές τους.

Η υλοποίηση του παραπάνω σεναρίου απαιτεί:

- Arduino Uno
- Xbee S1
- LDR
- LED
- MQ7 για τη μέτρηση CO και ενεργοποίηση alarm
- MQ135 για τη μέτρηση ποιότητας του αέρα
- DTH11 για τη μέτρηση θερμοκρασίας, υγρασίας
- Real Clock module για την καταγραφή της ώρας μέτρησης

Παρακάτω στην εικόνα 6-4 παρουσιάζεται η συνδεσμολογία του σεναρίου:



Εικόνα 6-4 Συνδεσμολογία σεναρίου Arduino

Σχετικά με τον κώδικα του Arduino UNO (παράρτημα A.1.1) χρειάζεται να εισαχθούν οι αντίστοιχες βιβλιοθήκες που απαιτούνται για την λειτουργία των αισθητήρων που θα χρησιμοποιηθούν. Συγκεκριμένα έχει εισαχθεί η βιβλιοθήκη "DHT.h" η οποία είναι αναγκαία για την λειτουργία του αισθητήρα DHT11. Έχει εισαχθεί η βιβλιοθήκη "SoftwareSerial.h" η οποία είναι αναγκαία για την λειτουργία του Xbee S1, μέσω αρχικοποίησης των ακίδων 4,5 ως RX-TX. Η βιβλιοθήκη "ArduinoJson.h" είναι αναγκαία για τη διαδικασία σχηματισμού μορφότυπου JSON για την αποστολή των δεδομένων στον κεντρικό κόμβο, με τελικό αποδέκτη το REST Server οποίος είναι υπεύθυνος για την καταχώριση των δεδομένων στη βάση δεδομένων. Η βιβλιοθήκη "MQ135.h" είναι αναγκαία για τη λειτουργία του αισθητήρα MQ135, ο οποίος λαμβάνει τις μετρήσεις ποιότητας αέρα. Η βιβλιοθήκη "Si1145.h" καθώς και η βιβλιοθήκη "Adafruit_Sensor.h" είναι αναγκαίες για τη λειτουργία του αισθητήρα υπεριώδους ακτινοβολίας. Τέλος οι βιβλιοθήκες "TimeLib.h" και "virtuabotixRTC.h" είναι αναγκαίες για τη λειτουργία του ρολογιού πραγματικού χρόνου. Οι αρχικοποιήσεις μεταβλητών παρουσιάζονται στο παράρτημα A.1.2, ενώ οι αρχικοποιήσεις των αισθητήρων εκτελούνται στο παράρτημα A.1.3. ενώ η λήψη και αποστολή των δεδομένων παρουσιάζεται στο παράρτημα.. Στο παράρτημα A.1.4. γίνεται η διαμόρφωση του JSON για τα δεδομένα των αισθητήρων και

στο παράρτημα A.1.5. γίνεται η διαμόρφωση του JSON για τα δεδομένα αποστολής του alarm. Στο παράρτημα A.1.6. παρουσιάζεται η αποστολή των μυνημάτων προς τον κεντρικό κόμβο με χρήση των ακίδων RX-TX.

6.3.2 «Έξυπνη κολώνα» Raspberry Pico W - Xbee S1»

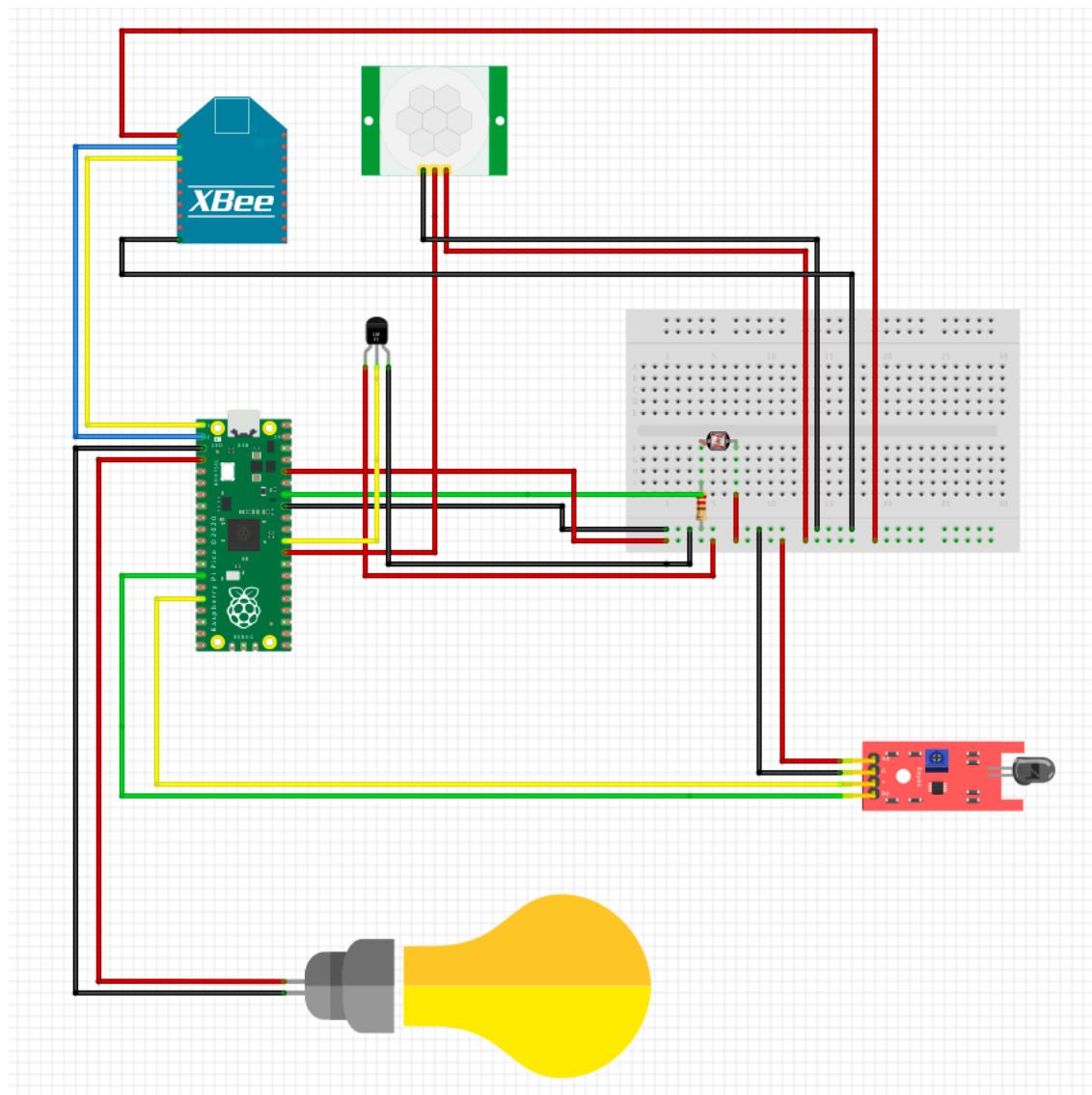
Στο συγκεκριμένο σενάριο η «έξυπνη κολώνα» λειτουργεί αυτόνομα ως προς τις λειτουργίες του φωτισμού με βάση τα δεδομένα εξωτερικού περιβάλλοντος. Συλλέγει συνεχώς πληροφορίες για το επίπεδο έντασης της ηλιακής ακτινοβολίας με χρήση φωτοάντιστασης (LDR) και ανάλογα με τις συγκεκριμένες τιμές ο controller επιλέγει αν θα ανάψει το φωτισμό LED καθώς και την ένταση λειτουργίας του, ώστε να πετύχει συνδυασμό ποιοτικού φωτισμού αλλά και εξοικονόμηση ενέργειας. Σε αντίθεση με το προηγούμενο σενάριο εξοικονόμησης ενέργειας, η συγκεκριμένη υλοποίηση περιλαμβάνει και τη λειτουργία αισθητήρα ανίχνευσης κίνησης στον περιβάλλοντα χώρο. Η ένταση του φωτός παραμένει στο 25% όταν η ένταση της ηλιακής ακτινολογίας πέσει κάτω από ένα ορισμένο επίπεδο. Όταν ο αισθητήρας κίνησης ενεργοποιηθεί τότε η ένταση παραμένει στο 100% για δύο λεπτά. Αν δεν υπάρξει στον ενδιάμεσο χρόνο κάποια κίνηση που θα επηρεάσει τον αισθητήρα κίνησης τότε η ένταση του φωτός επανέρχεται στο 25%. Επίσης η συγκεκριμένη «έξυπνη κολώνα» διαθέτει αισθητήρα ανίχνευσης ύπαρξης πυρκαγιάς. Όταν ενεργοποιείται ο συγκεκριμένος αισθητήρας τότε αποστέλλεται ειδικά διαμορφωμένο JSON για έγκαιρη εμφάνιση στην διαδικτυακή εφαρμογή, η λειτουργία της οποίας θα αναλυθεί στο επόμενο κεφάλαιο. Διαθέτει ακόμα αισθητήρα θερμοκρασίας και υγρασίας. Τα δεδομένα που αποστέλλονται από τους client, συλλέγονται μέσω του Xbee και αποστέλλονται με χρήση HTTP Request μέσω σύνδεσης WiFi. Τα δεδομένα από τις μετρήσεις που λαμβάνονται οι αισθητήρες μετασχηματίζονται σε κατάλληλη μορφή JSON και αποστέλλονται στον κεντρικό κόμβο κάθε δέκα λεπτά για τα δεδομένα των αισθητήρων και συνεχώς όταν ενεργοποιούνται τα alarm. Στη δομή του JSON περιέχονται τα δεδομένα σε μορφή ζευγαριών κλειδιού, τιμής. Κατά τη δημιουργία του μηνύματος στα δεδομένα περιέχονται το όνομα της κολώνας και τα ονόματα των αισθητήρων με τις αντίστοιχες τιμές τους.

Η υλοποίηση του παραπάνω σεναρίου απαιτεί:

- Raspberry Pico W

- XBee S1
- LDR
- LED
- DTH11 για τη μέτρηση θερμοκρασίας, υγρασίας
- Flame sensor για τον έλεγχο πυρκαγιάς
- Motion sensor για την ανίχνευση της κίνησης

Παρακάτω στην εικόνα 6-5 παρουσιάζεται η συνδεσμολογία του σεναρίου:



Εικόνα 6-5 Συνδεσμολογία σεναρίου Raspberry Pico

Σχετικά με τον κώδικα του Arduino UNO (παράρτημα B.1.1) χρειάζεται να εισαχθούν οι αντίστοιχες βιβλιοθήκες που απαιτούνται για την λειτουργία των αισθητήρων που θα χρησιμοποιηθούν. Συγκεκριμένα έχει εισαχθεί η βιβλιοθήκη " machine" η οποία είναι αναγκαία για την λειτουργία του μικροελεγκτή και συγκεκριμένα για τον καθορισμό των ακίδων και του τρόπου λειτουργίας τους. Έχει εισαχθεί η βιβλιοθήκη "urequests" η οποία είναι αναγκαία για την αποστολή των αιτημάτων http στην πλατφόρμα ThingSpeak και στο REST Server. Η βιβλιοθήκη "json" είναι αναγκαία για τη διαδικασία σχηματισμού μορφότυπου JSON για την αποστολή των δεδομένων στον κεντρικό κόμβο, με τελικό αποδέκτη το REST Server οποίος είναι υπεύθυνος για την καταχώριση των δεδομένων στη βάση δεδομένων. Η βιβλιοθήκη "network" είναι αναγκαία για τη σύνδεση του Raspberry pico W με το δίκτυο μέσω WiFi. Τέλος η βιβλιοθήκη "utime" είναι αναγκαία για την λήψη της τιμής πραγματικού χρόνου. Οι αρχικοποιήσεις μεταβλητών παρουσιάζονται στο παράρτημα B.1.2. Στο παράρτημα B.1.3. παρουσιάζεται η μέθοδος receive_data() η οποία διαβάζει τα δεδομένα που λαμβάνει από τους client και επιστρέφει το μήνυμα στη μορφή JSON. Στο παράρτημα B.1.4. παρουσιάζεται η μέθοδος send_json_to_server() η οποία αποστέλλει τα δεδομένα που λαμβάνει από τους αισθητήρες, στον REST Server. Στο παράρτημα B.1.6. παρουσιάζεται η μέθοδος manage_energy_saving() η οποία διαχειρίζεται τη λειτουργία του φωτισμού με χρήση της τιμής ηλιοφάνειας καθώς και του αισθητήρα κίνησης και προσαρμόζει ανάλογα την ένταση λειτουργίας του φωτισμού LED. Στο παράρτημα B.1.7. παρουσιάζεται η μέθοδος flame_alarm() η οποία ελέγχει την κατάσταση του alarm επικίνδυνων αερίων και σε περίπτωση ενεργοποίησης του αποστέλλει αποστέλλει JSON μήνυμα στον REST Server.

6. 3.3 «Έξυπνη κολώνα Arduino Uno – LoRa»

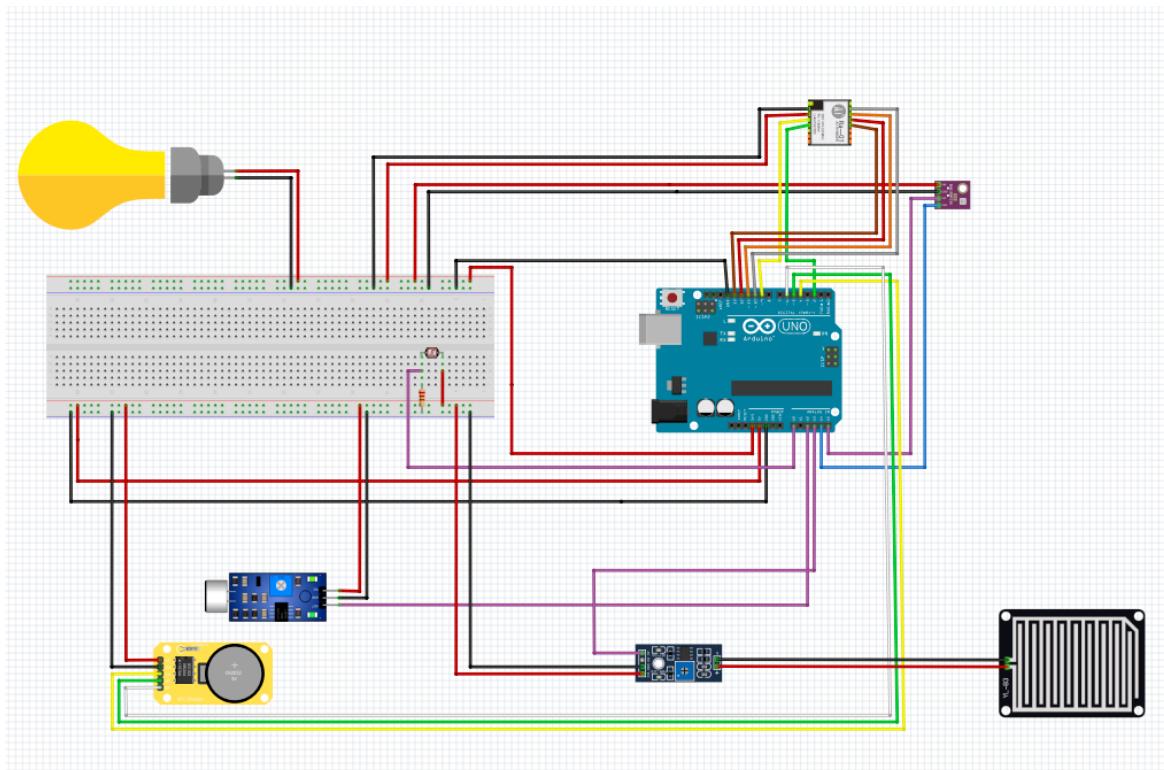
Στο συγκεκριμένο σενάριο η «έξυπνη κολώνα» λειτουργεί αυτόνομα ως προς τις λειτουργίες του φωτισμού με βάση τα δεδομένα εξωτερικού περιβάλλοντος. Συλλέγει συνεχώς πληροφορίες για το επίπεδο έντασης της ηλιακής ακτινοβολίας με χρήση φωτοάντιστασης (LDR) και ανάλογα με τις συγκεκριμένες τιμές το Arduino επιλέγει αν θα ανάψει το φωτισμό LED και προσαρμόζει την ένταση λειτουργίας του, ώστε να πετύχει συνδυασμό ποιοτικού φωτισμού αλλά και εξοικονόμηση ενέργειας. Η συγκεκριμένη «έξυπνη κολώνα» διαθέτει αισθητήρα μέτρησης θερμοκρασίας, μέτρησης υγρασίας,

μέτρησης ατμοσφαιρικής πίεσης, μέτρησης της έντασης του ήχου στον περιβάλλοντα χώρου σε dB, αισθητήρα ανίχνευσης βροχής και ρολόι πραγματικού χρόνου. Τα δεδομένα από τις μετρήσεις που λαμβάνουν οι αισθητήρες μετασχηματίζονται σε κατάλληλη μορφή JSON και αποστέλλονται στον κεντρικό κόμβο κάθε δέκα λεπτά για τα δεδομένα των αισθητήρων και συνεχώς όταν ενεργοποιούνται τα alarm. Στη δομή του JSON περιέχονται τα δεδομένα σε μορφή ζευγαριών κλειδιού, τιμής. Κατά τη δημιουργία του μηνύματος στα δεδομένα περιέχονται το όνομα της κολώνας και τα ονόματα των αισθητήρων με τις αντίστοιχες τιμές τους.

Η υλοποίηση του παραπάνω σεναρίου απαιτεί :

- Arduino Uno
- LoRa SX1278
- LDR
- LED
- Αισθητήρα μέτρησης θορύβου
- Αισθητήρα βροχής
- BME280 για τη μέτρηση θερμοκρασίας, υγρασίας και ατμοσφαιρικής πίεσης
- Real Clock module για την καταγραφή της ώρας μέτρησης

Παρακάτω στην εικόνα 6-6 παρουσιάζεται η συνδεσμολογία του σεναρίου:



Εικόνα 6-6 Συνδεσμολογία σεναρίου Arduino

Σχετικά με τον κώδικα του Arduino UNO (παράρτημα Γ) χρειάζεται να εισαχθούν οι αντίστοιχες βιβλιοθήκες που απαιτούνται για την λειτουργία των αισθητήρων που θα χρησιμοποιηθούν. Συγκεκριμένα έχει εισαχθεί η βιβλιοθήκη (παράρτημα Γ.1.1) "SPI.h" η οποία είναι αναγκαία για την για την αξιοποίηση της σειριακής επικοινωνίας. Έχει εισαχθεί η βιβλιοθήκη (παράρτημα Γ.1.1) "LoRa.h" η οποία είναι αναγκαία για την λειτουργία του LoRa, μέσω αρχικοποίησης των digital ακίδων 10, 11, 12 και 13. Η βιβλιοθήκη (παράρτημα Γ.1.1) "WiFi.h", "WiFiUdp.h" καθώς και "NTPClient.h" είναι αναγκαίες για τη σύνδεση του Raspberry pico W με το δίκτυο μέσω WiFi. Η βιβλιοθήκη (παράρτημα Γ.1.1) "ArduinoJson.h" είναι αναγκαία για τη διαδικασία σχηματισμού μορφότυπου JSON για την αποστολή των δεδομένων στο REST Server οποίος είναι υπεύθυνος για την καταχώριση των δεδομένων στη βάση δεδομένων. Η βιβλιοθήκη (παράρτημα Γ.1.1) "Adafruit_BMP280.h" είναι αναγκαία για τη λειτουργία του αισθητήρα BME280, ο οποίος λαμβάνει τις μετρήσεις θερμοκρασία, υγρασίας και ατμοσφαιρικής πίεσης. Τέλος, οι βιβλιοθήκες (παράρτημα Γ.1.1) "TimeLib.h" και "virtuabotixRTC.h" είναι αναγκαίες για τη λειτουργία του ρολογιού πραγματικού χρόνου. Οι αρχικοποίησεις μεταβλητών παρουσιάζονται στο παράρτημα Γ.1.2, ενώ οι αρχικοποίησεις των αισθητήρων εκτελούνται

στο παράρτημα Γ.1.3. ενώ η λήψη και αποστολή των δεδομένων παρουσιάζεται στο παράρτημα.. Στο παράρτημα Γ.1.4. γίνεται η διαμόρφωση του JSON για τα δεδομένα των αισθητήρων και στο παράρτημα Γ.1.5. γίνεται η διαμόρφωση του JSON για τα δεδομένα αποστολής του alarm. Στο παράρτημα Γ.1.6. παρουσιάζεται η αποστολή των μηνυμάτων προς τον κεντρικό κόμβο.

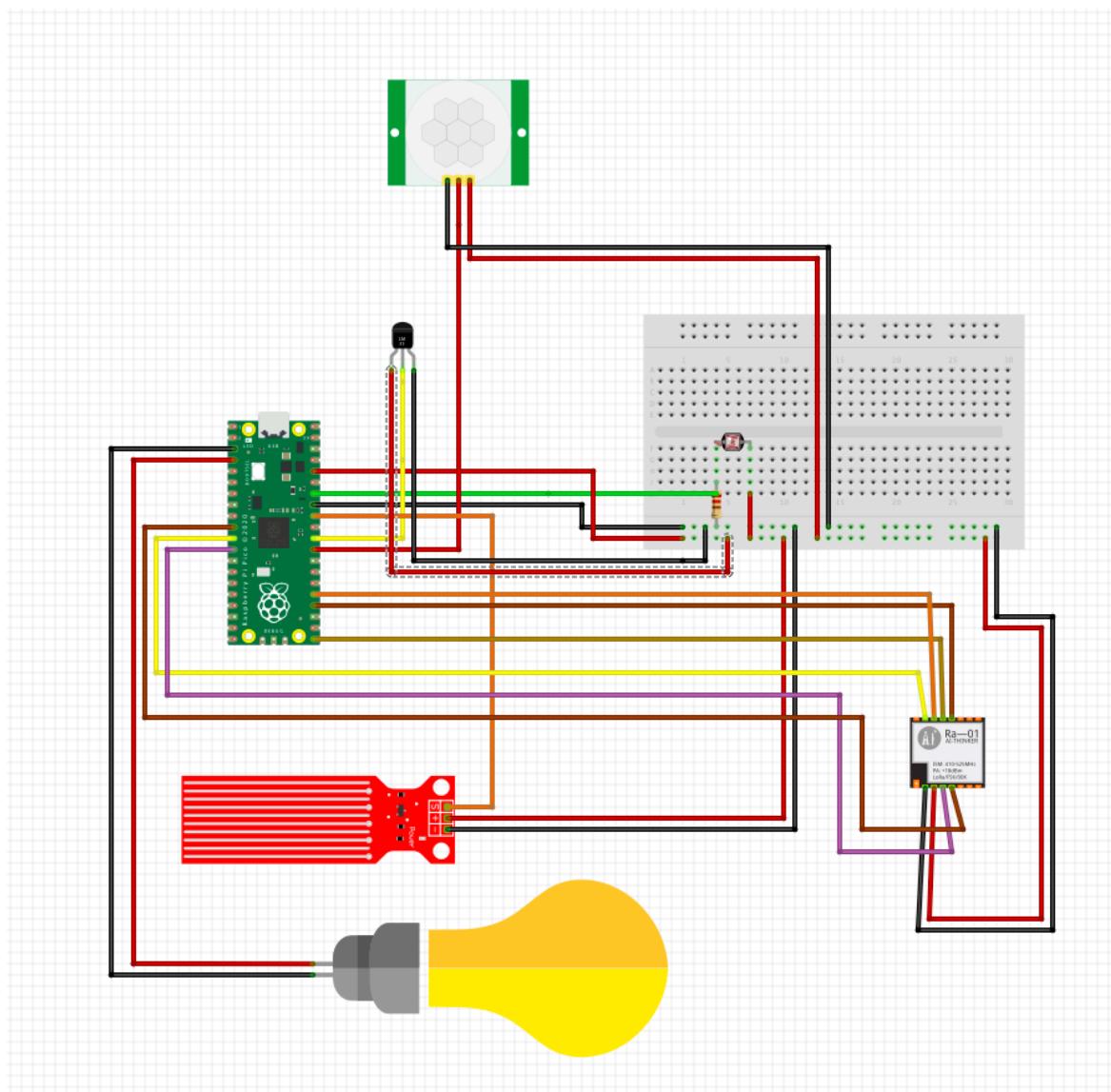
6. 3.4 «Έξυπνη κολώνα» Raspberry Pico W – LoRa

Στο συγκεκριμένο σενάριο η «έξυπνη κολώνα» λειτουργεί αυτόνομα ως προς τις λειτουργίες του φωτισμού με βάση τα δεδομένα εξωτερικού περιβάλλοντος. Συλλέγει συνεχώς πληροφορίες για το επίπεδο έντασης της ηλιακής ακτινοβολίας με χρήση φωτοαντίστασης (LDR) και ανάλογα με τις συγκεκριμένες τιμές o controller επιλέγει αν θα ανάψει το φωτισμό LED καθώς και την ένταση λειτουργίας του, ώστε να πετύχει συνδυασμό ποιοτικού φωτισμού αλλά και εξοικονόμηση ενέργειας. Σε αντίθεση με το προηγούμενο σενάριο εξοικονόμησης ενέργειας, η συγκεκριμένη υλοποίηση περιλαμβάνει και τη λειτουργία αισθητήρα ανίχνευσης κίνησης στον περιβάλλοντα χώρο. Η ένταση του φωτός παραμένει στο 25% όταν η ένταση της ηλιακής ακτινολογίας πέσει κάτω από ένα ορισμένο επίπεδο. Όταν ο αισθητήρας κίνησης ενεργοποιηθεί τότε η ένταση παραμένει στο 100% για δύο λεπτά. Αν δεν υπάρξει στον ενδιάμεσο χρόνο κάποια κίνηση που θα επηρεάσει τον αισθητήρα κίνησης τότε η ένταση του φωτός επανέρχεται στο 25%. Επίσης η συγκεκριμένη «έξυπνη κολώνα» διαθέτει αισθητήρα ανίχνευσης υπερχείλισης των φρεατίων όμβριων υδάτων. Όταν ενεργοποιείται ο συγκεκριμένος αισθητήρας τότε αποστέλλεται ειδικά διαμορφωμένο JSON για έγκαιρη εμφάνιση στην διαδικτυακή εφαρμογή, η λειτουργία της οποίας θα αναλυθεί στο επόμενο κεφάλαιο. Διαθέτει ακόμα αισθητήρα θερμοκρασίας. Τα δεδομένα που αποστέλλονται από τους client, συλλέγονται μέσω του LoRa και αποστέλλονται με χρήση HTTP Request μέσω σύνδεσης WiFi. Τα δεδομένα από τις μετρήσεις που λαμβάνουν οι αισθητήρες μετασχηματίζονται σε κατάλληλη μορφή JSON και αποστέλλονται στον κεντρικό κόμβο κάθε δέκα λεπτά για τα δεδομένα των αισθητήρων και συνεχώς όταν ενεργοποιούνται τα alarm. Στη δομή του JSON περιέχονται τα δεδομένα σε μορφή ζευγαριών κλειδιού, τιμής. Κατά τη δημιουργία του μηνύματος στα δεδομένα περιέχονται το όνομα της κολώνας και τα ονόματα των αισθητήρων με τις αντίστοιχες τιμές τους.

Η υλοποίηση του παραπάνω σεναρίου απαιτεί:

- Raspberry Pico W
- LoRa SX1278
- LDR
- LED
- TMP36 για τη μέτρηση θερμοκρασίας
- Liquid sensor για τον έλεγχο υπερχείλισης φρεατίων όμβριων υδάτων
- Motion sensor για την ανίχνευση της κίνησης

Παρακάτω στην εικόνα 6-7 παρουσιάζεται η συνδεσμολογία του σεναρίου:

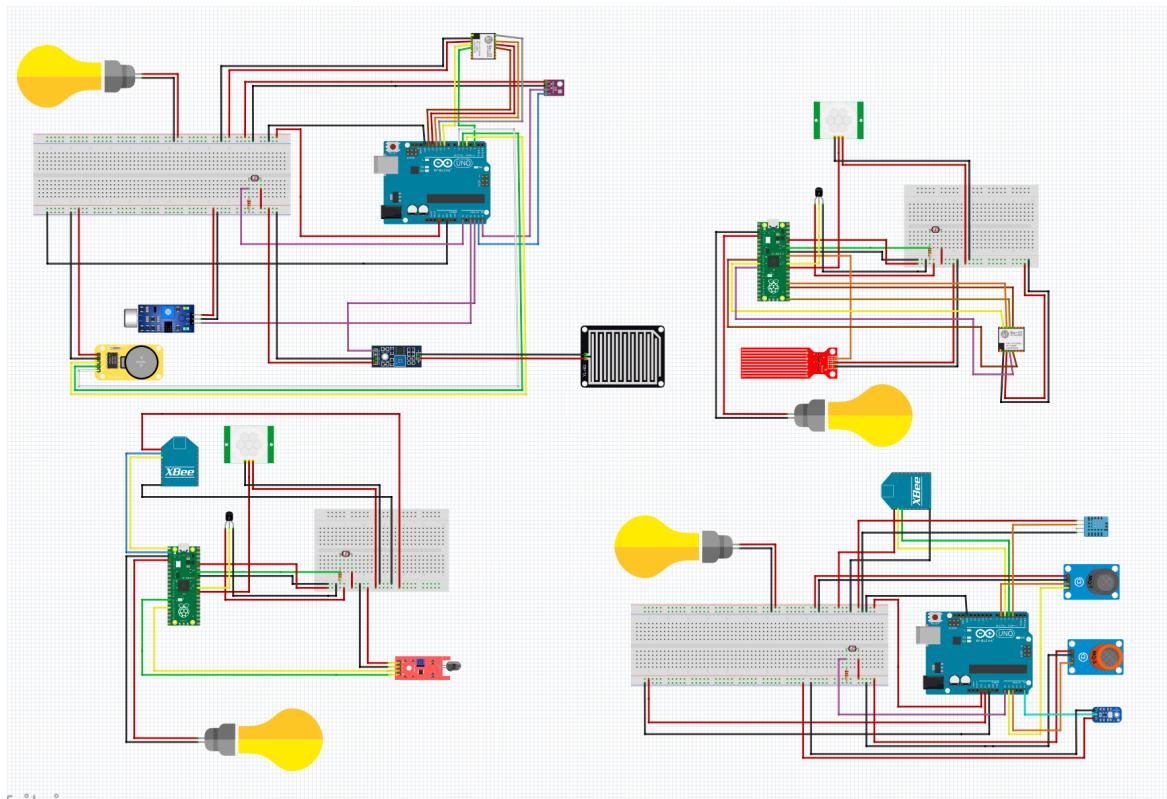


Εικόνα 6-7 Συνδεσμολογία Raspberry Pico

Σχετικά με τον κώδικα του Raspberry Pico W (παράρτημα Δ) χρειάζεται να εισαχθούν οι αντίστοιχες βιβλιοθήκες που απαιτούνται για την λειτουργία των αισθητήρων που θα χρησιμοποιηθούν. Συγκεκριμένα έχει εισαχθεί η βιβλιοθήκη (παράρτημα Δ.1.1) "SPI.h" η οποία είναι αναγκαία για την για την αξιοποίηση της σειριακής επικοινωνίας. Έχει εισαχθεί η βιβλιοθήκη (παράρτημα Δ.1.1) "LoRa.h" η οποία είναι αναγκαία για την λειτουργία του LoRa, μέσω αρχικοποίησης των digital ακίδων 8, 9 και 10. Η βιβλιοθήκη (παράρτημα B.1.1) "ArduinoJson.h" είναι αναγκαία για τη διαδικασία σχηματισμού μορφότυπου JSON για την αποστολή των δεδομένων στον κεντρικό κόμβο, με τελικό αποδέκτη το REST Server οποίος είναι υπεύθυνος για την καταχώριση των δεδομένων στη βάση δεδομένων. Τέλος, έχει εισαχθεί η βιβλιοθήκη (παράρτημα Δ.1.1) "HTTPClient.h" η οποία είναι αναγκαία για την αποστολή των αιτημάτων http στο REST Server. Οι αρχικοποιήσεις μεταβλητών παρουσιάζονται στο παράρτημα Δ.1.2. Στο παράρτημα Δ.1.3. παρουσιάζεται η μέθοδος receiveMessage() η οποία διαβάζει τα δεδομένα που λαμβάνει από τους client και επιστρέφει το μήνυμα στη μορφή JSON. Στο παράρτημα Δ.1.4. παρουσιάζεται η μέθοδος sendSensorDataToServer() η οποία αποστέλλει τα δεδομένα που λαμβάνει από τους client. Στο παράρτημα Δ.1.5. παρουσιάζεται η μέθοδος send_myData_to_server() η οποία αποστέλλει τα δεδομένα που λαμβάνει από τους αισθητήρες, στον REST Server. Στο παράρτημα Δ.1.6. παρουσιάζεται η μέθοδος manageEnergySaving() η οποία διαχειρίζεται τη λειτουργία του φωτισμού με χρήση της τιμής ηλιοφάνειας καθώς και του αισθητήρα κίνησης και προσαρμόζει ανάλογα την ένταση λειτουργίας του φωτισμού LED. Στο παράρτημα Δ.1.7. παρουσιάζεται η μέθοδος checkWaterSensor() η οποία ελέγχει την κατάσταση του alarm υπερχείλισης φρεατίου όμβριων υδάτων και σε περίπτωση ενεργοποίησης του αποστέλλει JSON μήνυμα στον REST Server.

6.4 Συνδεσμολογία πλακετών ανάπτυξης

Στην εικόνα 6-8 παρουσιάζεται η συνδεσμολογία των πλακετών ανάπτυξης όλου του συστήματος που υλοποιήθηκε. Η δημιουργία του σχεδίου έγινε με χρήση του λογισμικού Fritzing.



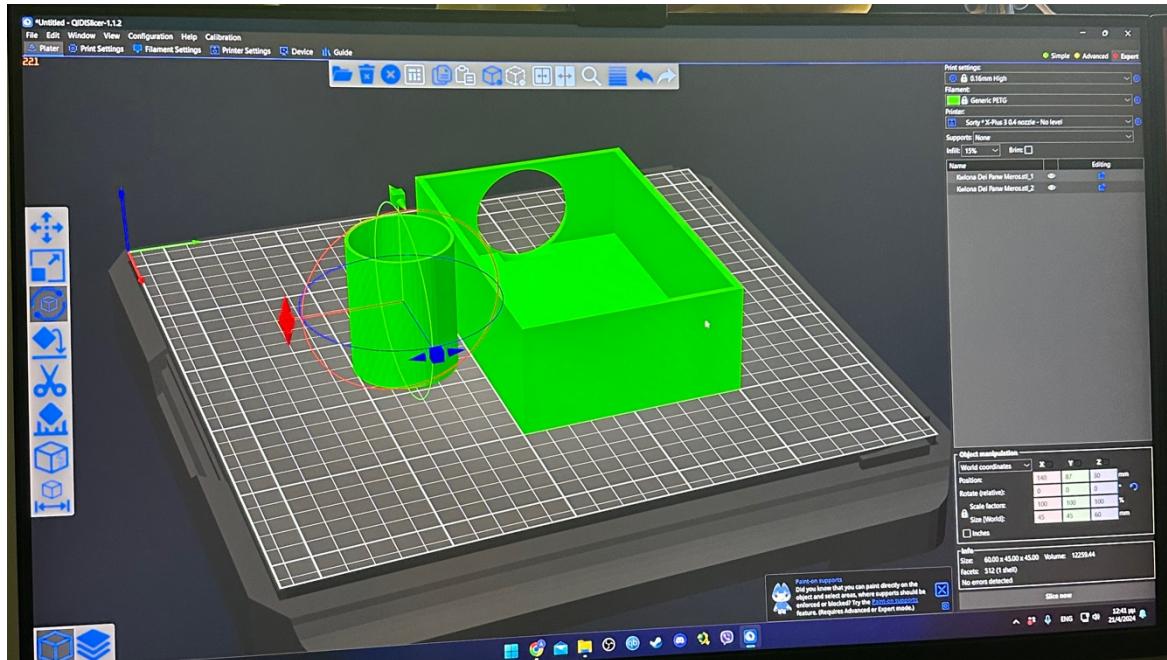
Εικόνα 6-8 Συνδεσμολογία συστήματος υλοποίησης

6.5 Κατασκευή Έξυπνων κολώνων Φωτισμού με χρήση τεχνολογίας εκτύπωσης 3D

Στην παρούσα πτυχιακή εργασία, η κατασκευή των θηκών που φιλοξένησαν τους αισθητήρες και τους μικροελεγκτές πραγματοποιήθηκε με τη χρήση τεχνολογίας εκτύπωσης 3D. Η επιλογή της τρισδιάστατης εκτύπωσης προσέφερε την ευελιξία και την ακρίβεια που απαιτούνταν για τη δημιουργία προσαρμοσμένων θηκών, οι οποίες ανταποκρίνονταν ακριβώς στις ανάγκες του έργου.

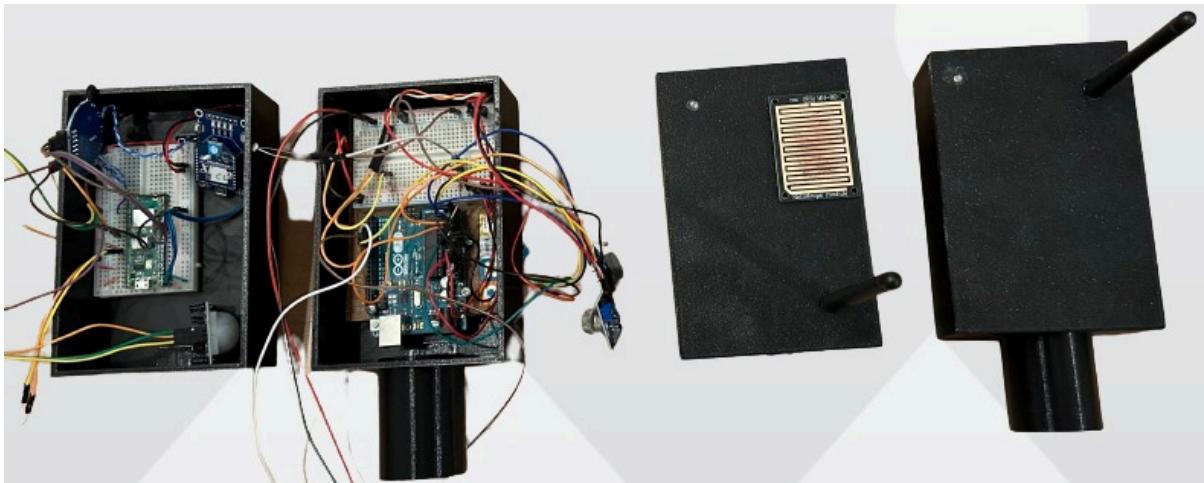
Προκειμένου να διασφαλιστεί η σωστή τοποθέτηση και προστασία των αισθητήρων και των μικροελεγκτών, σχεδιάστηκαν και υλοποιήθηκαν ορθογώνια κουτιά με ειδικές υποδοχές. Οι υποδοχές αυτές ήταν σχεδιασμένες έτσι ώστε να εξασφαλίζεται η σταθερή τοποθέτηση και η ασφάλεια των υλικών κατά την υλοποίηση και τη λειτουργία του συστήματος. Ο σχεδιασμός των κουτιών πραγματοποιήθηκε μέσω εξειδικευμένων λογισμικών ThinkerCAD και 3D-Builder, τα οποία επέτρεψαν την ακριβή διαμόρφωση των διαστάσεων και των χαρακτηριστικών των θηκών.

Η διαδικασία της εκτύπωσης 3D ξεκίνησε με τη μετατροπή των σχεδίων CAD σε μορφή αρχείων STL, τα οποία στη συνέχεια τροφοδοτήθηκαν στον εκτυπωτή 3D. Χρησιμοποιήθηκε θερμοπλαστικό υλικό υψηλής αντοχής PETG Filament, το οποίο εξασφάλισε την ανθεκτικότητα των θηκών. Η χρήση της τεχνολογίας εκτύπωσης 3D επέτρεψε την υλοποίηση θηκών για την τοποθέτηση των υλικών των συστημάτων, για την κάλυψη των απαιτήσεων του έργου. Στην εικόνα 6-9 παρουσιάζεται το σχέδιο από τα κουτιά που κατασκευάστηκαν για την φιλοξενία των υλικών του συστήματος.



Εικόνα 6-9 Σχέδιο εκτύπωσης με τεχνολογία 3D

6.6 Συνδεσμολογία πλακετών ανάπτυξης - Φωτογραφίες Υλοποίησης



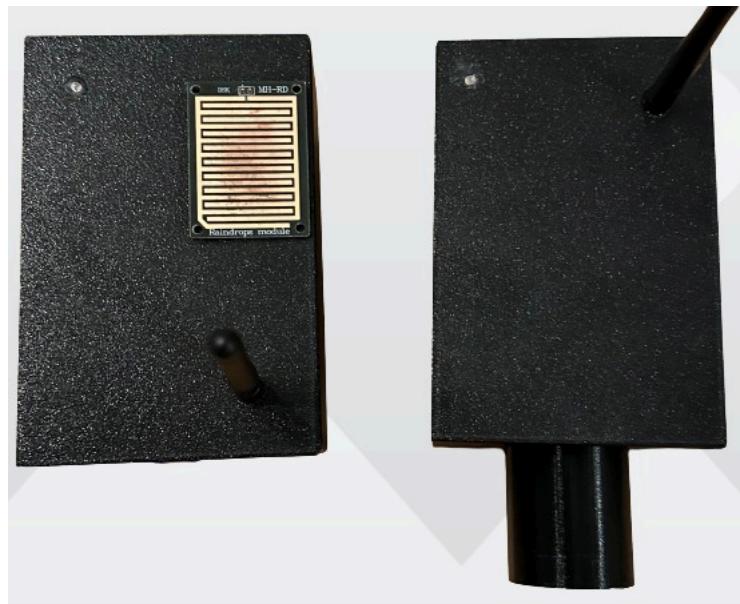
Εικόνα 6-10 Έξυπνες κολώνες υλοποίησης



Εικόνα 6-11 Αισθητήρας βροχής στο πάνω μέρος της κολώνας



Εικόνα 6-12 Εσωτερικό τμήμα της έξυπνης κολώνας



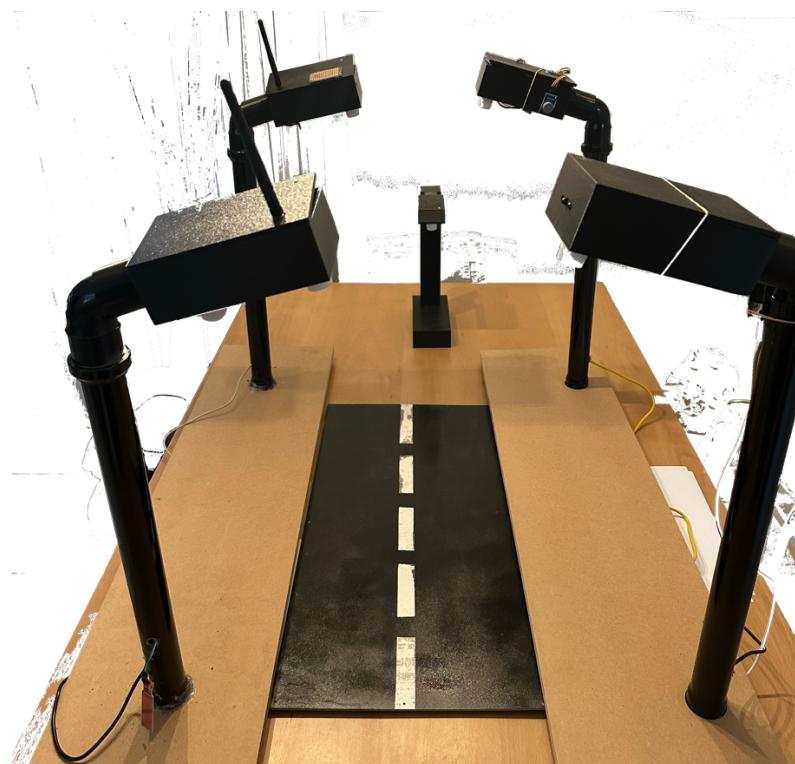
Εικόνα 6-13 Οι έξυπνες κολώνες με επικοινωνία LoRa



Εικόνα 6-14 Συνολική υλοποίηση της έξυπνης κολώνας



Εικόνα 6- 15 Έξυπνη κολώνα με Αισθητήρας κίνησης



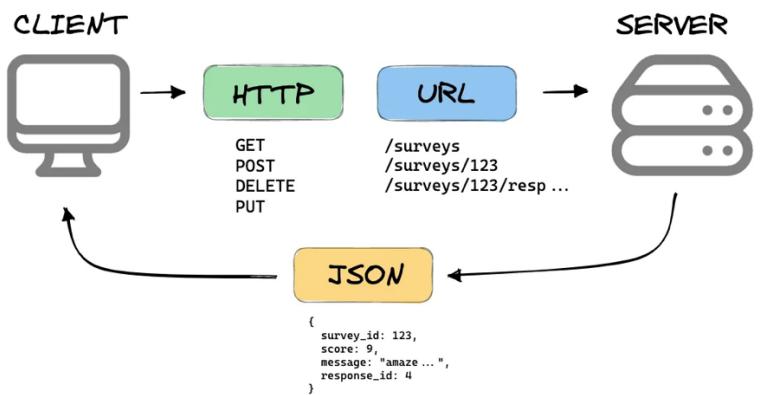
Εικόνα 6- 16 Μακέτα Δικτύου με έξυπνες Κολώνες

7. Υλοποίηση Διασύνδεσης

7.1 Εισαγωγή

Στο 7ο κεφάλαιο θα γίνει ανάλυση της υλοποίησης της διασύνδεσης του δικτύου των «Έξυπνων κολώνων» φωτισμού, με τον REST Server, την ιστοσελίδα, καθώς και την IoT πλατφόρμα ThingSpeak. Αρχικά θα αναλυθεί η δομή της βάσης δεδομένων, στην συνέχεια ο Rest Server, ο ThingSpeak API και τέλος η διαδικτυακή εφαρμογή. Στην εικόνα 7-1 παρουσιάζεται γενικό διάγραμμα από Rest API.

WHAT IS A REST API?

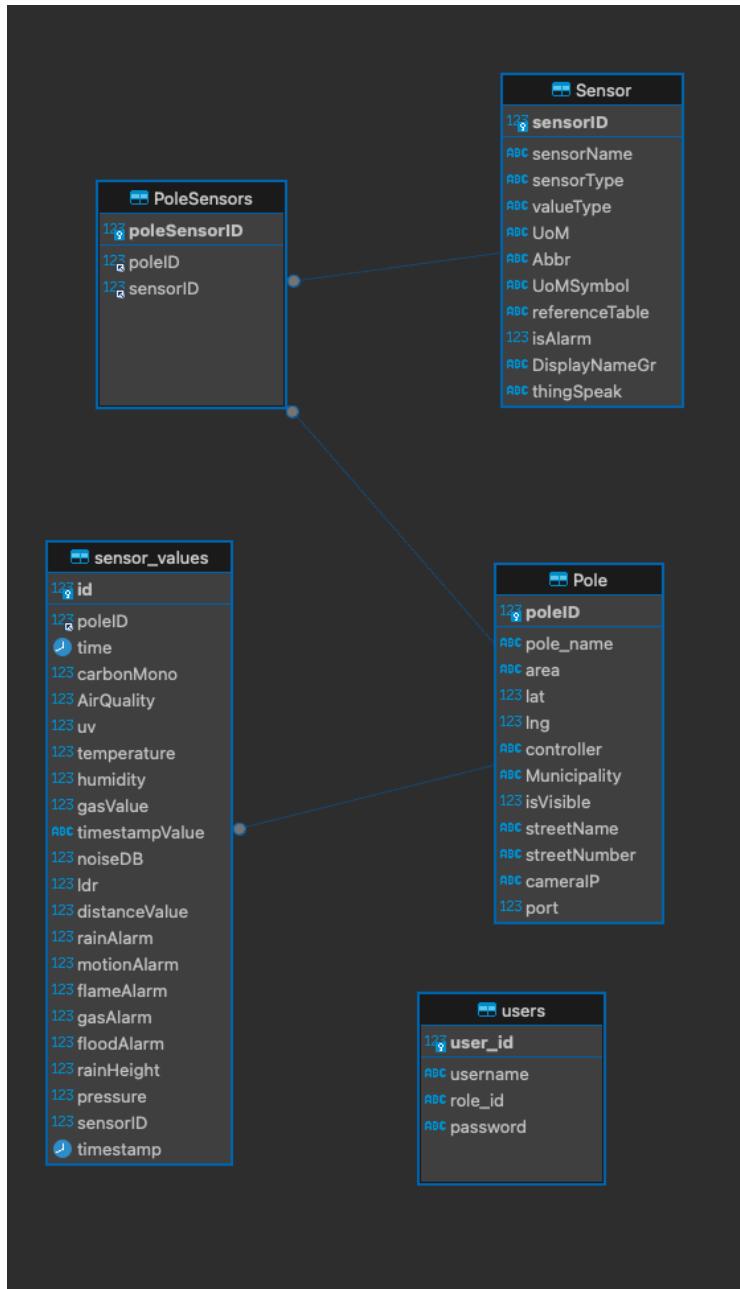


Εικόνα 7-1 Διάγραμμα REST API
(*Understanding REST API: A Comprehensive Guide* | Medium, 2024)

7.2 Βάση δεδομένων

Το σύστημα για την υλοποίησή του, χρησιμοποιεί βάση δεδομένων MariaDB (mySQL). Η βάση δεδομένων περιέχει 5 πίνακες. Ο πρώτος πίνακας "users" αναφέρεται στους χρήστες και τα δικαιώματα που κατέχουν για την πρόσβαση της ιστοσελίδας. Ο δεύτερος πίνακας "sensor_values" αναφέρεται στις εγγραφές των αισθητήρων και των alarm από όλες τις «έξυπνες κολώνες» του δικτύου, ο τρίτος πίνακας "pole" αναφέρεται σε κάθε «έξυπνη κολώνα» και περιέχει όλες τις πληροφορίες που σχετίζονται με αυτές, ο τέταρτος πίνακας "sensor" αναφέρεται στους αισθητήρες και τις ιδιότητές τους και τέλος ο πέμπτος πίνακας "PoleSensors" αναφέρεται στην υλοποίηση της σχέσης μεταξύ των «έξυπνων κολώνων»

και των αισθητήρων, εκφράζοντας τη σχέση «many – to – many». Παρακάτω στην εικόνα 7-2 παρουσιάζεται το διάγραμμα οντοτήτων - συσχετίσεων της βάσης δεδομένων:



Εικόνα 7- 2 Σχήμα βάσης Δεδομένων

7.3 Rest Server

Οι υπηρεσίες REST (Representational State Transfer) είναι ένας αρχιτεκτονικός τρόπος σχεδίασης APIs (Application Programming Interfaces), που επιτρέπουν την αλληλεπίδραση με διαδικτυακές υπηρεσίες. Δημιουργημένες από τον Roy Fielding, οι REST APIs

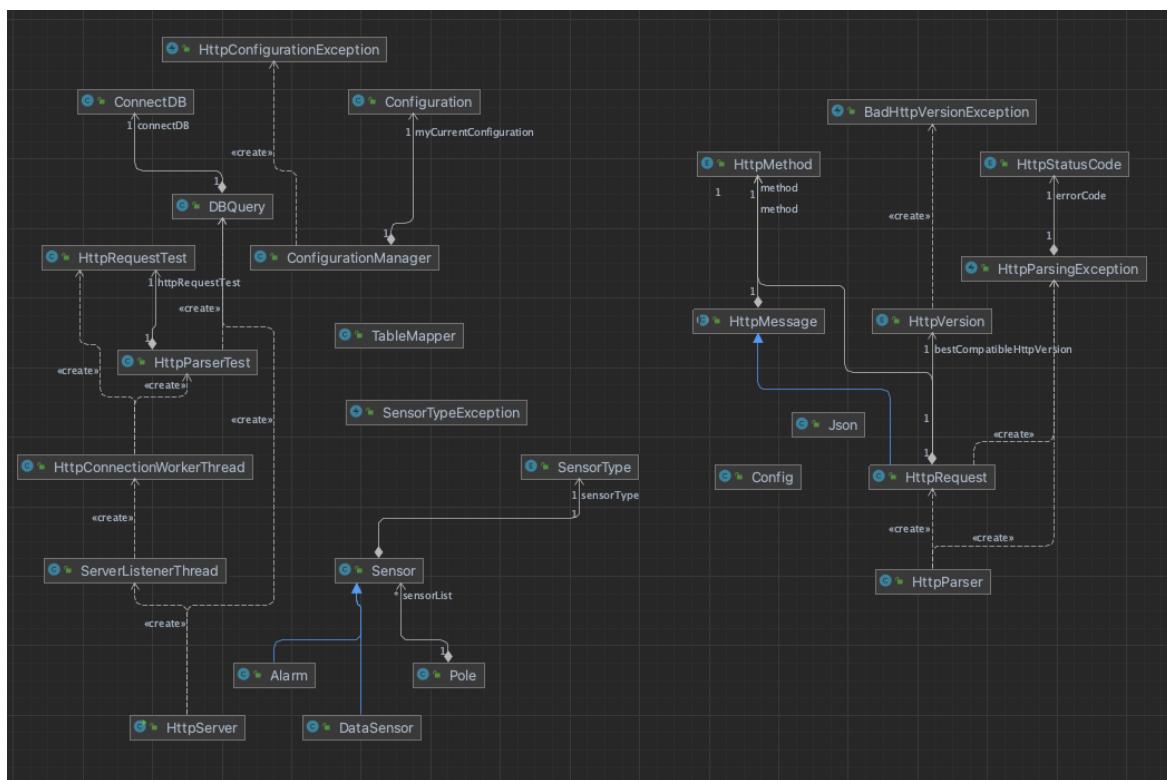


λειτουργούν μέσω αιτημάτων HTTP για την εκτέλεση CRUD λειτουργιών (δημιουργία, ανάγνωση, ενημέρωση, διαγραφή) σε πόρους. Ένα API γενικά είναι ένα σύνολο ορισμών και πρωτοκόλλων για την ανάπτυξη και ενσωμάτωση λογισμικού, λειτουργώντας ως μεσολαβητής μεταξύ των χρηστών και των πόρων που θέλουν να προσπελάσουν. Τα RESTful APIs χρησιμοποιούν αιτήματα HTTP για να μεταφέρουν πληροφορίες σε μορφές όπως JSON, HTML, XML ή απλό κείμενο. Αυτή η μορφή επικοινωνίας εξασφαλίζει ότι οι αιτήσεις και οι απαντήσεις περιέχουν όλες τις απαραίτητες πληροφορίες για την ολοκλήρωση των λειτουργιών, χωρίς να αποθηκεύουν την κατάσταση των αιτημάτων μεταξύ τους. Για να θεωρηθεί ένα API RESTful, πρέπει να τηρεί ορισμένα κριτήρια, όπως η αρχιτεκτονική client-server, η στατική επικοινωνία, η δυνατότητα αποθήκευσης στην cache, η ομοιόμορφη διεπαφή, το πολυεπίπεδο σύστημα και, προαιρετικά, η δυνατότητα αποστολής εκτελέσιμου κώδικα από τον server στον client. Τα REST APIs είναι ευέλικτα και ελαφριά, καθιστώντας τα ιδανικά για ανάπτυξη εφαρμογών Internet of Things (IoT) και κινητών εφαρμογών, σε αντίθεση με πρωτόκολλα όπως το SOAP, που είναι πιο πολύπλοκα και βαρύτερα. (What is a REST API? | Red Hat, n.d.).

Για την υλοποίηση του Rest Server, στη παρούσα εργασία θα χρησιμοποιηθεί το Maven Framework της Java. Για να μπορεί να εκτελεστεί ο server προϋποθέτει εγκατεστημένο το JDK 1.8 ή νεότερη έκδοση. Ο Rest Server λαμβάνει αιτήματα HTTP από τους κεντρικούς κόμβους των δικτύων για τα οποία έγινε αναφορά στον τρόπο λειτουργίας τους στο προηγούμενο κεφάλαιο. Τα συγκεκριμένα αιτήματα λαμβάνονται και επεξεργάζονται ώστε να γίνει η καταχώριση των δεδομένων στην βάση και να μπορούν να αξιοποιηθούν από τη διαδικτυακή εφαρμογή με σκοπό την προβολή τους προς το χρήστη. Ο Rest Server έχει φτιαχτεί με χρήση της τεχνολογίας νημάτων (thread). Κάθε φορά που λαμβάνεται ένα νέο αίτημα Http το Server αναθέτει τη διαχείριση του σε ένα thread το οποίο είναι υπεύθυνο για τη διαχείριση του. Συγκεκριμένα, ο Server με χρήση της κλάσης ServerListenerThread δημιουργεί ένα thread της κλάσης αυτής το οποίο δέχεται σε συγκεκριμένη πόρτα (port) για http αίτημα και δημιουργεί ένα thread της κλάσης HttpURLConnectionWorkerThread. Το συγκεκριμένο thread διαχωρίζει από το αίτημα την πληροφορία η οποία είναι σε μορφή JSON και αποδομεί το περιεχόμενο της ώστε να είναι ξεκάθαρες οι πληροφορίες που περιέχει και να μπορέσουν να εισαχθούν στη βάση. Στο κάθε JSON περιέχεται η πληροφορία σε μορφή κλειδιού – τιμής. Υπεύθυνη για την αποδόμηση της πληροφορίας είναι η κλάση HttpParser με χρήση καθορισμένης δομής από την κλάση TableMapper, η

οποία περιέχει την συσχέτιση μεταξύ των ονομάτων των αισθητήρων και των ονομάτων των αντίστοιχων στηλών του πίνακα sensor_values. Υπεύθυνες για την σύνδεση και την εισαγωγή των δεδομένων στη Βάση είναι οι κλάσεις ConnectDB και DBQuery. Οι τιμές που περιέχονται στο JSON καταχωρούνται στη βάση δεδομένων με το μοναδικό αριθμό που έχει η «έξυπνη κολώνα» (poleId) στον πίνακα sensor_values. Με την ολοκλήρωση της καταχώρισης των τιμών στο Server τα δεδομένα είναι διαθέσιμα από την διαδικτυακή εφαρμογή για προβολή στο χρήστη.

Στην εικόνα 7-3 παρουσιάζονται οι συναρτήσεις της εφαρμογής, καθώς στο Παράρτημα ΣΤ.1 παρουσιάζεται ο κώδικας του.



Εικόνα 7-3 Διάγραμμα εξαρτήσεων Rest Server

7.4 ThingSpeak API

Η πλατφόρμα ThingSpeak προσφέρει ένα εύχρηστο εργαλείο για την προβολή και ανάλυση δεδομένων, από συσκευές IoT. Επίσης τα δεδομένα παραμένουν αποθηκευμένα στο cloud με αποτέλεσμα να είναι προσβάσιμα από οποιοδήποτε σημείο βελτιστοποιώντας περαιτέρω τους τρόπους επεξεργασία και ανάλυσης των δεδομένων. Στην παρούσα εργασία επιλέχτηκε η χρήση της πλατφόρμας μέσω των δυνατοτήτων που παρέχει η δημιουργία

δωρεάν λογαριασμού πρόσβασης. Η διαδικασία δημιουργίας ενός καναλιού στην πλατφόρμα είναι απλή και περιλαμβάνει την εγγραφή από την ιστοσελίδα του ThingSpeak (<https://thingspeak.com>) όπου μετά τη συμπλήρωση των απαραίτητων στοιχείων γίνεται η δημιουργία λογαριασμού πρόσβασης. Στη συνέχεια από το μενού "Channels" και την επιλογή "My Channels" ο χρήστης μπορεί να δημιουργήσει ένα νέο κανάλι (εικόνα 7-4), συμπληρώνοντας την φόρμα με τα απαραίτητα πεδία, όπως το όνομα του καναλιού και την περιγραφή του. Τέλος, προσθέτοντας τα κατάλληλα ονόματα στα πεδία δεδομένων (Fields) που θα χρησιμοποιηθούν για την αποθήκευση των δεδομένων (εικόνα 7-5) το κανάλι είναι έτοιμο προς χρήση. Η δημιουργία και η παραμετροποίηση των διαγραμμάτων γίνεται από την φόρμα ρυθμίσεων του κάθε πεδίου (Field chart options) που παρουσιάζεται στην εικόνα 7-6. Στην εικόνα 7-7 παρουσιάζεται ένα διάγραμμα θερμοκρασίας που δημιουργείται μετά την παραμετροποίηση των απαραίτητων πεδίων στις φόρμα των ρυθμίσεων. Για τη διασύνδεση του καναλιού με συσκευές IoT καθώς και με ιστοσελίδες που θα προβάλουν τα δεδομένα του πραγματοποιείται μέσω Rest API. Η πλατφόρμα παρέχει δύο κλειδιά (API keys), ένα για δικαιώματα ανάγνωσης και ένα για δικαιώματα ανάγνωσης και εγγραφής (εικόνα 7-7) ώστε να παρέχεται η δυνατότητα σύνδεσης των widgets και των διαγράμματα με προσωπική ιστοσελίδα για τη δυναμική εμφάνιση των δεδομένων. Για την προβολή των widget ή των διαγραμμάτων, απαιτείται η ρύθμιση του καναλιού σε "Public View" από "Private View". Μέσω αυτής της επιλογής, παρέχεται η δυνατότητα του διαμοιρασμού και της ενσωμάτωσης των widget ή των διαγραμμάτων σε κώδικα HTML. Ο κώδικας ενσωμάτωσης (embed code) παρέχεται σε μορφή HTML iframe (εικόνα 7-8).

Name	Created	Updated			
PICO W	2023-09-21	2024-05-15 08:41			
Private	Public	Settings	Sharing	API Keys	Data Import / Export



Εικόνα 7-4 Δημιουργία καναλιού ThingSpeak

Channel Settings

Percentage Complete 50%

Channel ID 2277540

Name PICO W

Description PICO Xbee Coordinator

Field 1 ldrValue

Field 2 temperature

Field 3 humidity

Field 4 mq7Value

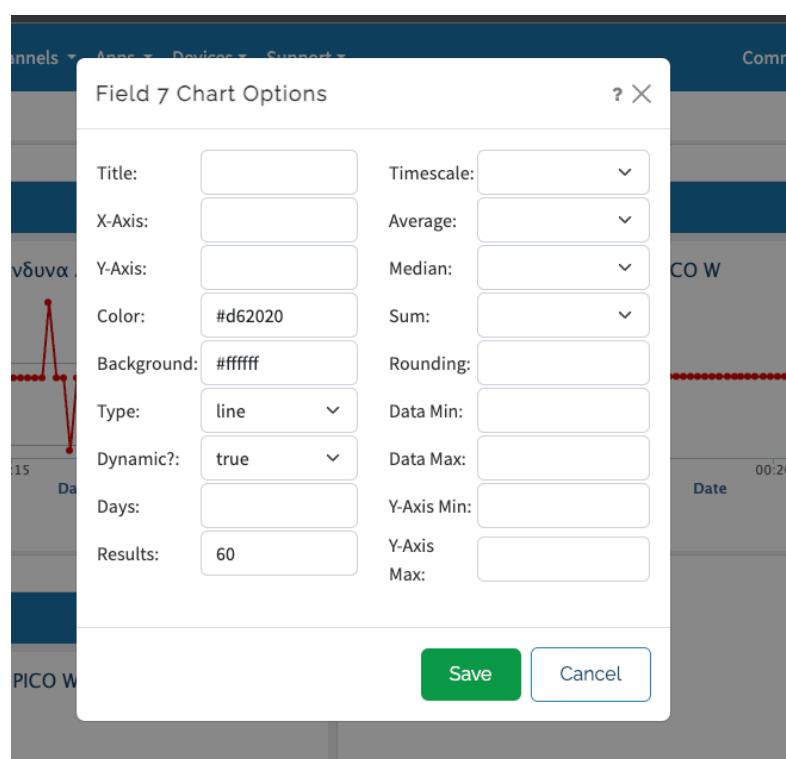
Field 5 co2Alarm

Field 6 PPM_MQ135

Field 7

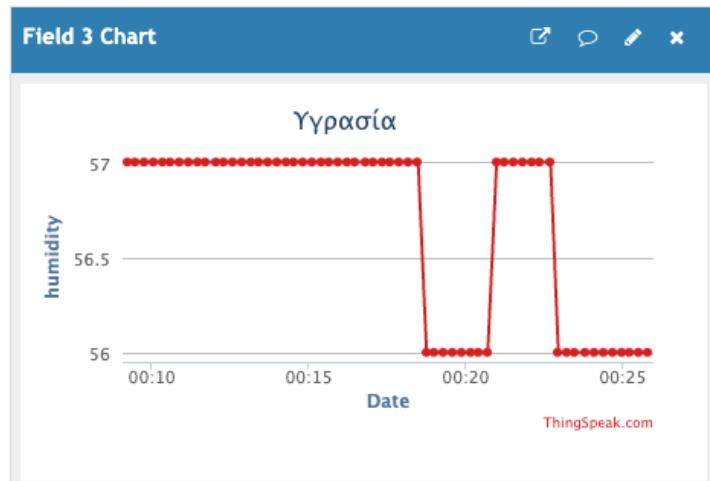
Field 8

Εικόνα 7-5 Ρυθμίσεις πεδίων





Εικόνα 7-6 Δημιουργία - Ρύθμιση Διαγράμματος



Εικόνα 7-7 Διάγραμμα Θερμοκρασίας

Private View Public View Channel Settings Sharing API Keys

Write API Key

Key

Generate New Write API Key

Read API Keys

Key

Note

Save Note

Delete API Key

Add New Read API Key

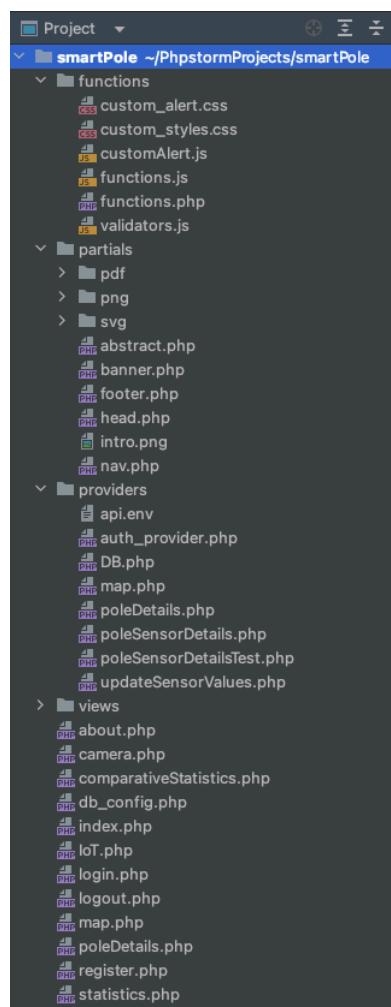
Εικόνα 7-8 Δημιουργία κλειδιών API



Εικόνα 7-9 Κώδικας HTML - iFrame

7.5 Διαδικτυακή εφαρμογή (PHP, ThingSpeak, Google charts)

Η ανάπτυξη της διαδικτυακής έγινε με χρήση της γλώσσας PHP και JavaScript. Η δομή του φακέλου της διαδικτυακής εφαρμογής παρουσιάζεται στην εικόνα 7-8.



Εικόνα 7-10 Δομή φακέλου Διαδικτυακής Εφαρμογής



Συγκεκριμένα στο φάκελο functions περιέχονται τα αρχεία PHP και JavaScript, στον φάκελο css τα αρχεία CSS, στον φάκελο partials διάφορα αρχεία PHP που χρησιμοποιεί η κάθε ιστοσελίδα, όπως είναι τα στοιχεία heap, header, footer και nav, καθώς επίσης εικόνες, εικονίδια και αρχεία pdf. Στο φάκελο providers περιέχονται αρχεία PHP τα οποία εκτελούν διεργασίες στο παρασκήνιο ώστε να υπάρχουν τα δεδομένα από τη βάση διαθέσιμα κατά την εκκίνηση των ιστοσελίδων που τα χρειάζονται. Στο φάκελο views είναι όλα τα αρχεία PHP που περιέχουν τον κώδικα HTML των σελίδων της εφαρμογής. Τέλος στην ρίζα του φακέλου βρίσκονται τα αρχεία PHP, τα οποία είναι οι routers των αντίστοιχων αρχείων που υπάρχουν στο φάκελο views.

Η διαδικτυακή εφαρμογή χρησιμοποιεί τις βασικές προδιαγραφές ασφαλείας και επομένως για την είσοδο του χρήστη (εικόνα 7-9) στην εφαρμογή απαιτείται επιτυχής αυθεντικοποίηση (εικόνα 7-10) σε διαφορετική περίπτωση ο χρήστης μπορεί να κάνει εγγραφή για πρόσβαση συμπληρώνοντας τα αντίστοιχα πεδία στη φόρμα εγγραφής χρήστη (εικόνα 7-11).



Σχολή Θετικών Επιστημών και Τεχνολογίας

Τμήμα Πληροφορικής

Πτυχιακή Εργασία ΠΕ419

Σχεδιασμός και ανάπτυξη «Έξυπνης Κολώνας»
δημόσιου φωτισμού σε περιβάλλον μίας έξυπνης πόλης

Επιβλέπων Καθηγητής
Δρ. Τοπάλης Ευάγγελος

Φοιτητής
Κυριακίδης Αριστοκλής
137659

Εικόνα 7-11 Αρχική σελίδα εφαρμογής



Είσοδος Χρήστη

Όνομα Χρήστη

Κωδικός Πρόσβασης

Είσοδος

Δεν έχετε λογαριασμό; [Εγγραφή νέου χρήστη](#)

Όροι χρήστης Πολιτική απορρήτου

Εικόνα 7-12 Σελίδα εισόδου εφαρμογής

ΕΛΛΗΝΙΚΟ
ΑΝΟΙΚΤΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ

Αρχική Σελίδα Είσοδος

Εγγραφή Νέου Χρήστη

Πλήρες Όνομα Χρήστη

Username

Email

Κωδικός Πρόσβασης

Επιβεβαίωση Κωδικού

Εγγραφή

Έχετε ήδη λογαριασμό; [Είσοδος Χρήστη](#)

Όροι χρήστης Πολιτική απορρήτου

Εικόνα 7- 13 Σελίδα εγγραφής νέου χρήστη



Στην αρχική σελίδα της εφαρμογής παρέχονται πληροφορίες σχετικά με την πτυχιακή εργασία (εικόνα 7-12). Από το μενού πλοιόγησης ο χρήστης έχει την δυνατότητα να επιλέξει τη σελίδα του χάρτη από την επιλογή «Χάρτης». Στη σελίδα εμφανίζεται ο χάρτης της περιοχής στην οποία είναι τοποθετημένες οι κολώνες φωτισμού (εικόνα 7-13). Για την δημιουργία της συγκεκριμένης λειτουργίας χρησιμοποιήθηκε το maps API της Google. Τα στοιχεία που είναι τοποθετημένες οι κολώνες αντλούνται από τη βάση, όπου στον πίνακα Pole είναι διαθέσιμα τα στοιχεία των γεωγραφικών συντεταγμένων. Με την αιώρηση του κέρσορα πάνω από την «πινέζα» της κάθε κολώνας εμφανίζεται σε εικονίδιο η διεύθυνση της (εικόνα 7-14).



Σχολή Θετικών Επιστημών και Τεχνολογίας

Τμήμα Πληροφορικής

Πτυχιακή Εργασία ΠΕ419

Σχεδιασμός και ανάπτυξη «Έξυπνης Κολώνας»
δημόσιου φωτισμού σε περιβάλλον μίας έξυπνης πόλης

Επιβλέπων Καθηγητής
Δρ. Τοπάλης Ευάγγελος

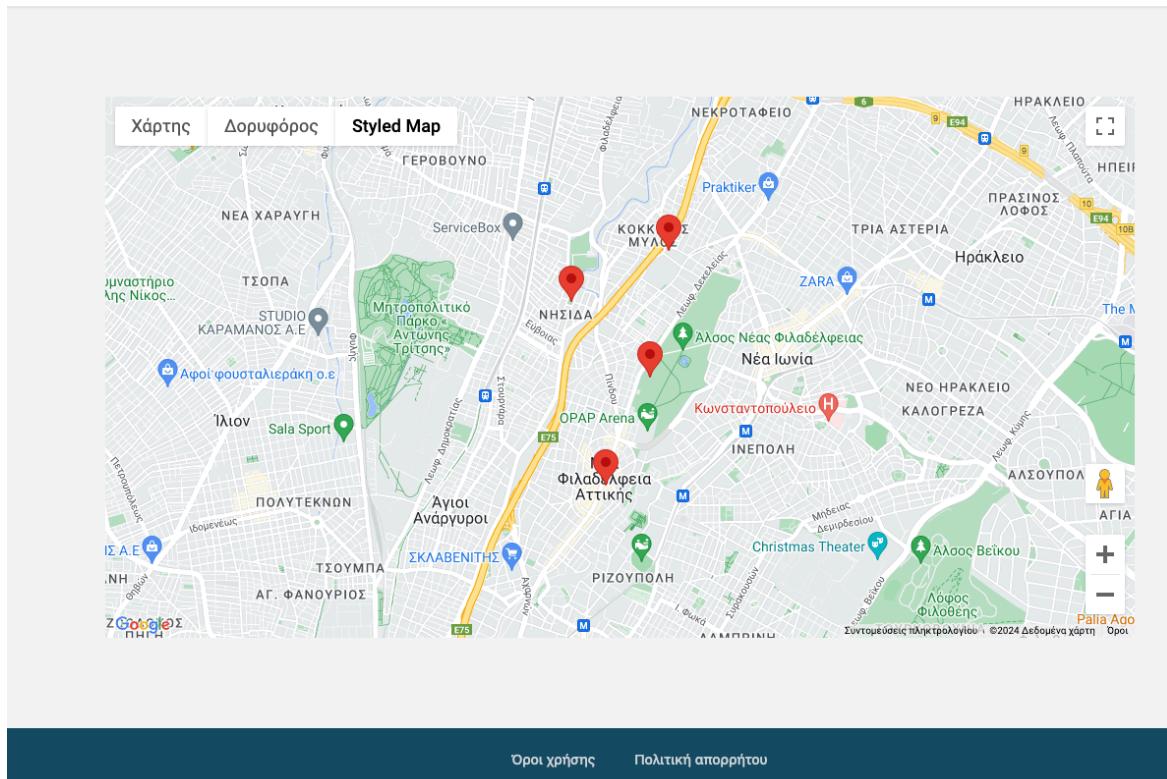
Φοιτητής
Κυριακίδης Αριστοκλής
137659



Εικόνα 7-14 Αρχική σελίδα εφαρμογής

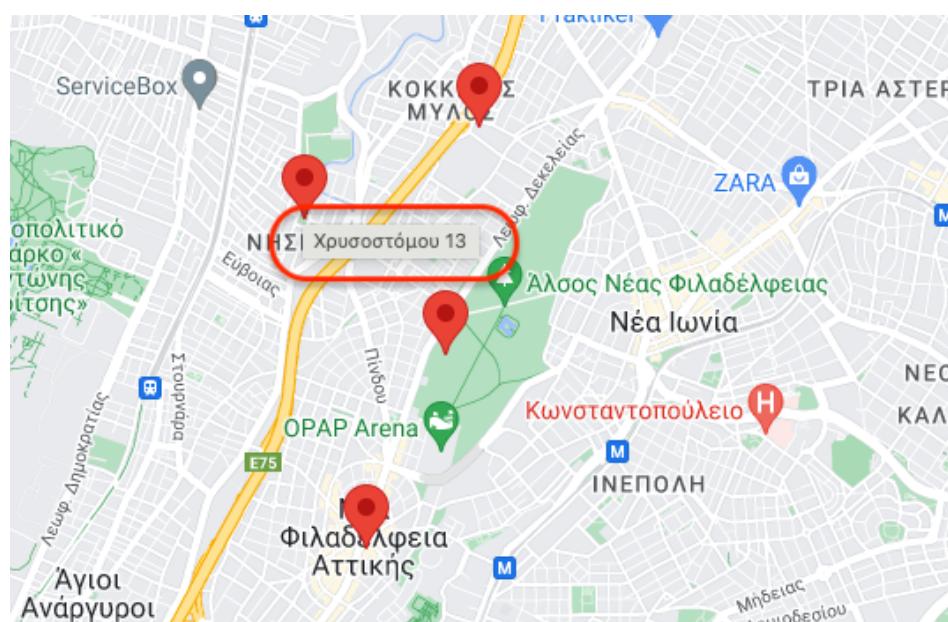


Χάρτης δικτύου «Έξυπνων Κολώνων» Φωτισμού



Όροι χρήσης Πολιτική απορρήτου

Εικόνα 7-15 Σελίδα χάρτη επιλογών



Εικόνα 7-16 Εμφάνιση Διεύθυνσης στο Χάρτη



Με την επιλογή της κάθε πινέζας ο χρήστης μεταφέρεται στη σελίδα με τις λεπτομέρειες της κάθε κολώνας. Στην σελίδα λεπτομερειών (εικόνα 7-15) της κάθε κολώνας ο χρήστης μπορεί να λάβει πληροφορίες για τα στοιχεία της κολώνας, όπως ο Δήμος και η διεύθυνση στην οποία είναι τοποθετημένη καθώς και τις γεωγραφικές συντεταγμένες της. Επίσης σε πραγματικό χρόνο λαμβάνει τιμές για τις μετρήσεις από τους αισθητήρες της κολώνας καθώς και για τα alarm, τα οποία σε περίπτωση ενεργοποίησης τους μετατρέπουν την πράσινη λυχνία σε κόκκινη (εικόνα 7-16). Τέλος εμφανίζεται ζωντανή μετάδοση βίντεο από την κάμερα που είναι συνδεμένη με την κάθε κολώνα.

ΕΛΛΗΝΙΚΟ
ΑΝΟΙΚΤΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ

Αρχική Σελίδα Χάρτης Στατιστικά Camera IoT About Αποσύνδεση

Κολώνα στην οδό Δεκελείας 100

Στοιχεία Έξυπνης Κολώνας

Στην συγκεκριμένη σελίδα εμφανίζονται λεπτομέρειες για την κολώνα καθώς και συνέχης Live ενημέρωση από τους αισθητήρες και τα alarm που έχει συνδεμένα.

ID Κολώνας	5
Όνομα Κολώνας	picoLoRa
Δήμος	Νέα Φιλαδέλφεια
Περιοχή	Πλατεία Πατριάρχου
Διεύθυνση	Δεκελείας 100
Συντεταγμένες Τοποθεσίας	38.0334396 - 23.7379341

Live Feed

Δεδομένα

Alarm Στάθμης Ομβρίων Υδάτων	
Αισθητήρας Μέτρησης Φωτεινότητας	0
Αισθητήρας Θερμοκρασίας	18.800
Αισθητήρας Ύψους Βροχόπτωσης	78

Θροι χρήστης Πολιτική απαρρήτου

Εικόνα 7-17 Σελίδα λεπτομέρειών κολώνας



Δεδομένα

Alarm Στάθμης Ομβρίων Υδάτων

0

Αισθητήρας Μέτρησης Φωτεινότητας

0

Αισθητήρας Θερμοκρασίας

18.800

Αισθητήρας Ύψους Βροχόπτωσης

78

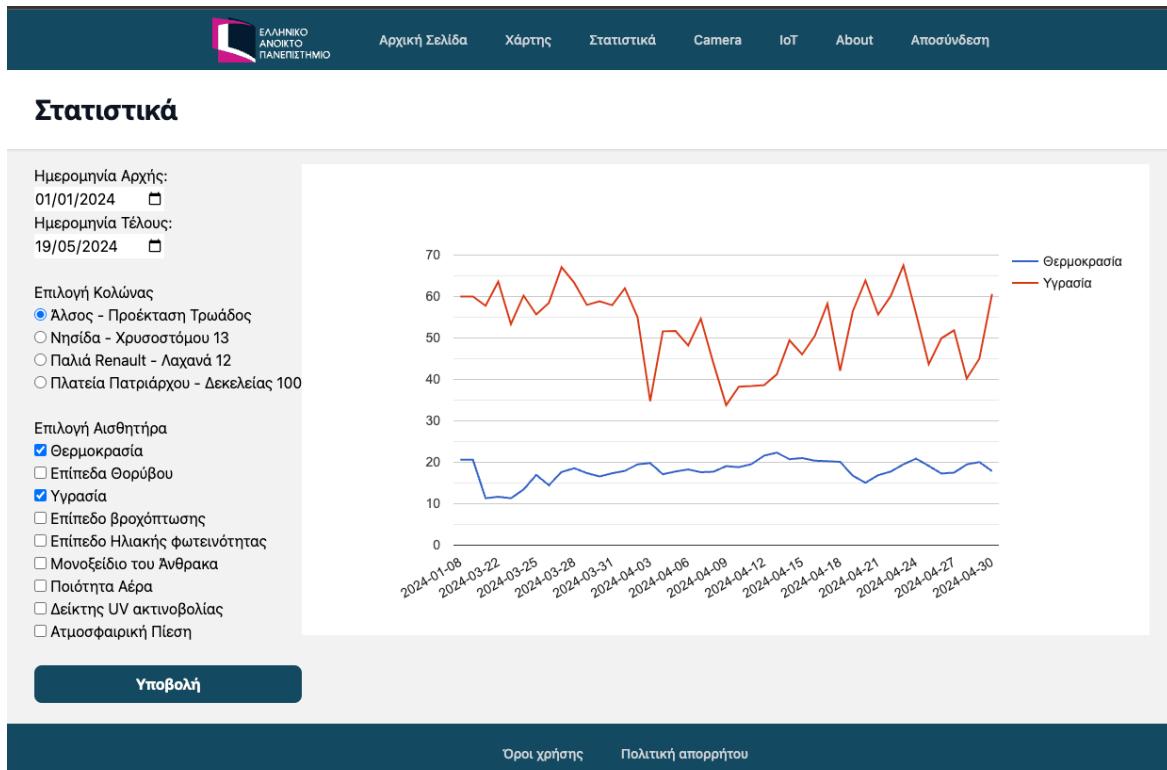
Εικόνα 7-18 Ενεργοποίηση Αισθητήρα

Από την επιλογή των στατιστικών ο χρήστης μπορεί να επιλέξει να πληροφορηθεί για στατιστικά στοιχεία των αισθητήρων κάθε κολώνας ή να πληροφορηθεί για συγκριτικά στατιστικά στοιχεία μεταξύ διαφορετικών έξυπνων κολώνων (εικόνα 7-17). Στη σελίδα Στατιστικά κολώνας (εικόνα 7-18) ο χρήστης επιλέγει την έξυπνη κολώνα για την οποία θέλει να δει πληροφορίες καθώς και τους αντίστοιχους αισθητήρες. Στη σελίδα Συγκριτικά Στατιστικά (εικόνα 7-19) ο χρήστης επιλέγει το μέγεθος για το οποίο θέλει να δει πληροφορίες καθώς και τις αντίστοιχες κολώνες.

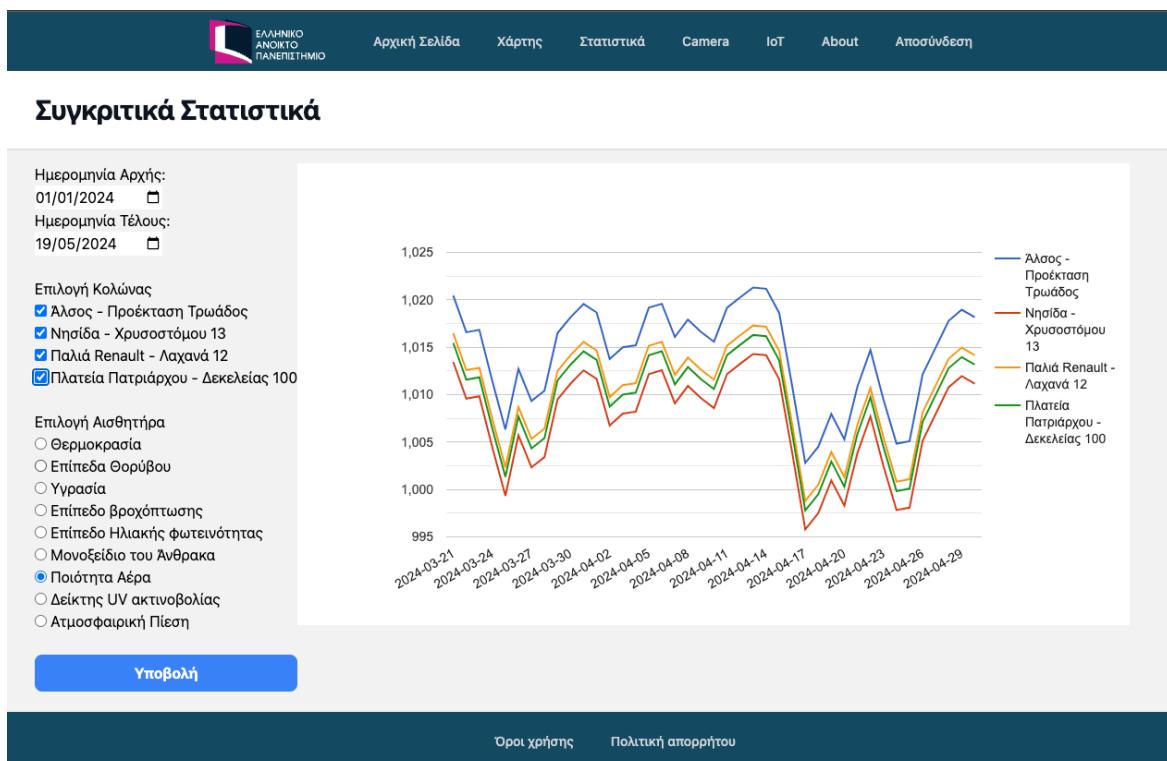
The screenshot shows a navigation bar at the top with the logo of the Hellenic Open University and links for "Αρχική Σελίδα", "Χάρτης", "Στατιστικά", "Camera", "IoT", "About", and "Αποσύνδεση". Below the navigation bar, there are two tabs: "Στατιστικά Κολώνας" (selected) and "Συγκρητικά Στατιστικά". The main content area displays a table with three columns: "Κολώνα", "Αισθητήρας Μέτρησης Φωτεινότητας", and "Αισθητήρας Θερμοκρασίας". The first row shows data for "Άνω Κολώνα": 0 for the first column, 18.800 for the second, and 78 for the third. A red circular icon with a white number '1' is located in the top right corner of the table area.

Κολώνα στην οδό Δεκελείας 100

Εικόνα 7-19 Επιλογές στατιστικών



Εικόνα 7-20 Σελίδα στατιστικών κολώνας



Εικόνα 7-21 Σελίδα Συγκριτικών στατιστικών



Στην σελίδα Camera (εικόνα 7-20) ο χρήστης έχει τη δυνατότητα να επιλέξει την προβολή ζωντανής μετάδοσης βίντεο από τις κολώνες που επιλέγει από το μενού επιλογών στην αριστερή πλευρά της σελίδας. Οι επιλογές εμφανίζονται σε μορφή πλέγματος στη δεξιά πλευρά (εικόνα 7-21). Από το κουμπί Camera options ο χρήστης μπορεί να επιλέξει να διαμορφώσει τις ρυθμίσεις της κάθε κάμερας από το αντίστοιχο μενού της το οποίο ανοίξει σε νέα σελίδα (εικόνα 7-22).

The screenshot shows a user interface for camera selection. At the top, there is a navigation bar with the Hellenic Open University logo and links for Αρχική Σελίδα, Χάρτης, Στατιστικά, Camera, IoT, About, and Αποσύνδεση. Below the navigation bar, the main content area has a title "Προβολή και Διαχείριση Live Video από Κάμερες". A sidebar on the left contains the heading "Επιλογή Έξυπνης Κολώνας" and a list of camera options with checkboxes:

- Άλσος - Προέκταση Τρωάδος
- Νησίδα - Χρυσοστόμου 13
- Παλιά Renault - Λαχανά 12
- Πλατεία Πατριάρχου - Δεκελείας 100

A blue "Υποβολή" button is located at the bottom of the sidebar.

At the bottom of the main content area, there is a footer bar with links for Όροι χρήσης and Πολιτική απορρήτου.

Εικόνα 7-22 Σελίδα Camera



Προβολή και Διαχείριση Live Video από Κάμερες

Επιλογή Έξυπνης Κολώνας

- Άλσος - Προέκταση Τρωάδος
- Νησίδα - Χρυσοστόμου 13
- Παλιά Renault - Λαχανά 12
- Πλατεία Πατριάρχου - Δεκελείας 100

Υποβολή

Προβολή Επιλογών

Άλσος - Προέκταση Τρωάδος



Camera Options

Νησίδα - Χρυσοστόμου 13



Camera Options

Παλιά Renault - Λαχανά 12



Camera Options

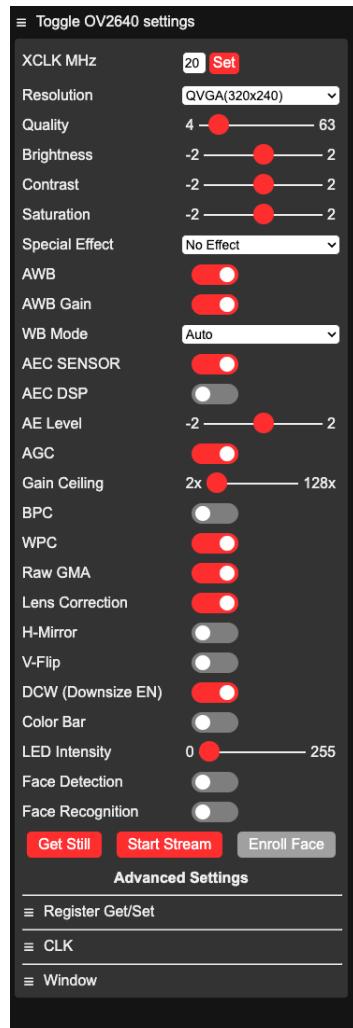
Πλατεία Πατριάρχου - Δεκελείας 100



Camera Options

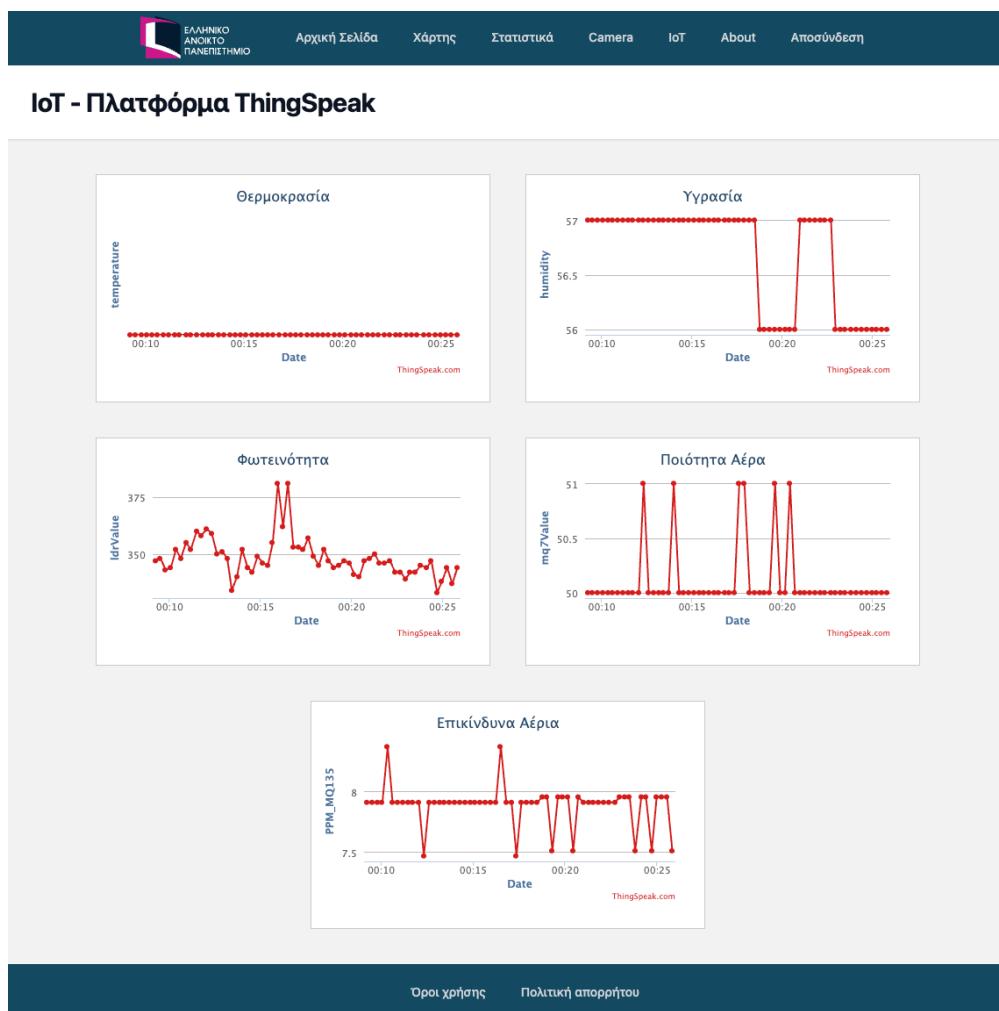
Όροι χρήσης Πολιτική απορρήτου

Εικόνα 7-23 Προβολή πλέγματος με ζωντανή μετάδοση



Εικόνα 7-24 Σελίδα ρυθμίσεων κάμερας

Στη σελίδα IoT (εικόνα 7-23) γίνεται εμφάνιση των διαγραμμάτων από την πλατφόρμα ThingSpeak. Η δωρεάν έκδοση περιορίζει το πλήθος των διαγραμμάτων που μπορούν να προβληθούν σε οχτώ συνολικά, επομένως δεν υπήρχε η δυνατότητα να προβληθεί κάθε κολώνα ξεχωριστά. Λόγω του συγκεκριμένου περιορισμού επιλέχθηκε μια από τις κολώνες του δικτύου.



Εικόνα 7-25 Σελίδα IoT - ThingSpeak

8. Συμπεράσματα Προτάσεις

Μέσω της έρευνας που έγινε στο πλαίσιο εκπόνησης της παρούσας πτυχιακής εργασίας διαπιστώθηκε ότι οι απαίτησεις των σύγχρονων πόλεων καθιστούν όλο και περισσότερο αναγκαίο τον εκσυγχρονισμό του δημόσιου φωτισμού σε «έξυπνο» με τη χρήση κολώνων φωτισμού που συνδυάζουν τα οφέλη από την ενσωμάτωση αισθητήρων και μικροελεγκτών.

Ο αρχικός στόχος που τέθηκε να δημιουργηθεί ένα δίκτυο «έξυπνου» φωτισμού με συνδυασμό διαφορετικών τεχνολογιών σε κάθε επίπεδο της υλοποίησης του. Από τη συνολική υλοποίηση, το αποτέλεσμα πλησίασε σε αρκετά μεγάλο βαθμό τις αρχικές επιδιώξεις. Συγκεκριμένα, χρησιμοποιήθηκαν τρία διαφορετικά πρωτόκολλα ασύρματης επικοινωνίας για να καλύψουν την ανάγκη δικτύωσης των συστημάτων, τα οποία ήταν το Zigbee, το LoRa και το WiFi. Για την κάλυψη της ανάγκης διαχείρισης της πληροφορίας καθώς και τη λήψη λογικών αποφάσεων χρησιμοποιήθηκαν τρεις τύποι μικροελεγκτών, οι οποίοι ενσωματώνουν διαφορετικές τεχνολογίες, το Arduino Uno, το ESP32 και το Raspberry Pico W. Για τη συγγραφή του κώδικα των μικροελεγκτών επιλέχτηκαν δύο γλώσσες προγραμματισμού, η C++ και η MicroPython. Για την υλοποίηση του REST Server επιλέχτηκε η γλώσσα προγραμματισμού JAVA και η υλοποίηση του έγινε με τη χρήση τεχνολογίας νημάτων (threads), ώστε να καλύψει την ανάγκη ενός δικτύου που μπορεί συνεχώς να επεκτείνεται. Η διαδικτυακή εφαρμογή στηρίχτηκε στις γλώσσες PHP και JavaScript, ενώ για την καλύτερη μορφοποίηση της ιστοσελίδας χρησιμοποιήθηκε η Tailwind CSS. Η υλοποίηση κάθε «έξυπνης κολώνας» αποτέλεσε και μια διαφορετική υλοποίηση με χρήση διαφορετικών αισθητήρων στην κάθε μία. Συγκεκριμένα, στις υλοποιήσεις των σεναρίων χρησιμοποιήθηκε ένα σύνολο από αισθητήρες που περιελάμβανε αισθητήρες θερμοκρασίες, αισθητήρες υγρασίας, αισθητήρα ατμοσφαιρικής πίεσης, αισθητήρα έντασης ήχου σε dB, αισθητήρα μέτρησης δείκτη UV, αισθητήρα ύπαρξης μονοξειδίου του Άνθρακα, αισθητήρα ύπαρξης επικίνδυνων αερίων, αισθητήρα μέτρησης ποιότητας του αέρα, αισθητήρα ανίχνευσης βροχής, αισθητήρα ανίχνευσης πυρκαγιάς καθώς και αισθητήρα στάθμης υδάτων. Η επιλογή του συνδυασμού όλων αυτών των διαφορετικών τεχνολογιών έγινε προκειμένου να μελετηθούν και να υλοποιηθούν λειτουργικά υποσυστήματα, τα οποία θα αποτελέσουν μέρος ενός ευρύτερου δικτύου, όπως είναι το περιβάλλον μιας σύγχρονης «έξυπνης» πόλης.

Τα δεδομένα από τη συνεχή καταγραφή των κόμβων του δικτύου για τα διαφορετικά μεγέθη που μελετήθηκαν και η συνεχής μετάδοση τους στον κεντρικό server, καθώς και η άμεση αποστολή ειδοποιήσεων, όταν ενεργοποιούνται τα alarms, αναδεικνύουν τα οφέλη που μπορεί να έχουν παρόμοιες υλοποιήσεις με αξιοποίηση των δεδομένων αυτών σε επιστημονικές μελέτες. Ο συνδυασμός χρήσης συστημάτων εποπτείας και ελέγχου μέσω του διαδικτύου μπορεί να είναι προσφέρει πολλαπλάσια οφέλη σε επίπεδο εξοικονόμησης ενέργειας. Επίσης, η μετάδοση των δεδομένων σε πραγματικό χρόνο μέσω της ιστοσελίδας, καθώς και των ιστορικών δεδομένων και των συγκριτικών αποτελεσμάτων από διάφορες χρονικές περιόδους, προσφέρουν μια πλήρη εικόνα των περιβαλλοντικών συνθηκών σε πραγματικό χρόνο, παρέχοντας στους πολίτες κίνητρα για περιβαλλοντική ευαισθητοποίηση τους.

Κατά τη διάρκεια της ανάπτυξης και υλοποίησης της παρούσας πτυχιακής εργασίας, προέκυψαν αρκετές προκλήσεις, αλλά και κάποια προβλήματα που επηρέασαν τη ροή της συνολικής διαδικασίας και μέχρι ένα βαθμό καθόρισαν το τελικό αποτέλεσμα. Αυτές οι προκλήσεις ήταν τόσο τεχνικές όσο και προκλήσεις στο λογισμικό, καθεμία από τις οποίες απαιτούσε διαφορετικές προσεγγίσεις και λύσεις με γνώμονα, όμως, την υλοποίηση του τελικού συστήματος στο πλαίσιο των αρχικών επιδιώξεων.

Μία από τις πιο σημαντικές προκλήσεις ήταν η τεχνική πολυπλοκότητα που εμπλέκεται στην ενσωμάτωση των διαφόρων στοιχείων, όπως το Arduino Uno, το Raspberry Pico W, το ESP32, το Xbee S1 (Zigbee) και τα LoRa modules σε ένα ολοκληρωμένο δίκτυο. Η εξασφάλιση της συνεχούς επικοινωνίας και του συγχρονισμού αυτών των συσκευών απαιτούσε αρκετά σενάρια ελέγχων τόσο για την αποσφαλμάτωση όσο και για την αντιμετώπιση των προβλημάτων. Η ενσωμάτωση των διαφορετικών πρωτοκόλλων επικοινωνιών που επιλέχθηκαν ανέβασαν την πολυπλοκότητα του συστήματος υλοποίησης. Από πλευράς διαχείρισης και συλλογής δεδομένων από τους αισθητήρες των διαφορετικών «έξυπνων κολώνων» ήταν ακόμα μια πρόκληση, καθώς η δομή της βάσης δεδομένων έπρεπε να φιλοξενεί διάφορους τύπους δεδομένων και να εξασφαλίζει ενημερώσεις σε πραγματικό χρόνο χωρίς καθυστερήσεις, παρέχοντας πληροφορίες για κάθε «έξυπνη κολώνα» του δικτύου. Κατά τη διάρκεια της ανάπτυξης του παραπάνω συστήματος, έπρεπε να υλοποιηθεί κώδικας για την αποστολή και λήψη των μηνυμάτων για το υποδίκτυο LoRa. Σε αντίθεση με την επιλογή των Xbee modules (Zigbee), για τα οποία το firmware τους

έχει ενσωματωμένο κώδικα που διαχειρίζεται τα μηνύματα σε επίπεδο byte frame, τα LoRa SX1278 που επιλέχτηκαν δεν είχαν ενσωματωμένο αντίστοιχο firmware, με αποτέλεσμα να γίνει υλοποίηση κώδικα για την συγκεκριμένη λειτουργία αυξάνοντας την πολυπλοκότητα του έργου. Η εξασφάλιση της δυναμικής λειτουργίας της διαδικτυακής εφαρμογής, ώστε να μπορεί να εμφανίζει δεδομένα σε πραγματικό χρόνο αποτελεσματικά για κάθε διαφορετική «έξυπνη κολώνα» με βάση τους συνδεδεμένους αισθητήρες, ήταν ένας παράγοντας που χρειάστηκε να γίνουν τροποποιήσεις στο σχεδιασμό της βάσης δεδομένων σε σχέση με τις αρχικές παραδοχές.

Βάσει των εμπειριών που αποκομίστηκαν και των προκλήσεων που αντιμετωπίστηκαν κατά τη διάρκεια της πτυχιακής εργασίας, μπορούν να γίνουν ορισμένες προτάσεις που αφορούν την επέκταση του συστήματος που υλοποιήθηκε, όπως:

1. Προηγμένες Τεχνικές Διαχείρισης Δεδομένων

Η χρήση πιο προηγμένων τεχνικών διαχείρισης και επεξεργασίας δεδομένων, όπως η ανάλυση δεδομένων σε πραγματικό χρόνο και οι αλγόριθμοι μηχανικής μάθησης, θα μπορούσε να ενισχύσει την ικανότητα του συστήματος να διαχειρίζεται μεγάλα σύνολα δεδομένων και να εξάγει σημαντικές πληροφορίες από τα δεδομένα των αισθητήρων. Η υλοποίηση λύσεων που βασίζονται στο cloud θα μπορούσε επίσης να προσφέρει μεγαλύτερη ευελιξία.

2. Βελτιωμένα Πρωτόκολλα Επικοινωνίας

Η διερεύνηση και η υιοθέτηση πιο προηγμένων πρωτοκόλλων επικοινωνίας που προσφέρουν υψηλότερη αξιοπιστία και εμβέλεια θα μπορούσε να περιορίσει τα ζητήματα συνδεσιμότητας. Τεχνολογίες όπως το 5G θα μπορούσαν να παρέχουν καλύτερες εναλλακτικές λύσεις σε σχέση με το Xbee (Zigbee) και το LoRa για συγκεκριμένες περιπτώσεις χρήσης.

3. Σχεδιασμός με Επίκεντρο τον Χρήστη

Η δημιουργία διαδραστικών εφαρμογών με ενσωμάτωση mobile τεχνολογιών με σκοπό την παροχή της πληροφορίας στους πολίτες της πόλης, θα μπορούσε να επιτύχει μεγαλύτερη αποτελεσματικότητα και ικανοποίηση των χρηστών.

Συνοψίζοντας, η έρευνα και η υλοποίηση της πτυχιακής εργασίας ανέδειξαν την αυξανόμενη ανάγκη για εκσυγχρονισμό του δημόσιου φωτισμού μέσω της χρήσης «έξυπνων» κολώνων με ενσωματωμένους αισθητήρες και δυνατότητες δικτύωσης. Η αλληλεπίδραση των διαφόρων τεχνολογιών, όπως το Arduino Uno, το Raspberry Pico W, το ESP32, το Xbee S1 (Zigbee) και τα LoRa modules, ανέβασε την πολυπλοκότητα του έργου, αποτέλεσε όμως ένα ιδιαίτερα πρόσφορο αντικείμενο μελέτης με ιδιαίτερο ενδιαφέρον. Η διαχείριση των δεδομένων από τους αισθητήρες και η υλοποίηση διαδικτυακής εφαρμογής με σκοπό την προβολή των δεδομένων αυτών, αποτέλεσαν επίσης, άλλο ένα σημαντικό αντικείμενο μελέτης. Οι διάφορες επεκτάσεις και βελτιώσεις για το σύστημα, όπως η χρήση προηγμένων τεχνικών διαχείρισης δεδομένων, η διερεύνηση βελτιωμένων πρωτοκόλλων επικοινωνίας και η δημιουργία διαδραστικών εφαρμογών με επίκεντρο τον χρήστη, θα βελτιώσουν και θα εξελίξουν το αποτέλεσμα της παρούσας πτυχιακής εργασίας. Οι προτάσεις αυτές στοχεύουν στη βελτίωση της αποδοτικότητας, της αξιοπιστίας και της ικανοποίησης των χρηστών στις μελλοντικές υλοποιήσεις «έξυπνων κολώνων» βασισμένων στο IoT.

Βιβλιογραφία

Ελληνόγλωσση

Αδαμόπουλος, Δ. (2008). Τηλεματική (Β' εκδ., Τόμ. Α'). Πάτρα: ΕΑΠ.

Δημητρίου, Ν. Δ. (2008). Ψηφιακές επικοινωνίες II Σήματα – Διαμόρφωση - Θόρυβος (Τόμ. Β' - Μέρ. Β'). Πάτρα: ΕΑΠ.

Ευρωπαϊκή Επιτροπή, επίσημος ιστότοπος. (2023). Ευρωπαϊκή Επιτροπή. Ανακτήθηκε από: https://commission.europa.eu/index_el

Ζορκάδης, Β. (2002). Θεωρία Πληροφορίας και Κωδικοποίησης (Τόμ. Α'). Πάτρα: ΕΑΠ.

Θραμπουλίδης, Κ. (2000). Γλώσσες Προγραμματισμού (Τόμ. Δ'). Πάτρα: ΕΑΠ.

Θραμπουλίδης, Κ. (2001). Γλώσσες Προγραμματισμού II (Αντικειμενοστρεφής Προγραμματισμός) (Τόμ. Γ'). Πάτρα: ΕΑΠ.

Καμέας, Α. Δ. (2008). Τεχνικές Προγραμματισμού (Β' εκδ., Τόμ. Β'). Πάτρα: ΕΑΠ.

Μεταξίδης, Ν. (2019). Αστικοποίηση (Παρουσίαση στο μάθημα Οικονομική Γεωγραφία). Ανακτήθηκε από: [https://oeclasse.hua.gr/eclasse/modules/document/\[Μάθημα9α.Ηπόλη.pdf\]](https://oeclasse.hua.gr/eclasse/modules/document/[Μάθημα9α.Ηπόλη.pdf])

Νικόλος, Δ. (2008). Αρχιτεκτονική Υπολογιστών I (Τόμ. Β'). Πάτρα: ΕΑΠ.

Ξένος, Μ. (2000). Βάσεις Δεδομένων (Τόμ. Γ'). Πάτρα: ΕΑΠ.

Φιτσίλης, Π., Ξένος, Μ., & Σταμέλος, Ι. (2008). Προγραμματισμός Έργων Πληροφορικής - Αντικειμενοστρεφείς Μεθοδολογίες (Τόμ. Δ'). Πάτρα: ΕΑΠ.

Φούσκας, Γ. (2002). Δίκτυα Υπολογιστών I (Τόμ. Γ'). Πάτρα: ΕΑΠ.

Φούσκας, Γ. (2003). Εισαγωγή στην Τεχνολογία του Παγκόσμιου Ιστού (Εγχειρίδιο Μελέτης εκδ., Τόμ. Β'). Πάτρα: ΕΑΠ.

Φούσκας, Γ. (2008). Ψηφιακές επικοινωνίες I (Β' εκδ., Τόμ. Β' - Μέρ. Α'). Πάτρα: ΕΑΠ.

Χατζηλυγερούδης, Ι. (2008). Δομές Δεδομένων (Β' εκδ., Τόμ. Γ'). Πάτρα: ΕΑΠ.

Deitel, P., & Deitel, H. (2015). Java: Προγραμματισμός (Δέκατη Έκδοση). Γκιούρδας.

Kurose, J.F., & Ross, K.W. (2018). Δικτύωση Υπολογιστών Προσέγγιση από Πάνω προς τα Κάτω (7^η εκδ.). Αθήνα: Μ.Γκιούρδας.

Ξενόγλωσση

Adafruit. (n.d.). *Adafruit_SI1145_Library/Adafruit_SI1145.h* at *master* · *adafruit/Adafruit_SI1145_Library*. Ανακτήθηκε στις 02 Μαρτίου 2024, από GitHub. https://github.com/adafruit/Adafruit_SI1145_Library/blob/master/Adafruit_SI1145.h

Aiswarya Dev Goswami, Basudeb Das, and Saswati Mazumdar (2022). Development of a GSM based Arduino controlled smart street lighting system.

Apache Software Foundation, (n.d.). Apache HTTP Server. Ανακτήθηκε από τη διεύθυνση: <https://httpd.apache.org/>

Arduino Docs | Arduino Built in Examples, (n.d.) Ανακτήθηκε στις 02 Μαρτίου 2024, από <https://docs.arduino.cc/built-in-examples/>

Arduino Docs | Arduino Documentation. (n.d.), Ανακτήθηκε στις 02 Μαρτίου 2024, από <https://docs.arduino.cc/hardware/>
Arduino Project Hub. (n.d.). projecthub.arduino.cc. Ανακτήθηκε στις 02 Μαρτίου 2024, από <https://projecthub.arduino.cc/>

Arduino. (n.d.). Arduino - Πλατφόρμα κατασκευής ηλεκτρονικών πρωτοτύπων ανοικτού κώδικα. Ανακτήθηκε στις 26 Ιανουαρίου 2024, από <https://www.arduino.cc>.

Babu D Vijendra, Nisha A SahayaAnselin, Dhasan D Bharathi, Venkatesan M, Karthikeyan C, (2021), Intelligent High Tech Street Lightning Pole for Smart City, Article in Annals of the Romanian Society for Cell Biology · April 2021

Buyya, R., & Dastjerdi, A. V. (2016). Internet of things: Principles and paradigms. Morgan Kaufmann Publisher, 2016.

Connolly, T. M., & Begg, C. E. (2014). Database Systems: A Practical Approach to Design, Implementation, and Management. Pearson.

Crockford, D. Introducing Json (2024), Ανακτήθηκε από τη διεύθυνση: <https://www.json.org/json-en.html>

Daely Philip Tobianto, Reda Haftu Tasew, Satrya Gandeva Bayu, Kim Jin Woo, Shin Soo Young (2017). Design of Smart LED Streetlight System for Smart City with Web-Based Management System.

Damirchi, M. (2021, June 19). *Interfacing MQ-7 Smoke Gas Sensor Module with Arduino*.

Electropeak. Ανακτήθηκε στις 02 Μαρτίου 2024, από <https://electropeak.com/learn/interfacing-mq-7-smoke-gas-sensor-module-with-arduino/>

De Bakker, B. (2024, January 30). *How to use HC-SR501 PIR Motion Sensor with Arduino*.

Makerguides.com. Ανακτήθηκε στις 02 Μαρτίου 2024, από <https://www.makerguides.com/hc-sr501-arduino-tutorial/>

European Commission: Energy and smart cities, (2023). Ανακτήθηκε από: https://energy.ec.europa.eu/topics/research-and-technology/energy-and-smart-cities_en

Extensible Markup Language (XML) 1.0 (Fifth Edition). (n.d.) W3, Ανακτήθηκε από τη διεύθυνση: <https://www.w3.org/TR/REC-xml/>

Fritzing, (n.d.). Fritzing - Making Electronics Accessible. Ανακτήθηκε από τη διεύθυνση: <https://fritzing.org/>

Gehlot A., Alshamrani S., Singh R., Rashid M., Akram S., AlGhamdi A. and Albogamy F. (2021). Internet of Things and Long-Range-Based Smart Lampposts for Illuminating Smart Cities

Google Charts. (n.d.). Google Developers. Ανακτήθηκε από τη διεύθυνση: <https://developers.google.com/chart>

Gouthami C, Santosh C, Pavan Kumar, Karthik A, Ramya.K.R (2016).Design and

Implementation of Automatic Street Light Control System using Light Dependent Resistor

Grobotronics. (n.d.) Ανακτήθηκε στις 02 Μαρτίου 2024, από <https://grobotronics.com/>

Hackster.io. (2023, November 27). Interfacing SX1278 (Ra-02) LORA Module with Arduino. <https://www.hackster.io/shathiralakdilu/interfacing-sx1278-ra-02-lora-module-with-arduino-db8e1f>

Hostinger, 2024. Hostinger tutorials: What Is a Web Server? Ανακτήθηκε από τη διεύθυνση: <https://www.hostinger.com/tutorials/what-is-a-web-server>

How to use the Flame sensor with Arduino. (n.d.). Ardumotive Arduino Greek Playground. <https://www.ardumotive.com/how-to-use-the-flame-sensoren.html>



Jackson, J. C. (2007). Web technologies: A computer science perspective. Upper Saddle River, NJ: Pearson/Prentice Hall.

Kiran V., Nityanand Sai R., Ozair Khan M., Tiwari J. (2020). Smart Street Light Based On Arduino.

Lab, M. (2024, March). ESP32-CAM AI-Thinker Board – All about GPIO Pins. Microcontrollers Lab. <https://microcontrollerslab.com/esp32-cam-ai-thinker-pinout-gpio-pins-features-how-to-program/>

LAMP Stack., (n.d.), LAMP (software bundle). Ανακτήθηκε από τη διεύθυνση: [https://en.wikipedia.org/wiki/LAMP_\(software_bundle\)](https://en.wikipedia.org/wiki/LAMP_(software_bundle))

LME Editorial Staff. (2022, October 16). How Water Level Sensor Works and Interface it with Arduino. Last Minute Engineers. Ανακτήθηκε στις 02 Μαρτίου 2024, από <https://lastminuteengineers.com/water-level-sensor-arduino-tutorial/>

LoRa Alliance. (n.d.). What is LoRaWAN? Ανακτήθηκε από τη διεύθυνση <https://lora-alliance.org/lorawan-for-developers/what-is-lorawan>.

Madureira, O. (2023). Smart City Observatory 2023 - IMD business school for management and leadership courses. IMD Business School for Management and Leadership Courses. Ανακτήθηκε από: <https://www.imd.org/smart-city-observatory/home/>

MDN, (2024). HTTP Request Methods - MDN Web Docs, 2 Jan. 2024, Ανακτήθηκε από τη διεύθυνση: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>

MDN, (n.d.). CSS: Cascading style sheets: MDN. MDN Web Docs. Ανακτήθηκε από τη διεύθυνση: <https://developer.mozilla.org/en-US/docs/Web/CSS>

MDN, (n.d.). JS: JavaScript: MDN, MDN Web Docs. Ανακτήθηκε από τη διεύθυνση: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

MDN, (n.d.). What is a Web Server? : MDN, MDN Web Docs. Ανακτήθηκε από τη διεύθυνση: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server

Medium, (2018). Medium: How the ‘humble lamppost’ means smarter streets, through London’s big collaboration with fellow European cities. Ανακτήθηκε από:



<https://chiefdigitalofficer4london.medium.com/the-humble-lamppost-means-smarter-streets-through-london-s-big-collaboration-with-fellow-european-f3f25975c93e>

Medium, 2024. Build and Deploy LAMP Server on AWS, Ανακτήθηκε από τη διεύθυνση:
<https://medium.com/@jacqmkindi/build-and-deploy-lamp-server-on-aws-3b72d4aafb2f>

Merlino Giovanni, Bruneo Dario, Distefano Salvatore, Longo Francesco,

Microcontrollers lab. (n.d.). Microcontrollers Lab. Ανακτήθηκε στις 02 Μαρτίου 2024, από
<https://microcontrollerslab.com/>

MySQL. (n.d.). MySQL :: MySQL Community Downloads. Ανακτήθηκε από τη διεύθυνση: <https://dev.mysql.com/downloads/>

NodeMCU-32S Core Development Board | Ai-Thinker. (n.d.). https://docs.ai-thinker.com/en/esp32/boards/nodemcu_32s

OpenStack-Powered Infrastructure. (Article on Sensors 2015)

Puliafito Antonio, Al-Anbuky Adnan (2015). A Smart City Lighting Case Study on an

Raspberry Pi Foundation. (n.d.). Raspberry Pi. Ανακτήθηκε στις 02 Μαρτίου 2024, από
<https://www.raspberrypi.org>.

Raspberry Pi Foundation. (n.d.). Raspbian. Ανακτήθηκε από τη διεύθυνση:
<https://www.raspberrypi.org/software/>

ResearchGate, 2024.Researchgate: Wireless technologies for IoT, Ανακτήθηκε από τη διεύθυνση: https://www.researchgate.net/figure/Wireless-technologies-for-IoT_fig3_322876769

Sandsoftwaresound (n.d.). - Ανακτήθηκε στις 02 Μαρτίου 2024, από
<https://sandsoftwaresound.net/wp-content/uploads/2020/11/Cortex-A72BlockDiagram.jpg>

Santos, S., & Santos, S. (2019, April 2). Guide for Microphone Sound Sensor Arduino | Random Nerd Tutorials. Random Nerd Tutorials. Ανακτήθηκε στις 02 Μαρτίου 2024, από
<https://randomnerdtutorials.com/guide-for-microphone-sound-sensor-with-arduino/>

Sei Ping Lau, Geoff V. Merrett, Alex S. Weddell, Neil M. White (2015). A Traffic-Aware Street Lighting Scheme for Smart Cities using Autonomous Networked Sensors

Shweta D, Shruthi M, Shreya A, Shwetha B, Sujay M, Chaitanya K (2019). Arduino Based Smart Street Light System.

SparkFun Electronics. (n.d.). Ανακτήθηκε από τη διεύθυνση:
<https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Manual.pdf>

Sravani Annareddy, Malarvezhi P., Dayana R. (2018). Design and implementation of dimmer based smart street lighting system using raspberry Pi and IoT.

ThingSpeak. (n.d.). ThingSpeak for IoT Projects. Ανακτήθηκε από τη διεύθυνση:
<https://thingspeak.com/>

UV Sensor (B) - Waveshare Wiki. (n.d.-b). Ανακτήθηκε στις 02 Μαρτίου 2024, από
[https://www.waveshare.com/wiki/UV_Sensor_\(B\)](https://www.waveshare.com/wiki/UV_Sensor_(B))

Verma Anshul, Verma Pradeepika, Farhaoui Yousef, and Zhihan Lv (2022). Emerging Real-World Applications of Internet of Things (9 IoT-enabled smart street light control for demand response applications, by Hettiarachchi Aroshana, Naz Islam Shama

Water level sensor for Arduino - Devobox. (n.d.). Devobox. Ανακτήθηκε στις 02 Μαρτίου 2024, από <https://www.devobox.com/el/enviromental/415-water-level-sensor-for-arduino.html>

Welling, L. & Thomson, L., (2017). PHP and MySQL Web Development. Hoboken, NJ, Addison-Wesley, 2017.

Wikipedia Foundation. (2023, December 30). Wi-Fi. Wikipedia. Ανακτήθηκε από τη διεύθυνση: <https://en.wikipedia.org/wiki/Wi-Fi>

XBee (ZigBee) Interfacing with Arduino UNO | Arduino. (n.d.). © 2018 ElectronicWings. Ανακτήθηκε στις 02 Μαρτίου 2024, από <https://www.electronicwings.com/arduino/xbee-s2-zigbee-interfacing-with-arduino-uno>

XCTU, (n.d.). Digi XCTU. Ανακτήθηκε από τη διεύθυνση:
<https://www.digi.com/products/embedded-systems/digi-xbee/digi-xbee-tools/xctu>

Red Hat, n.d - What is a REST API? | Red Hat, n.d., Ανακτήθηκε από τη διεύθυνση:
<https://www.redhat.com/en/topics/api/what-is-a-rest-api>

Medium, 2024 - Understanding REST API: A Comprehensive Guide | Medium, 2024,
Ανακτήθηκε από τη διεύθυνση: <https://medium.com/@MakeComputerScienceGreatAgain/understanding-rest-api>

Yang, Lee, Chen, Yang, Huang , Hou (2016).An Implementation of High Efficient Smart Street Light Management System for Smart City.

Zakas, Nickolas, (2012). Professional javascript for web developers. Wrox Press.

Παράρτημα Α: «Arduino UNO – Xbee module»

```

/*****
Author: Kyriakidis Aris
email: a.kyriakidis@hotmail.com
*****/

#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
#include <DHT.h>
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
#include <TimeLib.h>
#include <MQ135.h>
//#include "Adafruit_SI1145.h"
#include "Si1145.h"
#include <virtuabotixRTC.h>

// ***** Pin connections *****
// LDR and LED

const int lightSensorPin = A0;           // Analog pin for light sensor
const int ledPin = 7;                     // Digital pin for LED

// MQ-7 Carbon Monoxide Sensor
const int AOUT_MQ7pin = A1;              //the AOUT pin of the CO sensor goes into
analog pin A1
const int DOUT_MQ7pin = 6;                //the DOUT pin of the CO sensor goes into
digital pin D8

// MQ-135 Air quality Sensor

#define PIN_MQ135 A2

// Variables
float temperature = 0.0;                 // Variable to store the temperature value
float humidity = 0.0;                    // Variable to store the humidity value
unsigned long switchOnTime = 0;          // Variable to store LED switch on time
unsigned long switchOffTime = 0;          // Variable to store LED switch off time
const String poleId = "arduinoXbee";    // the pole identification parameter
String currentTime = "";

int limitMQ7Sensor;
int valueMQ7Sensor;
int alarmMQ7Sensor = 0;

```

```

// Light threshold values
const int lightThreshold = 500;           // Adjust this value to set the
sensitivity

// DHT11 sensor
#define DHTPIN 3                      // Digital pin for DHT11 sensor
#define DHTTYPE DHT11                  // DHT sensor type (DHT11)

// Create DHT object
DHT dht(DHTPIN, DHTTYPE);

// Create json object
StaticJsonDocument<128> jsonDocument; // JSON document for storing Alarm data
//DynamicJsonDocument jsonDocument(128); // JSON document for storing Alarm data
//StaticJsonDocument<128> jsonDocThing; // JSON document for storing Alarm data

// Create Software serial object to establish xbee communication
SoftwareSerial xbeeSerial(4, 5); // RX, TX pins for XBee module

// create the UV object
Si1145 uv = Si1145();
//Adafruit_SI1145 uv = Adafruit_SI1145();

// Create MQ135 sensor object
MQ135 mq135_sensor(PIN_MQ135);

// Real clock Module Pin assignement
virtuabotixRTC myRTC(8, 9, 10); //If you change the wiring change the pins here
also

// Timing variables for periodic actions
unsigned long previousMillis = 0; // Stores the last time the action was
performed
const unsigned long interval = 60000; // Interval for action (1 minute in
milliseconds)

String poleName = "arduinoXbee";
String alarmName = "gasAlarm";

void startUvSensor(){
    Serial.println("Adafruit SI1145 test");
    if (! uv.begin()) {
        Serial.println("Didn't find Si1145 !\r\n");
}

```

```

while (1);

}

Serial.println("Si1145 Init success !\r\n");
}

float checkUvSensor(){
    Serial.println("=====");
    Serial.print("Vis: "); Serial.println(uv.readVisible());
    Serial.print("IR: "); Serial.println(uv.readIR());

    // Uncomment if you have an IR LED attached to LED pin!
    //Serial.print("Prox: "); Serial.println(uv.readProx());

    float UVindex = uv.readUV();
    // the index is multiplied by 100 so to get the
    // integer index, divide by 100!
    UVindex /= 100.0;

    Serial.print("UV: "); Serial.println(UVindex);
    return UVindex;
}

void setInitialDayTime(){
    // ONE TIME ONLY EXECUTION
    // Set the current date, and time in the following format:
    // seconds, minutes, hours, day of the week, day of the month, month, year
    myRTC.setDS1302Time(00, 18, 14, 1, 14, 7, 2024); //Here you write your actual
    time/date as shown above
    //but remember to "comment/remove" this function once you're done
    //The setup is done only one time and the module will continue counting it
    automatically
}

void checkClockModule(){
    // This allows for the update of variables for time or accessing the
    individual elements.
    myRTC.updateTime();

    /* Start printing elements as individuals
    Serial.print("Current Date / Time: ");
    Serial.print(myRTC.dayofmonth); //You can switch between day and month if
you're using American system
    Serial.print("/");
    Serial.print(myRTC.month);
    Serial.print("/");
    Serial.print(myRTC.year);
}

```

```

Serial.print(" ");
Serial.print(myRTC.hours);
Serial.print(":");
Serial.print(myRTC.minutes);
Serial.print(":");
Serial.println(myRTC.seconds);
*/
currentTime = String(myRTC.dayofmonth) + "/" + String(myRTC.month) + "/" +
String(myRTC.year) + " " + String(myRTC.hours) + ":" + String(myRTC.minutes) +
":" + String(myRTC.seconds);
}

void printMQ135Values(float rzero, float correctedRZero, float resistance, float
ppm, float correctedPPM, float temperature, float humidity){

Serial.print("MQ135 RZero: ");
Serial.print(rzero);
Serial.print("\t Corrected RZero: ");
Serial.print(correctedRZero);
Serial.print("\t Resistance: ");
Serial.print(resistance);
Serial.print("\t PPM: ");
Serial.print(ppm);
Serial.print("\t Corrected PPM: ");
Serial.print(correctedPPM);
Serial.println("ppm");
}

void checkLightStatus(){
// Read the light sensor value
int lightValue = analogRead(lightSensorPin);

// Check if it's dark based on the light sensor reading
if (lightValue < lightThreshold) {
    // It's dark, turn on the LED
    digitalWrite(ledPin, HIGH);
    switchOnTime = millis(); // Record the LED switch on time
} else {
    // It's light, turn off the LED
    digitalWrite(ledPin, LOW);
    switchOffTime = millis(); // Record the LED switch off time
}
}

void checkDHT11Status(){
// Read temperature and humidity from DHT11 sensor
temperature = dht.readTemperature();
}

```



```
humidity = dht.readHumidity();  
}  
  
void checkMQ7Status(){  
    valueMQ7Sensor = analogRead(AOUT_MQ7pin); //reads the analaoг value from the  
CO sensor's AOUT pin  
    limitMQ7Sensor = digitalRead(DOUT_MQ7pin); //reads the digital value from the  
CO sensor's DOUT pin  
    if (valueMQ7Sensor > 200){  
        alarmMQ7Sensor = 1; //if limit of 200ppm has been  
reached, the alarm value will become 1  
    } else {  
        alarmMQ7Sensor = 0; //if limit has not been reached,  
the alarm value will remain 0;  
    }  
}  
  
String createThingSpeakJson(int ldr, float temperature, float humidity, float  
valueMQ7Sensor, int alarmMQ7Sensor, float correctedPPM) {  
  
    jsonDocument.clear();  
    JsonObject sensorDataObj = jsonDocument.createNestedObject("ThingSpeak");  
    sensorDataObj["ldr"] = ldr;  
    sensorDataObj["temperature"] = temperature;  
    sensorDataObj["humidity"] = humidity;  
    sensorDataObj["mq7"] = valueMQ7Sensor;  
    sensorDataObj["gasAlarm"] = alarmMQ7Sensor;  
    sensorDataObj["mq135"] = correctedPPM;  
    // Serialize the JSON object to a string  
    String jsonString1 = "";  
    serializeJson(jsonDocument, jsonString1);  
  
    return jsonString1;  
}  
  
String createSensorDataJSON(String poleName, String timestamp, float  
temperature, float humidity, float valueMQ7Sensor, float correctedPPM, float  
UVindex, int ldr) {  
    //StaticJsonDocument<128> jsonDoc;  
    jsonDocument.clear();  
    // Create nested objects and set values  
    JsonObject sensorDataObj = jsonDocument.createNestedObject("SensorData");  
    sensorDataObj["poleName"] = poleName;  
    sensorDataObj["timestamp"] = timestamp;  
    JsonObject valuesObj = sensorDataObj.createNestedObject("values");  
    valuesObj["temperature"] = temperature;  
    valuesObj["humidity"] = humidity;  
    valuesObj["mq7"] = valueMQ7Sensor;
```

```

valuesObj["mq135"] = correctedPPM;
valuesObj["ldr"] = ldr;
valuesObj["uv"] = UVindex;
delay(500);
// Serialize JSON to string
String jsonString = "";
serializeJson(jsonDocument, jsonString);
return jsonString;
}

String createAlarmJSON(String poleName, String timestamp, String alarmName, int
alarmMQ7Sensor) {
    jsonDocument.clear();
    // Create nested objects and set values
    JsonObject sensorDataObj = jsonDocument.createNestedObject("AlarmData");
    sensorDataObj["poleName"] = poleName;
    sensorDataObj["timestamp"] = timestamp;
    sensorDataObj["sensorName"] = alarmName;
    sensorDataObj["value"] = alarmMQ7Sensor;
    // Serialize JSON to string
    String jsonString;
    serializeJson(jsonDocument, jsonString);
    return jsonString;
}

// Function to create and send an alarm JSON to node 02 whenever the gas alarm
is activated
void checkAndSendGasAlarm(String poleName, String timestamp, String alarmName) {
    if (alarmMQ7Sensor == 1) {
        // Create JSON object
        String alarmJson = createAlarmJSON(poleName, timestamp,
alarmName,alarmMQ7Sensor);
        Serial.println("Gas Alarm JSON Data: ");
        Serial.println(alarmJson);
        // Send the JSON message
        xbeeSerial.println(alarmJson);
        delay(1000);
    } else {
        // Create JSON object
        String alarmJson = createAlarmJSON(poleName, timestamp, alarmName,
alarmMQ7Sensor);
        Serial.println("Gas Alarm JSON Data: ");
        Serial.println(alarmJson);
        // Send the JSON message
        xbeeSerial.println(alarmJson);
        delay(1000);
    }
}

```



```
// Function to perform an action every 10 minutes
void performActionEvery10Minutes(String poleName, String currentTime) {
    unsigned long currentMillis = millis();
    if (currentMillis - previousMillis >= interval) {
        previousMillis = currentMillis;
        // Action to perform every 10 minutes
        int ldr = analogRead(lightSensorPin);
        float UVindex = checkUvSensor();
        // Get the MQ135 sensor values
        float rzero = mq135_sensor.getRZero();
        float correctedRZero = mq135_sensor.getCorrectedRZero(temperature,
        humidity);
        float resistance = mq135_sensor.getResistance();
        float ppm = mq135_sensor.getPPM();
        float correctedPPM = mq135_sensor.getCorrectedPPM(temperature, humidity);
        // void printMQ135Values(rzero, correctedRZero, resistance, ppm,
        correctedPPM, temperature, humidity)

        /*
        // Create JSON object
        String jsonThingSpeak = createThingSpeakJson(ldr, temperature, humidity,
        valueMQ7Sensor, alarmMQ7Sensor, correctedPPM);
        Serial.println("ThingSpeak Json:");
        Serial.println(jsonThingSpeak);
        // Send from XBee module
        xbeeSerial.println(jsonThingSpeak);
        delay(3000);
        */

        // Create JSON object
        String jsonDataToServer = createSensorDataJSON(poleName, currentTime,
        temperature, humidity, valueMQ7Sensor, correctedPPM,UVindex,ldr);
        Serial.println("Json Data to Server:");
        Serial.println(jsonDataToServer);
        // Send from XBee module
        xbeeSerial.println(jsonDataToServer);
        delay(3000);

    }
}

void setup() {
    pinMode(ledPin, OUTPUT);          // Set LED pin as output
    pinMode(DOUT_MQ7pin, INPUT);      //sets the pin of MQ7 sensor as an input
    to the arduino
}
```

```

Serial.begin(9600);           // Initialize serial communication
xbeeSerial.begin(9600);
dht.begin();                 // Initialize DHT11 sensor
startUvSensor();            // Initialize UV sensor
}

void loop() {
    //setInitialDayTime();

    checkLightStatus();        // check the environmental light
    checkDHT11Status();        // Read temperature and humidity from
    DHT11 sensor

    //time_t currentTime = now();          // Get current timestamp
    checkClockModule();

    checkMQ7Status();             // Get MQ7 status

    // check enviromental conditions and sen them to central node
    performActionEvery10Minutes(poleName, currentTime);

    // if alarm is activated send the message
    checkAndSendGasAlarm(poleName,currentTime,alarmName);
    /*
    if (Serial.available()) {
        char data = Serial.read();
        xbeeSerial.write(data);
    }
    if (xbeeSerial.available()) {
        char data = xbeeSerial.read();
        Serial.write(data);
    }
    */
    delay(3000); // Delay for 3 seconds
}

```

Παράρτημα Β: «Raspberry Pico – Xbee module»

```

...
Author: Kyriakidis Aris
email: a.kyriakidis@hotmail.com
...

import machine
import utime
import urequests
import network
import json
from machine import Pin, PWM

# Configure UART pins
tx_pin = machine.Pin(0) # TX on GP0
rx_pin = machine.Pin(1) # RX on GP1

# Configure UART
uart = machine.UART(0, baudrate=9600, tx=tx_pin, rx=rx_pin)

# Configure LED and LDR pins
led_Pin = Pin(18)          # Change to the appropriate GPIO pin
led_pwm = PWM(led_Pin)     # create a PWM object on that pin
led_pwm.freq(1000)

# Configure ADC for LDR
ldr_pin = 27
ldr_threshold = 2000
ldr = machine.ADC(machine.Pin(ldr_pin)) # initialize an ADC object for pin 27

# Configure pin for PIR sensor
Pir_Pin = 22
pir = Pin(Pir_Pin, Pin.IN, Pin.PULL_DOWN)

# Configure Flame Sensor
flame_Sensor_1 = Pin(10, Pin.IN)
flame_Sensor_2 = Pin(11, Pin.IN)
flame_Sensor_3 = Pin(12, Pin.IN)

#variables

pole_name = 'picoXbee'
alarm_name = 'flame'

```



```
# Networking settings
ssid = ''
password = ''
THINGSPEAK_WRITE_API_KEY = ''

# Configure Pico W as Station
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)

if not sta_if.isconnected():
    print('Connecting to network...')
    sta_if.connect(ssid, password)
    while not sta_if.isconnected():
        pass
    print('Network config:', sta_if.ifconfig())

def receive_data():
    try:
        data_buffer = b""
        while uart.any() > 0:
            data_chunk = uart.read(512) # Adjust the chunk size as needed
            if data_chunk:
                data_buffer += data_chunk
                if b'}' in data_buffer:
                    start_idx = data_buffer.find(b'{')
                    end_idx = data_buffer.find(b'}', start_idx)
                    if start_idx != -1 and end_idx != -1:
                        data = data_buffer[start_idx:end_idx + 2]
                        data_buffer = data_buffer[end_idx + 2:]
                        return data.strip()
    return None
except Exception as e:
    print("UART Error:", e)
    return None

def rename_specific_keys(input_dict):
    try:
        #data = json.loads(input_json)

        # Access the nested ThingSpeak dictionary
        thingSpeak_data = input_dict.get("ThingSpeak", {})
        print(thingSpeak_data)

        # Rename specific keys within the ThingSpeak dictionary
```



```
renamed_data = {
    "field1": thingSpeak_data.get("ldr", None),
    "field2": thingSpeak_data.get("temperature", None),
    "field3": thingSpeak_data.get("humidity", None),
    "field4": thingSpeak_data.get("mq7", None),
    "field5": thingSpeak_data.get("gasAlarm", None),
    "field6": thingSpeak_data.get("mq135", None)
}

return json.dumps(renamed_data)
except json.JSONDecodeError as e:
    print("ThingSpeakJSON Decoding Error:", e)
    return None


def send_json_to_server(json_data):
    try:
        server_ip = "192.168.2.112"
        server_port = 3360

        # Specify the server URL with a concrete endpoint
        server_url = "http://{}:{}{}".format(server_ip, server_port) # Replace
        "api/data" with your actual endpoint

        # Set up HTTP headers
        headers = {'Content-Type': 'application/json'}

        # Convert JSON data to a string
        json_str = json.dumps(json_data)
        #debug
        print(json_str)

        # Make the HTTP GET request
        response = urequests.post(server_url, data=json_str, headers=headers)

        print("Request sent..")
        # Print the server's response (optional)
        print("Server Response:", response.text)

        # Close the request
        response.close()

    except Exception as e:
        print("Error:", e)


def check_Pir_status():
    pir_value = pir.value()
    print(pir_value)
```

```

if pir_value == 1:
    print("Movement Detected")
else:
    print("Waiting for movement")
return pir_value


def manage_energy_saving():
    ldr_value = ldr.read_u16()
    print("LDR Value:", ldr_value)
    pir_status = check_Pir_status()

    # Check PIR status and adjust LED brightness accordingly
    if ldr_value > 5000: # Bright day or full sunlight environment
        led_pwm.duty_u16(0) # Keep LED OFF
    elif pir_status == 0: # If motion is not detected
        if ldr_threshold < ldr_value <= 5000: # Afternoon or cloudy without
sunlight environment
            led_pwm.duty_u16(int(65535 * 0.3)) # Dim light to 30%
        elif ldr_value <= ldr_threshold: # Darkness
            led_pwm.duty_u16(int(65535 * 0.4)) # Keep LED ON at 40% brightness
    else: # Motion is detected
        led_pwm.duty_u16(65535) # Turn on LED to 100% brightness
    print("Motion Detected - LED at 100% brightness")

    light_percentage = round(ldr_value / 65535 * 100, 2)
    return light_percentage


def flame_alarm():
    if flame_Sensor_1.value() == 1 or flame_Sensor_2.value() == 1 or
flame_Sensor_3.value() == 1:
        print("Flame detected")
        # Sending alarm data example (replace "sensorName" with the actual
sensor name)
        send_alarm_data_to_server(pole_name, alarm_name,1)
        utime.sleep(1)
    else:
        print("No flame")
        send_alarm_data_to_server(pole_name, alarm_name,0)
        utime.sleep(1)

# Function to get the current date and time adjusted to Greece, Athens Summer
Time
def getTime():
    # Get current epoch time
    current_time = utime.time()

```

```

# Convert epoch time to local time
local_time = utime.localtime(current_time)

# Format the time as a string
time_str = "{:04d}-{:02d}-{:02d} {:02d}:{:02d}:{:02d}".format(
    local_time[0], local_time[1], local_time[2], local_time[3],
local_time[4], local_time[5]
)

return time_str

# Function to send sensorData to Server

def send_sensor_data_to_server(pole_name, values):
    timestamp = getTime()
    sensor_data = {
        "SensorData": {
            "poleName": pole_name,
            "timestamp": timestamp,
            "values": values
        }
    }
    send_json_to_server(sensor_data)

# Function to send alarmData to Server

def send_alarm_data_to_server(pole_name, sensor_name,value):
    timestamp = getTime()
    alarm_data = {
        "AlarmData": {
            "poleName": pole_name,
            "timestamp": timestamp,
            "sensorName": sensor_name,
            "value":value
        }
    }
    send_json_to_server(alarm_data)

while True:
    check_Pir_status()
    manage_energy_saving()
    flame_alarm()

    received_data = receive_data()

    if received_data is not None:

```



```
try:  
    decoded_data = received_data.decode('utf-8')  
    print(decoded_data)  
    json_data = json.loads(decoded_data)  
  
    if "SensorData" in json_data:  
        send_json_to_server(json_data)  
    elif "AlarmData" in json_data:  
        send_json_to_server(json_data)  
    elif "ThingSpeak" in json_data:  
        thingSpeak_data = rename_specific_keys(json_data)  
        transformed_json = json.loads(thingSpeak_data)  
        if transformed_json is not None:  
            print("Received data:", transformed_json)  
            request = urequests.post(  
                'http://api.thingspeak.com/update?api_key=' +  
                THINGSPEAK_WRITE_API_KEY,  
                json=transformed_json,  
                headers={'Content-Type': 'application/json'}  
            )  
            request.close()  
  
    except ValueError as e:  
        print("JSON Decoding Error:", e)  
    except Exception as e:  
        print("MainError:", e)  
  
    utime.sleep(1)
```

Παράρτημα Γ: «Arduino Uno – LoRa module»

```

/*****
Author: Kyriakidis Aris
email: a.kyriakidis@hotmail.com
*****/

#include <SPI.h>
#include <LoRa.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_BMP280.h>
#include <ArduinoJson.h>
#include <TimeLib.h>
#include <i2cdetect.h>
#include <virtuabotixRTC.h>
/*
#define BMP_SCK  (13)
#define BMP_MISO (12)
#define BMP_MOSI (11)
#define BMP_CS   (10)
*/
#define BMP280_ADDRESS 0x76 // if address is not 0x76 the sensor won't work

Adafruit_BMP280 bmp; // I2C

// Real clock Module Pin assignment
virtuabotixRTC myRTC(4, 5, 6); //If you change the wiring change the pins here
also

// Pin connections
const int lightSensorPin = A0;          // Analog pin for light sensor
const int ledPin = 3;                    // Digital pin for LED
const int soundSensorPin = A2;          // pin connected to pin 2 module sound
sensor

// Variables

String poleName = "arduinoLoRa";
String sensorName = "rain";
//StaticJsonDocument<100> jsonDocumentAlarm;
//StaticJsonDocument<200> jsonDocumentSensor;

//const byte MasterNode = 0xFF;    // Master Node address # raspberry pi 4
const byte Node1 = 0xBB;          // Master Node1 address # arduino uno

```

```

const byte Node2 = 0xCC;           // Master Node2 address # pico w
const byte delimiter = 0x7F;       // the frame start delimiter
/*
===== This is Node01 -- Arduino Uno -- and has static address 0xBB
=====
===== in case that the address has to change then the code to =====
===== all devices has to be modified accordingly =====
*/
String currentTime = "";

// lowest and highest for rain & fire sensors readings:

const int sensorMin = 0;          // sensor minimum
const int sensorMax = 1024;        // sensor maximum
int rainStatus = 0;                // rain status

// msg counter should be removed in the final version -- for tests only
//int counter = 0;

// timing parameter for transmission sensorData

unsigned long previousMillis = 0; // Variable to store the previous time
const unsigned long interval = 60000;//600000; // Interval in milliseconds (10
minutes)

// Light values
const int lightThreshold = 500;    // Adjust this value to set the
sensitivity
int lightValue = 0;                // Variable to store the light sensor
value

// BMP280 sensor return values
float temperature = 0.0;
float pressure = 0.0;

// Sound sensor decibel level
float noiseDB = 0.0;
const int sampleWindow = 50;        // Sample window width in mS (50 mS =
20Hz)
unsigned int sample;

void sendMessage(const String& message, byte recipient, byte sender) {
    byte frameData[256]; // Adjust the buffer size as needed
    int frameDataLength = 0;

    // Construct the frame

```

```

// Add the start delimiter to the frame
frameData[frameDataLength++] = delimiter;

// Add the destination address to the frame
frameData[frameDataLength++] = recipient; // using one of the addresses

// Add the sender address to the frame
frameData[frameDataLength++] = sender; // using one of the addresses

// Convert the message to bytes
int messageLength = message.length();
for (int i = 0; i < messageLength; i++) {
    frameData[frameDataLength++] = message.charAt(i);
}

// Calculate and add the checksum
byte checksum = 0;
for (int i = 0; i < frameDataLength; i++) {
    checksum += frameData[i];
    //Serial.println(checksum,HEX); // for debug
}
frameData[frameDataLength++] = checksum;

// Send the LoRa frame
LoRa.beginPacket();
LoRa.write(frameData, frameDataLength);
LoRa.endPacket();
}

void checkBMP280(){
    temperature = bmp.readTemperature();
    pressure = bmp.readPressure();
}

int checkRainStatus(){

    int sensorReading = analogRead(A3);
    int range = map(sensorReading, sensorMin, sensorMax, 0, 3);
    switch (range) {
    case 0:    // Sensor getting wet
        Serial.println("Flood");
        return 2;
        break;
    case 1:    // Sensor getting wet
        Serial.println("Rain Warning");
        return 1;
        break;
    }
}

```

```

case 2:      // Sensor dry – To shut this up delete the " Serial.println("Not
Raining"); " below.
    //Serial.println("Not Raining");
    return 0;
    break;
}
}

void setInitialDayTime(){
    // ONE TIME ONLY EXECUTION
    // Set the current date, and time in the following format:
    // seconds, minutes, hours, day of the week, day of the month, month, year
    myRTC.setDS1302Time(00, 02, 12, 5, 12, 7, 2024); //Here you write your actual
time/date as shown above
    //but remember to "comment/remove" this function once you're done
    //The setup is done only one time and the module will continue counting it
automatically
}

void checkClockModule(){
    // This allows for the update of variables for time or accessing the
individual elements.
    myRTC.updateTime();
    currentTime = String(myRTC.dayofmonth) + "/" + String(myRTC.month) + "/" +
String(myRTC.year) + " " + String(myRTC.hours) + ":" + String(myRTC.minutes) +
":" + String(myRTC.seconds);
}

void sendJsonToNode02(String jsonString){
    // Print the JSON string to the serial monitor
    Serial.println("JSON Data: ");
    Serial.println(jsonString);
    // from here start the transmission of the json
    sendMessage(jsonString, Node2, Node1);
}

void checkLightStatus(){
    lightValue = analogRead(lightSensorPin);           // Read the light sensor value
    if (lightValue < lightThreshold) {                // Check if it's dark based on
the light sensor reading
        digitalWrite(ledPin, HIGH);                   // It's dark, turn on the LED
        //switchOnTime = millis();                    // Record the LED switch on
time
    } else {                                         // It's light, turn off the LED
        digitalWrite(ledPin, LOW);
    }
}

```

```

        //switchOffTime = millis();                                // Record the LED switch off
time
    }
}

void checkSoundStatus(){

    unsigned long startMillis= millis();                      // Start of sample
window
    float peakToPeak = 0;                                     // peak-to-peak level
    unsigned int signalMax = 0;                               //minimum value
    unsigned int signalMin = 1024;                            //maximum value

    // collect data for 50 mS
    while (millis() - startMillis < sampleWindow) {
        sample = analogRead(soundSensorPin);                  //get reading from
microphone
        if (sample < 1024) {                                  // toss out spurious
readings
            if (sample > signalMax) {                         // save just the max
                signalMax = sample;
levels
            } else if (sample < signalMin) {                  // save just the min
                signalMin = sample;
levels
            }
        }
        peakToPeak = signalMax - signalMin;                  // max - min = peak-
peak amplitude
        noiseDB = map(peakToPeak,20,900,49.5,90);           //calibrate for
deciBels
        //Serial.print("Sound Level: ");
        //Serial.println(noiseDB);
    }

void createSensorDataJSON(String poleName, String timestamp, float temperature,
float pressure, float noiseDB) {
    StaticJsonDocument<128> jsonDocument;
    // Create nested objects and set values
    JsonObject sensorDataObj = jsonDocument.createNestedObject("SensorData");
    sensorDataObj["poleName"] = poleName;
    sensorDataObj["timestamp"] = timestamp;

    JsonObject valuesObj = sensorDataObj.createNestedObject("values");
    valuesObj["temperature"] = temperature;
    valuesObj["pressure"] = pressure;
    valuesObj["noiseDB"] = noiseDB;
    delay(500);
}

```

```

// Serialize JSON to string
String jsonString;
serializeJson(jsonDocument, jsonString);
sendJsonToNode02(jsonString);

}

void createAlarmJSON(String poleName, String timestamp, String sensorName) {
    StaticJsonDocument<100> jsonDocument;
    // Create nested objects and set values
    JsonObject sensorDataObj = jsonDocument.createNestedObject("AlarmData");
    sensorDataObj["poleName"] = poleName;
    sensorDataObj["timestamp"] = timestamp;
    sensorDataObj["sensorName"] = sensorName;
    // Serialize JSON to string
    String jsonString;
    serializeJson(jsonDocument, jsonString);
    delay(200);
    sendJsonToNode02(jsonString);
    delay(200);
}

void setup() {
    Serial.begin(9600);                                // initialize serial
    communication @ 9600 baud:
    while (!Serial);
    pinMode(ledPin, OUTPUT);                          // Set LED pin as output
    //setInitialDayTime();                           // only for the first time
    pinMode (soundSensorPin, INPUT); // Set the signal pin as input
    Serial.println(F("BMP280 test"));
    unsigned status;
    status = bmp.begin(BMP280_ADDRESS);
    if (!status) {
        Serial.println(F("Could not find a valid BMP280 sensor, check wiring or "
                         "try a different address!"));
        Serial.print("SensorID was: 0x"); Serial.println(bmp.sensorID(),16);
        Serial.print("      ID of 0xFF probably means a bad address, a BMP 180
or BMP 085\n");
        Serial.print("      ID of 0x56-0x58 represents a BMP 280,\n");
        Serial.print("      ID of 0x60 represents a BME 280.\n");
        Serial.print("      ID of 0x61 represents a BME 680.\n");
        while (1) delay(10);
    }

    /* Default settings from datasheet. */
    bmp.setSampling(Adafruit_BMP280::MODE_NORMAL,          /* Operating Mode. */
                   Adafruit_BMP280::SAMPLING_X2,           /* Temp. oversampling */
                   Adafruit_BMP280::SAMPLING_X16,          /* Pressure oversampling */

```

```

Adafruit_BMP280::FILTER_X16,      /* Filtering. */
Adafruit_BMP280::STANDBY_MS_500); /* Standby time. */

Serial.println("LoRa Sender");
if (!LoRa.begin(868E6)) { // Set your desired frequency (868 MHz in this
example)
    Serial.println("Starting LoRa failed!");
    while (1);
}
}

void loop() {

//setInitialDayTime();
checkClockModule();           // get the current time
checkLightStatus();           // check the environmental light
checkBMP280();                // check temperature and pressure
checkSoundStatus();           // get the decibel level

// transmit the sensorData every 10 minutes
unsigned long currentMillis = millis(); // Get current time

// Check if 10 minutes have elapsed since the last execution
if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis; // Update the previous time
    // Call createSensorDataJSON method with desired parameters
    createSensorDataJSON(poleName,currentTime,temperature,pressure,nois
create json to send
    delay(200);
}

rainStatus = checkRainStatus();    // check rain status implemt here the
alarm send method for rain
if (rainStatus != 0) {
    // Call createAlarmJSON method with desired parameters
    createAlarmJSON(poleName, currentTime, sensorName); // create json to send
    delay(200);
}
}

```

Παράρτημα Δ: «Raspberry Pico – LoRa module»

```
/*
*****
Author: Kyriakidis Aris
email: a.kyriakidis@hotmail.com
*****
```

```
#include <SPI.h>
#include <LoRa.h>
#include <WiFi.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>

#define nss 8
#define rst 9
#define dio0 7

const char* ssid = "";
const char* password = "";
String poleName = "picoLoRa";
String alarmName = "floodAlarm";
int lastAlarmState = 0;

// the temperature value from tmp36 sensor
float temperature = 0.0;

unsigned long lastSensorDataToServer = millis();

// the REST Server variables

const char* server_ip = "";
const int server_port = 3360;

const byte MasterNode = 0xFF; // Master Node address # raspberry pi 4
const byte Node1 = 0xBB; // Master Node1 address # arduino uno
const byte Node2 = 0xCC; // Master Node2 address # pico w
/*
===== This is Node02 -- Pico W -- and has static address 0xCC =====
===== in case that the address has to change then the code to =====
===== all devices has to be modified accordingly =====
*/
const byte delimiter = 0x7F; // the frame start delimiter
```

```

String receivedMessage = "";

// Pin configuration for LDR and LED
const int ldrPin = 28; // Analog pin for LDR
const int ledPin = 2; // Digital pin for LED

// Pin configuration for water sensor
const int waterSensorPin = 27; // Digital pin for water sensor

// Pin configuration for PIR motion sensor
const int pirPin = 22; // Digital pin for PIR motion sensor

// Adjust this threshold according to your ambient light conditions
const int threshold = 400;

// Define NTP Client to get time
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);

// Define the energy saving variables

bool isLedOn = false;
unsigned long ledActivationTime = 0;
bool isEnergySavingMode = false;

// Create json object
DynamicJsonDocument jsonDocument(254); // JSON document for storing data

// variables to store flags for manageEnergySaving()
unsigned long previousMotionTime = millis(); // Variable to store the previous
motion detection time
bool isMotionDetected = false; // Flag to indicate if motion is currently
detected

// Analog pin connected to the TMP36 sensor on Raspberry Pi Pico
const int tmp36Pin = 26;

void setup() {
    Serial.begin(9600);

    pinMode(ledPin, OUTPUT);
    pinMode(waterSensorPin, INPUT_PULLUP); // Assuming the water sensor provides
    LOW signal when water is detected
    pinMode(pirPin, INPUT); // PIR motion sensor pin
}

```

```

while (!Serial);

LoRa.setPins(nss, rst, dio0);
Serial.println("LoRa Receiver");
while (!LoRa.begin(868E6)) {    // Set your desired frequency (868 MHz in this
example)
    Serial.println("Starting LoRa failed!");
    while (1);
}

Serial.println("LoRa init succeeded.");

// Connect to Wi-Fi
WiFi.begin(ssid, password);
Serial.println("Connecting to WiFi");
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("WiFi connected");

// Initialize NTP client
timeClient.begin();
timeClient.setTimeOffset(7200); // Adjust this offset according to your
timezone (in seconds)
}

String receiveMessage(byte recipient, byte sender) {
    byte frameData[256]; // Adjust the buffer size as needed
    int frameDataLength = 0;

    int packetSize = LoRa.parsePacket();
    if (packetSize) {
        while (LoRa.available()) {
            if (delimiter == LoRa.read()) {
                // Add the start delimiter to the frame
                frameData[frameDataLength++] = delimiter;
                byte receivedRecipient = LoRa.read();
                // Add the recipient to the frame
                frameData[frameDataLength++] = receivedRecipient;
                byte receivedSender = LoRa.read();
                // Add the sender to the frame
                frameData[frameDataLength++] = receivedSender;

                if (receivedRecipient == recipient && receivedSender == sender) {

                    while (LoRa.available()) {

```



```
frameData[frameDataLength++] = LoRa.read();
}

// the checksum from sender
byte checksum = frameData[frameDataLength - 1];
//Serial.println(checksum,HEX); // for debug

// Verify the checksum
byte calculatedChecksum = 0;
for (int i = 0; i < frameDataLength - 1; i++) {
    calculatedChecksum += frameData[i];
    //Serial.println(calculatedChecksum,HEX); // for debug
}

if (calculatedChecksum == checksum) {
    receivedMessage = "";
    for (int i = 3; i < frameDataLength - 1; i++) {
        char receivedChar = (char)frameData[i];
        receivedMessage += receivedChar;
    }
    return receivedMessage;
    //Serial.print("[");
    //printTime();
    //Serial.print("] Received message: ");
    //Serial.print(receivedMessage);
    //Serial.print("' from Node ");
    //Serial.println(receivedSender);
} else {
    Serial.println("Checksum error. Message ignored.");
    return ""; // Return empty string if checksum error
}
} else {
    Serial.println("Unknown recipient. Message ignored.");
    return ""; // Return empty string if checksum error
}
}

}

return "";
}

// Function to print current time with timezone adjustment for Athens, Greece
// (Summer Time)
String printTime() {
    timeClient.update(); // Update time from NTP server

    // Get current time from NTP client
    time_t rawTime = timeClient.getEpochTime();
```

```

struct tm *timeinfo;
timeinfo = gmtime(&rawTime); // Get time in UTC

// Adjust for Athens, Greece (Summer Time)
timeinfo->tm_hour += 1; // Add 1 hour for Athens timezone
timeinfo->tm_isdst = 1; // Set Daylight Saving Time (DST) to true

// Convert adjusted time to epoch
time_t adjustedTime = mktime(timeinfo);

// Create a string to hold the formatted time
char buffer[20]; // Allocate space for the formatted time
strftime(buffer, sizeof(buffer), "%d/%m/%Y %H:%M:%S", timeinfo); // Format the
time
return String(buffer); // Convert the formatted time to a String and return it
}

// Function to check water sensor and sent alarm message to server if water is
detected
int checkWaterSensor() {
if (digitalRead(waterSensorPin)) {
return 1;
} else {
//Serial.print("[ ");
//printTime();
//Serial.print(" ]");
//Serial.print(" - Everything cool!");
//Serial.println();
return 0;
}
}

String createAlarmJSON(String poleName, String timestamp, String alarmName, int
value) {

StaticJsonDocument<200> jsonDocument;
// Create nested objects and set values
JsonObject sensorDataObj = jsonDocument.createNestedObject("AlarmData");
sensorDataObj["poleName"] = poleName;
sensorDataObj["timestamp"] = timestamp;
sensorDataObj["sensorName"] = alarmName;
sensorDataObj["value"] = value;
// Serialize JSON to string
String jsonString;
serializeJson(jsonDocument, jsonString);
jsonDocument.clear();
return jsonString;
}

```

}

```

String createSensorDataJSON(String poleName, String timestamp, float
temperature) {

    StaticJsonDocument<200> jsonDocument;
    // Create nested objects and set values
    JsonObject sensorDataObj = jsonDocument.createNestedObject("SensorData");
    sensorDataObj["poleName"] = poleName;
    sensorDataObj["timestamp"] = timestamp;

    JsonObject valuesObj = sensorDataObj.createNestedObject("values");
    valuesObj["temperature"] = temperature;

    // Serialize JSON to string
    String jsonString;
    serializeJson(jsonDocument, jsonString);
    jsonDocument.clear();
    return jsonString;
}

void sendSensorDataToServer() {
    unsigned long currentMillis = millis();
    unsigned long interval = 60000; // 10 minutes in milliseconds
    String timestamp = printTime();

    // Check if 10 minutes have elapsed since the last sensor data was sent
    if (currentMillis - lastSensorDataToServer >= interval) {
        // Update the last sent time to the current time
        lastSensorDataToServer = currentMillis;

        // Call the createSensorDataJSON method to create the JSON string
        String json_data = createSensorDataJSON(poleName, timestamp, temperature);
        // Assuming you have this method to create sensor data JSON
        // print the json to the server
        Serial.println("Sensor data to send to the server:");
        Serial.println(json_data);
        sendJsonToServer(json_data);
    }
}

void measureTemperature() {
    // Read the raw analog voltage from the TMP36 sensor
    int sensorValue = analogRead(tmp36Pin);

    // Convert the analog value to voltage (in millivolts)
    float voltage = sensorValue * (3300.0 / 1023.0); // 3300 mV (3.3V) is the
    Raspberry Pi Pico reference voltage
}

```

```

// Convert the voltage to temperature in Celsius
temperature = (voltage - 500.0) / 10.0;
}

// Function to manage energy saving mode

void manageEnergySaving() {
    int ldrValue = analogRead(ldrPin); // Read LDR value

    if (ldrValue < threshold) {
        // If LDR value is less than threshold, set LED brightness to 25%
        analogWrite(ledPin, 64); // 25% duty cycle for brightness
    } else {
        // If LDR value is greater than or equal to threshold, turn off the LED
        analogWrite(ledPin, 0); // Turn off LED
        isMotionDetected = false; // Reset motion detected flag
        return; // Exit the function early as there's no need to continue if LDR
value is high
    }

    if (digitalRead(pirPin)) {
        // If motion is detected
        if (!isMotionDetected) {
            // If motion was not previously detected, record the current time
            previousMotionTime = millis();
            isMotionDetected = true; // Set motion detected flag
        }

        // Check if 1 minute has elapsed since motion was detected
        if (millis() - previousMotionTime >= 60000) {
            // If 2 minutes have passed, set LED brightness back to 25%
            analogWrite(ledPin, 64); // Set LED brightness back to 25%
            isMotionDetected = false; // Reset motion detected flag
        } else {
            // If less than 2 minutes have passed, set LED brightness to 100%
            analogWrite(ledPin, 255); // Set LED brightness to 100% (full brightness)
        }
    }
}

void sendJsonToServer(String json_data) {

    // Specify the server URL with a concrete endpoint
    String server_url = "http://" + String(server_ip) + ":" +
String(server_port) + "/";

    // Set up HTTP headers
    HTTPClient http;
    http.begin(server_url);
}

```

```

http.addHeader("Content-Type", "application/json");

// Make the HTTP POST request
int httpResponseCode = http.POST(json_data);

// Print the HTTP response code
Serial.print("HTTP Response code: ");
Serial.println(httpResponseCode);

// Print the server's response (optional)
String response = http.getString();
Serial.println("Server Response:");
Serial.println(response);

// End the request
http.end();

}

void loop() {

    // Receive messages from Node1
    String receivedMsg = receiveMessage(Node2, Node1);
    Serial.print(receivedMsg);

    // Check if the received message is not empty
    if (receivedMsg.length() > 0) {
        sendJsonToServer(receivedMsg); // Send the received message to the server
    }

    // Check water sensor and send message to server if water is detected
    int currentState = checkWaterSensor();

    if (lastAlarmState != currentState) {
        String timestamp = printTime();
        String json = createAlarmJSON(poleName, timestamp, alarmName, currentState);
        sendJsonToServer(json);
        lastAlarmState = currentState;
    }

    // measure Temperature and store it to global variable temrerature
    measureTemperature();

    // Manage energy saving mode
    manageEnergySaving();

    // Check sensor and send message every 10 minutes to server
}

```



```
sendSensorDataToServer();
```

```
}
```

Παράρτημα Ε: «REST Server»

E.1.1 Η κλάση HttpServer:

```
package com.httpserver;

import com.DB.ConnectDB;
import com.DB.DBQuery;
import com.DB.configurationDB.Config;
import com.google.gson.JsonArray;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;
import com.httpserver.config.Configuration;
import com.httpserver.core.ServerListenerThread;
import com.httpserver.config.ConfigurationManager;
import org.slf4j.*;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.Map;

public class HttpServer {

    private final static Logger LOGGER =
    LoggerFactory.getLogger(HttpServer.class);

    public static void main(String[] args) {
        LOGGER.info("Server starting....");

        ConfigurationManager.getInstance().loadConfigurationFile("src/main/resou
rces/http.json");
        Configuration configuration =
        ConfigurationManager.getInstance().getCurrentConfiguration();

        LOGGER.info("Using Port: " + configuration.getPort());
        LOGGER.info("Using WebRoot: " + configuration.getWebRoot());

        try {
            ServerListenerThread serverListenerThread = new
            ServerListenerThread(configuration.getPort(),
            configuration.getWebRoot());
            serverListenerThread.start();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

E.1.2 Η κλάση ServerListenerThread:

```

package com.httpserver.core;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class ServerListenerThread extends Thread {

    private final static Logger LOGGER =
    LoggerFactory.getLogger(ServerListenerThread.class);

    private int port;
    private String webroot;
    private ServerSocket serverSocket;

    public ServerListenerThread(int port, String webroot) throws
    IOException {
        this.port = port;
        this.webroot = webroot;
        this.serverSocket = new ServerSocket(this.port);
    }

    @Override
    public void run() {

        try {

            while ( serverSocket.isBound() && !serverSocket.isClosed() )
{
                Socket socket = serverSocket.accept();
                LOGGER.info(" * Connection accepted: " +
socket.getInetAddress());
                HttpConnectionWorkerThread workerThread = new
HttpConnectionWorkerThread(socket);
                workerThread.start();
            }

            } catch (IOException e) {
                LOGGER.error("Problem with setting socket", e);
            } finally {
                if (serverSocket != null) {
                    try {
                        serverSocket.close();
                    } catch (IOException e) {}
                }
            }
        }
    }
}

```

}

E.1.3 Η κλάση HttpConnectionWorkerThread:

```

package com.httpserver.core;
import com.http.HttpParser;
import com.http.HttpRequest;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.*;
import java.net.Socket;

public class HttpConnectionWorkerThread extends Thread {
    private final static Logger LOGGER =
    LoggerFactory.getLogger(HttpConnectionWorkerThread.class);
    private Socket socket;

    public HttpConnectionWorkerThread(Socket socket) {
        this.socket = socket;
    }

    @Override
    public void run() {
        InputStream inputStream = null;
        OutputStream outputStream = null;

        try {
            inputStream = socket.getInputStream();
            outputStream = socket.getOutputStream();

            HttpRequest httpRequestTest = new HttpRequest();
            httpRequestTest.httpSerializer(inputStream);

            //String html = "<html><head><title>Simple Java HTTP
Server</title></head><body><h1>This page was served using my Simple Java
HTTP Server</h1></body></html>";
            String message = "This is the server response!";

            final String CRLF = "\r\n"; // 13, 10

            String response =
                "HTTP/1.1 200 OK" + CRLF + // Status Line :
                "HTTP_VERSION RESPONSE_CODE RESPONSE_MESSAGE"
                "Content-Length: " +
            message.getBytes().length + CRLF + // HEADER
            CRLF +
            message +
            CRLF + CRLF;

            outputStream.write(response.getBytes());
            LOGGER.info(" * Connection Processing Finished.");
        }
        System.out.printf("Method : %s\nURL :
%s\n", httpRequestTest.getMethod(), httpRequestTest.getUrl());
    }
}

```



```
httpRequestTest.getHeaders().forEach((key,value)->
System.out.printf("%s : %s\n",key,value));
    System.out.println("=".repeat(50));
    System.out.printf("Http Request Body :
%s\n",httpRequestTest.getHttpBody());
    System.out.println("=".repeat(50));

    HttpParser httpParserTest = new HttpParser(httpRequestTest);

    // TODO the implementation for receiving the JSON and insert
in the DB
    //String test = "{\"mq7\": 43, \"mq135\": 7.573481, \"uv\":
0.02, \"temperature\": 20.6, \"humidity\": 60, \"ldr\": 624, \"flame\":
0, \"timestamp\": 'January 08, 2024 12:12', \"gas\": 0, \"poleId\":
\"arduinoXbee\"}";

httpParserTest.bodyToJsonConvert(httpRequestTest.getHttpBody());

    } catch (IOException e) {
        LOGGER.error("Problem with communication", e);
    } finally {
        if (inputStream != null) {
            try {
                inputStream.close();
            } catch (IOException e) {}
        }
        if (outputStream != null) {
            try {
                outputStream.close();
            } catch (IOException e) {}
        }
        if (socket != null) {
            try {
                socket.close();
            } catch (IOException e) {}
        }
    }
}

public String readSocketInput(InputStream inputStream) throws
IOException {

    InputStreamReader inputStreamReader = new
InputStreamReader(inputStream);
    BufferedReader bufferedReader = new
BufferedReader(inputStreamReader);

    StringBuilder stringBuilder = new StringBuilder();
    String line;

    while ((line = bufferedReader.readLine()) != null) {
        stringBuilder.append(line).append("\n");
    }

    return stringBuilder.toString();
}
```



E.1.4 Η κλάση HttpConfigurationException:

```
package com.httpserver.config;

public class HttpConfigurationException extends RuntimeException {

    public HttpConfigurationException() {
    }

    public HttpConfigurationException(String message) {
        super(message);
    }

    public HttpConfigurationException(String message, Throwable cause) {
        super(message, cause);
    }

    public HttpConfigurationException(Throwable cause) {
        super(cause);
    }

    public HttpConfigurationException(String message, Throwable cause,
boolean enableSuppression, boolean writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
    }
}
```

E.1.5 Η κλάση ConfigurationManager:

```
package com.httpserver.config;

import com.httpserver.util.Json;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;

import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class ConfigurationManager {

    private static ConfigurationManager myConfigurationManager;
    private static Configuration myCurrentConfiguration;

    private ConfigurationManager() {}

    // Singleton Pattern to get only one instance
    public static ConfigurationManager getInstance() {
        if(myConfigurationManager == null) {
            myConfigurationManager = new ConfigurationManager();
        }
        return myConfigurationManager;
    }
}
```

```
// is used to load the configuration file by the provided path

public void loadConfigurationFile (String filePath) {
    FileReader fileReader = null;
    try {
        fileReader = new FileReader(filePath);
    } catch (FileNotFoundException e) {
        throw new HttpConfigurationException(e);
    }
    StringBuffer stringBuffer = new StringBuffer();

    int i;
    try {
        while ( (i = fileReader.read()) != -1)
            stringBuffer.append((char) i);
    } catch (IOException e) {
        throw new HttpConfigurationException(e);
    }

    JsonNode configuration = null;
    try {
        configuration = Json.parse(stringBuffer.toString());
    } catch (IOException e) {
        throw new HttpConfigurationException("Error parsing the
Configuration File.");
    }
    try {
        myCurrentConfiguration = Json.fromJson(configuration,
Configuration.class);
    } catch (JsonProcessingException e) {
        throw new HttpConfigurationException("Error parsing the
configuration file, internal", e);
    }
}

// returns the current loaded configuration
public Configuration getCurrentConfiguration() {
    if(myCurrentConfiguration == null)
        throw new HttpConfigurationException("No current
Configuration set!");
    return myCurrentConfiguration;
}
```

E.1.6 Η κλάση Configuration:

```
package com.httpserver.config;

public class Configuration {
    private int port;
    private String webRoot;

    public int getPort() {
```

```

        return port;
    }

    public void setPort(int port) {
        this.port = port;
    }

    public String getWebRoot() {
        return webRoot;
    }
    public void setWebRoot(String webRoot) {
        this.webRoot = webRoot;
    }
}

```

E.1.7 Η κλάση HttpVersion:

```

package com.http;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public enum HttpVersion {

    HTTP_1_1("HTTP/1.1", 1, 1);

    public final String LITERAL;
    public final int MAJOR;
    public final int MINOR;

    HttpVersion(String LITERAL, int MAJOR, int MINOR) {
        this.LITERAL = LITERAL;
        this.MAJOR = MAJOR;
        this.MINOR = MINOR;
    }

    private static final Pattern httpVersionRegexPattern =
        Pattern.compile("^HTTP/(<major>\\d+).(<minor>\\d+)");

    public static HttpVersion getBestCompatibleVersion(String
literalVersion) throws BadHttpVersionException {
        Matcher matcher =
        httpVersionRegexPattern.matcher(literalVersion);
        if (!matcher.find() || matcher.groupCount() != 2) {
            // TODO check if this is the exception
            throw new BadHttpVersionException();
        }

        HttpVersion tempBestCompatible = null;
        int major = Integer.parseInt(matcher.group("major"));
        int minor = Integer.parseInt(matcher.group("minor"));

        for (HttpVersion verion : HttpVersion.values()) {
            if (verion.LITERAL.equals(literalVersion))
                return verion;
            else if (verion.MAJOR == major)

```



```
        if (verion.MINOR < minor)
            tempBestCompatible = verion;
    }
    return tempBestCompatible;
}
}
```

E.1.8 Η κλάση HttpStatusCode:

```
package com.http;

public enum HttpStatusCode {

    /*** CLIENT ERRORS ***/

    CLIENT_ERROR_400_BAD_REQUEST(400, "Bad Request"),
    CLIENT_ERROR_401_METHOD_NOT_ALLOWED(401, "Method not Allowed"),
    CLIENT_ERROR_414_BAD_REQUEST(414, "URI too long"),

    /*** SERVER ERRORS ***/

    SERVER_ERROR_500_INTERNAL_SERVER_ERROR(500, "Internal Server
Error"),
    SERVER_ERROR_501_NOT_IMPLEMENTED(501, "Not Implemented"),
    SERVER_ERROR_505_HTTP_VERSION_NOT_SUPPORTED(505, "HTTP Version Not
Supported");

    public final int STATUS_CODE;

    public final String MESSAGE;

    HttpStatusCode(int STATUS_CODE, String MESSAGE) {
        this.STATUS_CODE = STATUS_CODE;
        this.MESSAGE = MESSAGE;
    }
}
```

E.1.9 Η κλάση HttpRequest:

```
package com.http;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.HashMap;
import java.util.Map;

public class HttpRequest {

    private String method;
    private String url;
    private Map<String, String> headers = new HashMap<>();
```

```

private String httpBody;
public HttpRequest() {
}

public String getMethod() {
    return method;
}

public String getUrl() {
    return url;
}

public Map<String, String> getHeaders() {
    return headers;
}

public String getHttpBody() {
    return httpBody;
}

public void httpSerializer(InputStream inputStream) throws
IOException {
    InputStreamReader reader = new InputStreamReader(inputStream);
    BufferedReader bufferedReader = new BufferedReader(reader);
    parseMetaData(bufferedReader);
    parseBody(bufferedReader);
}

private void parseMetaData(BufferedReader bufferedReader) throws
IOException {
    String firstLine = bufferedReader.readLine();
    this.method = firstLine.split("\s+")[0];
    this.url = firstLine.split("\s+")[1];

    String headerLine;
    while ((headerLine = bufferedReader.readLine()) != null) {
        if (headerLine.trim().isEmpty()) {
            break;
        }
        String key = headerLine.split(":\\s")[0];
        String value = headerLine.split(":\\s")[1];
        this.headers.put(key,value);
    }
}

private void parseBody(BufferedReader bufferedReader) throws
IOException {
    StringBuilder bodyBuilder = new StringBuilder();
    int contentLength = getContentLength();
    if (contentLength > 0) {
        char[] bodyBuffer = new char[contentLength];
        bufferedReader.read(bodyBuffer, 0, contentLength);
        bodyBuilder.append(bodyBuffer);
    }
    this.httpBody = bodyBuilder.toString();
}

private int getContentLength() {

```



```
String contentLengthValue = this.headers.get("Content-Length");
if (contentLengthValue != null && !contentLengthValue.isEmpty())
{
    try {
        return Integer.parseInt(contentLengthValue);
    } catch (NumberFormatException e) {
        e.printStackTrace(); // Handle parsing error
    }
}
return 0;
}
```

E.1.10 Η κλάση HttpParseException:

```
package com.http;

public class HttpParseException extends Exception {

    private final HttpStatusCode errorCode;

    public HttpParseException(HttpStatusCode errorCode) {
        super(errorCode.MESSAGE);
        this.errorCode = errorCode;
    }

    public HttpStatusCode getErrorCode() {
        return errorCode;
    }
}
```

Η κλάση HttpParser:

```
package com.http;

import com.DB.ConnectDB;
import com.DB.DBQuery;
import com.DB.configurationDB.TableMapper;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.google.gson.*;

import java.util.HashMap;
import java.util.Map;

public class HttpParser {

    private HttpRequest httpRequestTest;

    private final static Logger LOGGER =
    LoggerFactory.getLogger(HttpParser.class);

    public HttpParser(HttpRequest httpRequestTest) {
        this.httpRequestTest = httpRequestTest;
```

```

}

public void bodyToJsonConvert(String json) {

    JsonElement jsonElement = JsonParser.parseString(json);
    if (jsonElement.isJsonPrimitive()) {
        json = jsonElement.getAsString();
        jsonElement = JsonParser.parseString(json);
    }

    if (jsonElement.isJsonObject()) {
        JsonObject jsonObject = jsonElement.getAsJsonObject();
        System.out.println("is object");
        handleData(jsonObject);
    }
}

private static void handleData(JsonObject sensorDataObject) {
    if (sensorDataObject != null) {

        // Extract poleID and timestamp
        String pole =
sensorDataObject.remove("poleId").getAsString();

        // Extract sensor values
        //JsonObject valuesObject =
sensorDataObject.getAsJsonObject("values");

        ConnectDB connectDB = ConnectDB.getConnectionInstance();
        DBQuery dbQuery = new DBQuery(connectDB);
        int poleIdentity = dbQuery.selectSpecificPole(pole);

        // Get the configMap from TableMapper
        HashMap<String, String[]> configMap =
TableMapper.getInstance().getConfigMap();

        StringBuilder insertPart = new StringBuilder("Insert into
sensor_values (poleID");
        StringBuilder valuesPart = new StringBuilder("values (" );
        valuesPart.append(poleIdentity);

        // Iterate over sensor values
        for (Map.Entry<String, JsonElement> entry :
sensorDataObject.entrySet()) {
            String sensorName = entry.getKey();

            // Check if sensorName exists in configMap
            if (configMap.containsKey(sensorName)) {
                insertPart.append(",");
                String columnName = configMap.get(sensorName)[0];
                insertPart.append(columnName);
                String sensorValue = entry.getValue().getAsString();
                switch (configMap.get(sensorName)[1]) {
                    case "String":
                        valuesPart.append(" , ");
                }
            }
        }
    }
}

```

```

valuesPart.append("'").append(sensorValue).append("'");
    break;
    default:
        valuesPart.append(" ,
") .append(sensorValue);
            break;
        }
    } else {
        System.out.println("Sensor Name: " + sensorName + "
not found in configMap.");
    }
}
insertPart.append(", timestamp");
valuesPart.append(", CURRENT_TIMESTAMP());
String insertQuery =
insertPart.append(valuesPart).toString();
System.out.println(insertQuery);
dbQuery.insertData(insertQuery);
}

private static void handleAlarmData(JsonObject alarmDataObject) {
    if (alarmDataObject != null) {

        // Extract poleID and timestamp
        String pole = alarmDataObject.get("poleName").getAsString();
        String timestamp =
alarmDataObject.get("timestamp").getAsString();
        String sensorName =
alarmDataObject.get("sensorName").getAsString();

        ConnectDB connectDB = ConnectDB.getConnectionInstance();
        DBQuery dbQuery = new DBQuery(connectDB);
        int poleIdentity = dbQuery.selectSpecificPole(pole);

        // Get the configMap from TableMapper
        HashMap<String, String[]> configMap =
TableMapper.getInstance().getConfigMap();
        if (configMap.containsKey(sensorName)) {
            String tableName = configMap.get(sensorName)[0];

        dbQuery.insertAlarmData(tableName,poleIdentity,sensorName,timestamp);
            System.out.printf("%s inserted in
%s\n",sensorName,tableName);
        } else {
            System.out.println("Sensor Name: " + sensorName + " not
found in configMap.");
        }
    }
}

private static void handleLightData(JsonObject lightDataObject) {
    if (lightDataObject != null) {

        // Extract poleID and timestamp
        String pole = lightDataObject.get("poleName").getAsString();

```

```

        String timestamp =
lightDataObject.get("timestamp").getAsString();
        JSONObject valuesObject =
lightDataObject.getAsJsonObject("values");

        // Extract values from the valuesObject
        String startTime =
valuesObject.get("StartTime").getAsString();
        String endTime = valuesObject.get("EndTime").getAsString();
        int value = valuesObject.get("value").getAsString();

        ConnectDB connectDB = ConnectDB.getConnectionInstance();
        DBQuery dbQuery = new DBQuery(connectDB);
        int poleIdentity = dbQuery.selectSpecificPole(pole);

        // Get the configMap from TableMapper
        HashMap<String, String[]> configMap =
TableMapper.getInstance().getConfigMap();
        if (configMap.containsKey("LightData")) {
            String tableName = configMap.get("LightData")[0];

dbQuery.insertLightData(tableName,poleIdentity,startTime,endTime,timestamp,value);
            System.out.printf("From %s, light data inserted in
%s\n",pole,tableName);
        } else {
            System.out.println("LightData not found in configMap.");
        }
    }
}

```

E.1.11 Η κλάση HttpMethod:

```
package com.http;

public enum HttpMethod {
    GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS;

    public static final int MAX_LENGTH;

    static {
        int tempMaxLength = -1;
        for (HttpMethod method : values())
            if (method.name().length() > tempMaxLength)
                tempMaxLength = method.name().length();
        MAX_LENGTH = tempMaxLength;
    }
}
```

Η κλάση `HttpMessage`:

```
package com.http;  
  
public abstract class HttpMessage {
```

```

private HttpMethod method;
private String requestTarget;
private String httpVersion;

HttpMessage() {
}

public HttpMethod getMethod() {
    return method;
}

public void setMethod(HttpMethod method) {
    this.method = method;
}
}
}

```

E.1.12 Η κλάση DBquery:

```

package com.DB;
import java.sql.*;

public class DBQuery {
    private ConnectDB connectDB;

    public DBQuery(ConnectDB connectDB) {
        this.connectDB = connectDB;
    }

    // TODO the query design for the scenarios

    public void insertNewData(String tableName, int poleId, float
mq7Value, float mq135Value, float uvValue, float temperatureValue, float
humidityValue, float gasValue, String timestampValue) {

        try {
            Connection connection = connectDB.connect();
            String insertSQL = "Insert into " + tableName + " (poleID,
mq7Value, mq135Value, uvValue, temperatureValue, humidityValue,
gasValue, timestampValue) values(?,?,?,?,?,?,,?)";
            PreparedStatement preparedStatement =
connection.prepareStatement(insertSQL);
            preparedStatement.setInt(1, poleId);
            preparedStatement.setFloat(2, mq7Value);
            preparedStatement.setFloat(3, mq135Value);
            preparedStatement.setFloat(4, uvValue);
            preparedStatement.setFloat(5, temperatureValue);
            preparedStatement.setFloat(6, humidityValue);
            preparedStatement.setFloat(7, gasValue);
            preparedStatement.setString(8, timestampValue);

            ExecuteQuery(tableName, connection, preparedStatement);
            // System.out.println("Done!"); // for BackEnd
        } catch (SQLException sqlException) {
            System.out.println(sqlException.getLocalizedMessage());
        }
    }
}

```



```
        }

    }

public void insertData(String insertSQL) {

    try {
        Connection connection = connectDB.connect();
        Statement stmt = connection.createStatement();
        stmt.executeUpdate(insertSQL);
        // System.out.println("Done!"); // for BackEnd
    } catch (SQLException sqlException) {
        System.out.println(sqlException.getLocalizedMessage());
    }
}

// method to select poleID on Pole table by pole_name

public int selectSpecificPole(String poleName) {
    Integer poleId = null;
    try {
        Connection connection = connectDB.connect();
        String selectSQL = "SELECT poleID from Pole where pole_name
= ?;";
        PreparedStatement preparedStatement =
connection.prepareStatement(selectSQL);
        preparedStatement.setString(1, poleName);
        ResultSet rs = preparedStatement.executeQuery();
        rs.next();
        poleId = rs.getInt("poleID");
        preparedStatement.close();
        connection.close();
        return (int) poleId;
    } catch (SQLException throwables) {
        System.out.println(throwables.getLocalizedMessage());
    }
    return (int) poleId;
}

public void insertSensorData(String tableName, int poleId, float
value, String timestampValue) {

    try {
        Connection connection = connectDB.connect();
        String insertSQL = "Insert into " + tableName + " (poleID,
value, timestampValue) values(?, ?, ?)";
        PreparedStatement preparedStatement =
connection.prepareStatement(insertSQL);
        preparedStatement.setInt(1, poleId);
        preparedStatement.setFloat(2, value);
        preparedStatement.setTimestamp(3,
Timestamp.valueOf(timestampValue));
        ExecuteQuery(tableName, connection, preparedStatement);
    } catch (SQLException sqlException) {
        System.out.println(sqlException.getLocalizedMessage());
    }
}
```



```
public void insertAlarmData(String tableName, int poleId, String
sensorName, String activationTime) {
    try {
        Connection connection = connectDB.connect();
        String insertSQL = "Insert into " + tableName + " (poleID,
ActivationTime, alarmName) values(?, ?, ?)";
        PreparedStatement preparedStatement =
connection.prepareStatement(insertSQL);
        preparedStatement.setInt(1, poleId);
        preparedStatement.setTimestamp(2,
Timestamp.valueOf(activationTime));
        preparedStatement.setString(3, sensorName);
        ExecuteQuery(tableName, connection, preparedStatement);
    } catch (SQLException sqlException) {
        System.out.println(sqlException.getLocalizedMessage());
    }
}

public void insertLightData(String tableName, int poleId, String
startTime, String endTime, String timeStamp, int value) {

    try {
        Connection connection = connectDB.connect();
        String insertSQL = "Insert into " + tableName + " (poleID,
StartTime, EndTime, timestamp, value) values(?, ?, ?, ?)";
        PreparedStatement preparedStatement =
connection.prepareStatement(insertSQL);
        preparedStatement.setInt(1, poleId);
        preparedStatement.setTimestamp(2,
Timestamp.valueOf(startTime));
        preparedStatement.setTimestamp(3,
Timestamp.valueOf(endTime));
        preparedStatement.setTimestamp(4,
Timestamp.valueOf(timeStamp));
        preparedStatement.setInt(5, value);
        ExecuteQuery(tableName, connection, preparedStatement);
    } catch (SQLException sqlException) {
        System.out.println(sqlException.getLocalizedMessage());
    }
}

private void ExecuteQuery(String tableName, Connection connection,
PreparedStatement preparedStatement) throws SQLException {
    int count = preparedStatement.executeUpdate();
    if (count > 0) {
        System.out.printf("The %s table was updated\n", tableName);
        System.out.printf("%d row(s) inserted", count);
    } else {
        System.out.println("Something went wrong. Check the
exception"); // for BackEnd
    }
    preparedStatement.close();
    connection.close();
}
```

```
}
```

E.1.13 Η κλάση ConnectDB:

```
package com.DB;

import com.DB.configurationDB.Config;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class ConnectDB {

    //create an object of ConnectDB
    private static ConnectDB connectionInstance = new ConnectDB();

    //make the constructor private so that this class cannot be
instantiated
    private ConnectDB() {
    }

    //Get the only object available
    public static ConnectDB getConnectionInstance() {
        if (connectionInstance == null)
            connectionInstance = new ConnectDB();
        return connectionInstance;
    }
    public Connection connect() {

        String connectionString =
"jdbc:mysql://" + Config.ADDRESS + ":" + String.valueOf(
Config.PORT) + "/testGetPost?useSSL=false"; // this should
change to the final DB name
        Connection connection = null;
        try {
            connection = DriverManager.getConnection(connectionString,
Config.USER, Config.PASSWORD);
            if (connection != null)
                System.out.println("Connection to DataBase
established!");
        } catch (SQLException e) {
            e.printStackTrace();
        }
        // Step 2:Create a statement using connection object
        //Statement statement = connection.createStatement();

        // Step 3: Execute the query or update query
        //statement.execute(createTableSQL);

        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
    }
}
```

```

        return connection;
    }
}

```

E.1.14 Η κλάση TableMapper:

```

package com.DB.configurationDB;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.lang.reflect.Type;
import java.util.HashMap;

public class TableMapper {

    private static TableMapper myTableManager;
    private final String filePath =
"src/main/java/com/DB/configurationDB/mapper.json";

    private TableMapper() {}

    // Singleton Pattern with thread safety
    public static synchronized TableMapper getInstance() {
        if (myTableManager == null) {
            myTableManager = new TableMapper();
        }
        return myTableManager;
    }

    public HashMap<String, String[]> getConfigMap() {
        String jsonString = readFile(filePath);

        if (jsonString != null) {
            // Using Gson to parse JSON string into HashMap
            Gson gson = new Gson();
            Type type = new TypeToken<HashMap<String,
String[]>>().getType();
            return gson.fromJson(jsonString, type);
        } else {
            System.err.println("Failed to read the JSON from the
file.");
            return null;
        }
    }

    // Function to read file content and return as string
    private String readFile(String filePath) {
        StringBuilder content = new StringBuilder();
        try (BufferedReader reader = new BufferedReader(new
FileReader(filePath))) {
            String line;
            while ((line = reader.readLine()) != null) {
                content.append(line);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return content.toString();
    }
}

```



```
        }
    } catch (IOException e) {
        System.err.println("Error reading file: " + e.getMessage());
        return null;
    }
    return content.toString();
}
}
```

Παράρτημα ΣΤ: «Διαδικτυακή Εφαρμογή - PHP»

ΣΤ.1 Controllers

```
<?php

require 'functions/functions.php';
$heading = "Αρχική";
require "views/index.view.php";

<?php

$heading = 'Στατιστικά';
require 'functions/functions.php';
require "views/statistics.view.php";

<?php

$heading = 'Register';
require 'functions/functions.php';
require "views/register.view.php";

<?php

$heading = 'Pole Details';
require 'functions/functions.php';
require "views/poleDetails.view.php";

<?php

require 'functions/functions.php';
$heading = "Χάρτης";
require "views/map.view.php";

<?php

$heading = 'logout';
require 'functions/functions.php';
require "views/logout.view.php";

<?php

$heading = 'Login';
require 'functions/functions.php';
require "views/login.view.php";

<?php

require 'functions/functions.php';
$heading = "IoT";
require "views/IoT.view.php";

<?php
```

```

require 'functions/functions.php';
$heading = "About";
$headingGR = "Περίληψη";
require "views/about.view.php";

<?php

$heading = 'Συγκριτικά Στατιστικά';
require 'functions/functions.php';
require 'views/comparativeStatistics.view.php';

<?php

$heading = 'Κάμερα';
require 'functions/functions.php';
require "views/camera.view.php";

```

ΣΤ.2 Views

```

<?php

require_once 'providers/auth_provider.php';
require_once 'providers/DB.php';
$mapData = require_once 'providers/map.php';
include 'partials/head.php';

// Navigation bar
// include 'partials/nav.php';
?>
<body class="h-full min-h-[100vh] flex flex-col">
    <?php
    include 'partials/nav.php';
    ?>
    <!-- Header -->
    <div class="bg-white shadow h-full overflow-auto">
        <div class="mx-auto max-w-7xl px-4 py-6 sm:px-6 lg:px-8 h-full flex
flex-col justify-center">
            <div>
                <a href="index.php">
                    
                </a>
            </div>
            <div class="text-center mt-4">
                <h1 class="text-3xl font-bold tracking-tight text-gray-900
px-4 py-2">Σχολή Θετικών Επιστημών και Τεχνολογίας</h1>
                <h2 class="text-3xl font-bold tracking-tight text-gray-900
mt-2 px-4 py-2">Τμήμα Πληροφορικής</h2>
                <h3 class="text-2xl font-bold tracking-tight text-gray-900
mt-2 px-4 py-2">Πτυχιακή Εργασία ΠΕ419</h3>
                <p class="text-2xl mt-4 text-gray-900">Σχεδιασμός και  
ανάπτυξη «Έξυπνης Κολώνας»<br>δημόσιου φωτισμού σε περιβάλλον μίας  
έξυπνης πόλης</p>
                <h4 class="text-2xl font-bold tracking-tight text-gray-900
mt-6">Επιβλέπων Καθηγητής</h4>
                <p class="text-lg text-gray-900">Δρ. Τοπάλης Ευάγγελος</p>

```

```

<h5 class="text-2xl font-bold tracking-tight text-gray-900
mt-6">Φοιτητής</h5>
    <p class="text-lg text-gray-900"><strong>Κυριακίδης
Αριστοκλής</strong></p>
    <p class="text-lg text-gray-900">137659</p>
</div>
</div>
<!-- End of Header -->
<!-- Main -->
<main>
    <?php
    if (!isAuthenticated()) {
        ?>
        <div class="mx-auto max-w-7xl py-2 sm:px-6 lg:px-8 flex
justify-center space-x-8">
            <a href="login.php" >
                <button class="bg-cyan-900 hover:bg-cyan-800 text-
white font-bold py-2 px-4 rounded">Είσοδος</button>
            </a>
            <a href="register.php" >
                <button class="bg-cyan-900 hover:bg-cyan-800 text-
white font-bold py-2 px-4 rounded">Εγγραφή</button>
            </a>
        </div>
        <?php
    }
    ?>
</main>
<!-- End of Main -->
<!-- Footer -->
<?php include 'partials/footer.php';?>
<!-- End of footer -->

</body>
</html>

<?php

require_once 'providers/auth_provider.php';
require_once 'providers/DB.php';
include 'partials/head.php';
$poleID = isset($_GET['id']) ? $_GET['id'] : null;
// TODO the details for the page
// $poleDetails = "";
$data = require_once 'providers/poleSensorDetailsTest.php';
// Initialize an empty array for gauges
$gaugesAlarm = array();
$gaugesSensor = array();

// Add header for gauges
$gaugesAlarm[] = ['Label', 'Value'];
$gaugesSensor[] = ['Label', 'Value'];

// Loop through the data to build the gauges array
foreach ($data['dataAlarm'] as $gauge) {
    if ($poleID == $gauge['poleID']) {
        // TODO dynamic values for the sensors

```

```

        $gaugesAlarm[] = [$gauge['label'], 0];
    }
}

// Loop through the data to build the gauges array
foreach ($data['dataSensor'] as $gauge) {
    if ($poleID == $gauge['poleID']) {
        // TODO dynamic values for the sensors
        //dd($gauge['value']);
        $gaugesSensor[] = [$gauge['label'], 40];
    }
}

// Convert $gauges array to JSON format
$gaugesAlarmJson = json_encode($gaugesAlarm);
$gaugesSensorJson = json_encode($gaugesSensor);

?>

<script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript">

    google.charts.load('current', {'packages':['gauge']});
    google.charts.setOnLoadCallback(drawChart);

    function drawChartTest() {

        var data = google.visualization.arrayToDataTable([
            ['Label', 'Value'],
            ['test', 80],
        ]);

        var options = {
            width: 400, height: 120,
            redFrom: 90, redTo: 100,
            yellowFrom: 75, yellowTo: 90,
            minorTicks: 5
        };

        var chart = new
google.visualization.Gauge(document.getElementById('test'));

        chart.draw(data, options);

        setInterval(function() {
            data.setValue(0, 1, 40 + Math.round(60 * Math.random()));
            chart.draw(data, options);
        }, 13000);
    }

    function drawChart() {

        var gaugesAlarm = <?= $gaugesAlarmJson ?>;
        var gaugesSensor = <?= $gaugesSensorJson ?>;
    }
}

```



```
var dataAlarm =  
google.visualization.arrayToDataTable(gaugesAlarm);  
var dataSensor =  
google.visualization.arrayToDataTable(gaugesSensor);  
  
var optionsAlarm = {  
    width: 400, height: 120,  
    redFrom: 75, redTo: 100,  
    yellowFrom: 50, yellowTo: 74,  
    minorTicks: 5  
};  
  
var optionsSensor = {  
    width: 800, height: 120,  
    redFrom: 90, redTo: 100,  
    yellowFrom: 75, yellowTo: 90,  
    minorTicks: 5  
};  
  
var chartAlarm = new  
google.visualization.Gauge(document.getElementById('chart_Alarms'));  
var chartSensor = new  
google.visualization.Gauge(document.getElementById('chart_Sensors'));  
  
chartAlarm.draw(dataAlarm, optionsAlarm);  
chartSensor.draw(dataSensor, optionsSensor);  
  
setInterval(function() {  
    // Generate a random number between 0 and 1  
    var random = Math.random();  
    // Set the value based on the random number  
    var value = random < 0.5 ? 0 : 100;  
    // Set the value in the data table  
    data.setValue(0, 1, value);  
    // Draw the chart with the updated data  
    chart.draw(data, options);  
}, 5000);  
setInterval(function() {  
    // Generate a random number between 0 and 1  
    var random = Math.random();  
    // Set the value based on the random number  
    var value = random < 0.5 ? 0 : 100;  
    // Set the value in the data table  
    chartAlarm.setValue(0, 1, value);  
    // Draw the chart with the updated data  
    chartAlarm.draw(data, options);  
}, 5000);  
setInterval(function() {  
    // Generate a random number between 0 and 1  
    var random = Math.random();  
    // Set the value based on the random number  
    var value = random < 0.5 ? 0 : 100;  
    // Set the value in the data table  
    chartAlarm.setValue(0, 1, value);  
    // Draw the chart with the updated data  
    chartAlarm.draw(data, options);  
}, 5000);
```

```

setInterval(function() {
    chartSensor.setValue(0, 1, 40 + Math.round(60 *
Math.random()));
    chartSensor.draw(data, options);
}, 5000);
setInterval(function() {
    chartSensor.setValue(1, 1, 40 + Math.round(60 *
Math.random()));
    chartSensor.draw(data, options);
}, 5000);
setInterval(function() {
    chartSensor.setValue(2, 1, Math.random() < 0.5 ? 0 : 100);
    chartSensor.draw(data, options);
}, 5000);
}
</script>

```

```

<body class="h-full min-h-full">
<!-- Navigation bar -->
<?php
include 'partials/nav.php';
?>
<!-- End of Navigation bar -->
<!-- Header -->

<header class="bg-white shadow">
    <div class="mx-auto max-w-7xl px-4 py-6 sm:px-6 lg:px-8">
        <h1 class="text-3xl font-bold tracking-tight text-gray-900">Τίτλος - <?= $poleID . "#" . $heading?> </h1>
    </div>
</header>
<!-- End of Header -->
<!-- Main -->
<main>
    <main>
        <div class="mx-auto max-w-7xl py-6 sm:px-6 lg:px-8">
            <h2 class="text-lg font-semibold mb-4">Gauges Alarm</h2>
            <div id="chart_Alarms" style="width: 400px; height:
120px;"></div>

            <h2 class="text-lg font-semibold mb-4 mt-8">Gauges
Sensor</h2>
            <div id="chart_Sensors" style="width: 800px; height:
120px;"></div>

            <h2 class="text-lg font-semibold mb-4 mt-8">Test Sensor</h2>
            <div id="test" style="width: 800px; height: 120px;"></div>
        </div>
    </main>
    <?php
    if (!isAuthenticated()) {
        ?>
        <div class="mx-auto max-w-7xl py-2 sm:px-6 lg:px-8 flex justify-
center space-x-8">
            <a href="login.php" >

```

```

        <button class="bg-blue-500 hover:bg-blue-700 text-white
font-bold py-2 px-4 rounded">Είσοδος</button>
        </a>
        <a href="register.php" >
            <button class="bg-blue-500 hover:bg-blue-700 text-white
font-bold py-2 px-4 rounded">Εγγραφή</button>
            </a>
        </div>
        <?php
    }
    ?>
</main>
<!-- End of Main --&gt;
<!-- Footer --&gt;
&lt;?php include 'partials/footer.php';?&gt;
<!-- End of footer --&gt;

&lt;/body&gt;
&lt;/div&gt;

&lt;?php

require_once 'providers/auth_provider.php';
require_once 'providers/DB.php';
include 'partials/head.php';

$poleID = isset($_GET['id']) ? $_GET['id'] : null;

$data = require_once 'providers/poleSensorDetails.php';
?&gt;

&lt;body class="min-h-[100vh] h-full flex flex-col"&gt;
<!-- Navigation bar --&gt;
&lt;?php
include 'partials/nav.php';
?&gt;
<!-- End of Navigation bar --&gt;
<!-- Header --&gt;

&lt;script src="https://code.jquery.com/jquery-3.6.0.min.js"&gt;&lt;/script&gt;
&lt;script&gt;
    // Function to update table data
    function updateTableData(colName) {
        // Make AJAX call to retrieve updated data

        $.ajax({
            url: 'providers/updateSensorValues.php?id=&lt;?= $poleID
?&gt;&amp;colName=' + colName,
            type: 'POST', // You can use GET or POST depending on your
preference
            dataType: 'json', // Specify that the response will be JSON
            success: function(response) {
                $('#table_' + colName).text(response);

                var currentClass = $('#led_' + colName).className;

                // Update LED indicator color based on the response
                value
            }
        })
    }
&lt;/script&gt;
</pre>

```

```

        if (response == 0) {
            $('#led_' + colName).removeClass().addClass("green
led");
        } else if (response == 1) {
            $('#led_' + colName).removeClass().addClass("red
led");
        }
    },
    error: function(xhr, status, error) {
        console.error(xhr.responseText);
    }
});
}

// Call the updateTableData function for each alarm and sensor data
every second
$(document).ready(function() {
<?php

// Update alarm data every second
foreach ($data['dataAlarm'] as $alarm) {
    ?>
    setInterval(function() {
        updateTableData("<?= $alarm['colName'] ?>");
    }, 1000);
    <?php
}
?>

<?php

// Update sensor data every 1200000 milliseconds (20 minutes)
foreach ($data['dataSensor'] as $sensor) {
    ?>
    setInterval(function() {
        updateTableData("<?= $sensor['colName'] ?>");
    }, 1200000);
    <?php
}
?>
});

</script>

<header class="bg-white shadow-lg mx-10 border border-gray-400 rounded-
3xl mt-10">
    <div class="mx-auto max-w-7xl px-4 py-6 sm:px-6 lg:px-8 text-
center">
        <h1 class="text-3xl font-bold tracking-tight text-gray-
900">Κολώνα στην οδό <?= $data['poleDetails']['address']?> </h1>
    </div>
</header>
<!-- End of Header --&gt;
<!-- Main --&gt;
&lt;main class="h-full p-10"&gt;
    &lt;div class="grid grid-rows-2 grid-cols-3 grid-flow-col gap-4"&gt;
        &lt;div class="col-span-1 row-span-2 self-center"&gt;</pre>

```

```

<div class="flex flex-col w-full p-5 border border-gray-400 rounded-2xl shadow-xl bg-white gap-4">
    <h2 class="text-2xl font-semibold">Στοιχεία Έξυπνης Κολώνας</h2>
    <p class="text-sm">Στην συγκεκριμένη σελίδα εμφανίζονται λεπτομέρειες για την κολώνα καθώς και συνεχής Live ενημέρωση από τους αισθητήρες και τα alarm που έχει συνδεμένα.</p>
    <div class="w-full h-full flex flex-col gap-2 divide-y">
        <div class="w-full py-2 flex items-center">
            <p class="w-1/2 font-semibold">ID Κολώνας</p>
            <p class="w-1/2"><?= $data['poleDetails']['poleID']; ?></p>
        </div>
        <div class="w-full py-2 flex items-center">
            <p class="w-1/2 font-semibold">Όνομα Κολώνας</p>
            <p class="w-1/2"><?= $data['poleDetails']['poleName']; ?></p>
        </div>
        <div class="w-full py-2 flex items-center">
            <p class="w-1/2 font-semibold">Δήμος</p>
            <p class="w-1/2"><?= $data['poleDetails']['municipality']; ?></p>
        </div>
        <div class="w-full py-2 flex items-center">
            <p class="w-1/2 font-semibold">Περιοχή</p>
            <p class="w-1/2"><?= $data['poleDetails']['area']; ?></p>
        </div>
        <div class="w-full py-2 flex items-center">
            <p class="w-1/2 font-semibold">Διεύθυνση</p>
            <p class="w-1/2"><?= $data['poleDetails']['address']; ?></p>
        </div>
        <div class="w-full py-2 flex items-center">
            <p class="w-1/2 font-semibold">Συντεταγμένες Τοποθεσίας</p>
            <p class="w-1/2"><?= $data['poleDetails']['lat'] . " - " . $data['poleDetails']['lng']; ?></p>
        </div>
    </div>
    <div class="col-span-2 row-span-2 flex flex-col gap-4">
        <div class="row-span-1">
            <div class="flex flex-col w-full p-5 border border-gray-400 rounded-2xl shadow-xl bg-white gap-4">
                <p class="text-center text-2xl font-semibold">Live Feed</p>
                <div>
                    
                </div>
            </div>
            <div class="row-span-2">
                <div class="flex flex-col w-full p-5 border border-gray-400 rounded-2xl shadow-xl bg-white gap-4">
                    <h2 class="text-2xl font-semibold">Δεδομένα</h2>

```

```

<div class="w-full h-full flex flex-col gap-2
divide-y">
    <?php foreach ($data['dataAlarm'] as $alarm) : ?>
        <?php if ($alarm['poleID'] == $poleID) : ?>
            <div class="w-full flex items-center">
                <p class="w-1/2 font-semibold"><?=&
$alarm['DisplayNameGr']; ?></p>
                <div class="green led" id="led_<?=&
$alarm['colName'] ?>"></div>
            </div>
            <?php endif; ?>
        <?php endforeach; ?>
        <?php foreach ($data['dataSensor'] as $sensor) : ?>
            <?php if ($sensor['poleID'] == $poleID) : ?>
                <div class="w-full py-2 flex items-
center">
                    <p class="w-1/2 font-semibold"><?=&
$sensor['DisplayNameGr']; ?></p>
                    <p class="w-1/2" id="table_<?=&
$sensor['colName'] ?>"><?= $sensor['value'];
                </div>
                <?php endif; ?>
            <?php endforeach; ?>
        </div>
        <!-- <div>
            <table class="w-full text-sm text-left rtl:text-
right text-gray-500 dark:text-gray-400">
                <tbody>
                    </tr>
                    <?php foreach ($data['dataAlarm'] as $alarm) : ?>
                        <?php if ($alarm['poleID'] == $poleID) : ?>
                            <tr class="bg-white border-b
dark:border-gray-700">
                                <th scope="row" class="px-6 py-4
font-medium text-gray-900 whitespace nowrap dark:text-white">
                                    <?=$alarm['DisplayNameGr']; ?>
                                </th>
                                <td class="px-6 py-4">
                                    <div class="green led"
id="led_<?=$alarm['colName'] ?>"></div>
                                </td>
                            </tr>
                            <?php endif; ?>
                        <?php endforeach; ?>
                    </tbody>
                </table>
            </div>
            <table class="w-full text-sm text-left rtl:text-
right text-gray-500 dark:text-gray-400">
                <tbody>
                    </tr>
                    <?php foreach ($data['dataSensor'] as $sensor) : ?>
                        <?php if ($sensor['poleID'] == $poleID) : ?>

```

```

<tr class="bg-white border-b
dark:border-gray-700">
    <th scope="row" class="px-6 py-4
font-medium text-gray-900 whitespace nowrap dark:text-white">
        <?= $sensor['DisplayNameGr']; ?>
    </th>
    <td class="px-6 py-3"><?=
$sensor['DisplayNameGr']; ?></td>
    <td class="px-6 py-3" id="table_<?=
$sensor['colName'] ?>"><?= $sensor['value']; ?></td>
    </>
    <?php endif; ?>
    <?php endforeach; ?>
    </tbody>
</table>
</div> -->
</div>
</div>
<?php
if (!isAuthenticated()) {
?
<div class="mx-auto max-w-7xl py-2 sm:px-6 lg:px-8 flex justify-
center space-x-8">
    <a href="login.php" >
        <button class="bg-cyan-900 hover:bg-cyan-800 text-white
font-bold py-2 px-4 rounded">Είσοδος</button>
    </a>
    <a href="register.php" >
        <button class="bg-cyan-900 hover:bg-cyan-800 text-white
font-bold py-2 px-4 rounded">Εγγραφή</button>
    </a>
</div>
<?php
}
?>
</main>
<!-- End of Main --&gt;
<!-- Footer --&gt;
&lt;?php include 'partials/footer.php'; ?&gt;
<!-- End of footer --&gt;

&lt;/body&gt;
&lt;/div&gt;

&lt;?php

require_once 'providers/auth_provider.php';
require_once 'providers/DB.php';
$mapData = require_once 'providers/map.php';
include 'partials/head.php'
?&gt;

&lt;script&gt;
google.charts.load('current', {
    'packages': ['map'],
    'mapsApiKey': '&lt;?= $mapData['key'] ?&gt;'
}) ;
</pre>

```



```
google.charts.setOnLoadCallback(drawMap);
function drawMap() {

    var mapData = <?= $mapData['mapData'] ?>;
    var data = google.visualization.arrayToDataTable(mapData);
    var options = {
        mapType: 'styledMap',
        zoomLevel: 14,
        showTooltip: true,
        showInfoWindow: true,
        useMapTypeControl: true,
        maps: {
            styledMap: {
                name: 'Styled Map',
                styles: [
                    {featureType: 'poi.attraction',
                     stylers: [{color: '#fce8b2'}]},
                    {featureType: 'road.highway',
                     stylers: [{hue: '#0277bd'}, {saturation: -50}]},
                    {featureType: 'road.highway',
                     elementType: 'labels.icon',
                     stylers: [{hue: '#000'}, {saturation: 100}, {lightness: 50}]},
                    {featureType: 'landscape',
                     stylers: [{hue: '#259b24'}, {saturation: 10}, {lightness: -22}]}
                ]
            }
        }
    };

    var map = new
    google.visualization.Map(document.getElementById('map_div'));

    google.visualization.events.addListener(map, 'select',
    function() {
        var selection = map.getSelection();
        if (selection.length > 0) {
            var item = selection[0];
            if (item.row != null) {
                // Get the ID of the selected row from the data
                table
                var poleID = data.getValue(item.row, 3); // Assuming
                the first column contains the ID
                // Redirect to poledetails.php with the selected ID
                window.location.href = 'poledetails.php?id=' +
                poleID;
            }
        }
    });

    map.draw(data, options);
};

</script>
```

```

<body class="min-h-[100vh] h-full flex flex-col">
    <!-- Navigation bar -->
    <?php
    include 'partials/nav.php';
    ?>
    <!-- End of Navigation bar -->
    <!-- Header -->

    <header class="bg-white shadow">
        <div class="mx-auto max-w-7xl px-4 py-6 sm:px-6 lg:px-8">
            <h1 class="text-3xl font-bold tracking-tight text-gray-900">Χάρτης δικτύου «Εξυπνων Κολώνων» Φωτισμού</h1>
        </div>
    </header>
    <!-- End of Header -->
    <!-- Main -->
    <main class="h-full flex justify-center items-center">
        <div id="map_div" class="w-3/4" style="height: 75%;"></div>
        <?php
        if (!isAuthenticated()) {
            ?>
            <div class="mx-auto max-w-7xl py-2 sm:px-6 lg:px-8 flex justify-center space-x-8">
                <a href="login.php" >
                    <button class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded">Είσοδος</button>
                </a>
                <a href="register.php" >
                    <button class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded">Εγγραφή</button>
                </a>
            </div>
            <?php
        }
        ?>
    </main>
    <!-- End of Main -->
    <!-- Footer -->
    <?php include 'partials/footer.php';?>
    <!-- End of footer -->

</body>
</html>

<?php

require_once 'providers/auth_provider.php';

logout();
header("Location: index.php");
exit();

<?php

require_once 'providers/auth_provider.php';
require_once 'providers/DB.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {

```

```

$username = $_POST['username'];
$password = $_POST['password'];

if (login($username, $password)) {
    header('Location: index.php');
    exit();
}
else {
    $error_message = "Invalid username or password";
}
}

if (isAuthenticated()) {
    header('Location: index.php');
    exit();
}
?>

<?php
require 'partials/head.php';
?>

<body class="h-[100vh] flex flex-col">
    <!-- Navigation bar -->
    <?php
    include 'partials/nav.php';
    ?>
    <!-- End of Navigation bar -->
    <!-- Main -->
    <main class="flex justify-center items-center h-full">
        <div class="w-full max-w-xs">
            <h1 class="text-3xl font-bold text-center mb-8">Είσοδος
Χρήστη</h1>
            <?php
            if (isset($error_message)) {
                ?>
                <div class="bg-red-100 border border-red-400 text-red-700 px-4 py-3 rounded relative" role="alert">
                    <span class="block sm:inline"><?php echo
$error_message; ?></span>
                </div>
            <?php
            }
            ?>
            <form class="space-y-4" action="#" method="POST">
                <div>
                    <label for="username" class="block text-sm font-medium text-gray-700">Όνομα Χρήστη</label>
                    <input type="text" id="username" name="username" required class="mt-1 p-2 w-full border rounded-md">
                </div>
                <div>
                    <label for="password" class="block text-sm font-medium text-gray-700">Κωδικός Πρόσβασης</label>
                    <input type="password" id="password" name="password" required class="mt-1 p-2 w-full border rounded-md">
                </div>
                <div>

```

```

        <button type="submit" class="bg-cyan-900 hover:bg-cyan-800 text-white font-bold py-2 px-4 rounded-lg w-full">Είσοδος</button>
    </div>
</form>
<p class="text-center mt-4">Δεν έχετε λογαριασμό; <a href="register.php" class="text-cyan-800 hover:underline">Εγγραφή νέου χρήστη</a></p>
</div>
</main>
<!-- End of Main -->
<!-- Footer -->
<?php
include 'partials/footer.php';
?>
<!-- End of footer -->
</body>
</html>

<?php

require_once 'providers/auth_provider.php';
require_once 'providers/DB.php';
include 'partials/head.php'
?>

<body class="h-[100vh] flex flex-col">
<!-- Navigation bar -->
<?php
include 'partials/nav.php';
?>
<!-- End of Navigation bar -->
<!-- Header -->

<header class="bg-white shadow">
    <div class="mx-auto max-w-7xl px-4 py-6 sm:px-6 lg:px-8">
        <h1 class="text-3xl font-bold tracking-tight text-gray-900"><?= $heading?> - Πλατφόρμα ThingSpeak</h1>
    </div>
</header>
<!-- End of Header -->
<!-- Main -->
<main class="h-full py-10">
    <div class="flex flex-wrap gap-10 mx-auto max-w-[1510px] px-10 justify-center">
        <iframe width="450" height="260" style="border: 1px solid #cccccc;" src="https://thingspeak.com/channels/2277540/charts/2?bgcolor=%23ffffff&color=%23d62020&dynamic=true&results=60&title=%CE%98%CE%B5%CF%81%CE%BC%C E%BF%CE%BA%CF%81%CE%B1%CF%83%CE%AF%CE%B1&type=line"></iframe>
        <iframe width="450" height="260" style="border: 1px solid #cccccc;" src="https://thingspeak.com/channels/2277540/charts/3?bgcolor=%23ffffff&color=%23d62020&dynamic=true&results=60&title=%CE%A5%CE%B3%CF%81%CE%B1%C F%83%CE%AF%CE%B1&type=line"></iframe>

```

```

<iframe width="450" height="260" style="border: 1px solid
#cccccc;" src="https://thingspeak.com/channels/2277540/charts/1?bgcolor=%23ffffffff&
color=%23d62020&dynamic=true&results=60&title=%CE%A6%CF%89%CF%84%CE%B5%C
E%B9%CE%BD%CF%8C%CF%84%CE%B7%CF%84%CE%B1&type=line"></iframe>
    <iframe width="450" height="260" style="border: 1px solid
#cccccc;" src="https://thingspeak.com/channels/2277540/charts/4?bgcolor=%23ffffffff&
color=%23d62020&dynamic=true&results=60&title=%CE%A0%CE%BF%CE%B9%CF%8C%C
F%84%CE%B7%CF%84%CE%B1+%CE%91%CE%AD%CF%81%CE%B1&type=line"></iframe>
        <iframe width="450" height="260" style="border: 1px solid
#cccccc;" src="https://thingspeak.com/channels/2277540/charts/6?bgcolor=%23ffffffff&
color=%23d62020&dynamic=true&results=60&title=%CE%95%CF%80%CE%B9%CE%BA%C
E%AF%CE%BD%CE%B4%CF%85%CE%BD%CE%B1+%CE%91%CE%AD%CF%81%CE%B9%CE%B1&type=l
ine"></iframe>
    </div>
    <?php
    if (!isAuthenticated()) {
        ?>
        <div class="mx-auto max-w-7xl py-2 sm:px-6 lg:px-8 flex justify-
center space-x-8">
            <a href="login.php" >
                <button class="bg-blue-500 hover:bg-blue-700 text-white
font-bold py-2 px-4 rounded">Είσοδος</button>
            </a>
            <a href="register.php" >
                <button class="bg-blue-500 hover:bg-blue-700 text-white
font-bold py-2 px-4 rounded">Εγγραφή</button>
            </a>
        </div>
        <?php
    }
    ?>
</main>
<!-- End of Main --&gt;
<!-- Footer --&gt;
&lt;?php include 'partials/footer.php';?&gt;
<!-- End of footer --&gt;

&lt;/body&gt;
&lt;/div&gt;

&lt;?php

global $conn;
require_once 'providers/auth_provider.php';
require_once 'providers/DB.php';
include 'partials/head.php';

// Navigation bar
$data = require_once 'providers/poleDetails.php';
?&gt;
&lt;script&gt;var customAlert = new CustomAlert();&lt;/script&gt;

&lt;?php
// TODO create a file for this
</pre>

```

```

if (isset($_POST['selection']) && isset($_POST['poleID'])) { // Αν γίνεται τημα
    $selected_measurements = $_POST['selection']; // Αποθήκευση πίνακα επιλογής μετρήσεων που θα προβληθούν
    $poleIDs = $_POST['poleID']; // Αποθήκευση πίνακα επιλογής id που θα προβληθούν
    $dateFrom = mysqli_real_escape_string($conn, $_POST['dateFrom']); // Αποθήκευση ημερομηνίας αρχής
    $dateEnd = mysqli_real_escape_string($conn, $_POST['dateEnd']); // Αποθήκευση ημερομηνίας τέλους
}
else { // Αν δεν υπάρχει αίτημα τότε αρχικοποίηση στις παρακάτω τιμές
    $selected_measurements = "temperature";
    $dateFrom = mysqli_real_escape_string($conn, date("Y-01-01"));
    $dateEnd = mysqli_real_escape_string($conn, date("Y-m-d"));
    $poleIDs = [4, 5];
} // Ανάκτηση δεδομένων από την βάση, του διαστήματος και των πεδίων που έχουν επιλεγεί

$sql_query = "SELECT DATE(`timestamp`) as Date, ";
foreach ($poleIDs as $poleID) {
    $sql_query .= "AVG(CASE WHEN poleID = $poleID THEN $selected_measurements END) as '$poleID' , ";
}
$sql_query = rtrim($sql_query, ", ");
$sql_query .= " FROM sensor_values ";
$where = " WHERE $selected_measurements is NOT NULL AND ";
$where .= "(timestamp >= '$dateFrom' AND timestamp <= '$dateEnd') and poleID in ";
$poleID = "(";
foreach ($poleIDs as $item) {
    $poleID .= $item . ",";
}
$sql_query .= $where;
$poleID = rtrim($poleID, ", ");
$poleID .= ")";
$sql_query .= $poleID . " GROUP BY DATE(`timestamp`) ORDER BY Date ASC;";

// dd($sql_query);
$result = mysqli_query($conn, $sql_query);
// dd($poleIDs);
// Check if $result is null
if ($result->num_rows == 0) {
    // Output a JavaScript alert message
    echo '<script>
        customAlert.alert("Δεν βρέθηκαν καταχωρημένα στοιχεία για αυτές τις επιλογές.", "Κάτι πήγε στραβά...");
    </script>';
}
?>

<body class="min-h-[100vh] h-full flex flex-col">
<?php include 'partials/nav.php'; ?>

<header class="bg-white shadow">
    <div class="mx-auto max-w-7xl px-4 py-6 sm:px-6 lg:px-8">

```

```

<h1 class="text-3xl font-bold tracking-tight text-gray-900"><?= $heading?> </h1>
</div>
</header>
<!-- End of Header -->
<!-- Main -->
<main class="h-full">
    <div class="mx-auto max-w-7xl py-4 sm:px-6 lg:px-8" style="display: flex;">
        <div style="flex">
            <form name="line_chart" method="post" id="line_chart_form">
                <h2>Ημερομηνία Αρχής:</h2>
                <input style="font-size: 16px;" type="date" id="dateFrom" name="dateFrom" value=<?= !empty($_POST['dateFrom']) ? $_POST['dateFrom'] : date("Y") . "-01-01"; ?>">
                <h2>Ημερομηνία Τέλους:</h2>
                <input style="font-size: 16px;" type="date" id="dateEnd" name="dateEnd" value=<?= date("Y-m-d"); ?>">
                <br><br>
                <h2>Επιλογή Κολώνας</h2>
                <?php foreach ($data as $pole) {
                    ?>
                    <input type="checkbox" id="poleID" name="poleID[]" value=<?php echo $pole['poleID']; ?>">
                    <!-- Area and Street Name -->
                    <label for="poleID"><?= $pole['area'] . " - " . $pole['address']; ?></label><br>
                    <?php
                }
                ?>
                <!-- TODO dynamic sensors list -->
                <br>
                <h2>Επιλογή Αισθητήρα</h2>
                <input type="radio" id="temperature" name="selection" value="temperature">
                    <label for="temperature">Θερμοκρασία</label><br>
                    <input type="radio" id="noiseDB" name="selection" value="noiseDB">
                        <label for="noiseDB">Επίπεδα Θορύβου</label><br>
                        <input type="radio" id="humidity" name="selection" value="humidity">
                            <label for="humidity">Υγρασία</label><br>
                            <input type="radio" id="rainHeight" name="selection" value="rainHeight">
                                <label for="rainHeight">Επίπεδο βροχόπτωσης</label><br>
                                <input type="radio" id="ldr" name="selection" value="ldr">
                                    <label for="ldr">Επίπεδο Ηλιακής φωτεινότητας</label><br>
                                    <input type="radio" id="carbonMono" name="selection" value="carbonMono">
                                        <label for="carbonMono">Μονοξείδιο του Ανθρακα</label><br>
                                        <input type="radio" id="AirQuality" name="selection" value="AirQuality">
                                            <label for="AirQuality">Ποιότητα Αέρα</label><br>
                                            <input type="radio" id="uv" name="selection" value="uv">
                                                <label for="uv">Δείκτης UV ακτινοβολίας</label><br>

```

```

<input type="radio" id="pressure" name="selection"
value="pressure">
    <label for="pressure">Ατμοσφαιρική Πίεση</label><br>
    <br><button class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded-lg w-full" type="submit"> Υποβολή </button>
    </form>
</div>
<div style="flex">
    <script>
        google.charts.load('current', {
            'packages': ['corechart']
        });
        google.charts.setOnLoadCallback(comparativeChart);

        function comparativeChart() { //Η συνάρτηση κατασκευής
φραφήματος
            var data = new google.visualization.DataTable();
            data.addColumn('string', 'Date');
            <?php // Στήλες του πίνακα δεδομένων του
γραφήματος βάσει επιλογών
                foreach ($poleIDs as $poleID) {
                    $input_label = "";
                    foreach ($data as $item) {
                        if ($item['poleID'] == $poleID) {
                            $input_label = $item['area'] . " - " .
$item['address'];
                        }
                    }
                    echo "data.addColumn('number',
'$input_label');\n";
                }
            ?>
            data.addRows([
                <?php
                while ($row = mysqli_fetch_array($result)) {
                    echo "[{$row['Date']}], ";
                    foreach ($poleIDs as $poleID) {
                        echo "{$row[$poleID]}, ";
                    }
                    echo "],\n";
                }
            ?>
        ]);
    }

    var options = { //Μορφοποίηση εμφάνισης γραφήματος
        title: '',
        hAxis: {title: ''},
        vAxis: {title: ''},
    };
    //Εμφάνιση του γραφήματος μέσα στο στοιχείο <div>
    var chart = new
google.visualization.LineChart(document.getElementById('div_chart'));
    chart.draw(data, options);
}

</script>

```

```

        <!-- Chart area -->
        <div id="div_chart" style="width: 900px; height:
500px"></div>
    </div>
<?php
if (!isAuthenticated()) {
?
<div class="mx-auto max-w-7xl py-2 sm:px-6 lg:px-8 flex justify-
center space-x-8">
    <a href="login.php" >
        <button class="bg-cyan-900 hover:bg-cyan-800 text-white
font-bold py-2 px-4 rounded">Είσοδος</button>
    </a>
    <a href="register.php" >
        <button class="bg-cyan-900 hover:bg-cyan-800 text-white
font-bold py-2 px-4 rounded">Εγγραφή</button>
    </a>
</div>
<?php
}
?
</main>
<!-- End of Main -->
<!-- Footer -->
<?php include 'partials/footer.php';?>
<!-- End of footer -->

</body>
</div>

<?php

require_once 'providers/auth_provider.php';
require_once 'providers/DB.php';
include 'partials/head.php';
$data = require_once 'providers/poleDetails.php';
?>

<body class="h-[100vh] flex flex-col">
<!-- Navigation bar -->
<?php
include 'partials/nav.php';
?>
<!-- End of Navigation bar -->
<!-- Header -->

<header class="bg-white shadow">
    <div class="mx-auto max-w-7xl px-4 py-6 sm:px-6 lg:px-8">
        <h1 class="text-3xl font-bold tracking-tight text-gray-
900">Προβολή και Διαχείριση Live Video από Κάμερες</h1>
    </div>
</header>
<!-- End of Header -->
<!-- Main -->
<main class="h-full">
    <div class="mx-auto max-w-7xl py-6 sm:px-6 lg:px-8 flex gap-5">
        <!-- Left side div for checkbox selection -->

```

```

<div>
    <div class="flex flex-col p-5 border border-gray-400
rounded-2xl shadow-xl bg-white gap-4">
        <h2 class="text-lg font-semibold mb-4">Επιλογή Έξυπνης
Κολώνας</h2>
        <?php foreach ($data as $camera) : ?>
            <div>
                <input type="checkbox" id="camera_<?php echo
$camera['poleID']; ?>" value="<?php echo $camera['poleID']; ?>"
class="camera-checkbox">
                    <label for="camera_<?php echo $camera['poleID'];
?>"><?php echo $camera['area'] . " - " . $camera['address']; ?></label>
                </div>
            <?php endforeach; ?>
            <br><button id="submitBtn" class="bg-cyan-900 hover:bg-
cyan-800 text-white font-bold py-2 px-4 rounded-lg w-full" type="submit"
value="Submit"> Υποβολή </button>
        </div>
    </div>

    <!-- Right side div for displaying selected elements in a grid
layout -->
    <div class="flex flex-col items-center w-full gap-4"
id="selectedCamerasGrid">
        <!-- Selected cameras will be displayed here -->
    </div>
</div>
</main>

<script>
    document.getElementById('submitBtn').addEventListener('click',
function() {
    // Clear previous entries in the grid
    document.getElementById('selectedCamerasGrid').innerHTML =
'<h2 class="bg-white shadow-lg border border-gray-400 rounded-2xl w-full
text-2xl py-5 text-center font-semibold">Προβολή Επιλογών</h2><div
class="flex gap-4 flex-wrap w-full justify-center" id="camera-
grid"></div>';

    // Get all checked checkboxes
    var checkboxes = document.getElementsByClassName('camera-
checkbox');
    var count = 0;
    for (var i = 0; i < checkboxes.length; i++) {
        if (checkboxes[i].checked) {
            // Get the corresponding camera data
            var cameraId = checkboxes[i].value;
            var camera = <?php echo json_encode($data);
?>.find(function(camera) {
                return camera.poleID == cameraId;
            });

            // Create a div for each selected camera
            var newDiv = document.createElement('div');
            newDiv.className = 'flex flex-col p-5 border border-
gray-400 rounded-2xl shadow-xl bg-white gap-4';
            newDiv.innerHTML =
`
```

```

<label class="text-lg font-semibold text-center">${camera.area} - ${camera.address}</label>
    
        <!-- TODO the button functionality is for superAdmin
-->
        <button type="button" formtarget="_blank" class="bg-cyan-900 hover:bg-cyan-800 text-white font-bold py-2 px-4 rounded-lg w-full" onclick="location.href='${camera.cameraOptions}'">Camera
Options</button>
    `;
    document.getElementById('camera-grid').appendChild(newDiv);
    count++;
}
)
);
</script>
</div>
<?php
if (!isAuthenticated()) {
?
<div class="mx-auto max-w-7xl py-2 sm:px-6 lg:px-8 flex justify-center space-x-8">
    <a href="login.php" >
        <button class="bg-cyan-900 hover:bg-cyan-800 text-white font-bold py-2 px-4 rounded">Είσοδος</button>
    </a>
    <a href="register.php" >
        <button class="bg-cyan-900 hover:bg-cyan-800 text-white font-bold py-2 px-4 rounded">Εγγραφή</button>
    </a>
</div>
<?php
}
?
<!-- End of Main -->
<!-- Footer -->
<?php include 'partials/footer.php';?>
<!-- End of footer -->
</body>

<?php
require_once 'providers/auth_provider.php';
require_once 'providers/DB.php';
include 'partials/head.php';
$abstract = require_once 'partials/abstract.php';
?>

<body class="h-full min-h-full flex flex-col">
<!-- Navigation bar -->
<?php
include 'partials/nav.php';
?>
<!-- End of Navigation bar -->
<!-- Header -->

```

```

<header class="bg-white shadow">
    <div class="mx-auto max-w-7xl px-4 py-6 sm:px-6 lg:px-8">
        <h1 class="text-3xl font-bold tracking-tight text-gray-900"><?= $headingGR ?> </h1>
    </div>
</header>
<!-- End of Header -->
<!-- Main -->
<main class="h-full">
    <div class="mx-auto max-w-7xl py-6 sm:px-6 lg:px-8">
        <div>
            <?= $abstract ?>
        </div>
    </div>
    <?php
    if (!isAuthenticated()) {
        ?>
        <div class="mx-auto max-w-7xl py-2 sm:px-6 lg:px-8 flex justify-
center space-x-8">
            <a href="login.php" >
                <button class="bg-blue-500 hover:bg-blue-700 text-white
font-bold py-2 px-4 rounded">Είσοδος</button>
            </a>
            <a href="register.php" >
                <button class="bg-blue-500 hover:bg-blue-700 text-white
font-bold py-2 px-4 rounded">Εγγραφή</button>
            </a>
        </div>
    <?php
    }
    ?>
</main>
<!-- End of Main -->
<!-- Footer -->
<?php include 'partials/footer.php';?>
<!-- End of footer -->

</body>
</div>

```

ΣΤ.3 Providers

```

<?php

require '../functions/functions.php';
require '../db_config.php';

$poleID = isset($_GET['id']) ? $_GET['id'] : null;
$label = isset($_GET['colName']) ? $_GET['colName'] : null;

global $conn;

$queryLatest = "SELECT {$label}
    FROM sensor_values

```

```

WHERE {$label} IS NOT NULL AND poleID = {$poleID}
ORDER BY timestamp DESC
LIMIT 1";

$resultLatest = mysqli_query($conn, $queryLatest);

$value = 0;

$row = mysqli_fetch_row($resultLatest);
if ($row != null)
    $value = $row[0];

echo json_encode($value);

<?php

global $conn;

$queryAlarm = "select p.poleID as poleID,
                    p.pole_name as poleName,
                    s.sensorName as label,
                    p.area as area,
                    s.isAlarm as alarm
               from PoleSensors ps
              inner join Pole p on p.poleID = ps.poleID
              inner join Sensor s on s.sensorID = ps.sensorID
             where s.isAlarm = 1";

$querySensor = "select p.poleID as poleID,
                    p.pole_name as poleName,
                    s.sensorName as label,
                    p.area as area,
                    s.isAlarm as alarm
               from PoleSensors ps
              inner join Pole p on p.poleID = ps.poleID
              inner join Sensor s on s.sensorID = ps.sensorID
             where s.isAlarm = 0";

$resultAlarm = mysqli_query($conn, $queryAlarm);
$resultSensor = mysqli_query($conn, $querySensor);

// Store the fetched data in an array
$dataAlarm = array();
$dataSensor = array();

while ($row = mysqli_fetch_assoc($resultAlarm)) {
    $dataAlarm[] = $row;
}

while ($row = mysqli_fetch_assoc($resultSensor)) {
    $dataSensor[] = $row;
}

$data = [
    "dataAlarm" => $dataAlarm,
    "dataSensor" => $dataSensor
];

```

```

return $data;

<?php

$currentPole = [];

$pole = require_once 'providers/poleDetails.php';

foreach ($pole as $item) {
    if ($item[ 'poleID' ] == $poleID) {
        $currentPole = $item;
        break;
    }
}

global $conn;

$queryAlarm = "select      p.poleID as poleID,
                           p.pole_name as poleName,
                           s.sensorName as label,
                           s.Abbreviation as colName,
                           s.DisplayNameGr as DisplayNameGr,
                           s.UoMSymbol as UoM,
                           p.area as area,
                           s.isAlarm as alarm,
                           s.thingSpeak as iot
                  from PoleSensors ps
                 inner join Pole p on p.poleID = ps.poleID
                 inner join Sensor s on s.sensorID = ps.sensorID
                where s.isAlarm = 1 and p.poleID = {$poleID};";

$querySensor = "select      p.poleID as poleID,
                           p.pole_name as poleName,
                           s.sensorName as label,
                           s.Abbreviation as colName,
                           s.DisplayNameGr as DisplayNameGr,
                           s.UoMSymbol as UoM,
                           p.area as area,
                           s.isAlarm as alarm,
                           s.rangeTo as rangeTo
                  from PoleSensors ps
                 inner join Pole p on p.poleID = ps.poleID
                 inner join Sensor s on s.sensorID = ps.sensorID
                where s.isAlarm = 0 and p.poleID = {$poleID}";

$resultAlarm = mysqli_query($conn, $queryAlarm);
$resultSensor = mysqli_query($conn, $querySensor);

// Store the fetched data in an array
$dataAlarm = array();
$dataSensor = array();

while ($row = mysqli_fetch_assoc($resultAlarm)) {
    $dataAlarm[] = $row;
}

```

```

while ($row = mysqli_fetch_assoc($resultSensor)) {
    $dataSensor[] = $row;
}

$data = [
    "poleDetails" => $currentPole,
    "dataAlarm" => $dataAlarm,
    "dataSensor" => $dataSensor
];

//dd($data["dataAlarm"]);

foreach ($data["dataAlarm"] as $key => $value) {
    $label = $value['colName'];
    $poleId = $value['poleID'];
    $queryLatest = "SELECT {$label}
                    FROM sensor_values
                    WHERE {$label} IS NOT NULL AND
                          poleID = {$poleId}
                    ORDER BY timestamp DESC
                    LIMIT 1";
    $resultLatest = mysqli_query($conn, $queryLatest);
    $row = mysqli_fetch_row($resultLatest);
    if ($row == null){
        $insertValue = 0;
    } else {
        $insertValue = $row[0];
    }
    $data["dataAlarm"][$key]["value"] = $insertValue;
}

foreach ($data["dataSensor"] as $key => $value) {
    $label = $value['colName'];
    $poleId = $value['poleID'];
    $queryLatest = "SELECT {$label}
                    FROM sensor_values
                    WHERE {$label} IS NOT NULL AND
                          poleID = {$poleId}
                    ORDER BY timestamp DESC
                    LIMIT 1";
    //dd($queryLatest);
    $resultLatest = mysqli_query($conn, $queryLatest);
    $row = mysqli_fetch_row($resultLatest);
    if ($row == null){
        $insertValue = 0;
    } else {
        $insertValue = $row[0];
    }
    $data["dataSensor"][$key]["value"] = $insertValue;
}

return $data;
<?php

```

```

global $conn;

$query = "select      poleID as poleID,
                     pole_name as poleName,
                     area as area,
                     Municipality as municipality,
                     controller as controller,
                     CONCAT(streetName, ' ', COALESCE(streetNumber, ''))

AS address,
                     lat,
                     lng,
                     CONCAT('http://',cameraIP, ':', port, '/stream') as
cameraIP,
                     CONCAT('http://',cameraIP) AS cameraOptions
from Pole where isVisible = 1 order by area;";

$result = mysqli_query($conn, $query);

$data = array();

while ($row = mysqli_fetch_assoc($result)) {
    $data[] = [
        "poleID"=> $row[ 'poleID' ],
        "poleName" => $row[ 'poleName' ],
        "area" => $row[ 'area' ],
        "municipality" => $row[ 'municipality' ],
        "controller" => $row[ 'controller' ],
        "address" => $row[ 'address' ],
        "lat" => $row[ 'lat' ],
        "lng" => $row[ 'lng' ],
        "cameraIP" => $row['cameraIP'],
        "cameraOptions" => $row['cameraOptions']
    ];
}

return $data;
}

<?php

$mapsApiKey = require_once 'providers/api.env';
global $conn;
$query = "SELECT lat, lng, CONCAT(streetName, ' ',
COALESCE(streetNumber, '')) AS address, poleID  FROM Pole where
isVisible = 1";
$result = mysqli_query($conn, $query);

// Store the fetched data in an array
$data = array();
$data[] = ['Lat', 'Long', 'Name', 'poleID'];
while ($row = mysqli_fetch_assoc($result)) {
    // Convert lat and lng values to floats
    $lat = floatval($row['lat']);
    $lng = floatval($row['lng']);
    // Add converted values to $data array
    $data[] = array($lat, $lng, $row['address'], $row['poleID']);
}
// Encode the data as JSON

```

```
$json_data = json_encode($data);

$maps = [
    "key" => $mapsApiKey,
    "mapData" => $json_data
];

//dd($json_data);
return $maps;

<?php

session_start();

require_once 'db_config.php';

function login($username, $password)
{
    global $conn;

    $query = "SELECT * FROM users WHERE username = '$username' AND
password = '$password'";
    $result = $conn->query($query);

    if ($result->num_rows == 1) {
        // Fetch the user data
        $user_data = $result->fetch_assoc();

        // Store user data in session variables
        $_SESSION['user_id'] = $user_data['user_id'];
        $_SESSION['authenticated'] = true;
        $_SESSION['username'] = $username;
        $_SESSION['role_id'] = $user_data['role_id'];
        return true;
    }
    else {
        return false;
    }
}

function logout()
{
    session_unset();
    session_destroy();
}

function isAuthenticated()
{
    return isset($_SESSION['authenticated']) &&
$_SESSION['authenticated'] === true;
}
```

ΣΤ.4 Partials

```
// the navigation bar
```

```

<header>
    <nav class="bg-cyan-900">
        <div class="mx-auto max-w-7xl px-4 sm:px-6 lg:px-8">
            <div class="flex h-16 items-center justify-center">
                <div class="flex items-center">
                    <div class="flex-shrink-0">
                        <a href="./"></a>
                    </div>
                    <div class="ml-10 flex items-baseline space-x-4">
                        <!-- Current: "bg-gray-900 text-white",  

                        Default: "text-gray-300 hover:bg-cyan-800 hover:text-white" -->
                        <a href="./" class="text-gray-300 hover:bg-cyan-800 hover:text-white rounded-md px-3 py-2 text-sm font-medium">Αρχική Σελίδα</a>
                        <?php
                            if (isset($_SESSION['username'])) {
                                switch ($_SESSION['role_id']) {
                                    // user (χρήστης πλατφόρμας από το
                                    Δήμο
                                    case 1:
                                        echo '<a href="map.php" class="text-gray-300 hover:bg-cyan-800 hover:text-white rounded-md px-3 py-2 text-sm font-medium">Χάρτης</a>';
                                        // echo '<a href="statistics.php" class="text-gray-300 hover:bg-cyan-800 hover:text-white rounded-md px-3 py-2 text-sm font-medium">Στατιστικά</a>';
                                        echo '<div class="relative">
                                                <a href="#" class="text-gray-300 hover:bg-cyan-800 hover:text-white rounded-md px-3 py-2 text-sm font-medium" id="user-menu-button">Στατιστικά</a>
                                            <!-- Dropdown menu -->
                                            <div class="absolute right-0 z-10 mt-2 w-48 origin-top-right rounded-md bg-white py-1 shadow-lg ring-1 ring-black ring-opacity-5 focus:outline-none hidden"
                                                role="menu" aria-orientation="vertical" aria-labelledby="user-menu-button" tabindex="-1">
                                                <!-- Active: "bg-gray-100", Not Active: "" -->
                                                <a href="statistics.php" class="block px-4 py-2 text-sm text-cyan-800" role="menuitem" tabindex="-1">Στατιστικά Κολώνας</a>
                                                <a href="comparativeStatistics.php" class="block px-4 py-2 text-sm text-cyan-800" role="menuitem" tabindex="-1">Συγκρητικά Στατιστικά</a>
                                            </div>
                                        </div>';
                                        echo '<a href="camera.php" class="text-gray-300 hover:bg-cyan-800 hover:text-white rounded-md px-3 py-2 text-sm font-medium">Camera</a>';
                                        echo '<a href="IoT.php" class="text-gray-300 hover:bg-cyan-800 hover:text-white rounded-md px-3 py-2 text-sm font-medium">IoT</a>';
                            }
                        
```

```

echo '<a href="about.php"
class="text-gray-300 hover:bg-cyan-800 hover:text-white rounded-md px-3
py-2 text-sm font-medium">About</a>';
break;

// TODO extra capabilities for
super Admin
// platform admin
case 2:
echo'<a href="#" class="text-
gray-300 hover:bg-cyan-800 hover:text-white rounded-md px-3 py-2 text-sm
font-medium">Καταχώριση Κολώνας</a>';
echo'<a href="#" class="text-
gray-300 hover:bg-cyan-800 hover:text-white rounded-md px-3 py-2 text-sm
font-medium">Καταχώριση Χρήστη</a>';
break;

// για τους guest users
default:
echo '<a href="map.php"
class="text-gray-300 hover:bg-cyan-800 hover:text-white rounded-md px-3
py-2 text-sm font-medium">Χάρτης</a>';
}
?>
<a href="logout.php" class="text-gray-
300 hover:bg-cyan-800 hover:text-white rounded-md px-3 py-2 text-sm
font-medium">Αποσύνδεση</a>
<?php
}
else {
?>
<a href="login.php" class="text-gray-300
hover:bg-cyan-800 hover:text-white rounded-md px-3 py-2 text-sm font-
medium">Είσοδος</a>
<?php
}
?>
</div>
</div>
</div>
</div>
</div>
</nav>
</header>

<script>
document.getElementById('user-menu-
button').addEventListener('click', function () {
    var menu = document.querySelector('.relative .absolute');
    menu.classList.toggle('hidden');
});
</script>

// the head

<!DOCTYPE html>
<html lang="en" class="h-full bg-gray-100">
<head>

```

```

<meta charset="UTF-8">
<title><?= $heading?> - SmartCity</title>
<script src="https://cdn.tailwindcss.com/"></script>
<script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
<link rel="stylesheet" href="css/custom_alert.css">
<link rel="stylesheet" href="css/custom_styles.css">
<script src="functions/customAlert.js"></script>
<script src="functions/functions.js"></script>
</head>

// the footer

<footer class="bg-cyan-900">
    <div class="mx-auto max-w-7xl px-4 sm:px-6 lg:px-8">
        <div class="flex h-16 items-center justify-center">
            <div class="ml-10 flex items-baseline space-x-4">
                <!-- Current: "bg-gray-900 text-white", Default: "text-
gray-300 hover:bg-cyan-800 hover:text-white" -->
                <a href=".partials/pdf/terms.pdf" target="_blank"
class="text-gray-300 hover:bg-cyan-800 hover:text-white rounded-md px-3
py-2 text-sm font-medium">Όποι ο χρήσης</a>
                <a href=".partials/pdf/gdpr.pdf" target="_blank"
class="text-gray-300 hover:bg-cyan-800 hover:text-white rounded-md px-3
py-2 text-sm font-medium">Πολιτική απορρήτου</a>
            </div>
        </div>
    </div>
</footer>

// the banner

<head>
    <meta charset="UTF-8">
    <title><?= $heading ?></title>
    <script src="https://cdn.tailwindcss.com"></script>
</head>

```



ΕΛΛΗΝΙΚΟ
ΑΝΟΙΚΤΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ

Κυριακίδης Αριστοκλής

*Σχεδιασμός και ανάπτυξη
«Έξυπνης Κολώνας» δημόσιου
φωτισμού σε περιβάλλον μίας
έξυπνης πόλης*

Παράρτημα Ζ: «Διαδικτυακή Κάμερα – ESP32»

```
*****  
Author: Kyriakidis Aris  
email: a.kyriakidis@hotmail.com  
*****
```

```
#include "esp_camera.h"  
#include <WiFi.h>  
#define CAMERA_MODEL_AI_THINKER // Has PSRAM  
#include "camera_pins.h"  
  
// -----  
// Enter your WiFi credentials  
// -----  
const char* ssid = "";  
const char* password = "";  
  
void startCameraServer();  
void setupLedFlash(int pin);  
  
void setup() {  
    Serial.begin(115200);  
    Serial.setDebugOutput(true);  
    Serial.println();  
  
    camera_config_t config;  
    config.ledc_channel = LEDC_CHANNEL_0;  
    config.ledc_timer = LEDC_TIMER_0;  
    config.pin_d0 = Y2_GPIO_NUM;  
    config.pin_d1 = Y3_GPIO_NUM;  
    config.pin_d2 = Y4_GPIO_NUM;  
    config.pin_d3 = Y5_GPIO_NUM;  
    config.pin_d4 = Y6_GPIO_NUM;  
    config.pin_d5 = Y7_GPIO_NUM;  
    config.pin_d6 = Y8_GPIO_NUM;  
    config.pin_d7 = Y9_GPIO_NUM;  
    config.pin_xclk = XCLK_GPIO_NUM;  
    config.pin_pclk = PCLK_GPIO_NUM;  
    config.pin_vsync = VSYNC_GPIO_NUM;  
    config.pin_href = HREF_GPIO_NUM;  
    config.pin_sccb_sda = SIOD_GPIO_NUM;  
    config.pin_sccb_scl = SIOC_GPIO_NUM;  
    config.pin_pwdn = PWDN_GPIO_NUM;  
    config.pin_reset = RESET_GPIO_NUM;  
    config.xclk_freq_hz = 20000000;
```

```

config.frame_size = FRAMESIZE_UXGA;
config.pixel_format = PIXFORMAT_JPEG; // for streaming
//config.pixel_format = PIXFORMAT_RGB565; // for face detection/recognition
config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
config.fb_location = CAMERA_FB_IN_PSRAM;
config.jpeg_quality = 12;
config.fb_count = 1;

// if PSRAM IC present, init with UXGA resolution and higher JPEG quality
//                                for larger pre-allocated frame buffer.
if(config.pixel_format == PIXFORMAT_JPEG){
    if(psramFound()){
        config.jpeg_quality = 10;
        config.fb_count = 2;
        config.grab_mode = CAMERA_GRAB_LATEST;
    } else {
        // Limit the frame size when PSRAM is not available
        config.frame_size = FRAMESIZE_SVGA;
        config.fb_location = CAMERA_FB_IN_DRAM;
    }
} else {
    // Best option for face detection/recognition
    config.frame_size = FRAMESIZE_240X240;
}

#if CONFIG_IDF_TARGET_ESP32S3
    config.fb_count = 2;
#endif
}

#if defined(CAMERA_MODEL_ESP_EYE)
    pinMode(13, INPUT_PULLUP);
    pinMode(14, INPUT_PULLUP);
#endif

// camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
}

sensor_t * s = esp_camera_sensor_get();
// initial sensors are flipped vertically and colors are a bit saturated
if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
}
// drop down frame size for higher initial frame rate
if(config.pixel_format == PIXFORMAT_JPEG){

```

```

    s->set_framesize(s, FRAMESIZE_QVGA);

}

#if defined(CAMERA_MODEL_M5STACK_WIDE) || defined(CAMERA_MODEL_M5STACK_ESP32CAM)
    s->set_vflip(s, 1);
    s->set_hmirror(s, 1);
#endif

#if defined(CAMERA_MODEL_ESP32S3_EYE)
    s->set_vflip(s, 1);
#endif

// Setup LED Flash if LED pin is defined in camera_pins.h
#if defined(LED_GPIO_NUM)
    setupLedFlash(LED_GPIO_NUM);
#endif

WiFi.begin(ssid, password);
WiFi.setSleep(false);

while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

startCameraServer();

Serial.print("Camera Ready! Use 'http://");
Serial.print(WiFi.localIP());
Serial.println("' to connect");
}

void loop() {
    // Do nothing. Everything is done in another task by the web server
    delay(10000);
}

```



Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν.1599/1986, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης.