ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΥΣΤΗΜΑ ΑΣΦΑΛΕΙΑΣ ΚΑΙ ΣΥΝΑΓΕΡΜΟΥ ΜΕ ΤΗΝ ΧΡΗΣΗ ARDUINO

Δημήτριος Χαρπαντίδης

Επιβλέπων: Ευριπίδης Γλαβάς

Καθηγητής ΔΕΠ

Άρτα, Φεβρουάριος, 2023

SECURITY AND ALARM SYSTEM USING ARDUINO

Εγκρίθηκε από τριμελή εξεταστική επιτροπή

Τόπος, Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Επιβλέπων καθηγητής

   Ευριπίδης Γλαβας,

2. Μέλος επιτροπής

   Νικόλαος Γιαννακέας,

3. Μέλος επιτροπής

   Αλέξανδρος Τζάλας,

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα μεταπτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για τη συγγραφή της περιλαμβάνονται στη βιβλιογραφία.

Επίθετο, Όνομα

Υπογραφή

ΠΕΡΙΛΗΨΗ

Η πτυχιακή εργασία αυτή θα επικεντρωθεί στην υλοποίηση ενός πρότυπου συστήματος ασφαλείας και συναγερμού οικίας, με τη χρήση του μικροελεγκτή ESP8266 και της ESP32-CAM. Το σύστημα θα περιλαμβάνει μια κάμερα ασφαλείας, έναν αισθητήρα κίνησης με δυνατότητα ενεργοποίησης συναγερμού, δυνατότητα ελέγχου φωτισμού 5 χώρων καθώς και ενημέρωση περιβαλλοντικών συνθηκών εντός της οικίας. Το σύστημα μπορεί να ελέγχεται και να παρακολουθείται από οπουδήποτε χωρίς περιορισμούς σύνδεσης σε τοπικό δίκτυο . Η πρόσβαση στο σύστημα θα γίνεται μέσω ιστοσελίδας, επιτρέποντας στον χρήστη να ελέγχει και να ρυθμίζει την ασφάλεια του σπιτιού του κάθε στιγμή. Συνεχίζοντας, η πτυχιακή εργασία θα ασχολείται επίσης με την υλοποίηση όλων των απαιτούμενων αλγόριθμων για την λειτουργία του συστήματος.

Λέξεις-κλειδιά: Ασφάλεια, Συναγερμός,Arduino,απομακρυσμένος έλεγχος.

ABSTRACT

This thesis will focus on the implementation of a prototype home security and alarm system using the ESP8266 microcontroller and the ESP32-CAM. The system will include a security camera, a motion sensor capable of triggering an alarm, the ability to control lighting in five areas, as well as monitoring of environmental conditions inside the house. The system is accessible and controllable from anywhere, without being restricted to a local network connection. Access will be provided through a web interface, allowing the user to monitor and manage their home security at any time. Furthermore, the thesis will also cover the implementation of all necessary algorithms required for the system's operation.

Keywords: Security, Alarm, Arduino, Remote control.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περιεχόμενα

1.ΕΙΣΑΓΩΓΗ

Αναπόσπαστο κομμάτι από την καθημερινότητά μας αποτελεί πλέον η υπεράσπιση-φύλαξη της ατομικής ακίνητης ιδιοκτησίας και πιο συγκεκριμένα της οικίας. Οι ιδιοκτήτες αναζητώντας ασφάλεια, και σε ένα ευρύτερο πλαίσιο ποιότητα ζωής, στρέφονται σε μία κορεσμένη πλέον αγορά, για να επιλέξουν ένα σύστημα ασφαλείας. Στα πλαίσια περιορισμένων γνώσεων επί του αντικειμένου, συχνά επιλέγουν προϊόντα που υπερκαλύπτουν τις ανάγκες τους με συνέπεια το τελικό αποτέλεσμα να υπερβαίνει το αρχικό οικονομικό πλάνο που υπολόγιζε ο εκάστοτε ιδιοκτήτης. Η παρούσα εργασία στοχεύει, στην ανάπτυξη ενός πρότυπου οικονομικού και επεκτάσιμου συστήματος ασφαλείας, βασιζόμενο σε προσβάσιμες τεχνολογίες. Πιο συγκεκριμένα με την χρήση ενός **ESP8266,** μίας ESP32-CAM, ενός VPS(Virtual Private Server) και μερικών αισθητήρων, μπορεί το σύστημα ασφαλείας να παρέχει στον χρήστη:

- Εποπτεία και έλεγχο απομακρυσμένα μέσω Web Interface.
- Ζωντανή παρακολούθηση του χώρου.
- Ενημέρωση περιβαλλοντικών συνθηκών εντός της οικίας.
- Ανίχνευση κίνησης με δυνατότητα ενεργοποίησης συναγερμού.
- Έλεγχο φωτισμού σε πέντε χώρους.

1.Εξοπλισμός και Τεχνολογίες

1.1 Εισαγωγή

Αντικείμενο αυτού του κεφαλαίου αποτελεί ο αναγκαίος εξοπλισμός και οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εργασίας. Τόσο σε φυσικό, όσο και σε λογικό επίπεδο.

1.2 Hardware

Για την υλοποίηση του συστήματος είναι απαραίτητος ο παρακάτω εξοπλισμός.

Είδος	Ποσότητα	Περιγραφή
ESP8266 (Node MCU)	1	Μικροεπεξεργαστής με δυνατότητα σύνδεσης Wi-Fi για έλεγχο I/O και HTTP polling
ESP32 CAM	1	Κάμερα με Wi-Fi και cardslot
DHT11	1	Αισθητήρας θερμοκρασίας/Υγρασίας
BUZZER	1	Ηχητική ειδοποίηση
HC-SR501	HC-SR501	Αισθητήρας κίνησης
BREADBOARD	1	Πίνακας διεπαφών
LEDS	5	Λυχνίες τύπου led
ΑΝΤΙΣΤΑΣΗ	1	Αντίσταση 10kΩ
SERIAL ADAPTER ή ARDUINO UNO	1	Για τον προγραμματισμό της ESP32
JUMPER WIRES		Καλώδια για τις συνδέσεις

Πίνακας 1.1 [Hardware]

1.3 Τεχνικά Χαρακτηρίστηκα & Περιγραφή Hardware

1.2.1 ESP8266

Ο ESP8266 είναι ένας μικροελεγκτής που παρέχει στον χρήστη, δυνατότητα ασύρματης σύνδεσης(Wi-Fi), το οποίο ήταν το κύριο κριτήριο για την επιλογή του. Ο συγκεκριμένος μικροελεγκτης, υπερκαλύπτει τις απαιτήσεις του project και προσφέρει αρκετές δυνατότητες αναβάθμισης. Πιο συγκεκριμένα:

- **GPIO**: 17 γενικής χρήσης ψηφιακές είσοδοι/έξοδοι.
- **Flash memory**: 4 MB.
- Ενσωματωμένο Wi-Fi (IEEE 802.11 b/g/n).
- Ενσωματωμένο USB-to-Serial για εύκολο Flashing.
- Τάση λειτουργίας 5v.

Εικόνα 1.1 ESP8266

1.2.2 ESP32-Cam

Για την ένταξη κάμερας ασφαλείας στην εργασία επιλέχθηκε η ESP32-CAM, η οποία προσφέρει:

- Κάμερα OV2640
- Ενσωματωμένο Wi-Fi (802.11 b/g/n)
- 4 MB flash
- Τάση λειτουργίας 5V

Εικόνα 1.1ESP32 Cam

1.2.3 DHT11

Για την καταγραφή περιβαλλοντικών παραμέτρων επιλέχθηκε ο DHT11

- Τάση λειτουργίας 3.3-5.5V
- Εύρος μέτρησης θερμοκρασίας 0°C ~ 50°C
- Εύρος μέτρησης υγρασίας 20%RH ~ 90%RH (25°C)

Εικόνα1.2 DHT11

1.2.4 PIR Motion Sensor & Buzzer

Για την λειτουργία του συναγερμού, με ανίχνευση κίνησης και άμεση ηχητική ειδοποίηση, επιλέχθηκε ο συνδυασμός ενός HC-SR501 με ένα Buzzer module.

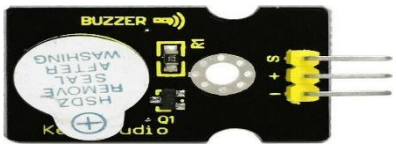
HC-SR501

- Τάση λειτουργίας 3.3 V
- Εύρος ανίχνευσης κίνησης 6m

Buzzer

- Τάση λειτουργίας 3.3-5 V

Εικόνα 1. 4 HC-SR501



Εικόνα 1. 3.1 BUZZER

1.2.5 Arduino Uno, Breadboard, Καλώδια, LEDs & Αντιστάσεις

Για την σύνδεση και τον πειραματικό έλεγχο του κυκλώματος χρησιμοποιήθηκαν τα παρακάτω:

- Breadboard: Πλακέτα διασύνδεσης χωρίς κολλήσεις απαραίτητη για το project αφού το τελικό αποτέλεσμα επιτεύχθηκε έπειτα από πολλούς πειραματισμούς.
- Καλώδια(jumper wires): Male to male & male to female καλώδια για τις συνδέσεις όλων των κομματιών του hardware.
- LEDs: Χρησιμοποιήθηκαν για την προσομοίωση ελέγχου φωτισμού.
- Αντιστάσεις: Για την ορθή και ασφαλή λειτουργία του κυκλώματος.
- Arduino Uno: Χρησιμοποιήθηκε σαν USB to serial adapter για τον προγραμματισμό της ESP32-CAM.

1.4 Γλώσσες Προγραμματισμού

Η εργασία αποτελείται από ένα συνδυασμό γλωσσών προγραμματισμού. Για να είναι εύκολο στην κατανόηση η εργασία χωρίστηκε στα εξής τμήματα:

- Frontend
- Backend
- Προγραμματισμός Hardware

1.3.1 Frontend

Κάθε ιστοσελίδα χρειάζεται το γραφικό περιβάλλον που θα μπορεί ο χρήστης να αλληλοεπιδρά με το «πίσω» κομμάτι τις ιστοσελίδας(backend). Πιο συγκεκριμένα το Frontend είναι η ορατή πλευρά μιας εφαρμογής η μιας ιστοσελίδας, που βλέπει ο χρήστης και μπορεί να αλληλοεπιδρά μαζί της. Δημιουργείται και σχεδιάζεται ένα γραφικό περιβάλλον για να παρέχει στον χρήστη εμπειρία. Στην παρούσα εργασία για την δημιουργία του γραφικού περιβάλλοντος χρησιμοποιήθηκαν και συνδυάστηκαν η HTML, η CSS και η JavaScript.

1.3.1.α HTML

Η **HTML** (Hypertext Markup Language),η οποία στα ελληνικά μεταφράζεται ως «*γλώσσα σήμανσης υπερκειμένου*», είναι η τυπική γλώσσα σήμανσης για την δημιουργία ιστοσελίδων που έχουν σχεδιαστεί με σκοπό την εμφάνισή τους σε πρόγραμμα περιήγησης. Με την HTML καθορίζουμε το περιεχόμενο και τη δομή της σελίδας. Συχνά υποστηρίζεται από τεχνολογίες όπως η CSS για την τροποποίηση της εμφάνισης και η JavaScript για την λειτουργικότητα και την αλληλεπίδραση. (Wikipedia HTML, 2025)

1.3.1.β CSS

Η **CSS** (***Cascading Style Sheets***) είναι μια γλώσσα που χρησιμοποιείται για καθορισμό της εμφάνισης μιας ιστοσελίδας που έχει γραφτεί με HTML.Με την χρήση της CSS έχουμε περισσότερες δυνατότητες εμφάνισης σε σχέση με την HTML.Μπορούμε να επεξεργαστούμε την στοίχιση, τα χρώματα να δημιουργήσουμε γραφικά στοιχεία κ.α. Είναι δηλαδή μια γλώσσα που εξυπηρετεί αποκλειστικά στην στιλιστική ανάπτυξη μιας ιστοσελίδας. (Wikipedia CSS, 2025)

1.3.1.γ JavaScript

Η JavaScript είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται κυρίως για την ανάπτυξη διαδραστικών λειτουργιών σε ιστοσελίδες. Λειτουργεί από την πλευρά του χρήστη (client-side), επιτρέποντας την αλληλεπίδραση των χρηστών με τα στοιχεία μιας ιστοσελίδας, όπως τη διαχείριση φόρμας, την εμφάνιση δυναμικού περιεχομένου και τη

δημιουργία κινούμενων γραφικών. Ωστόσο, η JavaScript μπορεί επίσης να λειτουργεί από την πλευρά του διακομιστή (server-side) με την πλατφόρμα Node.js, επεκτείνοντας τη χρήση της σε πλήρεις εφαρμογές ιστού. (Wikipedia JavaScript, 2025)

1.3.2 Backend

Το backend σε μια εφαρμογή η μια ιστοσελίδα αναφέρεται στο τμήμα που δεν είναι ορατό στον χρήστη, αλλά είναι απαραίτητο για την ορθή λειτουργία. Αυτό περιλαμβάνει τον κώδικα, την υποδομή, τη διαχείριση των δεδομένων, τις βάσεις δεδομένων και τα APIs που είναι απαραίτητα για να λειτουργήσει η εφαρμογή. Στην παρούσα εργασία για την δομή του backend, χρησιμοποιήθηκαν οι γλώσσες PHP και SQL.

1.3.2.α PHP (Hypertext Preprocessor)

Είναι μια γλώσσα προγραμματισμού που λειτουργεί από την πλευρά του διακομιστή και χρησιμοποιείται κυρίως για τη δημιουργία διαδραστικών ιστοσελίδων. Έχει τη δυνατότητα να ενσωματώνεται απευθείας σε HTML, επιτρέποντας τη δημιουργία περιεχομένου που προσαρμόζεται στις ανάγκες των χρηστών, όπως φόρμες, βάσεις δεδομένων και διαχείριση συνεδριών. (Wikipedia PHP, 2025)

1.3.2.β SQL

Η SQL (Structured Query Language) είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται για τη διαχείριση και τον χειρισμό δεδομένων σε συστήματα διαχείρισης βάσεων δεδομένων. Μέσω της SQL, οι χρήστες μπορούν να δημιουργούν, να διαβάζουν, να ενημερώνουν και να διαγράφουν δεδομένα σε βάσεις δεδομένων, όπως η MySQL. (WIkipedia SQL, 20252)

1.3.2 Προγραμματισμός Hardware

Για την ανάπτυξη του κώδικα χρησιμοποιήθηκε η γλώσσα προγραμματισμού **C/C++**, προσαρμοσμένη για μικροελεγκτές.

1.5 Εργαλεία

- Arduino IDE: Χρησιμοποιήθηκε για την ανάπτυξη του κώδικα που αφορά τον προγραμματισμό των πλακετών. Καθώς και για τον έλεγχο λειτουργίας του συστήματος.
- Visual studio code: Χρησιμοποιήθηκε για την ανάπτυξη κώδικα που αφορά το frontend & backend τμήμα της εργασίας
- VPS: Χρησιμοποιήθηκε για την φιλοξενία της εργασίας σε server.

2.VPS

2.1 Εισαγωγή

Ένας **Virtual Private Server (VPS)** είναι ένας εικονικός ιδιωτικός διακομιστής που παρέχει στους χρήστες πλήρη πρόσβαση σε ένα απομονωμένο περιβάλλον διακομιστή. Πρόκειται για μια τεχνολογία που επιτρέπει τη φιλοξενία και τη λειτουργία ιστοσελίδων, εφαρμογών και υπηρεσιών σε έναν διακομιστή που λειτουργεί ως ανεξάρτητη μονάδα.

2.2 Για ποιο λόγο χρησιμοποιούμε VPS στο Σύστημά μας

Στην παρούσα εργασία, ο VPS χρησιμοποιείται ως κεντρικός διακομιστής επικοινωνίας και διαχείρισης του συστήματος ασφαλείας. Οι βασικοί λόγοι χρήσης του VPS περιλαμβάνουν:

- Απομακρυσμένη Πρόσβαση: Ο VPS επιτρέπει στον χρήστη να διαχειρίζεται το σύστημα ασφαλείας μέσω ενός Web Interface.
- Συλλογή και Επεξεργασία Δεδομένων: Ο VPS λαμβάνει δεδομένα από τον ESP8266 (π.χ., δεδομένα αισθητήρων) και τον ESP32-CAM (εικόνες ή βίντεο).
- Αποθήκευση Δεδομένων: Ο VPS μπορεί να αποθηκεύει αρχεία καταγραφής (logs) από τις συσκευές και να παρέχει ιστορικά δεδομένα.
- Ανταλλαγή Εντολών: Ο ESP8266 λαμβάνει εντολές από τον VPS μέσω HTTP polling, επιτρέποντας την απομακρυσμένη διαχείριση.

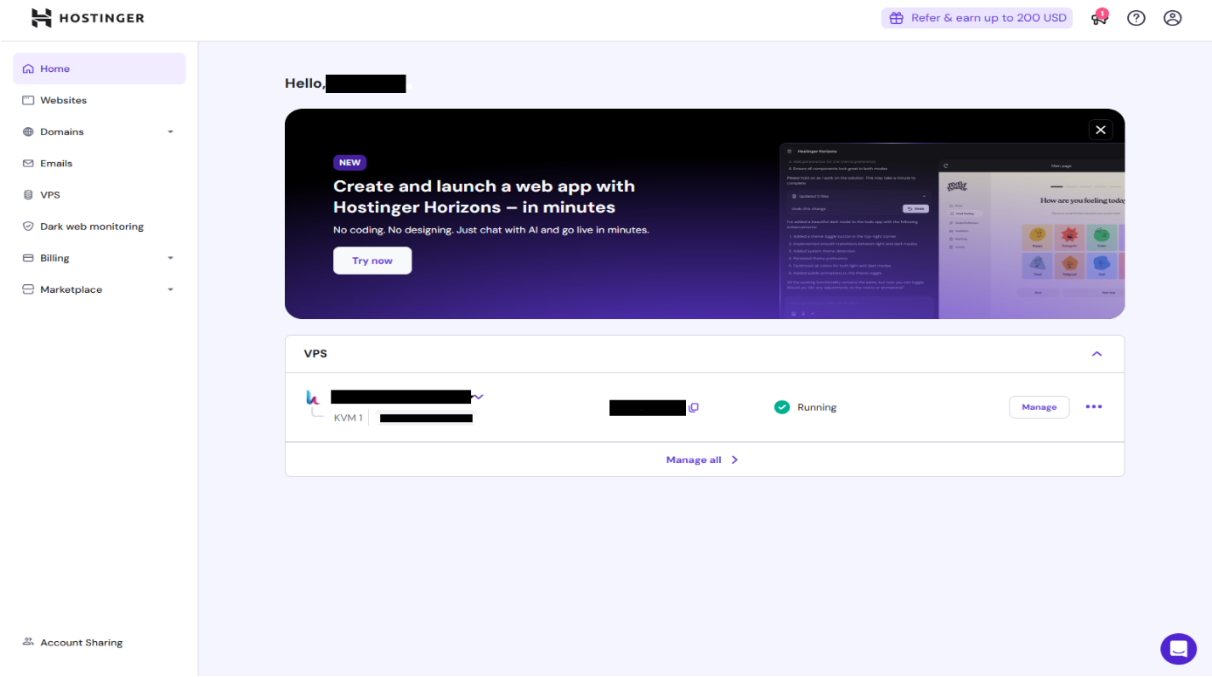
2.3 Επιλογή και Διαμόρφωση του VPS

Για την φιλοξενία της εργασίας επιλέχθηκε ένας VPS της εταιρείας Hostinger. Σε αυτό το κεφάλαιο θα δούμε κάθε βήμα από την αρχή μέχρι την ολοκλήρωση της διαμόρφωσης.

Ο χρηστής έχει την δυνατοτητα να επιλέξει ανάμεσα σε αρκετα πακέτα συνδρομών ανάλογα με τις απαιτήσεις του εκάστοτε project. Για την συγκεκριμενη εργασία επιλέχθηκε το μικρότερο πακέτο που υπερκαλύπτει τις ανάγκες της εργασίας και παρέχει:

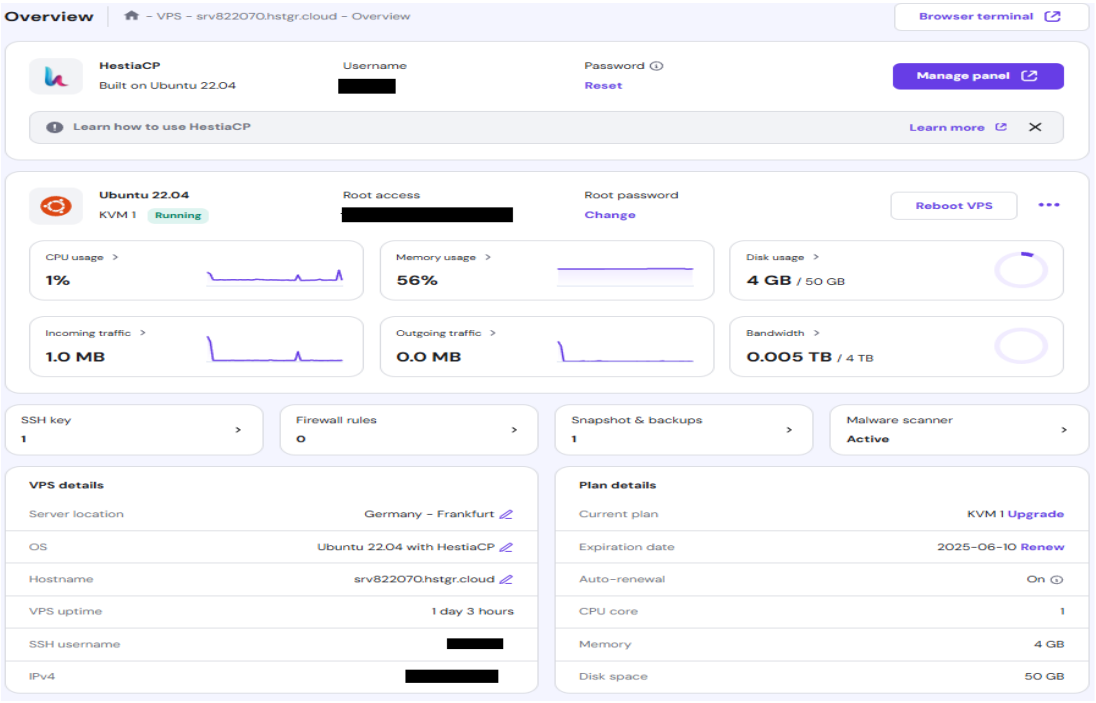
- CPU core 1
- Memory 4 GB
- Disk space 50 GB

Κατά την διάρκει της διαμόρφωσης του VPS, ο χρήστης επιλέγει την τοποθεσία και το λειτουργικο σύστημα του διακομιστή.Ως τοποθεσία επιλέχθηκε η κοντινότερη στην Ελλάδα και ο VPS διαθέτει λειτουργικό σύστημα Ubuntu 22.04 με προεγκατεστημένο το HestiaCP.Αφότου συμπληρωθούν όλα τα απαραίτητα πεδία, ο χρήστης μπορεί μεσα από την κεντρική σελίδα να πλοηγηθεί και διαχειρίστει τον VPS.



Εικόνα 2.1 Κεντρική Σελίδα VPS.

Πατώντας στην επιλογή «manage», όπως φαίνεται στην «εικόνα 2.1», ο χρήστης έχει μια πλήρη εικόνα του διακομιστή «εικόνα 2.2». Από αυτή την σελίδα μπορεί ο χρήστης να αντλήσει όλες τις απαραίτητες πληροφορίες για να ξεκινήσει να δημιουργεί και να ελέγχει την εφαρμογή του.



Εικόνα 2.2.Σελίδα Διαχείρισης

2.3 HestiaCp

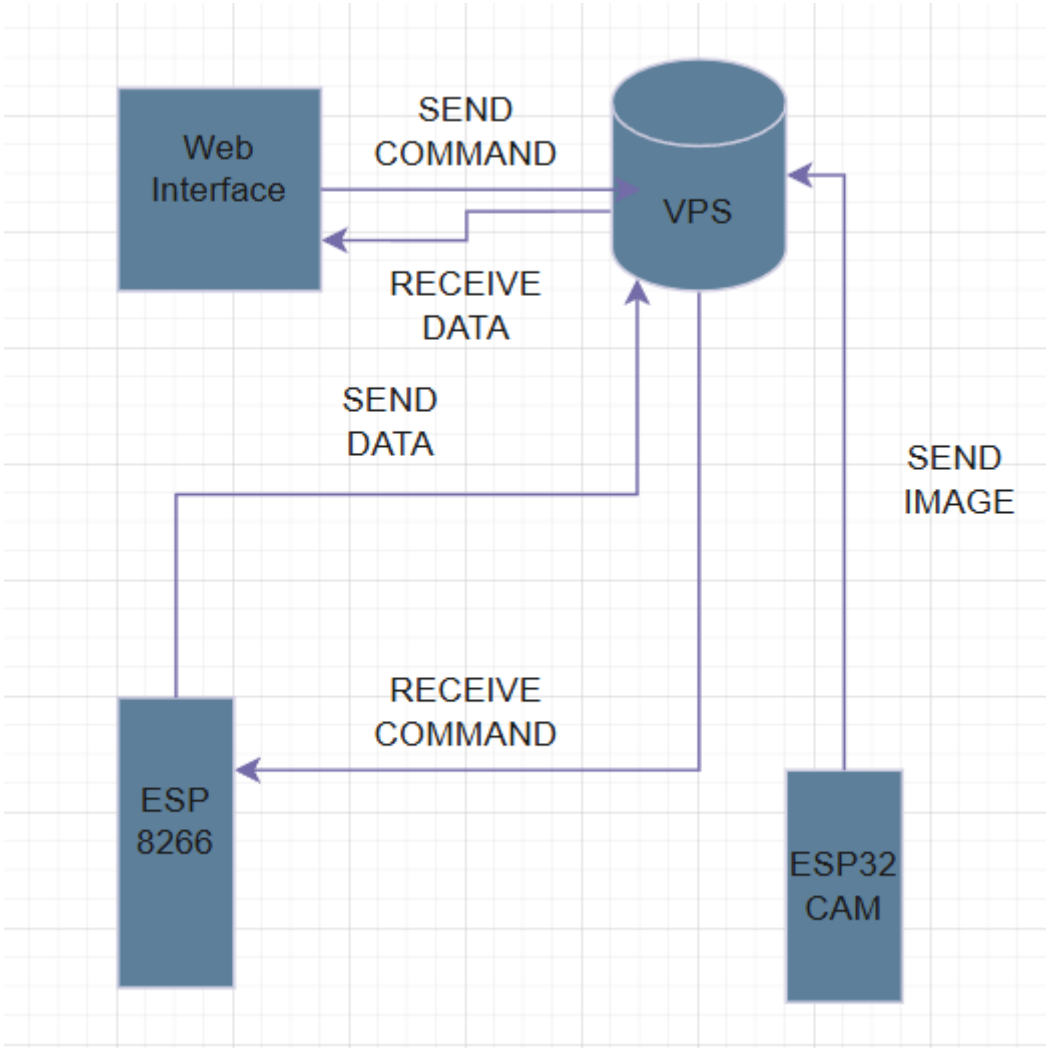
Το Hestia Control Panel είναι ένα δωρεάν εργαλείο που βοήθα στην διαχείριση Web Server. Απλοποιεί την διαδικασία με ένα περιβάλλον φιλικό προς τον χρήστη, ώστε η δημιουργία της ιστοσελίδας να γίνεται με ευκολία.

3.Τρόπος Λειτουργίας και Συνδεσμολογία Μικροελεγκτών

3.1 Εισαγωγή

Ξεκινώντας από την θεωρητική προσέγγιση της εργασίας, το σύστημα σε φυσικό επίπεδο, θα πρέπει αρχικά να συλλέγει και να διαχειρίζεται πληροφορίες από τους αισθητήρες. Στη συνέχεια αυτά τα δεδομένα θα πρέπει να μεταβιβάζονται στον VPS, διαμέσου της επικοινωνίας HTTP Polling.Ο διακομιστής θα επεξεργάζεται τα δεδομένα, θα τα προβάλει στην ιστοσελίδα και ταυτόχρονα θα προωθεί πίσω την κατάσταση των LEDs και του συναγερμού .Για καλύτερη κατανόηση θα διαχωρίσουμε την εργασία σε τέσσερις βασικές εφαρμογές και θα τις αναλύσουμε ξεχωριστά.

- Έλεγχος φωτισμού
- Περιβαλλοντικές μετρήσεις
- Έλεγχος συναγερμού
- Κάμερα

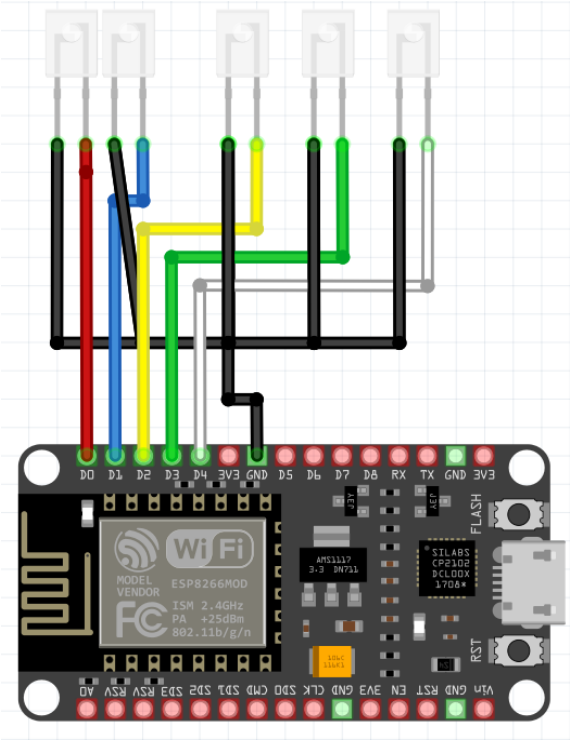


Εικόνα 3 Σχεδιάγραμμα επικοινωνίας

3.2 Ανάλυση Εφαρμογών

3.2.1 Έλεγχος φωτισμού

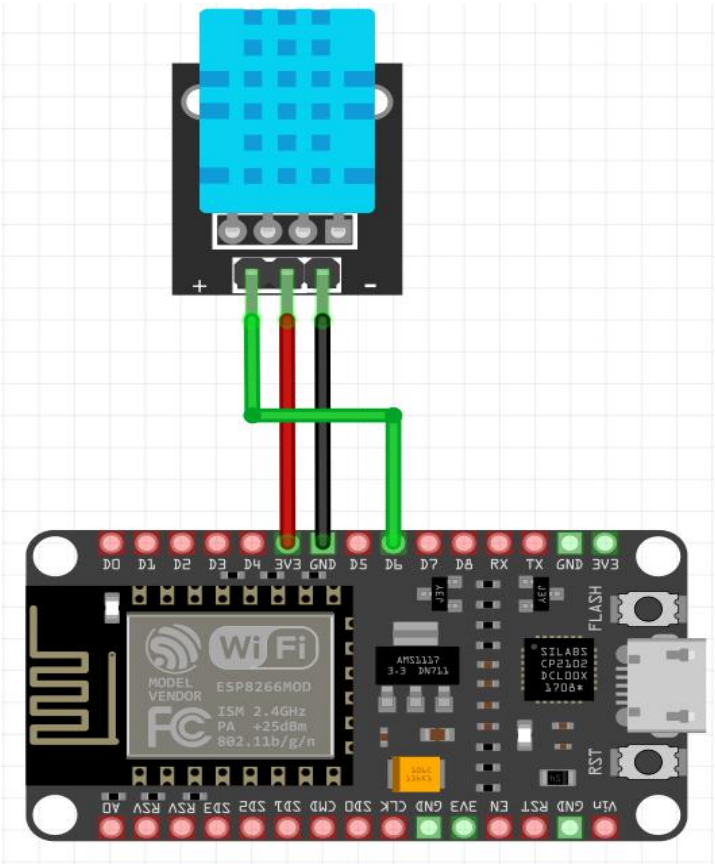
Το παρόν σύστημα παρέχει την δυνατότητα ελέγχου φωτισμού σε πέντε χώρους. Για την ανάγκη της προσομοίωσης θα χρησιμοποιήσουμε πέντε λαμπτήρες Led, κατάλληλα συνδεδεμένους με τον ESP8266 ώστε, η λειτουργία τους να γίνεται αποκλειστικά από την ιστοσελίδα. Στην «εικόνα 3.1» αποτυπώνεται αυτή η συνδεσμολογία όπου έχουν οριστεί 5 ψηφιακές έξοδοι από D0 έως D4 για την τροφοδοσία των λαμπτήρων καθώς και με μαύρο χρώμα απεικονίζονται οι απαραίτητες γειώσεις.



Εικόνα 3.1 Συνδεσμολογία λαμπτήρων

3.2.2 Περιβαλλοντικές Μετρήσεις.

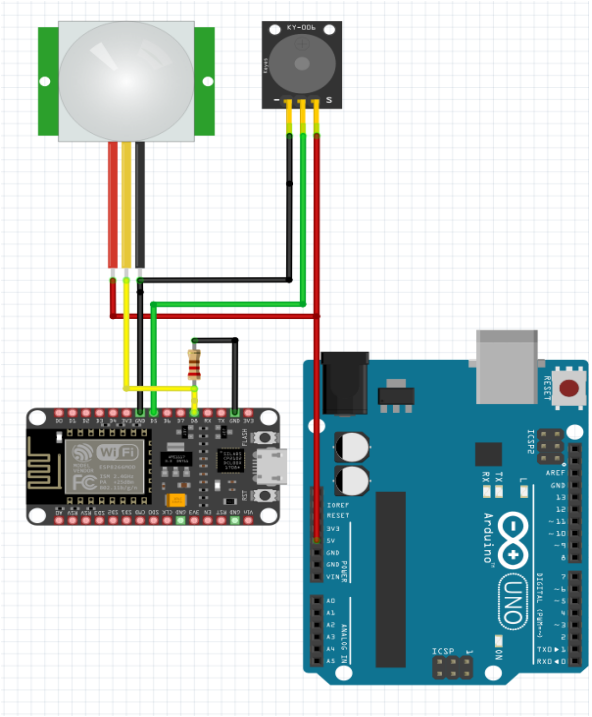
Η προσθήκη του DHT11 στο σύστημα παρέχει την δυνατότητα μέτρησης περιβαλλοντικών παραμέτρων όπως η θερμοκρασία και η υγρασία του χώρου. Για να λειτουργήσει σωστά ο αισθητήρας χρειάζεται τροφοδοσία 3,3-5V, μια γείωση και ένα διαθέσιμο pin στον ESP8266. Στο συγκεκριμένο παράδειγμα έχει οριστεί το pin D6 όπου, θα λειτουργεί σαν είσοδο για την λήψη δεδομένων από τον αισθητήρα στον μικροελεγκτή. Στην «εικόνα 3.2» διακρίνεται η απλότητα της σύνδεσης.



Εικόνα 3.2 .Συνδεσμολογία DHT

3.3.2 Έλεγχος Συναγερμού.

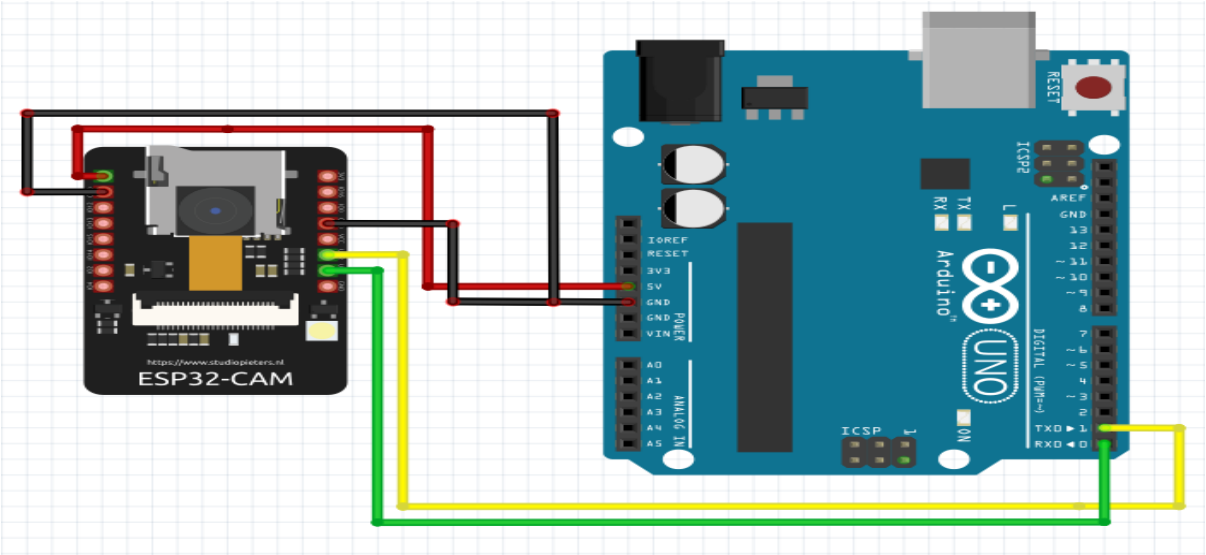
Η λειτουργία του συναγερμού αποτελείται από τον αισθητήρα κίνησης HC-SR501 και ένα Buzzer.Ως πηγή τροφοδοσίας θα χρησιμοποιηθεί ένας Arduino UNO ο οποίος έχει την δυνατότητα παροχής 5V και θα καλύψει τις ανάγκες λειτουργίας του HC-SR501 και του buzzer.Ως ψηφιακή είσοδος για την επικοινωνία με τον αισθητήρα κίνησης ορίστηκε το pin D7.Για να αποφευχθούν λανθασμένες τιμές εισόδου λόγω floating pin, όσο ο αισθητήρας είναι απενεργοποιημένος, τοποθετήθηκε μία αντίσταση 10kΩ ώστε η κατάσταση του D7 να παραμένει Low . Με αυτόν τον τρόπο ουσιαστικά επιτυγχάνουμε την ενεργοποίηση του buzzer μόνο όταν ο συναγερμός είναι ενεργός από το web interface και ο αισθητήρας εντοπίσει κίνηση.



Εικόνα 3.3 Συνδεσμολογία Συναγερμού

3.4.2 Κάμερα

Με την ESP32 Cam θα υλοποιηθεί η ζωντανή παρακολούθηση του χώρου. Δεδομένου ότι η κάμερα δεν διαθέτει ενσωματωμένο USB to Serial adapter, Θα χρησιμοποιηθεί ένας Arduino Uno για να επιτευχθεί ο προγραμματισμός της. Όπως φαίνεται και στην «εικόνα 7» εκτός από την παροχή και την γείωση που είναι απαραίτητα για την λειτουργία της κάμερας. Υπάρχουν ακόμα τρείς συνδέσεις που συμβάλλουν στον προγραμματισμός της. Για την αποστολή του αλγόριθμου αλλά και την λήψη μηνυμάτων στο Serial Monitor, πρέπει να συνδεθούν τα δυο pin αποστολής και λήψης δεδομένων μεταξύ των δύο πλακετών. Πιο συγκεκριμένα το UOT της ESP32 συνδέεται με το TX του Arduino και αντίστοιχα το UOR με το RX. Για να οριστεί η ESP32 σε κατάσταση προγραμματισμού πρέπει το pin IO0 να συνδεθεί με την γείωση. Αφότου ολοκληρωθεί η αποστολή του αλγορίθμου, πρέπει να διακοπεί η τροφοδοσία και να αποσυνδεθεί το pin IO0 από την γείωση. Με αυτόν τον τρόπο διακόπτεται η λειτουργία προγραμματισμού που είχε επιβληθεί στην πλακέτα.



Εικόνα 3.4 Συνδεσμολογία Προγραμματισμού Κάμερας

3.3Τελική Συνδεσμολογία

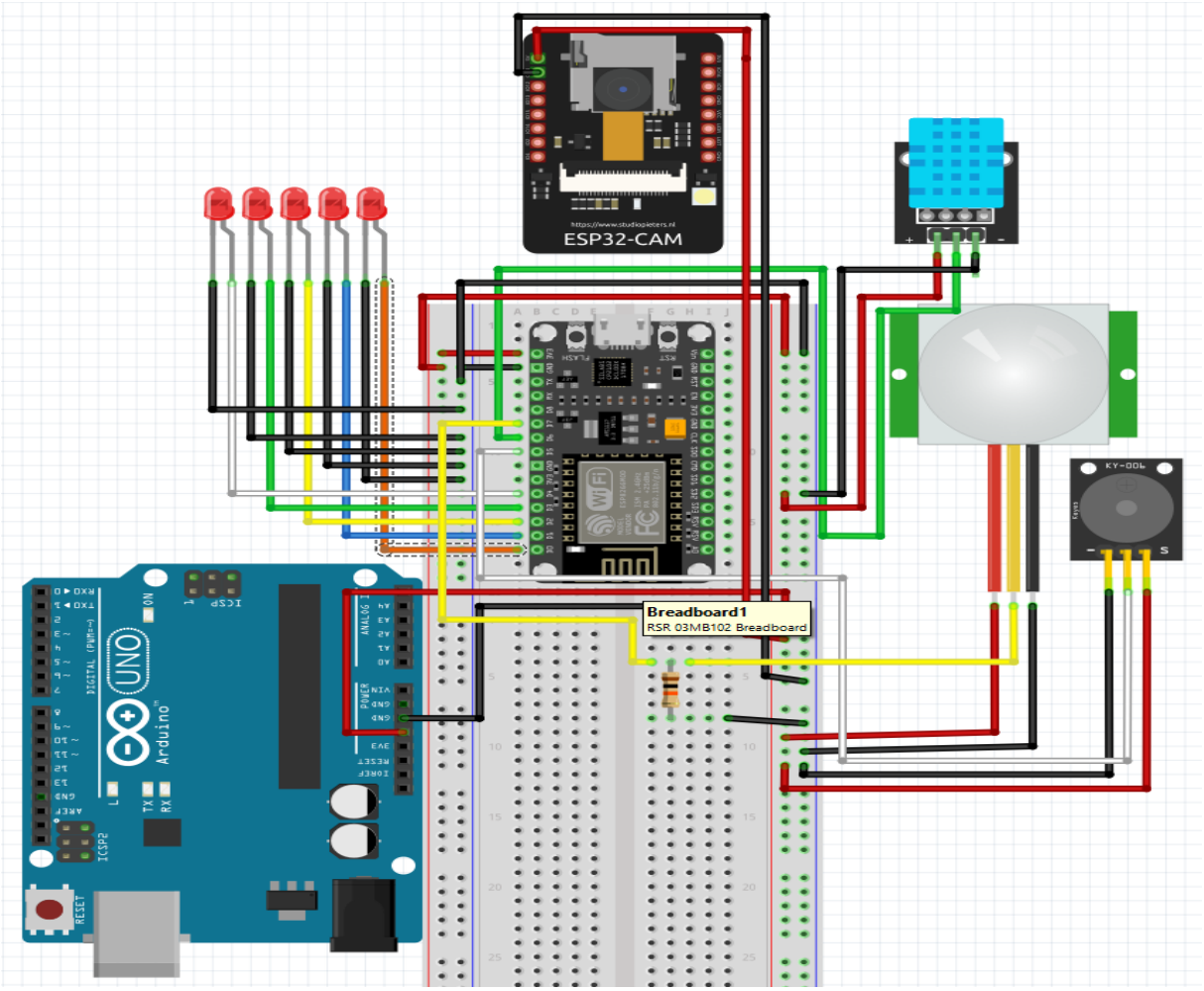
Έχοντας πραγματοποιηθεί η συνδεσμολογία των εφαρμογών ξεχωριστά και έχοντας ελεγχθεί η λειτουργικότητα τους, είναι εφικτό πλέον να πραγματοποιηθεί η συνένωση τους σε ένα ενιαίο σύστημα. Το σύστημα έχει δυο πηγές παροχής ενέργειας, η πρώτη είναι από τον ESP8266 με τάση στα 3.3V και η δεύτερη από τον Arduino Uno στα 5V τάση. Το Breadboard είναι χωρισμένο σε δύο ανεξάρτητα τμήματα με το καθένα να λαμβάνει διαφορετική τάση. Η τροφοδοσία κάθε εξαρτήματος συμβολίζεται με την χρήση κόκκινου καλωδίου και η γείωση με μαύρο.

Για τον χειρισμό των LEDs έχουν οριστεί πέντε ψηφιακές έξοδοι, από D0 έως D4, και χρησιμοποιήθηκαν καλώδια διαφορετικού χρωματισμού για να είναι ευδιάκριτη η αντιστοιχία του κάθε Led με την εκάστοτε ψηφιακή έξοδο.

Ο DHT 11 με την χρήση πράσινου καλωδίου συνδέεται στην ψηφιακή είσοδο D6. Από την συγκεκριμένη είσοδο ο ESP8266 θα λαμβάνει τις περιβαλλοντικές μετρήσεις και έπειτα θα τις αποστέλλει στον VPS.

Ο συναγερμός χρησιμοποιεί μία ψηφιακή είσοδο και μία ψηφιακή έξοδο. Στο D7 ο αισθητήρας κίνησης στέλνει λογικό 0 όταν δεν υπάρχει κίνηση στον χώρο και λογικό 1 όταν εντοπίζει κίνηση. Το D5 λειτουργεί σαν έξοδος και τροφοδοτεί το buzzer όταν στην D7 έχει ληφθεί η τιμή ένα.

Η ESP32 CAM ως ξεχωριστός μικροελεγκτής λειτουργεί ανεξάρτητα από το υπόλοιπο σύστημα, λαμβάνοντας μόνο την απαραίτητη τροφοδοσία των 5V.



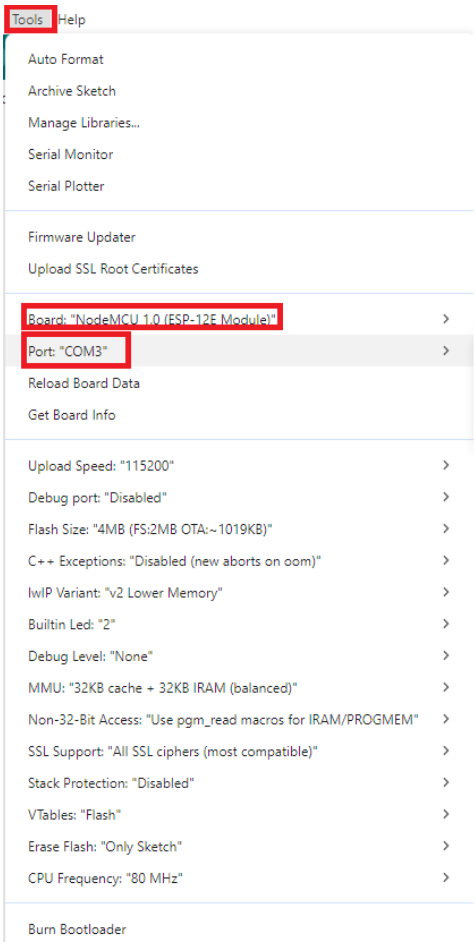
Εικόνα 3.5 Ολοκληρωμένη Συνδεσμολογία

4.Προγραμματισμός ESP8266

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο παρουσιάζονται όλοι οι αλγόριθμοι που συντέλεσαν στην δημιουργία του ολοκληρωμένου λειτουργικού κώδικα του ESP8266. Με την εκτεταμένη ανάλυση και περιγραφή του αντίστοιχου αλγορίθμου γίνεται πλήρως κατανοητή η λειτουργία της κάθε εφαρμογής. Για τον προγραμματισμό των μικροελεγκτών χρησιμοποιήθηκε το λογισμικό Arduino IDE με τις απαραίτητες βιβλιοθήκες που προσφέρει και θα αναλυθούν μετέπειτα. Ο προγραμματισμός κάθε εφαρμογής έγινε ξεχωριστά ώστε να είναι εύκολη η διαχείριση και η διόρθωσή της.

Για να ξεκινήσει ο προγραμματισμός θα πρέπει ο ESP8266 να είναι συνδεδεμένος με τον Η/Υ με την χρήση USB καλωδίου. Ανοίγοντας το Arduino IDE μέσα από το πεδίο «Tools» πρέπει να επιλεχθεί η θύρα USB που είναι συνδεδεμένος ο μικροελεγκτής καθώς και ο τύπος του «εικόνα 4.1».



Εικόνα 4.1

4.2 Βιβλιοθήκες

Για την υλοποίηση του αλγορίθμου που θα επιτρέπει την απομακρυσμένη διαχείριση του συστήματος είναι απαραίτητη η χρήση ορισμένων βιβλιοθηκών. Η προσθήκη τους απλοποιεί την συγγραφή κώδικα με τις συναρτήσεις που προσφέρουν.

4.2.1 ESP8266WiFi

Η σύνδεση του ESP8266 με τον δρομολογητή επιτυγχάνεται μέσω των συναρτήσεων που παρέχονται από την βιβλιοθήκη “ESP8266WiFi”. Εφόσον έχουν οριστεί το SSID και το Password του δρομολογητή, μπορεί να κληθεί συνάρτηση WiFi.begin(ssid, password) και ο μικροελεγκτής να συνδεθεί με τον δρομολογητή. Με την χρήση της συνάρτησης “WiFi.status()” γίνεται ο έλεγχος της κατάστασης σύνδεσης. Παρακάτω δίνεται ένα απλό παράδειγμα σύνδεσης:

- Συμπερίληψη βιβλιοθήκης.
- Ορισμός στοιχείων δρομολογητή.

- Εκκίνηση σειριακής επικοινωνίας για προβολή μηνυμάτων στο Serial Monitor.
- Κλήση της WiFi.Begin().
- Συνθήκη : Όσο WiFi.status() δεν είναι συνδεδεμένο εμφάνισε «Προσπάθεια Σύνδεσης».
- Μόλις γίνει η σύνδεση η συνθήκη While “σπάει” και εμφανίζονται τα μηνύματα σύνδεσης.

```
#include <ESP8266WiFi.h>

const char* ssid     = "SSID"; // Όνομα Δρομολογητή
const char* password = "Pass"; // Κωδικός

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  Serial.print("Σύνδεση σε ");
  Serial.print(ssid);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print("\nΠροσπάθεια Σύνδεσης");
  }
  Serial.println("\nΣυνδέθηκε!");
  Serial.print("IP διεύθυνση: ");
  Serial.println(WiFi.localIP());
}
```

```
Προσπάθεια Σύνδεσης
Προσπάθεια Σύνδεσης
Προσπάθεια Σύνδεσης
Προσπάθεια Σύνδεσης
Συνδέθηκε!
IP διεύθυνση: 192.168.0.205
```

Εικόνα 4.2 Κώδικας σύνδεσης σε τοπικό δίκτυο

4.2.2 ESP8266HTTPClient

Για την επικοινωνία με τον διακομιστή χρησιμοποιείται η βιβλιοθήκη ”ESP8266HTTPClient” όπου παρέχει συναρτήσεις για την αποστολή και λήψη HTTP αιτημάτων. Για να είναι διαθέσιμες προς χρήση οι συναρτήσεις, θα πρέπει αρχικά να δημιουργηθεί ένα αντικείμενο με την εντολή “ HTTPClient Όνομα_Αντικειμένου;”. Εφόσον το αντικείμενο έχει δημιουργηθεί μπορεί να κληθεί η συνάρτηση

“Όνομα_Αντικειμένου.begin();” που παίρνει δύο ορίσματα για το «πως» και «που» θα στέλνονται τα αιτήματα. Με την συνάρτηση “Όνομα_Αντικειμένου.GET()” είναι δυνατό να ελεγχθεί η κατάσταση της σύνδεσης αφού η συγκεκριμένη συνάρτηση επιστρέφει κωδικούς πχ 200 η σύνδεση είναι εντάξει, 500 εσωτερικό πρόβλημα στον server κ.α. Οι κωδικοί αναφοράς όταν είναι θετικοί σημαίνει ότι έχει ληφθεί απάντηση από τον διακομιστή. Αντίθετα όταν είναι αρνητικοί σημαίνει ότι δεν υπάρχει απάντηση από τον διακομιστή και οι αιτίες ποικίλουν. Εφόσον η συνάρτηση .GET() επιστρέψει HTTP_CODE_OK(ορισμένη σταθερά από την βιβλιοθήκη και έχει τιμή 200), όπου αυτό συνεπάγεται ότι υπάρχει σύνδεση με τον διακομιστή , μπορεί να κληθεί η συνάρτηση .getString(). Με την χρήση της .getString() γίνεται η ανάγνωση του σώματος της απάντησης από τον διακομιστή.

4.2.3 ArduinoJson.h

Η αποστολή και η λήψη δεδομένων μεταξύ του μικροελεγκτή και του διακομιστή γίνεται με την μορφή JSON. Η βιβλιοθήκη “ArduinoJson.h” είναι απαραίτητη για την διαχείριση αυτών των δεδομένων. Αρχικά με την συνάρτηση “DynamicJsonDocument Έγγραφο(1024)” δημιουργείται ένας χώρος ώστε να τοποθετηθεί το επεξεργασμένο περιεχόμενο από την απάντηση του διακομιστή. Η επεξεργασία της απάντησης από τον διακομιστή που λαμβάνεται από την .getString() πραγματοποιείται με την χρήση της deserializeJson(Έγγραφο, Απάντηση Διακομιστή). Τέλος για τον έλεγχο της μεταγλώττισης μπορεί να χρησιμοποιηθεί μια μεταβλητή τύπου DeserializationError.

4.2.4 DHT.h

Η DHT.h θα συμβάλει με τις συναρτήσεις της στην λήψη περιβαλλοντικών μετρήσεων από τον DHT11. Με την .begin() αρχίζει η διαδικασία όπου ο ESP μπορεί να διαβάζει δεδομένα από τον αισθητήρα. Για την ανάγνωση της θερμοκρασίας χρησιμοποιείται η readTemperature() ενώ για την ανάγνωση της υγρασία η readHumidity().

4.3 Έλεγχος Φωτισμού

Ο συγκεκριμένος κώδικας χρησιμοποιώντας τις βιβλιοθήκες που αναλύθηκαν παρέχει την δυνατότητα απομακρυσμένης διαχείρισης πέντε LED. Αξιοποιώντας την δυνατότητα σύνδεσης Wi-Fi του ESP8266 και το πρωτόκολλο επικοινωνίας HTTP, επιτυγχάνεται η σύνδεση με τον απομακρυσμένο διακομιστή. Ο μικροελεγκτής στέλνει περιοδικά αιτήματα στον διακομιστή ώστε να λαμβάνει δεδομένα για την κατάσταση των LED σε μορφή JSON.

Στην αρχή του κώδικα δηλώνονται οι απαραίτητες βιβλιοθήκες και ορίζονται οι απαραίτητες μεταβλητές. Για την σύνδεση με τον δρομολογητή είναι απαραίτητα το όνομα και ο κωδικός της συσκευής. Το URL του διακομιστή καθώς και ένα επιπλέον κλειδί ασφάλειας έχουν δηλωθεί σε μεταβλητές που θα χρησιμοποιηθούν για την επικοινωνία με τον διακομιστή. Για την λειτουργία των LED ορίζονται οι ακροδέκτες που είναι συνδεδεμένα, ένας πίνακας που έχει το όνομα κάθε λυχνίας καθώς και ένας μετρητής. Ολοκληρώνοντας το πρώτο κομμάτι του κώδικα ορίζονται δυο μεταβλητές που θα συμβάλουν στην εκτέλεση περιοδικών αιτημάτων από τον ESP στον διακομιστή.

```
1   #include <ESP8266WiFi.h>
2   #include <ESP8266HTTPClient.h>
3   #include <ArduinoJson.h>
4
5   const char* ssid     = "SSID";
6   const char* password = "PASS";
7
8   const char* host   = "VPS_URL";
9   const char* apiKey = "API_KEY";
10
11  const uint8_t LED_PINS[]  = { D0, D1, D2, D3, D4 };
12  const char*   LED_NAMES[] = { "L1", "L2", "L3", "L4", "L5" };
13  const uint8_t NUM_LEDS    = 5;
14
15  unsigned long lastTime = 0;
16  const unsigned long del   = 5000;
```

Εικόνα 4.3.1 Αρχικές δηλώσεις ελέγχου φωτισμού

Προχωρώντας στο δεύτερο κόμματί του κώδικα μέσα στην συνάρτηση setup(), αρχικοποιείται η κατάσταση των LED σε Low δηλαδή σβηστά. Συνεχίζοντας ακολουθούν εντολές για την έναρξη σειριακής επικοινωνίας, για την σύνδεση στο δίκτυο καθώς και για μηνύματα που εμφανίζονται στο serial monitor του Arduino IDE που βοηθούν στην επιβεβαίωση ορθής λειτουργίας. Στην συνάρτηση loop() χρησιμοποιείται μια λίγο πιο σύνθετη μέθοδο επανάληψης που θα βοηθήσει την κάθε εφαρμογή της εργασίας να λειτουργεί σε διαφορετικούς χρόνους. Πιο συγκεκριμένα η συνάρτηση millis() επιστρέφει σε millisecond τον χρόνο που έχει περάσει από την εκκίνηση του ESP. Σε συνδυασμό με τις πράξεις, μεταξύ της συνάρτησης και των ορισμένων μεταβλητών που γίνονται μέσα στην συνθήκη “if” , επιτυγχάνεται η ζητούμενη καθυστέρηση των 5 δευτερολέπτων. Σε κάθε επανάληψη πραγματοποιείται και ένα αίτημα λήψης δεδομένων με την κλήση της συνάρτησης receivedata().

```
20  void setup() {
21
22    for (uint8_t i = 0; i < NUM_LEDS; i++) {
23      pinMode(LED_PINS[i], OUTPUT);
24      digitalWrite(LED_PINS[i], LOW);
25    }
26
27    Serial.begin(115200);
28    Serial.println("Προσπάθεια σύνδεσης στο δίκτυο");
29    WiFi.begin(ssid, password);
30
31    while (WiFi.status() != WL_CONNECTED) {
32      delay(500);
33      Serial.print('.');
34    }
35    Serial.println(" Επιτυχής Σύνδεση!");
36  }
37
38  void loop() {
39    if (millis() - lastTime >= del) {
40      receivedata();
41      lastTime = millis();
42    }
43  }
```

Εικόνα 4.3.2 Συνάρτηση setup() ελέγχου φωτισμού

Η συνάρτηση receivedata() έχει ως σκοπό την αποστολή αιτημάτων, την λήψη και διαχείριση των δεδομένων. Η συνάρτηση πριν ξεκινήσει την επικοινωνία με τον διακομιστή, ελέγχει την κατάσταση της σύνδεσης με το δίκτυο, αν δεν υφίσταται σύνδεση η συνάρτηση τερματίζεται. Στην συνέχεια δημιουργούνται δυο αντικείμενα (client, http) τα οποία κληρονομούν συναρτήσεις από τις κλάσεις τους και διευκολύνουν στην επικοινωνία. Παρακάτω σχηματίζεται ο σύνδεσμος επικοινωνίας, γίνεται η προσπάθεια σύνδεσης και με σειριακά μηνύματα γίνεται ο έλεγχος της σύνδεσης. Εφόσον επιτευχθεί η σύνδεση, αρχίζει η άντληση των δεδομένων με την χρήση της βιβλιοθήκης ArduinoJSON όπως έχει αναλυθεί. Μέσο της http.getString() λαμβάνονται τα δεδομένα και αφότου μεταγλωττιστούν από την deserializeJson() τοποθετούνται στο αρχείο “doc” που δημιουργήθηκε με την εντολή DynamicJsonDocument. Συνεχίζοντας στην δομή επανάληψης γίνεται η επεξεργασία του Json Array που έχει φορτωθεί στο αρχείο doc και δημιουργείται ένα προσωρινό αντικείμενο για κάθε στοιχείο. Το κάθε προσωρινό αντικείμενο έχει id και status το οποίο εκτυπώνεται στο serial monitor. Με την βοήθεια εμφωλευμένης δομής επανάληψης και της strcmp() γίνεται η σύγκριση του id με το όνομα στον πίνακα LED_NAMES. Αν αυτά τα δυο string είναι ίδια, ελέγχεται με τον ίδιο ακριβώς τρόπο αν η τιμή του status είναι “On” και μόνο τότε ο ESP θέτει τον αντιστοιχώ ακροδέκτη σε κατάσταση HIGH. Τέλος η http.end() τερματίζει την επικοινωνία με τον διακομιστή και αποδεσμεύει την προσωρινή μνήμη.

```
void receivedata() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Αποτυχία Σύνδεσης!");
    return;
  }
  WiFiClient client;
  HTTPClient http;
  String url = String("http://") + host + "URL=" + API_KEY;
  Serial.print("Σύνδεση στον διακομιστή ");

  http.begin(client, url);
  int code = http.GET();
  Serial.print("Κωδικός Αναφοράς: ");
  Serial.println(code);

  if (code == HTTP_CODE_OK) {
    String receivedData = http.getString();

    DynamicJsonDocument doc(1024);
    DeserializationError error = deserializeJson(doc, receivedData);
    if (error) {
      Serial.print("deserializeJson() Απέτυχε: ");
      Serial.println(error.c_str());
      return;
    }

    for (JsonObject obj : doc.as<JsonArray>()) {
      const char* id = obj["id"];
      const char* status = obj["status"];
      Serial.print("LED: ");
      Serial.print(id);
      Serial.print(" => ");
      Serial.println(status);

      for (uint8_t i = 0; i < NUM_LEDS; i++) {
        if (strcmp(id, LED_NAMES[i]) == 0) {
          bool on = strcmp(status, "On") == 0;
          digitalWrite(LED_PINS[i], on ? HIGH : LOW);
        }
      }
    }
  } else {
    Serial.print("HTTP error: ");
    Serial.println(http.errorToString(code).c_str());
  }
  http.end();
}
```

Εικόνα 4.3.3 Συνάρτηση διαχείρισης ελέγχου φωτισμού

Εν κατακλείδι, Ο ESP έχοντας τον παραπάνω κώδικα καταφέρνει να συνδεθεί στο δίκτυο, να επικοινωνήσει με ασφάλεια με τον διακομιστή και να δέχεται δεδομένα από αυτόν. Επεξεργάζεται τα δεδομένα και τροποποιεί την κατάσταση του κάθε LED.

4.4 Περιβαλλοντικές Μετρήσεις

Ο παρών κώδικας στοχεύει στον προγραμματισμό του ESP ώστε να επικοινωνεί και να διαβάζει δεδομένα από τον DHT11. Χρησιμοποιώντας την ίδια μεθοδολογία σύνδεσης στο τοπικό δίκτυον το διακομιστή καθώς και τον ίδιο τρόπο ελέγχου για πιθανά σφάλματα. Θα χρειαστούν δύο μικρές προσθήκες και μια συνάρτηση για την επίτευξη του εγχειρήματος. Ξεκινώντας από τις προσθήκες θα πρέπει αρχικά να οριστεί ο ακροδέκτης που είναι συνδεδεμένος ο αισθητήρας (D6), να δηλωθεί ο τύπος του

αισθητήρα (DHT11) καθώς και να δημιουργηθεί το αντικείμενο τύπου DHT. Στη συνέχεια προστίθεται μέσα στην συνάρτηση setup(), η dht.begin() για να ξεκινήσει η επικοινωνία.

```
14  #define DHTPIN    D6
15  #define DHTTYPE   DHT11
16  DHT dht(DHTPIN, DHTTYPE);
```

```
22  void setup() {
23    Serial.begin(115200);
24    dht.begin();
```

Εικόνα 4.4.1 Αρχικές δηλώσεις DHT

Η συνάρτηση που αναλάβει την εκτέλεση όλων των απαραίτητων ενεργειών θα ονομαστεί sendData(). Αρχικά γίνεται η ανάγνωση και η ανάθεση των τιμών σε δυο μεταβλητές την T και την H. Στην συνέχεια με την χρήση της isnan() «Is Not a Number» ελέγχεται αν η μεταβλητές έχουν πάρει σαν τιμή αριθμούς. Ακολουθεί η εκτύπωσή τους στην σειριακή οθόνη και η δημιουργία επικοινωνίας με τον διακομιστή.

```
41  void sendData() {
42    float T = dht.readTemperature();
43    float H = dht.readHumidity();
44    if (isnan(T) || isnan(H)) {
45      Serial.println("ΣΦΑΛΜΑ: Αποτυχία ανάγνωσης DHT");
46      return;
47    }
48    Serial.println("--- ΝΕΑ ΜΕΤΡΗΣΗ ---");
49    Serial.print("Θερμοκρασία: "); Serial.print(T); Serial.println("
50    Serial.print("Υγρασία:      "); Serial.print(H); Serial.println(
51
52    // 2) Σύνδεση HTTP & GET
53    HTTPClient http;
54    WiFiClient client;
55    String url = String("http://") + host + "URL"
56                 + "?key=" + apiKey
57                 + "&temp=" + String(T,1)
58                 + "&hum="  + String(H,1);
59    Serial.print("Σύνδεση στον διακομιστή ");
60    http.begin(client, url);
61    int code = http.GET();
```

```
62    Serial.print("Κωδικός Αναφοράς DHT: ");
63    Serial.println(code);
64
65
66    if (code == HTTP_CODE_OK) {
67      String answer = http.getString();
68      Serial.print("Απάντηση server: ");
69      Serial.println(answer);
70      if (code == HTTP_CODE_OK && answer == "OK") {
71        Serial.println("Αποστολή: ΕΠΙΤΥΧΗΣ");
72      } else {
73        Serial.println("Αποστολή: ΑΠΟΤΥΧΗΣ (non-OK response
74      }
75    } else {
76      Serial.print("ΣΦΑΛΜΑ HTTP GET: ");
77      Serial.println(http.errorToString(code));
78    }
79    http.end();
80    Serial.println("------------------");
81  }
```

Εικόνα 4.4.2 Συνάρτηση διαχείρισης DHT

Την επιθυμητό αποτέλεσμα ότι τα δεδομένα λήφθηκαν και στάλθηκαν με επιτυχία επιβεβαιώνεται από τα μηνύματα στην σειριακή οθόνη.

```
..... Επιτυχής Σύνδεση!
--- ΝΕΑ ΜΕΤΡΗΣΗ ---
Θερμοκρασία: 23.10 °C
Υγρασία:      52.00 %
Σύνδεση στον διακομιστή Κωδικός Αναφοράς DHT: 200
Απάντηση server: OK
Αποστολή: ΕΠΙΤΥΧΗΣ
-------------------
```

Εικόνα 4.4.3 Μήνυμα σειριακής επικοινωνίας

4.5 Έλεγχος Συναγερμού

Η τελευταία εφαρμογή που μένει να υλοποιηθεί για ολοκληρωθεί ο προγραμματισμός του ESP8266 είναι ο έλεγχος τους συναγερμού. Η συγγραφή του κώδικα έγινε στο ίδιο μοτίβο με τους προηγούμενους. Ο τρόπος σύνδεσης με το τοπικό δίκτυο και τον διακομιστή δεν τροποποιήθηκε καθόλου. Ουσιαστικά και σε αυτόν τον αλγόριθμο έγινα οι απαραίτητες προσθήκες και ορθή λειτουργία του συναγερμού.

Αρχικά προστέθηκε η δήλωση για τους αντίστοιχους ακροδέκτες που θα συνδεθούν ο αισθητήρας κίνησης(D7) και το buzzer(D5). Στην συνέχεια γίνεται ο ορισμός τριών μεταβλητών που θα βοηθήσουν για τον έλεγχο της κατάστασης τους συναγερμού και του buzzer καθώς και έναν μετρητή για τα περιοδικά αιτήματα.

```
11  const int motionPin = D7;
12  const int buzzerPin = D5;
13
14  bool alarmStatus = false;
15  bool buzzerActive = false;
16  unsigned long lastTime3 = 0;
```

Εικόνα 4.5.1 Αρχικές δηλώσεις συναγερμού

Στην συνέχει στην συνάρτηση setup() πραγματοποιείται η έναρξη της σειριακής επικοινωνίας, η σύνδεση στο τοπικό δίκτυο και ορισμός των ακροδεκτών. Πιο συγκεκριμένα ο ακροδέκτης που είναι συνδεδεμένος ο αισθητήρας κίνησης ορίστηκε σαν είσοδο, ενώ ο ακροδέκτης του buzzer σε έξοδο με κατάσταση Low.

```
18  void setup() {
19    Serial.begin(115200);
20    pinMode(motionPin, INPUT);
21    pinMode(buzzerPin, OUTPUT);
22    digitalWrite(buzzerPin, LOW);
23
24    WiFi.begin(ssid, password);
25    Serial.print("Σύνδεση στο WiFi");
26    while (WiFi.status() != WL_CONNECTED) {
27      delay(500);
28      Serial.print('.');
29    }
30    Serial.println("\nΕπιτυχής Σύνδεση!");
31  }
```

Εικόνα 4.5.2 Συνάρτηση setup() συναγερμού

Συνεχίζοντας στην συνάρτηση loop() εντοπίζεται μια πιο σύνθετη δομή επανάληψης από τις υπόλοιπες εφαρμογές. Ωστόσο αυτό που υλοποιεί στην πραγματικότητα είναι αρκετά απλοϊκό. Ουσιαστικά η κλήση της συνάρτησης επαναλαμβάνεται ανά δύο δευτερόλεπτα και πραγματοποιούνται οι εξής έλεγχοί.

- Αν η κατάσταση του συναγερμού είναι ενεργοποιημένη και η κατάσταση του buzzer δεν είναι ήδη ενεργοποιημένη σε συνδυασμό με ανίχνευση κίνησης από τον αισθητήρα. Τότε η μεταβλητή buzzerActive παίρνει την τιμή true και εμφανίζει το αντίστοιχο μήνυμα.
- Όσο ισχύει ότι το buzzerActive είναι true η ηχητική ειδοποίηση δεν σταματάει να λειτουργεί.
- Μόλις δοθεί η εντολή από την ιστοσελίδα για απενεργοποίηση συναγερμού, Η συνθήκη if δεν είναι πλέον αληθής και εκτελείτε η else που απενεργοποιεί την ηχητική ειδοποίηση και επαναφέρει την τιμή του buzzerActive σε false.

```
33  void loop() {
34    unsigned long now = millis();
35
36    if (now - lastTime3 > 2000) {
37      lastTime3 = now;
38      receivestatus();
39    }
40
41    if (alarmStatus) {
42      if (!buzzerActive && digitalRead(motionPin) == HIGH) {
43        buzzerActive = true;
44        Serial.println("Κίνηση ανιχνεύθηκε! Ηχητική ειδοποίηση!");
45      }
46      if (buzzerActive) {
47        digitalWrite(buzzerPin, HIGH);
48      } else {
49        digitalWrite(buzzerPin, LOW);
50      }
51    } else {
52      buzzerActive = false;
53      digitalWrite(buzzerPin, LOW);
54    }
55
56    if (WiFi.status() != WL_CONNECTED) {
57      WiFi.reconnect();
58    }
59  }
```

Εικόνα 4.5.3 Συνάρτηση loop() συναγερμού

Ολοκληρώνοντας τον αλγόριθμο εντοπίζεται η κομβική συνάρτηση receivestatus() η οποία εκτελείται όταν υπάρχει σύνδεση στο τοπικό δίκτυο. Στην συνέχεια στέλνει αίτημα στον διακομιστή, λαμβάνει τα δεδομένα και τα επεξεργάζεται μέσο του αντικειμένου cmd.

```
void receivestatus() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Αποτυχία Σύνδεσης!");
    return;
  }
  WiFiClient client;
  HTTPClient http;
  String url = String("http://") + host + "URL" + apiKey;
  http.begin(client, url);
  int code = http.GET();

  if (code == HTTP_CODE_OK) {
    String json = http.getString();

    DynamicJsonDocument doc(256);
    DeserializationError error = deserializeJson(doc, json)
    if (error) {
      Serial.print("deserializeJson() Απέτυχε: ");
      Serial.println(error.c_str());
      http.end();
      return;
    }
    for (JsonObject cmd : doc.as<JsonArray>()) {
      const char* state = cmd["state"];
      bool newStatus = (strcmp(state, "On") == 0);
      if (alarmStatus != newStatus) {
        alarmStatus = newStatus;
        Serial.print(" Ο συναγερμός είναι ");
        Serial.println(alarmStatus ? "Ενεργοποιημένος" : "Απενεργοποιημένος

        if (!alarmStatus) buzzerActive = false;
      }
    }
  } else {
    Serial.print("Σφάλμα σύνδεσης HTTP error: ");
    Serial.println(http.errorToString(code));
  }
  http.end();
}
```

Εικόνα 4.5.4 Συνάρτηση διαχείρισης συναγερμού

5.Προγραματισμός ESP32 CAM

5.1 Εισαγωγή

Η ανάπτυξη του αλγορίθμου της ESP32 πραγματοποιήθηκε στο περιβάλλον του λογισμικού εργαλείου Arduino IDE. Τα παραδείγματα που προσφέρει το συγκεκριμένο εργαλείο συνέβαλαν ουσιαστικά στην συγγραφή του κώδικα. Το έργο που θα εκτελεί ο κώδικας που υλοποιήθηκε είναι η προσομοίωση ζωντανής προβολής βίντεο με συνεχή αποστολή εικόνων. Η τεχνολογία επικοινωνίας που επιλέχθηκε(HTTP Polling) δεν παρέχει την δυνατότητα ζωντανής μετάδοσης βίντεο. Γι' αυτό τον λόγω η συγκεκριμένη υλοποίηση πραγματοποιεί λήψη εικόνας και αποστολή στον διακομιστή για προβολή. Η συχνότητα αποστολής είναι πέντε εικόνες ανά δευτερόλεπτο, ουσιαστικά αυτή η γρήγορη εναλλαγή εικόνων είναι που προσφέρει στον χρήστη την αίσθηση του βίντεο.

5.2 Βιβλιοθήκες

5.2.1 WiFi.h

Η παρούσα βιβλιοθήκη είναι αντίστοιχη της ESP8266WiFi που αναλύθηκε στο κεφάλαιο 4. Χρησιμοποιείται στην πλακέτα ESP32 για την επίτευξη σύνδεση στο τοπικό δίκτυο. Η διαδικασία και η χρήση των συναρτήσεων είναι ίδια με τις υπόλοιπες εφαρμογές.

5.2.2 Esp_camera.h & Camera_pins.h

Η Esp_camera.h παρέχει τις απαραίτητες εντολές και συναρτήσεις για την ενεργοποίηση και την λειτουργία της κάμερας. Αρχικά Μέσα από αυτή τη βιβλιοθήκη χρησιμοποιήθηκε η δομή camera_config_t, η οποία επιτρέπει τον καθορισμό όλων των ρυθμίσεων της κάμερας (αντιστοιχία ακροδεκτών, ανάλυση, format εικόνας κ.ά.). Επιπλέον χρησιμοποιήθηκαν οι εξής συναρτήσεις:

- esp_camera_init() ενεργοποιεί την κάμερα.

- esp_camera_fb_get() τραβάει μία φωτογραφία και επιστρέφει τα δεδομένα εικόνας.

- esp_camera_fb_return() ελευθερώνει τη μνήμη μετά την αποστολή.

Η βιβλιοθήκη camera_pins.h περιλαμβάνει τους ορισμούς των ακροδεκτών που χρησιμοποιούνται στην camera_config_t. Παρέχει δηλαδή τις μεταβλητές Y2_GPIO_NUM, XCLK_GPIO_NUM, SIOD_GPIO_NUM και πολλές άλλες που αντιστοιχούν σε συγκεκριμένα GPIO.

5.3 Υλοποίηση κώδικα

Ο κώδικας αρχίζει με την δήλωση του μοντέλου της κάμερα που θα χρησιμοποιηθεί και την συμπερίληψη των απαραίτητων βιβλιοθηκών. Στην συνέχεια προστίθενται οι απαραίτητες μεταβλητές για την σύνδεση δικτύου και διακομιστή.

```
#define CAMERA_MODEL_AI_THINKER

#include "esp_camera.h"
#include <WiFi.h>
#include "camera_pins.h"
#include <HTTPClient.h>

const char* ssid     = "ssid";
const char* password = "pass";
const char* serverUrl = "url";
const char* apikey = "apikey";
```

Εικόνα 5.3.1

Στην συνάρτηση setup() γίνεται η έναρξη της σειριακής επικοινωνίας για έλεγχο σφαλμάτων και καθορισμός της κάμερες μεσώ της δομής camera_config_t. Στη συνέχεια πραγματοποιείται ο ορισμός των στοιχείων που αφορούν την εικόνα(μέγεθος, τύπος, ποιότητα κ.α.). Πριν το κλείσιμο της συνάρτησης γίνεται η αρχικοποίηση της κάμερας με έλεγχο και σφάλματα καθώς και η σύνδεση στο τοπικό δίκτυο.

```
void setup() {
  Serial.begin(115200);

  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_QVGA;
  config.pixel_format = PIXFORMAT_JPEG;
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 35;
  config.fb_count = 1;

  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Η έναρξη της κάμερα απέτυχε", err);
    while (1) delay(1000);
  }


  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\n Επιτυχής σύνδεση στο τοπικό δίκτυο");
}
```

Εικόνα 5.3.2 Συνάρτηση setup() κάμερας

Ο κώδικας που περιέχεται μέσα στην συνάρτηση loop() εφόσον υπάρχει σύνδεση στο τοπικό δίκτυο προβαίνει σε λήψη φωτογραφίας. Αφότου ληφθεί η εικόνα προχωρά σε αποστολή μηνύματος στον διακομιστή(HTTT POST) όπου στο σώμα του μηνύματος περιέχεται η ληφθείσα φωτογραφία. Τέλος με την προσθήκη καθυστέρησης(delay(200)) επιτυγχάνεται η λήψη των πέντε καρέ ανά δευτερόλεπτο.

```
void loop() {
  if (WiFi.status() == WL_CONNECTED) {
    camera_fb_t *fb = esp_camera_fb_get();
    if (fb) {
      HTTPClient http;
      http.begin(serverUrl);
      http.addHeader("Content-Type", "image/jpeg");
      http.addHeader("X-API-KEY", apikey);
      int code = http.POST(fb->buf, fb->len);
      if (code == 200) {
        Serial.println("Αποστολή εικόνας");
      } else {
        Serial.printf("Αποτυχία αποστολής %d\n", code);
      }
      http.end();
      esp_camera_fb_return(fb);
    } else {
      Serial.println("Αποτυχία λήψης εικόνας");
    }
  } else {
    Serial.println("Αδυναμια σύνδεσης WIFI");
  }
  delay(200);
}
```

Εικόνα 5.3.3 Συνάρτηση loop() κάμερας

6.Προγραματισμός Frontend

6.1 Εισαγωγή

Ο έλεγχος του συστήματος πραγματοποιείται αποκλειστικά από την ιστοσελίδα. Το γραφικό περιβάλλον σχεδιάστηκε με σκοπό να συνδυάζει την απλότητα της εμφάνισης και την ευκολία χρήσης. Η ιστοσελίδα αποτελείται από δύο επιμέρους σελίδες, την σελίδα σύνδεσης χρήστη και την σελίδα ελέγχου. Η πρόσβαση στην σελίδα ελέγχου επιτρέπεται μόνο σε χρήστες που είναι εγγεγραμμένοι στην βάση δεδομένων από τον διαχειριστή. Σε αυτό το κεφάλαιο παρουσιάζονται οι αλγόριθμοι που συντέλεσαν στην διαμόρφωση της εμφάνισης

6.2 Σελίδα Σύνδεσης Χρήστη

Η σελίδα σύνδεσης αποτελείται από μια φόρμα συμπλήρωσης στοιχείων γραμμένη σε html, το αρχείο CSS για την εμφάνιση καθώς και δύο PHP scripts.

Ο κώδικας ξεκινάει με την ενσωμάτωση του αρχείου για την σύνδεση με την βάση δεδομένων και τις βασικές δηλώσεις όπως η γλώσσα της σελίδας, τον τύπο χαρακτήρων, τον τίτλο καθώς και τον σύνδεσμο για την σύνδεση με το αρχείο εμφάνισης.

```php
<?php
include 'dbcon.php';
?>
<!DOCTYPE html>
<html lang="el">
<head>
  <meta charset="utf-8">
  <title>Login</title>
  <link rel="stylesheet" href="logstyl.css">
</head>
<body>
  <div class="container">
    <?php if (!empty($error)): ?>
      <div class="error"><?= htmlspecialchars($error) ?></div>
    <?php endif; ?>
    <form class="login-form" method="post" action="index.php">
      <h1>Login</h1>
      <label>Χρήστης:</label>
      <input type="text" name="username"
       placeholder="Enter Username" required>
      <label>Κωδικός:</label>
      <input type="password" name="password"
       placeholder="Enter Password" required>
      <button type="submit">Είσοδος</button>
    </form>
  </div>
</body>
</html>
```

Εικόνα 6.2.1 Σελίδα σύνδεσης χρήστη.

Το πρώτο PHP script ενσωματώνει το αρχείο που δημιουργεί την σύνδεση με την βάση δεδομένων για τον έλεγχο των στοιχείων σύνδεσης.

Το δεύτερο PHP script έχει δύο ρόλους. Ο πρώτος είναι να εμφανίζει τα μηνύματα σφάλματός και ο δεύτερος μετατρέπει ειδικούς χαρακτήρες σε απλή μορφή. Ουσιαστικά

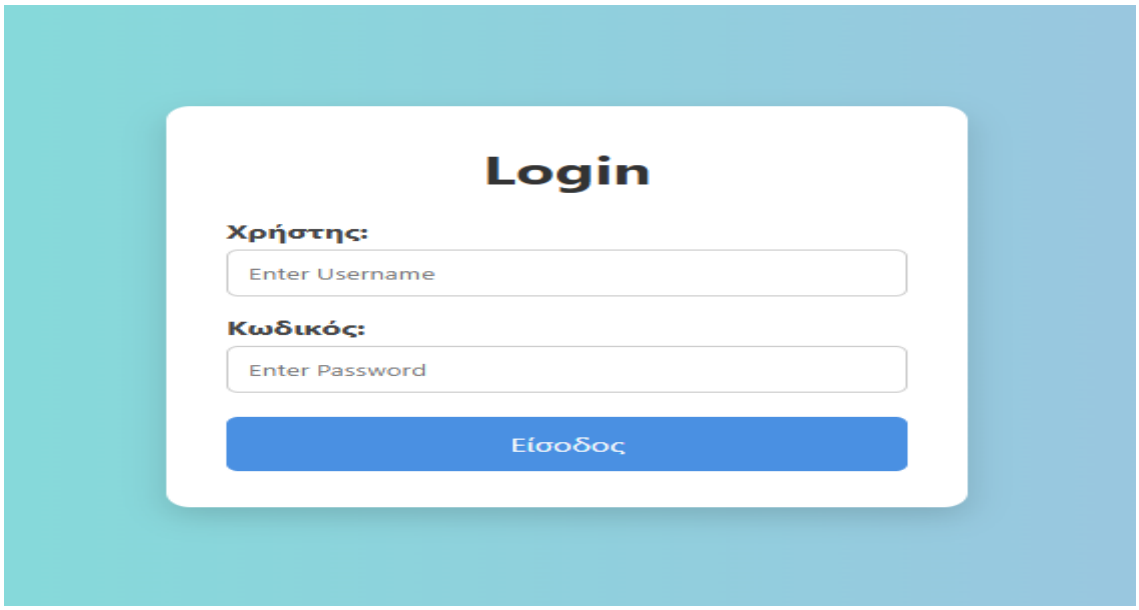
αποτρέπει ένα μην εξουσιοδοτημένο κακόβουλο χρήστη να εισάγει κάποιο script στην φόρμα συμπλήρωσης.

Στο αρχείο CSS καθορίζεται η εμφάνιση της κάθε κλάσης που έχει οριστεί στον κώδικα HTML.

```css
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

body {
  background: linear-gradient(to right, #74ebd5, #acb6e5);
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
}

.container {
  background-color: white;
  padding: 30px;
  border-radius: 12px;
  box-shadow: 0 8px 20px rgba(0, 0, 0, 0.15);
  width: 100%;
  max-width: 400px;
}

.login-form h1 {
  text-align: center;
  margin-bottom: 20px;
  color: #333;
}

.login-form label {
  display: block;
  margin-top: 15px;
  margin-bottom: 5px;
  color: #444;
  font-weight: bold;
}
```

```css
.login-form input {
  width: 100%;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 6px;
  transition: border 0.3s;
}

.login-form input:focus {
  border-color: #74ebd5;
  outline: none;
}

.login-form button {
  width: 100%;
  padding: 12px;
  background-color: #4a90e2;
  color: white;
  border: none;
  border-radius: 6px;
  font-size: 16px;
  margin-top: 20px;
  cursor: pointer;
  transition: background-color 0.3s
}

.login-form button:hover {
  background-color: #357ab8;
}

.error {
  background-color: #f8d7da;
  color: #842029;
  border: 1px solid #f5c2c7;
  padding: 10px;
  margin-bottom: 15px;
  border-radius: 6px;
  text-align: center;
}
```

Εικόνα 6.2.2 CSS σελίδας σύνδεσης χρήστη



Εικόνα 6.2.3 Εμφάνιση σελίδας σύνδεσης χρήστη

6.3 Σελίδα Διαχείρισης Συστήματος

Στην σελίδα αυτή ενσωματώνονται οι τέσσερις εφαρμογές του συστήματος. Στο γραφικό της περιβάλλον έχουν δημιουργηθεί 4 κυτία προβολής ώστε η κάθε εφαρμογή να διαθέτει ξεχωριστό χώρο ενσωμάτωσης. Κάθε εφαρμογή διαθέτει γραφική απεικόνιση που προβάλλεται στο αντίστοιχο κυτίο.

6.3.1 Γραφική απεικόνιση ελέγχου φωτισμού

Η διαχείριση του φωτισμού πραγματοποιείται με την χρήση checkbox όπου η κατάσταση On/Off καθορίζεται από το αν το αντίστοιχο checkbox είναι επιλεγμένο η όχι. Για τον έλεγχο της κατάστασης των checkbox δημιουργήθηκε ένα αρχείο JavaScript που διαβάζει αν είναι επιλεγμένο η όχι το κάθε κουτί και ενημερώνει την βάση δεδομένων. Το αρχείο CSS μεταμορφώνει την εμφάνιση δυο απλών κυτίων σε ρεαλιστικό ψηφιακό διακόπτη.

Το αρχείο JS εκτελεί ορισμένες ενέργειες με την παρακάτω σειρά:

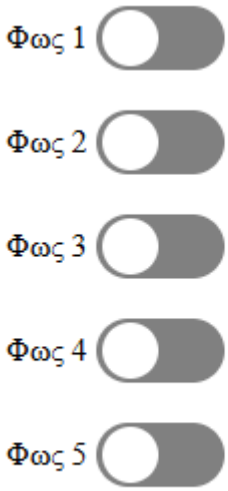
- Επιλέγει όλα τα checkbox που το όνομά τους αρχίζει από «L».
- Προσθέτει έναν ακροατή γεγονότων στο κάθε checkbox που ενεργοποιείται όταν κλικάρει ο χρήστης.
- Εμφανίζει μήνυμα στην κονσόλα για την αντίστοιχη αλλαγή και προετοιμάζει την αποστολή led & state.
- Στέλνει με post αίτημα τα δεδομένα στο αρχείο lightsupdate.php.
- Εμφανίζει μηνύματα επιτυχούς αποστολής η σφάλματος.

```
document.querySelectorAll('input[name^="L"]').forEach(chk => {
  chk.addEventListener('change', () => {
    console.log('Checkbox clicked:', chk.name, chk.checked);

    const led   = chk.name;
    const state = chk.checked ? 'On' : 'Off';

    fetch('lightsupdate.php', {
      method: 'POST',
      credentials: 'include',
      headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
      body: `led=${encodeURIComponent(led)}&state=${state}`
    })
    .then(res => {
      console.log('Response status:', res.status);
      if (!res.ok) throw new Error(`HTTP ${res.status}`);
      return res.json();
    })
    .then(json => {
      console.log('Response JSON:', json);
      if (!json.success) {
        alert(`Σφάλμα: ${json.error}`);
        chk.checked = !chk.checked;
      }
    })
    .catch(err => {
      console.error('Fetch error:', err);
      alert(`Σφάλμα δικτύου: ${err.message}`);
      chk.checked = !chk.checked;
    });
```

Εικόνα 6.3.1.1 Αρχείο JS ελέγχου φωτισμού

Στο αρχείο HTML εντοπίζεται το περιεχόμενο της σελίδας το αποτελείται από ένα πίνακα πέντε γραμμών και δύο στηλών. Η κάθε γραμμή περιέχει το όνομα και το checkbox του αντίστοιχου LED. Στον κώδικα αυτόν υπάρχει ένα PHP script όπου ελέγχει αν ο χρηστής είναι συνδεδεμένος ώστε να αποτρέψει την πρόσβαση σε μη εξουσιοδοτημένους χρήστες. Επίσης ενσωματώνονται τα αρχεία JS και CSS. Ο αναλυτικός κώδικας βρίσκεται στο ΠΑΡΑΡΤΗΜΑ Α.

Η τελική μορφή του ψηφιακού διακόπτη επιτυγχάνεται με τον κώδικα CSS και αποτυπώνεται στην εικόνα.



Εικόνα 6.3.1.2 Απεικόνιση διακοπτών

6.3.2 Γραφική απεικόνιση περιβαλλοντικών μετρήσεων.

Η προβολή των μετρήσεων θερμοκρασίας και υγρασίας πραγματοποιείται στο ίδιο πλαίσιο. Μια απλή εμφάνιση με τις μετρήσεις και ένα ψηφιακό ρολόι είναι το περιεχόμενο του HTML κώδικα. Με την χρήση της συνάρτησης updateClock ενημερώνεται το ρολόι ανά δευτερόλεπτο ενώ με την χρήση της receivedht πραγματοποιείται η λήψη των μετρήσεων.

```
<!DOCTYPE html>
<html lang="el">
<head>
  <meta charset="UTF-8">
  <title>Θερμοκρασία & Υγρασία</title>
  <link rel="stylesheet" href="dhtstyl.css">
</head>
<body>
  <h1>🕒 <span id="clock">--:--:--</span></h1>

  <div class="label">🌡️ Θερμοκρασία</div>
  <div class="value" id="temp">-- °C</div>

  <div class="label">💧 Υγρασία</div>
  <div class="value" id="hum">-- %</div>

  <script>

    function updateClock() {
      const now = new Date();
      const hh = String(now.getHours()).padStart(2, '0');
      const mm = String(now.getMinutes()).padStart(2, '0');
      const ss = String(now.getSeconds()).padStart(2, '0');
```

```
    setInterval(updateClock, 1000);
    updateClock();


    async function receivedht() {
      try {
        const res = await fetch('get.php');
        const data = await res.json();
        if (!data.error) {
          document.getElementById('temp').textContent = data.temp.toFixed(1) + ' °C';
          document.getElementById('hum').textContent  = data.hum.toFixed(1)  + ' %';
        }
      } catch (e) {
        console.error('Fetch error', e);
      }
    }
    setInterval(receivedht, 60000);
    receivedht();
  </script>
</body>
</html>
```

Εικόνα 6.3.2.1Κωδικας HTML DHT11

Με την προσθήκη του αρχείου CSS επιτυγχάνεται η τελική διαμόρφωση της εμφάνισης.

```css
body {
  background: #4a90e2;
  color: rgb(255, 255, 255);
  font-family: 'Courier New', monospace;
  text-align: center;
  margin: 0;
  padding: 1rem;
}

h1 {
  margin-bottom: 1rem;
}

.label {
  font-size: 1.2rem;
  margin-top: 0.5rem;
}

.value {
  font-size: 2rem;
  margin-bottom: 1rem;
}
```

Εικόνα 6.3.2.2 Απεικόνιση DHT

6.3.3 Γραφική απεικόνιση Συναγερμού.

Στο ίδιο πλαίσιο υλοποίησης δημιουργήθηκε και η γραφική απεικόνιση του συναγερμού. Δύο κουμπιά ενεργοποίησης/απενεργοποίησης που συνδυάζονται με ένα αρχείο JS για την λειτουργία του συναγερμού.

Το αρχείο JS εκτελεί συγκεκριμένες λειτουργίες με σκοπό την διαχείριση της ενεργοποίησης/απενεργοποίησης του συναγερμού. Αρχικά με την χρήση των συναρτήσεων loadStatus() και UpdateUI() λαμβάνει την τελευταία κατάσταση από την βάση δεδομένων και ενημερώνει το περιβάλλον χρήστη.

```
function loadStatus() {
  fetch('get_alarm_state.php', { credentials:'same-origin' })
    .then(r => r.json())
    .then(json => {
    updateUI(json.state);
  })
    .catch(console.error);
}

function updateUI(state) {
  if (state === 'On') {
    status.textContent = 'Ενεργοποιημένος';
    status.classList.add('on');
  } else {
    status.textContent = 'Απενεργοποιημένος';
    status.classList.remove('on');
  }
}
```

Εικόνα 6.3.3.1 Κώδικας loadstatus

Η συνάρτηση sendCommand(state) ενημερώνει την κατάσταση του συναγερμού στην βάση δεδομένων και καλεί την updateUI(). Καλείται κάθε φορά που ο χρήστης κλικαρεί ένα από τα διαθέσιμα κουμπιά.

Τέλος εκτελείται η loadstatus() μια φορά για την φόρτωση της κατάστασης

```
function sendCommand(state) {
  fetch('alarmdbupdate.php', {
    method: 'POST',
    credentials: 'same-origin',
    headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
    body: `temp=ALARM&state=${state}`
  })
  .then(r => r.json())
  .then(json => {
    if (json.success) updateUI(state);
    else              alert(`Σφάλμα: ${json.error}`);
  })
  .catch(err => {
    alert(`Network error: ${err.message}`);
  });
}

btnon.addEventListener('click', () => sendCommand('On'));
btnoff.addEventListener('click', () => sendCommand('Off'));

```

Εικόνα 6.3.3.2 Κώδικας sendCommand

Με την προσθήκη του σώματος HTML και του κώδικα CSS ολοκληρώνεται και η εμφάνιση της εφαρμογής του συναγερμού. Η αλγόριθμοι βρίσκονται στο ΠΑΡΑΡΤΗΜΑ Β .

Εικόνα 6.3.3.3 Απεικόνιση συναγερμού

6.3.4 Προβολή ζωντανής ροής εικόνων

Η προσομοίωση της ζωντανής μετάδοσης βίντεο όπως προαναφέρθηκε πραγματοποιείται με την γρήγορη λήψη και εμφάνιση εικόνων. Στο πλαίσιο αυτό σχηματίστηκε μια απλή ιστοσελίδα με ίδιο στυλ που εμφανίζει απλά την εικόνα.

```html
<!DOCTYPE html>
<html lang="el">
<head>
  <meta charset="UTF-8">
  <title>Live Camera</title>
  <style>
    body {
      background: #4a90e2;
      margin: 0;
      height: 100vh;
      display: flex;
      align-items: center;
      justify-content: center;
    }
    #camera-stream {
      width: 320px;
      height: 240px;
      background: #111;
      display: block;
      object-fit: contain;
      border: 2px solid #333;
      box-shadow: 0 2px 8px #0006;
    }
  </style>
</head>
<body>
  <img id="camera-stream" src="/uploads/latest.jpg"
   alt="Live Stream">
  <script>
    setInterval(() => {
      document.getElementById('camera-stream').src =
      '/uploads/latest.jpg?ts=' + Date.now();
    }, 500);
  </script>
</body>
</html>
```

Εικόνα 3.4.1 Κώδικας HTML κάμερας

Λόγω του τρόπου επικοινωνίας της ESP32 με τον διακομιστή δεν ήταν δυνατό να επιτευχθεί η αποστολή πολλών εικόνων ανά δευτερόλεπτο και σε καλή ποιότητα. Αναπόφευκτα πραγματοποιήθηκε έκπτωση στην ανάλυση ώστε να επιτυγχάνεται η σταθερή ροή εικόνων.

Εικόνα 6.3.4.2 Απεικόνιση κάμερας

6.3.5 Σελίδα διαχείρισης

Η σελίδα διαχείρισης συστήματος αποτελεί το σημείο ενσωμάτωσης όλων τον εφαρμογών. Με αυτόν τον τρόπο αντί ο χρήστης να αναζητά την σελίδα κάθε εφαρμογής, μπορεί να ελέγχει ολόκληρο το σύστημα μόνο από την σελίδα διαχείρισης. Η τελική διαμόρφωση της σελίδας μετά την ενσωμάτωση των εφαρμογών στα τέσσερα κυτία προβολής αποτυπώνεται στην εικόνα. Οι κώδικες βρίσκονται στα παραρτήματα Γ.

Εικόνα 6.3.5 Απεικόνιση δωματίου ελέγχου

7.Προγραμματισμός Backend

7.1 Εισαγωγή

Όπως έχει αναφερθεί για να λειτουργήσει η επικοινωνία μεταξύ των μικροελεγκτών και της ιστοσελίδας, υπάρχει ένα τμήμα από κώδικες που οι ενέργειές τους δεν είναι άμεσα ορατές στον χρήστη. Αυτό το τμήμα θα παρουσιαστεί στον παρόν κεφάλαιο.

Τα αρχεία που είναι υπεύθυνα για την αποστολή και λήψη δεδομένων ονομάζονται endpoints και κάθε εφαρμογή χρησιμοποιεί το δικό της. Έχουν σχεδιαστεί με παρόμοιο τρόπο εξυπηρετώντας τις ανάγκες κάθε εφαρμογής.

Σε όλες τις εφαρμογές πλην της κάμερας, γίνεται χρήση βάσης δεδομένων. Αυτό σημαίνει ότι το πρώτο βασικό αρχείο μετά τα endpoints, είναι το αρχείο που δημιουργεί αυτήν την σύνδεση με την βάση. Για την απλή σύνθεση του αρχείου χρειάζονται τα στοιχεία της βάσης δεδομένων και ένα αίτημα σύνδεσης.

```
10  define('DB_HOST', '');
11  define('DB_NAME', '');
12  define('DB_USER', '');
13  define('DB_PASS', '');
```

Εικόνα 7.1.1 Δήλωση στοιχείων βάσης δεδομένων

```
$pdo = new PDO(
    "mysql:host=".DB_HOST.";dbname=".DB_NAME.";charset=utf8mb4",
    DB_USER,
    DB_PASS,
    [ PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION ]
);
```

Εικόνα 7.1.2 Αίτημα σύνδεσης στην βάση δεδομένων

7.2 Έλεγχος φωτισμού

Το αρχείο JS που διαχειρίζεται την κατάσταση των διακοπτών, στέλνει τις αλλαγές στην βάση δεδομένων. Αυτό γίνεται με την βοήθεια του παρακάτω αρχείου.

Αρχικά ορίζεται ο τύπος απάντησης από τον διακομιστή και ενσωματώνεται το αρχείο σύνδεσης με την βάση. Στην συνέχεια απορρίπτονται αιτήματα που δεν είναι της μορφής POST και δηλώνονται οι μεταβλητές για την ανάγνωση των δεδομένων. Τέλος πραγματοποιείται έλεγχος εγκυρότητας των ληφθείσων παραμέτρων και η αποστολή στην βάση.

```php
<?php
header('Content-Type: application/json');
require_once 'lightscon.php';

if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    echo json_encode(['success' => false, 'error' => 'Invalid request method.']);
    exit;
}

$led   = $_POST['led']   ?? '';
$state = $_POST['state'] ?? '';

$valid_leds   = ['L1','L2','L3','L4','L5'];
$valid_states = ['On','Off'];
if (!in_array($led, $valid_leds) || !in_array($state, $valid_states)) {
    echo json_encode(['success' => false, 'error' => 'Invalid parameters.']);
    exit;
}

try {
    $stmt = $pdo->prepare("UPDATE leds SET status = :state WHERE id = :led");
    $stmt->execute([':state' => $state, ':led' => $led]);
    echo json_encode(['success' => true]);
} catch (PDOException $e) {
    echo json_encode(['success' => false, 'error' => $e->getMessage()]);
}
```

Εικόνα 7.2.1 Κώδικας ενημέρωσης βάσης δεδομένων

Με παρόμοιο τρόπο λειτουργεί ο endpoint της εφαρμογής. Οι διαφορές είναι ότι δέχεται αιτήματα GET, ελέγχει το κλειδί ασφαλείας και αντί να στέλνει δεδομένα πραγματοποιεί ανάκτηση από την βάση.

```php
<?php
header('Content-Type: application/json');
require_once 'lightscon.php';

if ($_SERVER['REQUEST_METHOD'] !== 'GET') {
    echo json_encode(['success' => false, 'error' => 'Invalid request method.']);
    exit;
}

$key = $_GET['key'] ?? '';
if ($key !== $apiKey) {
    echo json_encode(['success' => false, 'error' => 'Invalid API key.']);
    exit;
}

try {
    $stmt = $pdo->query("SELECT id, status FROM leds");
    $leds = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($leds);

} catch (PDOException $e) {
    echo json_encode(['success' => false, 'error' => $e->getMessage()]);
```

Εικόνα 7.2.2 Κώδικας απάντησης

7.2 Περιβαλλοντικές μετρήσεις

Με μικρές παραλλαγές στον κώδικα αλλά κρατώντας την ίδια μεθοδολογία λειτουργεί το backend της παρούσας εφαρμογής.

Ο ρόλος του endpoint είναι να λαμβάνει τις τιμές από τον αισθητήρα και να ενημερώνει την βάση. Πραγματοποιώντας τους ίδιους ελέγχους με την πρώτη εφαρμογή και προσθέτοντας τον κώδικα για την διαχείριση των απαιτούμενων δεδομένων, επιτυγχάνεται η ενημέρωση της βάσης.

```
if (!isset($_GET['temp'], $_GET['hum'])) {
    http_response_code(400);
    echo 'Missing parameters';
    exit;
}

$temp = floatval($_GET['temp']);
$hum  = floatval($_GET['hum']);

$sql = "REPLACE INTO dht (id, temp, hum, ts)
        VALUES (1, :temp, :hum, NOW())";
$stmt = $pdo->prepare($sql);
$stmt->execute([
    ':temp' => $temp,
    ':hum'  => $hum
]);
```

Εικόνα 7.2.1 Κώδικας λήψης μετρήσεων

Για την εξαγωγή δεδομένων τροποποιούμε το αίτημα προς στην βάση.

```
$stmt = $pdo->prepare('SELECT temp, hum, ts FROM dht WHERE id = 1');
$stmt->execute();
$row = $stmt->fetch(PDO::FETCH_ASSOC);

if ($row) {
    echo json_encode([
        'temp' => round($row['temp'], 1),
        'hum'  => round($row['hum'], 1),
        'ts'   => $row['ts']
    ], JSON_UNESCAPED_UNICODE);
} else {
    http_response_code(404);
    echo json_encode(['error' => 'No data'], JSON_UNESCAPED_UNICODE);
```

Εικόνα 7.2.2 Κώδικας εξαγωγής δεδομένων.

7.3 Έλεγχος Συναγερμού

Για την λειτουργία αυτής της εφαρμογής χρησιμοποιήθηκαν τρία αρχεία ώστε να γίνεται η ενημέρωση της βάσης, η ανάκτηση και η αποστολή από αυτήν προς τον ESP και η φόρτωση της τελευταίας κατάστασης.

Στο ίδιο πλαίσιο ενημέρωσης της βάσης τροποποιείται μόνο το τμήμα κώδικα για την διαχείριση των σωστών μεταβλητών.

```
$state = (isset($_POST['state']) && $_POST['state'] === 'On') ? 'On' : 'Off';

try {
    $stmt = $pdo->prepare("UPDATE alarm_commands SET state = ? WHERE id = 1");
    $stmt->execute([$state]);

    echo json_encode(['success' => true, 'state' => $state]);
} catch (Exception $e) {
    http_response_code(500);
    echo json_encode(['success' => false, 'error' => 'DB error']);
}
```

Εικόνα 7.3.1 Κώδικας ενημέρωσης βάσης δεδομένων

Το πρόσθετο αρχείο σε σχέση με τις υπόλοιπες εφαρμογές βοηθάει στην διατήρηση της κατάστασης του συναγερμού έπειτα από ανανέωση της ιστοσελίδας. Ουσιαστικά ανακτά από την βάση την καταχωρημένη κατάσταση.

```
<?php
require_once 'alarmcon.php';
header('Content-Type: application/json');
try {
    $stmt = $pdo->prepare("SELECT state FROM alarm_commands WHERE id = 1");
    $stmt->execute();
    $row = $stmt->fetch(PDO::FETCH_ASSOC);
    $state = $row ? $row['state'] : 'Off';
    echo json_encode(['state' => $state]);
} catch (Exception $e) {
    http_response_code(500);
    echo json_encode(['error' => 'DB error']);
}
```

Εικόνα 7.3.2 Κώδικας ελέγχου κατάστασης

Τέλος ο endpoint τροποποιημένος μόνο στο κομμάτι της απάντησης, αποστέλλει την κατάσταση του συναγερμού σε κάθε αίτημα του ESP

```php
try {
    $stmt = $pdo->prepare("SELECT state FROM alarm_commands WHERE id = 1");
    $stmt->execute();
    $row = $stmt->fetch(PDO::FETCH_ASSOC);

    $state = $row ? $row['state'] : 'Off';
    echo json_encode([['state' => $state]]);

} catch (Exception $e) {
    http_response_code(500);
    echo json_encode(['error' => 'DB error', 'message' => $e->getMessage()]);
}
```

Εικόνα 7.3.3 Κώδικας απάντησης

7.4 Κάμερα

Για την διαχείριση της κάμερα χρειάστηκε μόνο ένα αρχείο. Το αρχείο αυτό πραγματοποιεί έλεγχο του κλειδιού ασφαλείας και στην συνέχεια λαμβάνει την εικόνα και την αποθηκεύει ως latest.jpg. Τέλος δεν παραλείπονται οι εντολές για έλεγχο σφαλμάτων.

```php
<?php
$apikey = "[redacted]";
if (!isset($_SERVER['KEY']) || $_SERVER['KEY'] !== $apikey) {
    http_response_code(401);
    exit('Unauthorized');
}
$data = file_get_contents('php://input');
if (!$data) {
    http_response_code(400);
    exit('No image data');
}
$dir   = __DIR__ . '/uploads';
$final = $dir . '/latest.jpg';

if (file_put_contents($final, $data) === false) {
    http_response_code(500);
    exit('Failed to write image');
}
header('Content-Type: text/plain');
```

Εικόνα 7.4 Κώδικας αποστολής εικόνων

8.Συμπεράσματα

Η εργασία ολοκληρώθηκε με επιτυχία και πλέον το σύστημα λειτουργεί προσφέροντας όλα όσα είχαν τεθεί σαν στόχος. Η υλοποίηση της αποτέλεσε ιδιαίτερη πρόκληση, η διαδικασία προσπάθεια-αποτυχία-νέα προσπάθεια επαναλαμβανόταν συνεχώς συμβάλλοντας στης εξαγωγή πολλών συμπερασμάτων. Μέσα από ολόκληρη την διαδικασία αντιμετωπίστηκαν αρκετά προβλήματα, κάποια λυθήκαν και κάποια οδήγησαν σε εκπτώσεις από τον αρχικό σχεδιασμό. Η συνεχόμενη προσπάθεια για την επίλυση των προβλημάτων οδήγησε την έρευνα σε ένα ευρύτερο περιβάλλον, με αποτέλεσμα να δημιουργούνται συνεχώς ιδέες αναβάθμισης και βελτίωσης.

Το τελικό αποτέλεσμα δεδομένων του υπάρχοντος εξοπλισμού προσφέρει ένα αξιόλογο ελεγχόμενο εξ ’αποστάσεως σύστημα. Ο περιορισμός του δρομολογητή, να μην επιτρέπει Port Forward , οδήγησε στην χρήση HTTP polling κάτι που περιόρισε τις δυνατότητες του συστήματος.

Για κάθε δημιουργό συστημάτων που εκτίθενται στον παγκόσμιο ιστό, η ασφάλεια του συστήματος αποτελεί ιδιαίτερη πρόκληση. Οποιοδήποτε κενό ασφαλείας μπορεί να οδηγήσει σε απλή διακοπή λειτουργείας μέχρι ολική καταστροφή του συστήματος. Για το κομμάτι της ασφάλειας δεν επαρκεί μόνο η γνώση των τύπων κακόβουλων επιθέσεων. Πρέπει ο δημιουργός να διαθέτει την ικανότητα συγγραφής αλγορίθμων για την αποτροπή τέτοιων επιθέσεων. Στην παρούσα εργασία τα μέτρα ασφαλείας είναι περιορισμένα λόγω του ότι πρόκειται για πρότυπο και όχι για ολοκληρωμένο σύστημα ασφαλείας. Ωστόσο η αύξηση των μέτρων ασφαλείας θα μπορούσε να είναι η πρώτη αναβάθμιση.

Αδυναμία του συστήματος αποτελεί και η εξάρτηση του συνεχόμενη παροχή ρεύματος. Σε περίπτωση διακοπής ηλεκτροδότησης το σύστημα παύει να λειτουργεί. Μια πρακτική λύση είναι η χρήση μπαταρίας με ηλιακό φορτιστή. Το σύστημα λειτουργεί με μέγιστη τάση 5V και ένταση 2A αυτό σημαίνει ότι η μέγιστη κατανάλωση ισούται με 10 Wh. Ένα Power Bank 30.000 mAh με ηλιακό φορτιστή θα μπορούσε να καλύψει την τροφοδοσία του συστήματος με άνεση τους καλοκαιρινούς μήνες.

Το σύστημα παρόλο που ενσωματώνει κάμερα ασφάλειας δεν διαθέτει καλή ποιότητα ζωντανής ροής εικόνων. Η ESP32 μπορεί να ανταπεξέλθει σε λήψη πολλών

καρέ με μεγάλη ανάλυση ανά δευτερόλεπτο ωστόσο δεν κατέστη εφικτή η σωστή διαχείρισή τους στον διακομιστή. Για την διατήρηση της συνεχόμενης προβολής χωρίς να χάνονται εικόνες χρειάστηκε ο ρυθμός των εικόνων να μειωθεί σε μόλις δύο καρέ ανά δευτερόλεπτο και σε ανάλυση QQVGA.

Τέλος παρόλες τις αδυναμίες ή ελλείψεις του συστήματος, το τελικό αποτέλεσμα δεν παύει να είναι λειτουργικό με εξαιρετικές δυνατότητες αναβάθμισης.

ΠΑΡΑΡΤΗΜΑ

Α:

HTML

```
<?php

session_start();

if (!isset($_SESSION['user'])) {

    header('Location: ../index.php');

    exit;

}

?>

<!DOCTYPE html>

<html lang="el">

<head>

  <meta charset="UTF-8">

  <title>Διαχείριση Φώτων</title>

  <link rel="stylesheet" href="listyl.css">

</head>

<body>

  <table>

    <tr>

      <td>Φως 1</td>

      <td>
```

CSS

```
body {
  background: #4a90e2;
  color: rgb(255, 255, 255);
  font-family: 'Courier New', monospace;
  text-align: center;
  margin: 0;
  padding: 1rem;
}

input {
  margin-left: 15px;
}

.switch {
  position: relative;
  display: inline-block;
  width: 64px;
  height: 32px;
  margin: 8px 0px;
}

.slider {
  position: absolute;
  cursor: pointer;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: grey;
  transition: 0.4s;
  border-radius: 32px;
}

.switch input {
  opacity: 0;
  width: 0;
  height: 0;
}

.slider::before {
  position: absolute;
  content: "";
  height: 28px;
  width: 28px;
  left: 3.6px;
  bottom: 2.4px;
  background-color: white;
  transition: 0.4s;
  border-radius: 50px;
}
```

<label class="switch">
<input type="checkbox" name="L1">
 	
</label>
</td>
</tr>
<tr>
<td>Φως 2</td>
<td>
<label class="switch">
<input type="checkbox" name="L2">
 	
</label>
</td>
</tr>
<tr>
<td>Φως 3</td>
<td>
<label class="switch">
<input type="checkbox" name="L3">
 	
</label>
</td>
	input:checked + .slider {
	background-color: rgb(0, 255, 76);
	}
	
	input:checked + .slider::before {
	transform: translateX(30px);
	}

```
  </tr>
  <tr>
   <td>Φως 4</td>
   <td>
    <label class="switch">
     <input type="checkbox" name="L4">
     <span class="slider"></span>
    </label>
   </td>
  </tr>
  <tr>
   <td>Φως 5</td>
   <td>
    <label class="switch">
     <input type="checkbox" name="L5">
     <span class="slider"></span>
    </label>
   </td>
  </tr>
</table>
 <script src="lights.js"></script>
</body>
</html>
```

Β:

HTML	CSS

<?php
session_start();
if (!isset($_SESSION['user'])) {
header('Location: ../index.php');
exit;
}
?>
<!DOCTYPE html>
<html lang="el">
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="alarm.css">
</head>
<body>
 	
 Κατάσταση Συναγερμού
 Κατάσταση:
 	
<button id="alarm-enable"
class="buttonon">Ενεργοποίηση</button>
<button id="alarm-disable"
class="buttonoff">Απενεργοποίηση</button>
 	
<script src="alarm.js"></script>
</body>
</html>
	body {
	background: #4a90e2;
	color: white;
	font-family: 'Courier New', monospace;
	text-align: center;
	margin: 0;
	padding: 1rem;
	}
	
	h1 {
	font-size: 1.2rem;
	margin-top: 0.5rem;
	}
	
	.buttons {
	display: flex;
	flex-direction: column;
	align-items: center;
	gap: 1rem;
	margin-top: 2rem;
	}
	
	.buttonoff,
	.buttonon {
	color: white;
	font-size: 1.2rem;
	padding: 0.75rem 1.5rem;
	border: none;
	border-radius: 10px;
	cursor: pointer;
	width: 200px;
	text-align: center;
	}
	
	.buttonoff {
	background: #e74c3c;
	}
	
	.buttonon {
	background: #27ae60;
	}

Γ:

HTML	CSS

<?php
session_start();
if (empty($_SESSION['user'])) {
header('Location: index.php');
exit;
	* {
	box-sizing: border-box;
	margin: 0;

```
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<link rel="stylesheet" type="text/css" href="cpstyl.css" />
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<title>Δωμάτιο ελέγχου</title>
</head>

<body>
<!-- HEADER -->
<div class="header">
<h1>Δωμάτιο Ελέγχου</h1>
</div>

<!-- NAVBAR -->
<div class="navbar">
<div class="navbar-right">
<button><a href="index.php" class="right">Logout</a></button>
</div>
</div>

<!-- MAIN -->
<div class="page">
<!-- Φώτα -->
<div class="box">
<h2>Φώτα</h2>
<iframe src="/lights/lights.php"></iframe>
</div>

<!-- Κάμερα -->
<div class="box">
<h2>Κάμερα</h2>
<iframe src="stream.php" allowfullscreen></iframe>
</div>

<!-- Υγρασία-Θερμοκρασία -->
<div class="box">
<h2>Υγρασία - Θερμοκρασία</h2>
<iframe src="/dht/t&h.php"></iframe>
</div>

<!-- Συναγερμός -->
<div class="box">
<h2>Συναγερμός</h2>
<iframe src="alarm/alarm.php"></iframe>
</div>
```

```
 padding: 0;
 font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

body {
 background: linear-gradient(to right, #74ebd5, #acb6e5);
 height: 100vh;
 display: flex;
 flex-direction: column;
 overflow: hidden;
}

.header {
 background-color: #4a90e2;
 color: white;
 padding: 10px;
 text-align: center;
 font-size: 1.5rem;
}

.navbar {
 background-color: #357ab8;
 padding: 8px 20px;
 text-align: right;
}

.navbar-right button {
 background-color: #ffffff;
 color: #4a90e2;
 border: 2px solid #4a90e2;
 padding: 6px 12px;
 border-radius: 6px;
 cursor: pointer;
 transition: all 0.3s ease;
}

.navbar-right button:hover {
 background-color: #4a90e2;
 color: white;
}

.navbar-right a {
 text-decoration: none;
 color: inherit;
 font-weight: bold;
}

.page {
 flex: 1;
 padding: 10px 20px;
 display: grid;
 grid-template-columns: repeat(2, 1fr);
 grid-template-rows: repeat(2, 1fr);
 gap: 20px;
 overflow: hidden;
}

.box {
```

```
</div>

<!-- FOOTER -->
<div class="footer">
<p>Copyright &copy; 2025</p>
</div>
</body>
</html>
```

```
  background-color: white;
  border-radius: 10px;
  padding: 10px;
  box-shadow: 0 4px 12px rgba(0,0,0,0.1);
  display: flex;
  flex-direction: column;
  overflow: hidden;
}

.box h2 {
  font-size: 1rem;
  margin-bottom: 5px;
  color: #333;
}

.box iframe {
  flex-grow: 1;
  width: 100%;
  border: none;
  border-radius: 6px;
}

.footer {
  background-color: #4a90e2;
  color: white;
  text-align: center;
  padding: 8px;
  font-size: 0.85rem;
}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

Arduino. (n.d.). Retrieved from https://www.arduino.cc/?subject=Arduino

ESP32 Projects. (n.d.). Retrieved from https://randomnerdtutorials.com/projects-esp32-cam/

Esp8266 projects. (n.d.). Retrieved from https://www.youtube.com/@GetCert

Espressif ESP8266. (n.d.). Retrieved from https://www.espressif.com/sites/default/files/documentation/esp8266-technical_reference_en.pdf

W3school HTML. (n.d.). Retrieved from https://www.w3schools.com/html/default.asp

W3Schools CSS. (n.d.). Retrieved from https://www.w3schools.com/css/default.asp

W3Schools JavaScript. (n.d.). Retrieved from https://www.w3schools.com/js/default.asp

W3Schools PHP. (n.d.). Retrieved from https://www.w3schools.com/php/default.asp

W3Schools SQL. (n.d.). Retrieved from https://www.w3schools.com/sql/default.asp

Wikipedia CSS. (2025, 06 08). Retrieved from https://en.wikipedia.org/wiki/CSS

Wikipedia HTML. (2025, 06 08). Retrieved from https://en.wikipedia.org/wiki/HTML

Wikipedia JavaScript. (2025, 06 08). Retrieved from https://en.wikipedia.org/wiki/JavaScript

Wikipedia PHP. (2025, 06 08). Retrieved from https://en.wikipedia.org/wiki/PHP

WIkipedia SQL. (20252, 06 08). Retrieved from https://en.wikipedia.org/wiki/SQL

Ψούνης, Δ. (n.d.). *Βιντεοδιαλέξεις γλωσσών προγραμματισμού*. Retrieved from https://www.youtube.com/@psounis