ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

# ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

# ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

## ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

## «Σύστημα συναγερμού για οικιακή χρήση»


**Του φοιτητή**
**Γεώργιος Κλαψινάκης**
**Αρ. Μητρώου: 512049**

**Επιβλέπων**
**Ονοματεπώνυμο Βασίλειος**
**Βάσσιος**
**Βαθμίδα Ακαδημαϊκός Υπότροφος**


**Ημερομηνία 16/4/2024**

Τίτλος Δ.Ε. Σύστημα συναγερμού για οικιακή χρήση
Κωδικός Δ.Ε. 23354
Ονοματεπώνυμο φοιτητή Γεώργιος Κλαψινάκης
Ονοματεπώνυμο εισηγητή Βασίλειος Βάσσιος
Ημερομηνία ανάληψης Δ.Ε. 13/12/2023
Ημερομηνία περάτωσης Δ.Ε. 21/05/2024

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

*«Στην οικογένειά μου»*

# Πρόλογος

Για την εκπόνηση της πτυχιακής μου εργασίας επέλεξα θέμα με τίτλο «Σύστημα συναγερμού για οικιακή χρήση». Η ανάληψη αυτού του θέματος έγινε από την επιθυμία μου να μπορέσω να προγραμματίσω έναν μικροεπεξεργαστή και να κατασκευάσω την δική μου πλακέτα, μέσα από αυτήν την διαδικασία μου δόθηκε η ευκαιρία να το καταφέρω. Ένας ακόμη λόγος που επέλεξα αυτό το θέμα ήταν και η τωρινή μου εργασία που αντικείμενο έχει την εγκατάσταση και προγραμματισμό συστημάτων ασφαλείας και καμερών.

Αφού έφερα εις πέρας το πρακτικό μέρος της πτυχιακής μου, μπορώ να πω πως οι εμπειρίες από τον σχεδιασμό και την υλοποίηση της κατασκευής έως και τον προγραμματισμό της με το εργαλείο Arduino ήταν πολύτιμες. Επιπλέον, οι γνώσεις που αποκτήθηκαν κυρίως στο κομμάτι του προγραμματισμού του Arduino ήταν χρήσιμες και πιστεύω θα συμβάλλουν στην μετέπειτα πορεία μου.

# Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στον σχεδιασμό και την υλοποίηση ενός συστήματος συναγερμού για οικιακή χρήση βασισμένο σε Arduino. Συγκεκριμένα, ο χρήστης θα μπορεί να οπλίζει και να αφοπλίζει το σύστημα τοπικά με την χρήση του πληκτρολογίου αλλά και με δυνατότητα χειρισμού απομακρυσμένα μέσω διαδικτύου και να ειδοποιείται με την λήψη push notifications στο κινητό τηλέφωνο και με e-mail για την οποιαδήποτε παραβίαση. Αρχικά, στην εργασία αυτή γίνεται αναφορά στο τι είναι ένα σύστημα ασφαλείας και ο σκοπός του. Στην συνέχεια, θα γίνει ανάλυση του συστήματος συναγερμού καθώς και στα επιμέρους στοιχεία που το απαρτίζουν, πχ αισθητήρες, πληκτρολόγιο. Έπειτα, μια σύντομη περιγραφή για το Arduino Due, το Esp8266(Esp12-E) και την εφαρμογή Blynk που χρησιμοποιήθηκαν, το πρώτο για την κεντρική μονάδα, το δεύτερο για την σύνδεση του συστήματος στο διαδίκτυο και το τρίτο για την εξ αποστάσεως λειτουργεία. Ύστερα, σχεδιάζεται το κύκλωμα του συναγερμού και μέσω έρευνας που διενεργήθηκε γίνεται η καταγραφή των λειτουργειών και των χαρακτηριστικών των εξαρτημάτων που επέλεξα. Κατόπιν, ξεκινάει η υλοποίηση της πλακέτας και ο προγραμματισμός του Arduino Due , αργότερα η δημιουργία της μακέτας για την επίδειξη του συστήματος. Μετά, το επόμενο βήμα είναι η επεξήγηση της λειτουργείας του συναγερμού, πως οπλίζουμε και αφοπλίζουμε το σύστημα, καθώς και το πως χρησιμοποιούμε την εφαρμογή Blynk. Τέλος, αναφέρονται τα συμπεράσματα και οι προτάσεις βελτίωσης για την κατασκευή.

**ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ**

Συναγερμός, σύστημα ασφαλείας, Arduino, Blynk

«Home Security System»

«Giorgos Klapsinakis»

# Abstract

This undergraduate thesis focuses on the design and implementation of a home security system based on Arduino. Specifically, the user will be able to arm and disarm the system locally using the keypad, as well as remotely via the internet, and receive push notifications on their mobile phone and email for any breaches. Initially, the thesis introduces the concept of a security system and its purpose. Then, it analyzes the alarm system and its individual components, such as sensors and keypad. Subsequently, a brief description is provided for the Arduino Due, the Esp8266(Esp12-E), and the Blynk application used, the former for the central unit, the second for connecting the system to the internet, and the third for remote operation. Next, the alarm system circuit is designed, and through conducted research, the functions and characteristics of the selected components are recorded. Following this, the implementation of the circuit and programming of the Arduino Due begin, followed by the creation of the prototype for system demonstration. Afterwards, the operation of the alarm system is explained, including how to arm and disarm it, as well as how to use the Blynk application. Finally, conclusions are drawn, and suggestions for improvement of the construction are provided.

**KEY WORDS**

Alarm, security system, Arduino, Blynk

# Ευχαριστίες

Αρχικά, θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Βασίλειο Βάσσιο, για την ευκαιρία που μου έδωσε να υλοποιήσω αυτό που είχα στο μυαλό μου με την ανάθεση της συγκεκριμένης πτυχιακής εργασίας, όπως επίσης και για την καθοδήγησή του ώστε να έρθει εις πέρας όλο αυτό.

Μετέπειτα, θα ήθελα να ευχαριστήσω τον πολύ καλό μου φίλο Παναγιώτη Σιδερά που συνέβαλε και αυτός με την βοήθεια του στην 3D εκτύπωση αντικειμένων, συγκεκριμένα στις γωνίες που συγκρατούν την μακέτα, την βάση για την πλακέτα μου και τις πόρτες, με τις διαστάσεις και τα σχέδια που του παρείχα.

Στην συνέχεια, θα ήθελα να ευχαριστήσω θερμά την οικογένεια μου που με στήριξε και με βοήθησε σε όσες δύσκολες στιγμές είχα καθ' όλη την διάρκεια των σπουδών μου καθώς και στην πτυχιακή εργασία μου.

Τέλος, θέλω να πω ένα ευχαριστώ σε όλους τους φίλους μου που ήταν δίπλα μου και με υποστήριζαν με την αγάπη τους.

# Περιεχόμενα

# Κατάλογος Σχημάτων

# Κατάλογος Πινάκων

# Συντομογραφίες

| | |
|---|---|
| Η/Υ | Ηλεκτρονικός Υπολογιστής |
| ΚΛΣ | Κέντρο Λήψης Σημάτων |
| Μ.Ε | Μαγνητική Επαφή |

# Εισαγωγή

Η ασφάλεια στο σπίτι αποτελεί πρωταρχική ανησυχία για κάθε οικογένεια. Με την αύξηση των περιστατικών κλοπών και εισβολών, η ανάγκη για αποτελεσματικά συστήματα ασφαλείας γίνεται ολοένα και πιο επιτακτική. Στο πλαίσιο αυτό, η παρούσα πτυχιακή εργασία στοχεύει στην υλοποίηση ενός συστήματος συναγερμού για οικιακή χρήση, το οποίο θα παρέχει υψηλό επίπεδο προστασίας για τους χρήστες του.

Ο κύριος στόχος της εργασίας είναι η σχεδίαση και η υλοποίηση ενός συστήματος συναγερμού το οποίο θα είναι εύκολο στη χρήση, αξιόπιστο και θα προσφέρει ολοκληρωμένη προστασία στο σπίτι. Το σύστημα θα περιλαμβάνει διάφορους αισθητήρες όπως αισθητήρες κίνησης, αισθητήρες ανίχνευσης φωτός και μαγνητικές επαφές πόρτας/παραθύρου, ενώ θα ενσωματώνει επίσης λειτουργίες ειδοποίησης για επιπλέον ευκολία και ασφάλεια.

Μέσω της παρούσας έρευνας και ανάπτυξης, επιδιώκεται η δημιουργία ενός συστήματος που θα ανταποκρίνεται στις ανάγκες των χρηστών του, παρέχοντας τους την αίσθηση της ασφάλειας και την εμπιστοσύνη ότι το σπίτι τους προστατεύεται σε κάθε στιγμή.

Στο πρώτο κεφάλαιο της παρούσας πτυχιακής εργασίας γίνεται η παρουσίαση των συστημάτων ασφαλείας, τι ονομάζουμε σύστημα ασφαλείας και σε ποιες μεγάλες κατηγορίες χωρίζονται. Έπειτα, γίνονται αναφορές στα κύρια χαρακτηριστικά τους και τι χρειαζόμαστε για να έχουμε ένα ολοκληρωμένο σύστημα.

Στο δεύτερο κεφάλαιο γίνεται αναλυτική παρουσίαση του συστήματος συναγερμού, καθώς σε αυτό το σύστημα βασίζεται η πτυχιακή εργασία. Αναφέρεται η δομή του συστήματος και οι πιο βασικοί ανιχνευτές μαζί με την αρχή λειτουργίας τους.

Στο τρίτο και στο τέταρτο κεφάλαιο θα γίνει μια σύντομη γνωριμία με το Arduino και την εφαρμογή Blynk. Και τα δύο αποτελούν βασικά εργαλεία για την κατασκευή της πτυχιακής εργασίας, καθώς το Arduino θα παίζει τον ρόλο της κεντρικής μονάδας ενώ η εφαρμογή Blynk θα είναι αυτή που θα μας επιτρέψει να ελέγχουμε το σύστημα εξ αποστάσεως.

Το πέμπτο κεφάλαιο αφορά την σχεδίαση και την υλοποίηση του συστήματος συναγερμού, την κατασκευή της πλακέτας που θα συνοδεύει το Arduino καθώς και της μακέτας για την επίδειξή του. Γίνεται αναφορά στα εξαρτήματα που χρησιμοποιήθηκαν και αναλύονται ξεχωριστά, ενώ παράλληλα περιγράφεται και ο τρόπος σύνδεσής τους με το Arduino για την καλύτερη κατανόηση του κυκλώματος. Τέλος, παρουσιάζεται η διαδικασία δημιουργίας της πλακέτας και της μακέτας βήμα βήμα.

Το έκτο κεφάλαιο αφορά τον κώδικα του συστήματος και προγραμματίζουμε το Arduino Due με την χρήση του Arduino IDE. Έπειτα, δίνεται το διάγραμμα ροής του κώδικα της κατασκευής. Τέλος, τίθεται σε εφαρμογή το σύστημα συναγερμού και παρουσιάζεται με εικόνες η λειτουργία του, καθώς και η χρήση σεναρίων για να δούμε πως αντιδράει το σύστημα.

Στο έβδομο και τελευταίο κεφάλαιο παρουσιάζονται τα προβλήματα που προέκυψαν κατά την διάρκεια της πτυχιακής εργασίας αλλά και πώς αντιμετωπίστηκαν. Τέλος, παραχωρούνται τα συμπεράσματα και οι μελλοντικές βελτιώσεις που μπορούν να γίνουν στο σύστημα συναγερμού για οικιακή χρήση.

# Κεφάλαιο 1ο: Συστήματα ασφαλείας

## 1.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει ενημέρωση για τα συστήματα ασφαλείας, τις βασικές κατηγορίες, την βαθμίδα κατάταξης των συστημάτων και τα κριτήρια επιλογής τους. Τέλος θα δοθεί μια περίληψη για την κάθε κατηγορία.

## 1.2 Ηλεκτρονικά συστήματα ασφαλείας

### 1.2.1 Έννοια ηλεκτρονικών συστημάτων ασφαλείας

Σύστημα ασφαλείας θεωρείται κάθε σύστημα το οποίο μπορεί να προστατέψει τον ιδιοκτήτη προσφέροντας του προστασία ενάντια στην ανεπιθύμητη ανθρώπινη παρουσία που έχει ως σκοπό την διάρρηξη και την ληστεία όπως επίσης και από άλλους κινδύνους που μπορεί να προκύψουν πχ πυρκαγιά. Την σήμερον ημέρα τα συστήματα αυτά είναι κυρίως ηλεκτρονικά, δηλαδή βασίζουν την λειτουργία τους στα ηλεκτρονικά εξαρτήματα.[1]

### 1.2.2 Βασικές κατηγορίες

- Συστήματα συναγερμού (Security-Alarm systems)
- Συστήματα πυρανίχνευσης (Fire-Alarm systems)
- Συστήματα παρακολούθησης καμερών – CCTV (Closed Circuit Television)
- Συστήματα ελέγχου πρόσβασης (Access control systems)

### 1.2.3 Βαθμίδες κατάταξης

Τα Ευρωπαϊκά standards (BSEN 50131) αντικατέστησαν τα Βρετανικά standards (4737, 7042 και BS 6799 Wireless Systems) την 1η Οκτωβρίου του 2005. Οι βαθμίδες καθορίζουν την δυσκολία που θα αντιμετωπίσει ο διαρρήκτης κατά την προσπάθειά του να παραβιάσει τον προστατευόμενο χώρο. Όσο μεγαλύτερη η βαθμίδα τόσο καλύτερα θεωρείται προστατευμένος ο χώρος.[1]

- **Grade 1** – χαμηλής επικινδυνότητας: για διάρρηξη αρκεί το σπάσιμο μιας πόρτας ή ενός τζαμιού
- **Grade 2** – χαμηλής ή μεσαίας επικινδυνότητας: χρειάζονται ελάχιστες γνώσεις ή εμπειρία και απλά εργαλεία
- **Grade 3** – μεσαίας ή υψηλής επικινδυνότητας: χρειάζεται αρκετή εμπειρία και ολοκληρωμένος εξοπλισμός εργαλείων
- **Grade 4** – υψηλής επικινδυνότητας: χρειάζονται εξειδικευμένες γνώσεις και εξελιγμένα εργαλεία

### 1.2.4 Κριτήρια επιλογής

Βασικό κριτήριο επιλογής πολλές φορές είναι το κόστος αγοράς ενός συστήματος ασφαλείας, αν όμως αναλογιστούμε τα οφέλη που προσφέρει στον ιδιοκτήτη τότε η αγορά του το καθιστά αναγκαίο.

- Αξιοπιστία των συσκευών
- Δυνατότητα του συστήματος
- Επεκτασιμότητα του συστήματος
- Τεχνική υποστήριξη (συντήρηση, έλεγχος καλής λειτουργείας)
- 24ωρη παρακολούθηση από κάποιο κέντρο λήψης σημάτων

## 1.3 Συστήματα συναγερμού

Ένα σύστημα συναγερμού μπορεί να είναι ασύρματο ή ενσύρματο, όμως υπάρχει και μια κατηγορία που συνδυάζει και τα 2 παραπάνω σε ένα σύστημα, τα υβριδικά συστήματα συναγερμού.


Σχήμα 1.1: *(από αριστερά) Ενσύρματο σύστημα συναγερμού, Ασύρματο σύστημα συναγερμού*

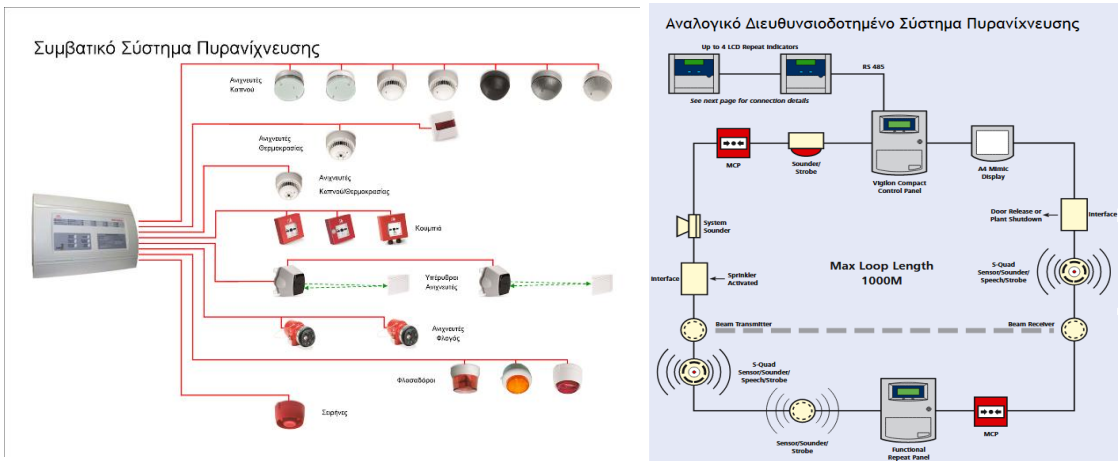
Πλεονεκτήματα ενσύρματου έναντι ασύρματου, το κόστος , δεν χρειάζεται συχνή αλλαγή μπαταριών, δεν σε περιορίζει η εμβέλεια, τα μεγέθη των ανιχνευτών είναι μικρότερα.

Μειονεκτήματα ενσύρματου έναντι ασύρματου, απαιτείται καλωδίωση που αν δεν έχει ληφθεί υπόψη από την αρχή της κατασκευής του κτιρίου τότε η καλαισθητική του χώρου χαλάει, η επέκταση και η απεγκατάσταση είναι δυσκολότερη.


Σχήμα 1.2: *Υβριδική πλακέτα επέκτασης*

## 1.4 Συστήματα πυρανίχνευσης

Σύστημα πυρανίχνευσης μπορεί να ονομαστεί ένα σύστημα το οποίο αποτελείται από συσκευές που έχουν ως σκοπό την έγκαιρη ανίχνευση μιας εστίας φωτιάς και να ενημερώσουν με ηχητικά, οπτικά και άλλα μέσα για τον κίνδυνο. Υπάρχουν 2 κατηγορίες που μπορεί να επιλέξει ο ιδιοκτήτης, το συμβατικό ή το διευθυνσιοδοτούμενο σύστημα, στο συμβατικό υπάρχουν ζώνες και σε κάθε ζώνες τοποθετούνται οι ανιχνευτές και τα κομβία (μπουτόν), ενώ από την άλλη τα διευθυνσιοδοτούμενα αποτελούνται από βρόγχους με μέγιστο μήκος και συνδέονται συσκευές πυρανίχνευσης που η καθεμία έχει την δική της ταυτότητα.[1]




Σχήμα 1.3: *(από αριστερά) Συμβατικό σύστημα πυρανίχνευσης, Διευθυνσιοδοτούμενο σύστημα πυρανίχνευσης*

Ένα ολοκληρωμένο σύστημα πυρανίχνευσης αποτελείται από τα παρακάτω:

- Κεντρικό πίνακα πυρανίχνευσης
- Εφεδρική τροφοδοσία
- Αισθητήρια πυρανίχνευσης
- Μπουτόν αναγγελίας φωτιάς
- Φωτεινούς επαναληπτές
- Σειρήνες πυρανίχνευσης


Σχήμα 1.4: *Ολοκληρωμένο σύστημα πυρανίχνευσης*

## 1.5 Συστήματα παρακολούθησης καμερών – CCTV

Σε ένα κλειστό κύκλωμα τηλεόρασης (Closed Circuit TV System) ο ιδιοκτήτης έχει την δυνατότητα να παρακολουθεί απομακρυσμένα από την οθόνη, τον υπολογιστή και το κινητό του, μέσω καμερών, διάφορα σημεία του χώρου του στα οποία μπορεί να μην υπάρχει εύκολη πρόσβαση, ακόμα όμως μπορεί να χρησιμοποιηθεί και για προστασία του χώρου από επίδοξους κλέφτες. Επίσης, σημαντικό είναι να μπορεί να γίνεται η καταγραφή των συμβάντων ώστε να υπάρχει η επιλογή της επαναπροβολής.[1]

Ένα σύστημα CCTV αποτελείται από:

- Κάμερες
- Οθόνη παρακολούθησης
- Μονάδα καταγραφής αναλογική (VCR) ή ψηφιακή (NVR, DVR, HVR)
- Μονάδα διαχείρισης εικόνων
- Στηρίγματα και προστατευτικά καμερών


Σχήμα 1.5: *Κλειστό κύκλωμα καμερών (CCTV)*

## 1.6 Συστήματα ελέγχου πρόσβασης

Τα συστήματα ελέγχου πρόσβασης έχουν στόχο να προστατέψουν χώρους που κυκλοφορούν συνεχώς εργαζόμενοι και πολλοί επισκέπτες, ενώ επιτρέπουν την είσοδο σε αδειοδοτημένους χρήστες και καταγράφουν δεδομένα. Χρησιμοποιούνται κυρίως σε επιχειρήσεις, ξενοδοχεία, αποθήκες και νοσοκομεία, με αυτόν τον τρόπο δίνεται η δυνατότητα να αυξηθεί η ασφάλεια σε χώρους ύψιστης σημασίας για την εταιρεία. Επιπλέον, μπορεί να λειτουργήσει και ως κάρτα ελέγχου ωραρίου εργαζομένων.[1]

Ένα σύστημα ελέγχου πρόσβασης αποτελείται:

- Κεντρική μονάδα ελέγχου με δυνατότητα σύνδεσης σε Η/Υ
- Τοπικό ελεγκτή
- Λογισμικό διαχείρισης κινήσεων και διαβάθμισης της προσβασιμότητας
- Κάρτες (Access Control) απλές ή προτυπωμένες (Μαγνητικές, proximity, κλπ.)


Σχήμα 1.6: *Σύστημα ελέγχου πρόσβασης*

## 1.7 Επίλογος

Όπως αναφέρθηκε παραπάνω, τα συστήματα ασφαλείας είναι σημαντικά για την προφύλαξη του χώρου μας και την αποτροπή κινδύνων. Μπορούμε να συνδυάσουμε τους τύπους των συστημάτων ανάλογα με τις ανάγκες του χώρου.

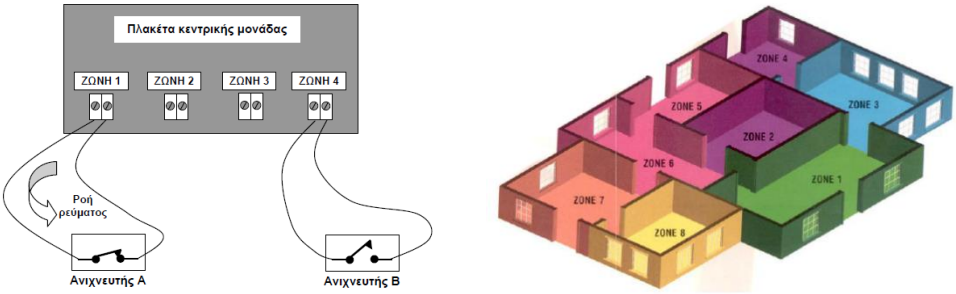
# Κεφάλαιο 2ο: Σύστημα συναγερμού

## 2.1 Εισαγωγή

Στο δεύτερο κεφάλαιο της πτυχιακής εργασίας , θα εμβαθύνουμε όπως λέει και ο τίτλος του κεφαλαίου, περισσότερο στα συστήματα συναγερμού διότι αυτός είναι και ο σκοπός του θέματος. Τέλος, θα αναλυθούν τα εξαρτήματα που απαρτίζουν ένα ολοκληρωμένο σύστημα συναγερμού.

## 2.2 Αρχή λειτουργίας

Κάθε σύστημα συναγερμού, απλό ή πολύπλοκο, βασίζεται σε βασικές αρχές που είναι κοινές, δηλαδή η κεντρική μονάδα συναγερμού επιτηρεί συνεχώς το σύνολο των αισθητήρων, και όταν είναι οπλισμένο αν έστω και ένας αισθητήρας διεγερθεί τότε θα ηχήσει η σειρήνα, θα ειδοποιηθεί το κέντρο λήψης σημάτων ή/και θα τηλεφωνεί στον ιδιοκτήτη μέσω ηχογραφημένου μηνύματος. Κάθε ανιχνευτής διαθέτει έναν αυτόματο διακόπτη, συνήθως μια NC (Normally closed) επαφή ενός ρελέ (ή συμπεριφέρεται έτσι αν πρόκειται για ηλεκτρονικό διακόπτη) ή μια επαφή reed relay αν πρόκειται για μαγνητική επαφή. Σε κατάσταση ηρεμίας οι διακόπτες είναι κλειστοί και στον βρόγχο υπάρχει ροή ρεύματος, ενώ όταν διεγερθεί ο ανιχνευτής ή ο αισθητήρας, ο διακόπτης ανοίγει και διακόπτεται η ροή ρεύματος στον βρόγχο. Τους βρόγχους τους ονομάζουμε ζώνες σε ένα σύστημα, έτσι είναι ευκολότερος ο εντοπισμός της παραβίασης. Κάθε ζώνη στην κεντρική μονάδα αποτελείται από 2 υποδοχές (κλέμες) που εκεί συνδέονται τα καλώδια των ανιχνευτών, συνήθως ένας αισθητήρας αντιστοιχεί σε μια ζώνη, αν όμως μας περιορίζουν οι διαθέσιμες ζώνες τότε μπορούμε να συνδέσουμε παραπάνω από έναν αισθητήρα, όλοι οι αισθητήρες στην ίδια ζώνη πρέπει να είναι σε σειρά.[1]




Σχήμα 2.1: (από αριστερά) Βρόγχος, Διαχωρισμός σπιτιού σε ζώνες

## 2.3 Δομή συστήματος συναγερμού

Τα βασικά στοιχεία που απαρτίζουν το σύστημα συναγερμού είναι:

- Κεντρική μονάδα συναγερμού
- Πληκτρολόγιο
- Είσοδοι του συστήματος
    - Αισθητήρες
        - Μαγνητικές επαφές
        - Ανιχνευτές κίνησης
        - Ανιχνευτές καπνού
        - Ανιχνευτές θραύσης κρυστάλλων
        - Ανιχνευτές κρούσης
        - Ανιχνευτές πλημμύρας
- Έξοδοι του συστήματος
    - Σειρήνες
        - Εσωτερικές
        - Εξωτερικές
    - Τηλεφωνητές

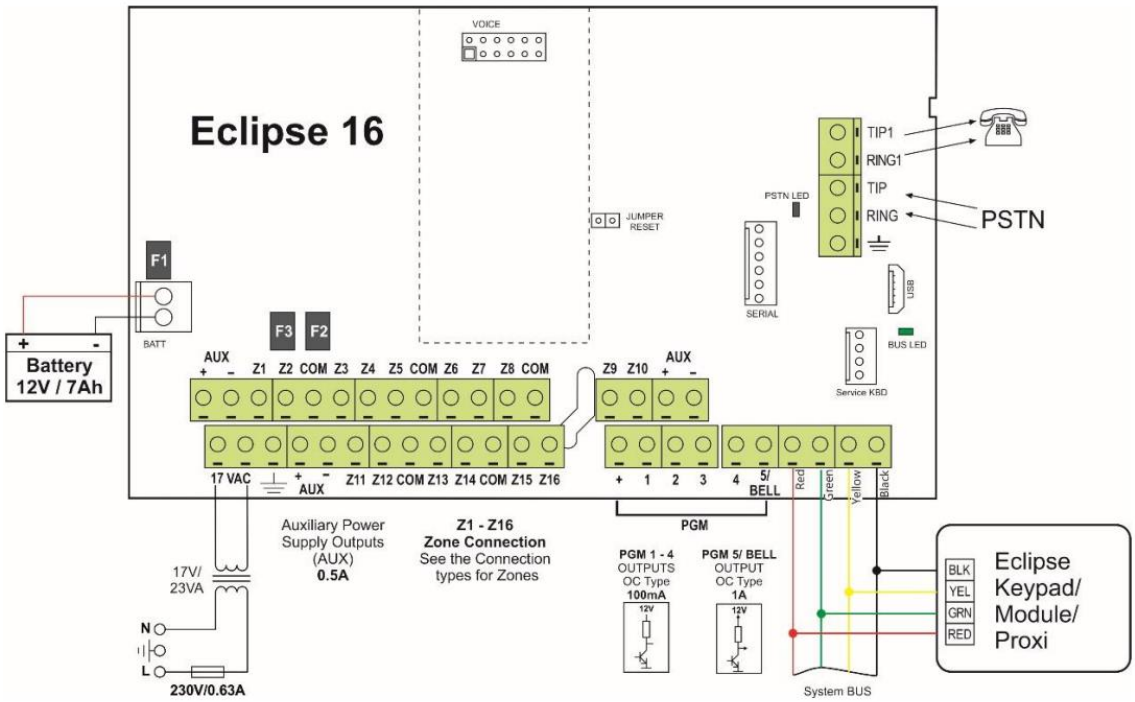
### 2.3.1 Κεντρική μονάδα ελέγχου

Η κεντρική μονάδα ελέγχου αποτελεί βασικό στοιχείο ενός συστήματος συναγερμού καθώς σε αυτήν συνδέονται όλες οι περιφερειακές συσκευές. Κάθε τύπος συστήματος έχει τις δικές του προδιαγραφές και οι δυνατότητές του είναι ανάλογες με τον προγραμματισμό που θα γίνει από τον τεχνικό της εγκατάστασης. Τέλος, σκοπός της είναι να λαμβάνει σήματα από τους αισθητήρες, να τα επεξεργάζεται και να διεγείρει την κατάλληλη έξοδο.[1]

Αποτελείται από:

- Πλακέτα κεντρικής μονάδας ελέγχου
    - Μικροελεγκτής
    - Τροφοδοτικό
    - Ρελέ τηλεφωνικών κλήσεων
    - Υποδοχές συνδέσεων
- Μετασχηματιστής υποβιβασμού
- Μπαταρία 12V/7Ah
- Μονάδες επέκτασης ζωνών ή εξόδων
- Συσκευές επικοινωνίας

Σχήμα 2.2: (από αριστερά) Κλειστό κουτί συναγερμού (Teletek Eclipse 16), Εσωτερικό κουτιού συναγερμού (Teletek Eclipse 16)




Σχήμα 2.3: Διάταξη PCB κεντρικής μονάδας ελέγχου Teletek Eclipse 16

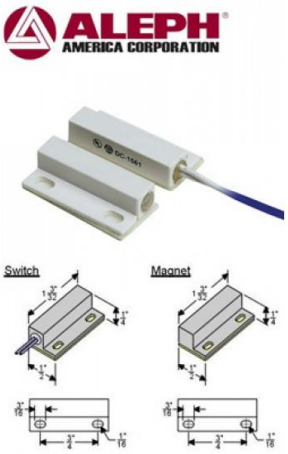
### 2.3.2 Πληκτρολόγιο

Το πληκτρολόγιο είναι απαραίτητο να βρίσκεται δίπλα στην κεντρική είσοδο του χώρου, μέσω του πληκτρολογίου ο εγκαταστάτης μπορεί να προγραμματίσει την κεντρική μονάδα ελέγχου, ενώ ο ιδιοκτήτης ελέγχει το σύστημα συναγερμού, μπορεί να οπλίζει και να αφοπλίζει το σύστημα με τον προσωπικό του κωδικό, να πληροφορηθεί για την κατάσταση των ζωνών, να ενημερωθεί για πιθανές βλάβες ( πχ αλλαγή μπαταρίας), να ενημερώσει με τα πλήκτρα κινδύνου ή με συνδυασμό 2 πλήκτρων το ΚΛΣ για φωτιά, ιατρική βοήθεια ή ληστεία. Κάθε πληκτρολόγιο έχει τον δικό του μικροελεγκτή και συνήθως μια επιπλέον ζώνη, συνδέεται με την κεντρική μονάδα ελέγχου μέσω 4 καλωδίων, τα 2 καλώδια για την τροφοδοσία του 12Vdc και τα άλλα 2 για την σειριακή επικοινωνία (Data-Clock).[1]


Σχήμα 2.4: Πληκτρολόγιο Teletek LCD 32GR

### 2.3.3 Μαγνητικές επαφές

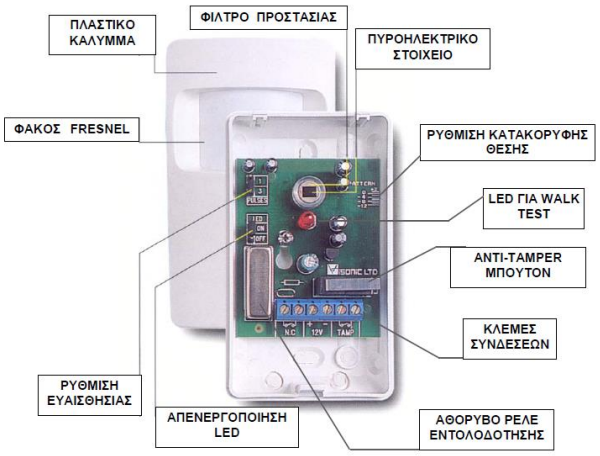
Οι μαγνητικές επαφές αποτελούνται από 2 κομμάτια, τον σταθερό μαγνήτη και τον μαγνητικό διακόπτη. Η τοποθέτησή τους γίνεται στο κούφωμα της πόρτας και του παραθύρου. Ο μαγνητικός διακόπτης αποτελείται από 2 παράλληλα σιδηρομαγνητικά ελάσματα τοποθετημένα σε γυάλινη αμπούλα με αέριο, τα ελάσματα έρχονται σε επαφή και κλείνουν κύκλωμα μόνο όταν είναι κοντά τους ο σταθερός μαγνήτης. Τέλος, δεν χρειάζονται τροφοδοσία για να λειτουργήσουν.[1]




Σχήμα 2.5: Μαγνητική επαφή

### 2.3.4 Ανιχνευτές κίνησης

Στους ανιχνευτές κίνησης έχουμε αρκετούς τύπους ανιχνευτών, ο σκοπός όλων είναι να επιτηρούν έναν συγκεκριμένο χώρο και όταν ανιχνεύσουν κίνηση να ενημερώσουν την κεντρική μονάδα ελέγχου, ο πιο διαδεδομένος από αυτούς είναι ο παθητικός ανιχνευτής υπέρυθρης ακτινοβολίας (PIR), η λειτουργεία του βασίζεται στην ανίχνευση της μεταβολής της εκπεμπόμενης υπέρυθρης ακτινοβολίας, από ένα σώμα, στο πεδίο της εμβελείας του. Ο πυροηλεκτρικός αισθητήρας που βρίσκεται στο τυπωμένο κύκλωμα του ανιχνευτή μετατρέπει την μεταβολή αυτή σε ηλεκτρική τάση.[1]




Σχήμα 2.6: Ανιχνευτής υπέρυθρης ακτινοβολίας (PIR)

### 2.3.5 Σειρήνες

Οι σειρήνες στα συστήματα συναγερμού είναι απαραίτητες διότι ενημερώνουν τους επίδοξους διαρρήκτες ότι ο χώρος προστατεύεται, αλλά και να ειδοποιήσουν τους κοντινούς κατοίκους ότι υπάρχει εισβολή στην γειτονιά τους. Διακρίνονται σε 2 τύπους, την εσωτερική και την εξωτερική, η εσωτερική είναι χαμηλότερης έντασης ενώ η εξωτερική είναι υψηλότερης έντασης, διαθέτει μπαταρία και φάρο για να είναι ξεκάθαρο ποια κατοικία δέχεται εισβολή. Συνδέονται στις υποδοχές για έξοδο της κεντρικής μονάδας ελέγχου, συνήθως υπάρχει συγκεκριμένη υποδοχή για σειρήνα καθώς χρειάζεται μεγαλύτερο ρεύμα.[1]

Σχήμα 2.7: Εξωτερική σειρήνα BS-1

## 2.4 Επίλογος

Μετά από την μελέτη που θα χρειαστεί να κάνει ο εγκαταστάτης του συστήματος συναγερμού, θα τοποθετήσει στα κατάλληλα σημεία την κεντρική μονάδα ελέγχου, τους ανιχνευτές, το πληκτρολόγιο και την σειρήνα, τέλος θα προγραμματίσει το σύστημα σύμφωνα με τις ανάγκες μας αλλά και του χώρου.

# Κεφάλαιο 3ο: Arduino

## 3.1 Εισαγωγή

Για την υλοποίηση του συστήματος συναγερμού για οικιακή χρήση, θα χρησιμοποιηθεί το εργαλείο Arduino. Σε αυτό το κεφάλαιο θα γίνει αναφορά στον ορισμό του, που βρίσκει εφαρμογή στον τομέα της ηλεκτρονικής και θα αναφερθούν ονομαστικά τα πιο διαδεδομένα μοντέλα ενώ θα γίνει μια πιο λεπτομερή αναφορά στο Arduino Due που θα χρησιμοποιηθεί στην κατασκευή της πτυχιακής.

## 3.2 Ορισμός

Το Arduino είναι μια πλατφόρμα ανοιχτού κώδικα (open-source) για το λογισμικό του (software) και το υλικό του (hardware) που χρησιμοποιείται για την ανάπτυξη πρωτότυπων ηλεκτρονικών έργων. Είναι ικανό να διαβάζει ηλεκτρονικά σήματα σαν εισόδους και να τα μετατρέπει σε ηλεκτρονικά σήματα εξόδων με την χρήση του μικροελεγκτή που έχει στην πλακέτα. Για τον προγραμματισμό του μικροελεγκτή χρησιμοποιείται η γλώσσα προγραμματισμού Arduino (βασισμένη στην Wiring), ενώ το περιβάλλον ανάπτυξης κώδικα είναι το Arduino Software (IDE), βασισμένο στο Processing.[2][3]

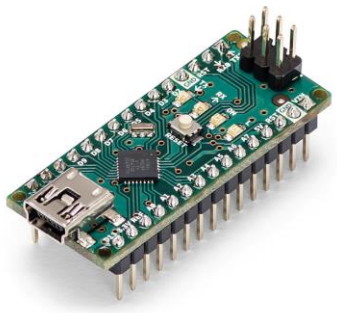
## 3.3 Εφαρμογές του Arduino

Η χρήση του Arduino στην καθημερινότητα βρίσκει εφαρμογή σε πολλά πεδία όπου η καινοτομία είναι επιθυμητή. Μερικές από αυτές είναι οι παρακάτω:[4]

- Οικιακός αυτοματισμός
- Παρακολούθηση υγείας
- Γεωργία και κτηνοτροφία
- Εκπαίδευση
- Βιομηχανικός αυτοματισμός
- Περιβαλλοντική παρακολούθηση

## 3.4 Πλακέτες Arduino

Η εταιρία χωρίζει τις πλακέτες της σε οικογένειες, έτσι είναι πιο εύκολη η αναζήτηση από τον χρήστη ανάλογα με το μέγεθος του έργου που επιθυμεί να υλοποιήσει. Κάθε οικογένεια έχει συγκεκριμένες δυνατότητες όσον αφορά την μνήμη, το μέγεθος της πλακέτας και τις διαθέσιμες εισόδους/εξόδους.



Σχήμα 3.1: Arduino Nano from nano family

Σχήμα 3.2: Arduino Uno from classic family

Σχήμα 3.3: Arduino Mega from mega family

## 3.5 Arduino Due

Το Arduino Due είναι μέρος της οικογένειας Arduino Mega και είναι ένας μικροελεγκτής με 32-bit ARM πυρήνα. Διαθέτει πληθώρα εισόδων/εξόδων και 4 UART θύρες επικοινωνίας, προσφέροντας ευελιξία για πολλές εφαρμογές. Με την τροφοδοσία λειτουργίας του να είναι στα 3.3V, είναι κατάλληλος για εφαρμογές που απαιτούν χαμηλή τάση. Ο μικροελεγκτής του, ο Atmel SAM3X8E, παρέχει αξιόπιστη απόδοση και είναι ιδανικός για έργα που απαιτούν υψηλή απόδοση και αξιοπιστία.[5][6]

Σχήμα 3.4: Arduino Due from mega family

## 3.6 Επίλογος

Παραπάνω αναφέρθηκαν οι λόγοι που κάποιος θα αποφασίσει να χρησιμοποιήσει το Arduino σαν βασικό εργαλείο για την ανάπτυξη και τον έλεγχο του πρωτότυπου ηλεκτρονικού έργου του. Τέλος, το Arduino Due ξεχωρίζει για την υψηλή απόδοσή του.

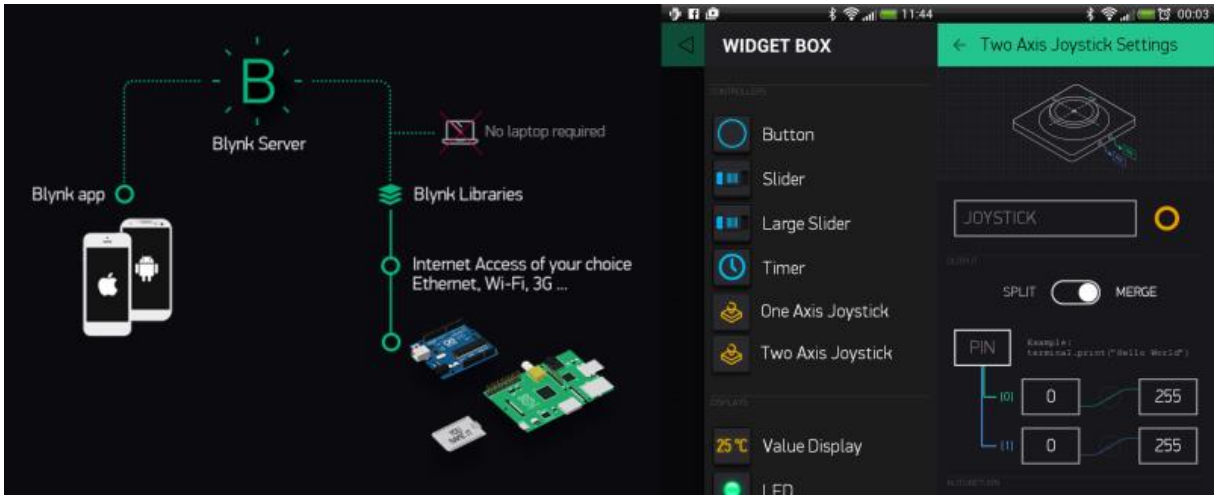
# Κεφάλαιο 4ο: Blynk

## 4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναφερθεί η πλατφόρμα Blynk που χρησιμοποιήθηκε για τον απομακρυσμένο έλεγχο του συστήματος συναγερμού της πτυχιακής εργασίας.

## 4.2 Σκοπός

Ο σκοπός του Blynk είναι να παρέχει ένα εύχρηστο και προσιτό περιβάλλον για τους χρήστες ώστε να δημιουργούν και να διαχειρίζονται εφαρμογές Internet of Things (IoT) χωρίς την ανάγκη προηγμένων γνώσεων προγραμματισμού. Με την χρήση του Blynk, οι χρήστες μπορούν να επικοινωνούν απευθείας με τις συσκευές τους, να συλλέγουν δεδομένα αισθητήρων και να ελέγχουν απομακρυσμένα τις συσκευές τους μέσω του διαδικτύου.[7]




Σχήμα 4.1: (από αριστερά) Επικοινωνία Εφαρμογής-Μικροελεγκτή, Εφαρμογή Blynk

## 4.3 Χαρακτηριστικά

Τα κύρια χαρακτηριστικά του Blynk περιλαμβάνουν:

- Απλή διεπαφή χρήστη για τη δημιουργία εφαρμογών IoT.
- Υποστήριξη ποικίλων πλατφορμών και συσκευών.
- Δυνατότητα απευθείας ελέγχου και παρακολούθησης συσκευών μέσω κινητών συσκευών.
- Προηγμένες λειτουργίες όπως ειδοποιήσεις στις κινητές συσκευές και στο email.

## 4.4 Επίλογος

Το Blynk απλοποιεί την ανάπτυξη εφαρμογών IoT, επιτρέποντας σε κάθε χρήστη να εκμεταλλευτεί τις δυνατότητες του απομακρυσμένου ελέγχου χωρίς περίπλοκο προγραμματισμό.

# Κεφάλαιο 5ο: Σχεδιασμός και Υλοποίηση Εργασίας

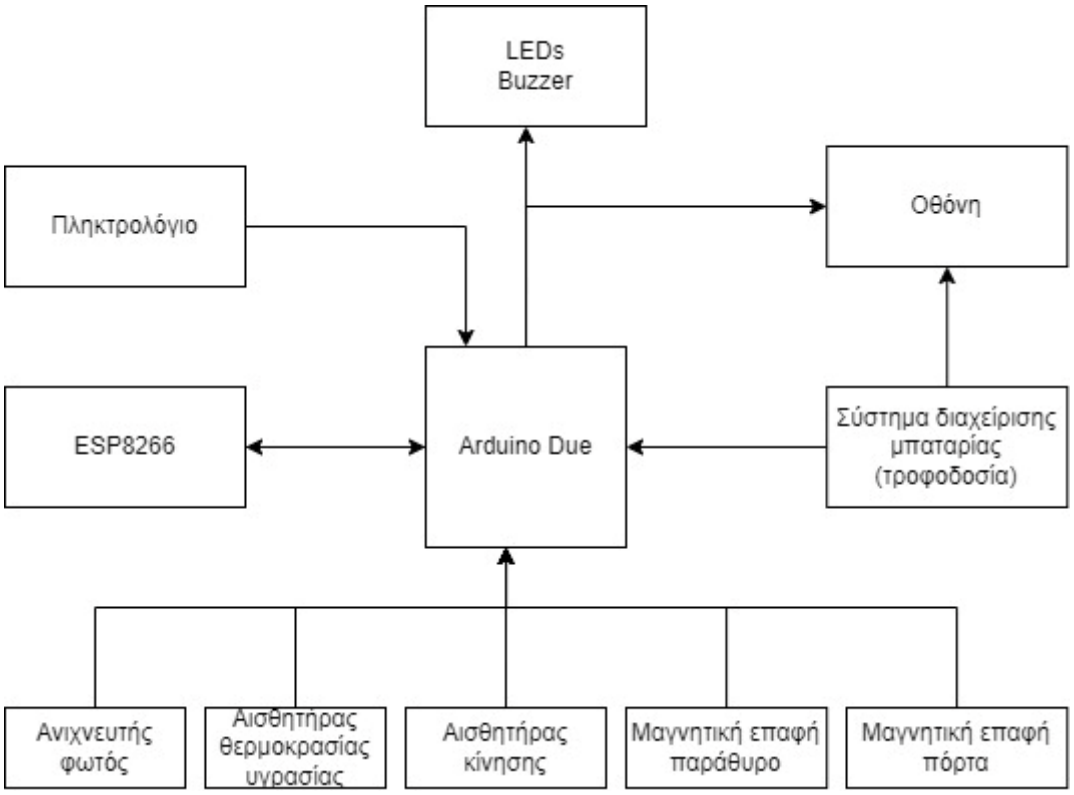
## 5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει ο σχεδιασμός και η υλοποίηση του συστήματος ασφαλείας για οικιακή χρήση, τέλος θα γίνει η κατασκευή της μακέτας ώστε να εγκατασταθεί το σύστημα.

## 5.2 Σχεδίαση κατασκευής

Για τον σχεδιασμό του συστήματος ασφαλείας, όπως αναφέρθηκε σε προηγούμενα κεφάλαια θα χρησιμοποιηθεί για κεντρική μονάδα ελέγχου το Arduino Due, ενώ θα υπάρχουν και άλλες συσκευές που θα το πλαισιώνουν και θα επεξηγηθούν παρακάτω.

Η σκέψη για τον σχεδιασμό του συστήματος, είναι να μπορεί ο χρήστης με την χρήση του πληκτρολογίου να εισάγει τον προσωπικό του κωδικό και έπειτα να οπλίζει τον συναγερμό, να έχει την δυνατότητα να επιτηρεί μέσω του κινητού τηλεφώνου του αν υπάρχει κάποια παραβίαση στον χώρο και να ενημερώνεται για αυτήν, ή να σιγουρευτεί ότι το σύστημα είναι οπλισμένο. Θα υπάρχει μια οθόνη ώστε να απεικονίζονται τυχόν ξεχασμένες ζώνες ανοιχτές πριν τον οπλισμό. Για να μπορεί το σύστημα να λειτουργεί και σε περίπτωση διακοπής ρεύματος, υπάρχει μια επαναφορτιζόμενη μπαταρία που το τροφοδοτεί, ενώ όταν αποκατασταθεί η διακοπή τότε έχει γίνει πρόβλεψη για την φόρτιση της μέσω ενός κυκλώματος. Οι ανιχνευτές του συστήματος για την επιτήρηση του χώρου είναι: 2 μαγνητικές επαφές, ένας ανιχνευτής κίνησης, ένας αισθητήρας υγρασίας και θερμοκρασίας, και τέλος ένας αισθητήρας φωτός. Για την ενημέρωση ύπαρξης συστήματος συναγερμού υπάρχει μια σειρά από LEDs που αναβοσβήσουν, ενώ για την παραβίαση του χώρου όταν το σύστημα θα είναι οπλισμένο, χρησιμοποιήθηκε ένας βομβητής (buzzer).

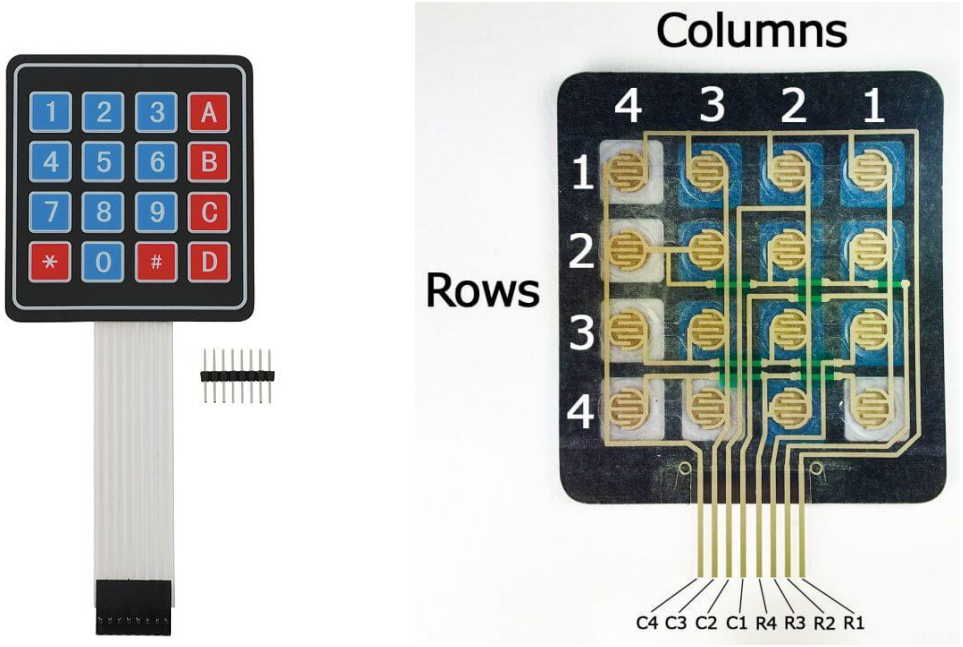



Σχήμα 5.1: Μπλόκ διάγραμμα συστήματος ασφαλείας

## 5.3 Υλοποίηση κατασκευής

### 5.3.1 Πληκτρολόγιο

Για να ελέγχει ο χρήστης το σύστημα συναγερμού, επιλέχθηκε ένα πληκτρολόγιο 16 πλήκτρων που είναι διατεταγμένα σε δομή πίνακα (matrix), αποτελείται από 4 γραμμές και 4 στήλες, κάθε στοιχείο του πίνακα είναι ένα πλήκτρο (διακόπτης) και αντιστοιχεί σε μια μοναδική στήλη και μια μοναδική γραμμή. Τα πλήκτρα είναι συνδεδεμένα μεταξύ τους ανά 4, ώστε με το ένα άκρο τους να σχηματίζουν τις 4 γραμμές και με το άλλο άκρο τους να σχηματίζουν τις 4 στήλες.

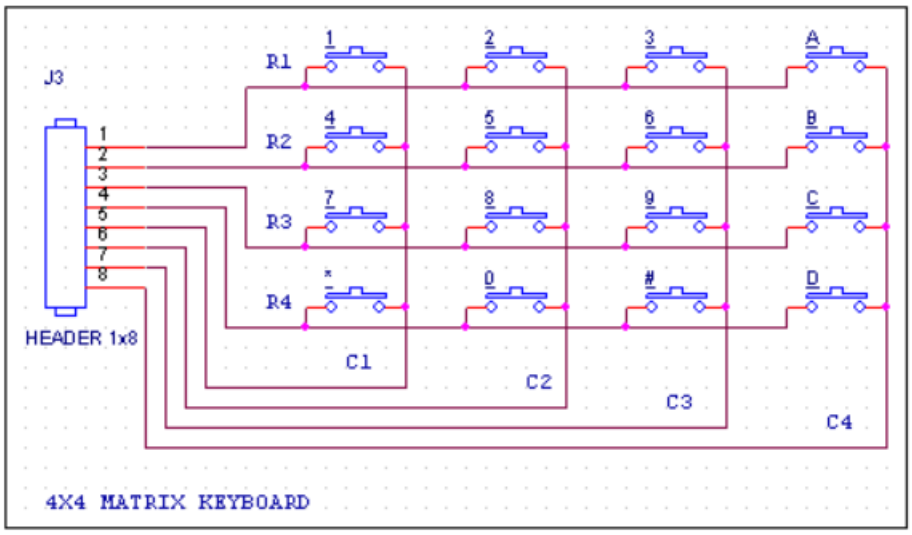



Σχήμα 5.2: (απο αριστερά) Πληκτρολόγιο μεμβράνης 4x4 matrix, Συνδεσμολογία πληκτρολογίου

#### 5.3.1.1 Αρχή λειτουργίας πληκτρολογίου

Η λειτουργία του πληκτρολογίου σε δομή πίνακα (Matrix) βασίζεται στην τεχνική της πολυπλεξίας, αυτή η τεχνική στον τομέα της ηλεκτρονικής αναφέρεται στην οικονομία του αριθμού των συνδέσεων που χρησιμοποιούνται, με διάφορους τρόπους. Σε αυτή την περίπτωση αναφέρεται στην τεχνική διάταξης ενδείξεων και πληκτρολογίων/διακοπτών σε έναν πίνακα προκειμένου να μειωθεί ο αριθμός των γραμμών εισόδου-εξόδου στο υλικό. Αν για παράδειγμα είχαμε αυτά τα 16 πλήκτρα και θέλαμε να τα ελέγχουμε κάθε ένα ξεχωριστά, θα χρειαζόμασταν 16 ακροδέκτες εισόδου στον μικροελεγκτή, ωστόσο με την χρήση της πολυπλεξίας μπορούμε να μειώσουμε αυτόν τον αριθμό.[8]

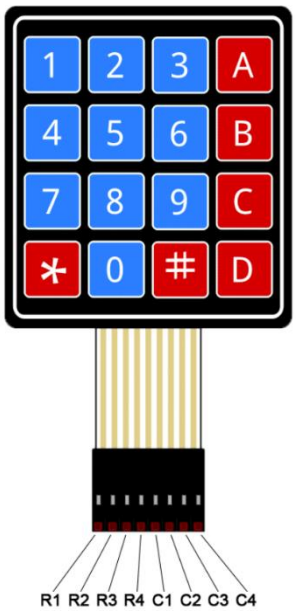
Ο έλεγχος για το ποιο πλήκτρο πατήθηκε γίνεται με την τεχνική της σάρωσης. Ο μικροελεγκτής στέλνει σε κάθε στήλη λογικό ‘0’ (LOW), και για κάθε στήλη τη φορά ελέγχει μία προς μία τις γραμμές. Αν κάποια γραμμή είναι σε λογικό ‘0’ (LOW), τότε μέσω του πίνακα καταλαβαίνει ποιο πλήκτρο πατήθηκε. Με αυτόν τον τρόπο, οι έξοδοι του μικροελεγκτή είναι οι στήλες, ενώ οι είσοδοι του μικροελεγκτή είναι οι γραμμές. Για να μπορεί ο μικροελεγκτής να γνωρίζει πάντα την κατάσταση των γραμμών και να μην είναι σε ενδιάμεση κατάσταση (floating), ενεργοποιούνται οι εσωτερικές αντιστάσεις έλξης (pull-up resistors). Με αυτόν τον τρόπο, οι γραμμές είναι σε κατάσταση ‘1’ (HIGH) όσο κάποιο πλήκτρο δεν έχει πατηθεί.




Σχήμα 5.3: 16 πλήκτρα (διακόπτες) διατεταγμένα σε μορφή πίνακα (matrix) 4x4

Ένα παράδειγμα για τη λειτουργία του πληκτρολογίου θα είναι όταν πατηθεί το πλήκτρο που αντιστοιχεί στον αριθμό 2. Ο μικροελεγκτής ξεκινά τη διαδικασία στέλνοντας λογικό ‘0’ (LOW) στην πρώτη στήλη (C1) και σαρώνει μία προς μία τις γραμμές (R1-R4) για λογικό ‘0’ (LOW). Εφόσον όλες οι γραμμές είναι σε λογικό ‘1’ (HIGH), δηλαδή κανένα πλήκτρο δεν πατήθηκε, επιλέγει τη δεύτερη στήλη (C2) και στέλνει λογικό ‘0’ (LOW). Κατά τη σάρωση της πρώτης γραμμής (R1), ανιχνεύει λογικό ‘0’ (LOW) και έτσι, μέσω του πίνακα, καταλαβαίνει ότι η στήλη C2 με τη γραμμή R1 αντιστοιχεί στον αριθμό 2. Για την διευκόλυνσή μας, θα χρησιμοποιηθεί στον κώδικα η βιβλιοθήκη <Keypad.h> από τους Mark Stanley και Alexander Brevig, όπου θα κάνει όλη αυτή την διαδικασία για εμάς.

### 5.3.1.2 Συνδεσμολογία πληκτρολογίου με το Arduino




Σχήμα 5.4: Ακροδέκτες Πληκτολογίου Matrix 4x4

Για την σύνδεση του πληκτρολογίου με το Arduino due, θα χρησιμοποιήσουμε τους ακροδέκτες 23, 25, 27, 29, 31, 33, 35 και 37.

Πίνακας 5.1: Συνέσεις πληκτρολογίου με το Arduino

| Ακροδέκτες Πληκτρολογίου Matrix 4x4 | Ακροδέκτες Arduino Due |
|---|---|
| ROW 1 | Ψηφιακός Ακροδέκτης 23 |
| ROW 2 | Ψηφιακός Ακροδέκτης 25 |
| ROW 3 | Ψηφιακός Ακροδέκτης 27 |
| ROW 4 | Ψηφιακός Ακροδέκτης 29 |
| COL 1 | Ψηφιακός Ακροδέκτης 31 |
| COL 2 | Ψηφιακός Ακροδέκτης 33 |
| COL 3 | Ψηφιακός Ακροδέκτης 35 |
| COL 4 | Ψηφιακός Ακροδέκτης 37 |

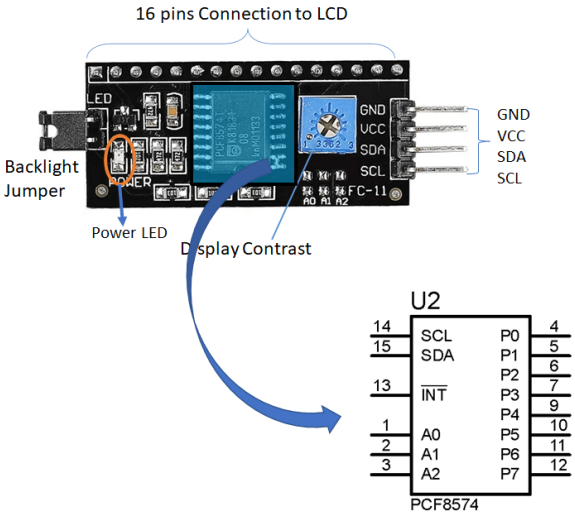
### 5.3.2 Οθόνη

Για την εμφάνιση δεδομένων του συστήματος συναγερμού, θα χρησιμοποιηθεί μια οθόνη υγρών κρυστάλλων (LCD - Liquid Crystal Display) με τον ελεγκτή HITACHI HD44780. Η οθόνη έχει μέγεθος 20 χαρακτήρων ανά 4 γραμμές (20x4). Η τάση λειτουργίας της είναι τα 5V, ενώ χρησιμοποιεί παράλληλη επικοινωνία και διαθέτει οπίσθιο φωτισμό.

Σχήμα 5.5: Οθόνη LCD 20x4 Character - White on Blue 5V

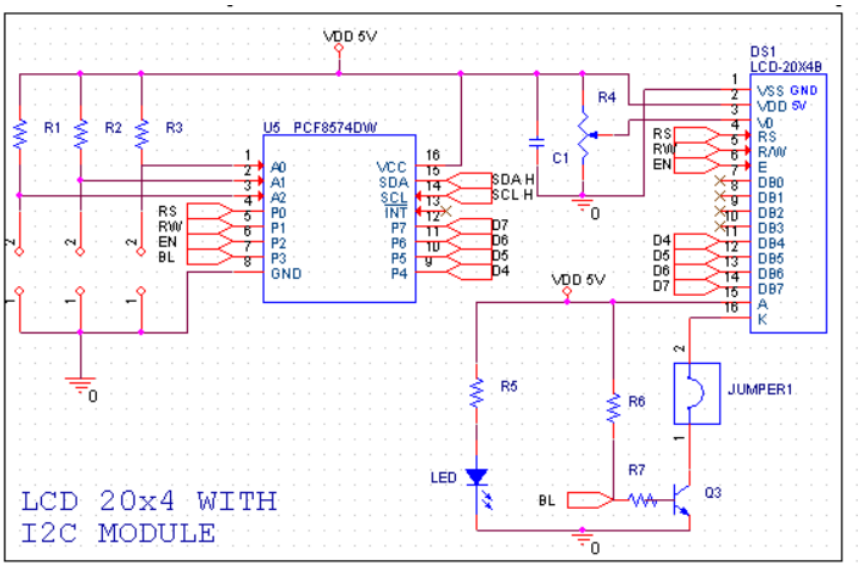
#### 5.3.2.1 Σύνδεση οθόνης με την συσκευή διεπαφής I2C

Η οθόνη για την σύνδεσή της περιλαμβάνει 16 ακίδες. Για την λειτουργία 4-bit χρειάζεται να συνδεθούν οι 12 από αυτές, ενώ για την λειτουργία 8-bit χρειάζονται και οι 16. Ανάλογα την λειτουργία που θα επιλέξουμε, θα πρέπει να χρησιμοποιήσουμε αντίστοιχα 7 ή 11 ψηφιακούς ακροδέκτες του μικροελεγκτή για την αποστολή δεδομένων προς την οθόνη[9][10]. Για να αποφύγουμε όλες αυτές τις συνδέσεις προς τον μικροελεγκτή, χρησιμοποιήθηκε μια συσκευή διεπαφής οθόνης υγρών κρυστάλλων με πρωτόκολλο επικοινωνίας I2C (LCD Display I2C Interface Module).




Σχήμα 5.6: Συσκευή διεπαφής οθόνης υγρών κρυστάλλων με πρωτόκολλο επικοινωνίας I2C (LCD Display I2C Interface Module)

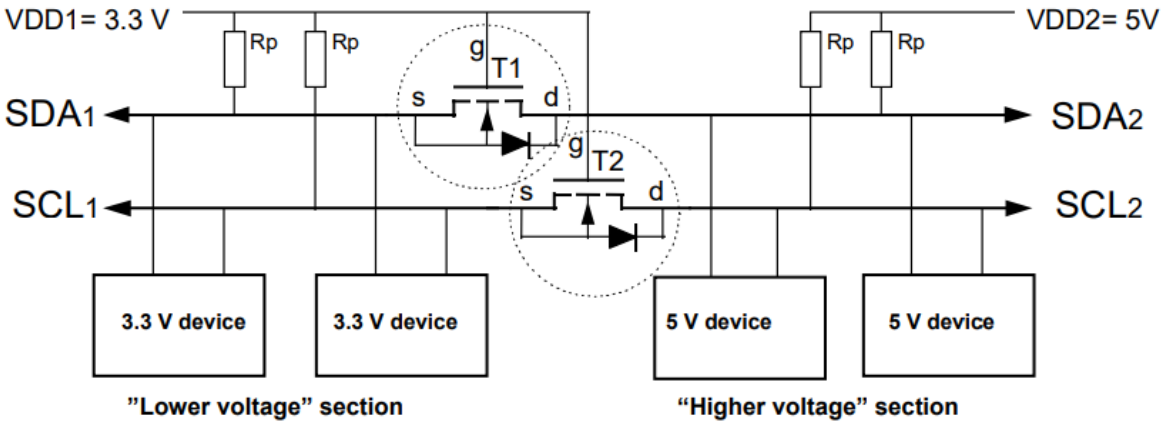
Η συσκευή διαθέτει ένα ενσωματωμένο κύκλωμα επέκτασης 8-bit I/O (Input/Output) για το πρωτόκολλο I2C (Inter-Integrated Circuit), το PCF8574. Αυτό το κύκλωμα μετατρέπει την σειριακή επικοινωνία σε παράλληλη και το αντίστροφο. Το πρωτόκολλο I2C είναι ένας τύπος σειριακής επικοινωνίας που χρησιμοποιεί δύο γραμμές μεταφοράς, την SDA (Serial Data Line) και την SCL (Serial Clock Line), για την μεταφορά δεδομένων ανάμεσα στις συσκευές. Κάθε συσκευή που χρησιμοποιεί το πρωτόκολλο I2C πρέπει να έχει μια μοναδική διεύθυνση για να την αναγνωρίζει ο μικροελεγκτής. Αυτές οι διευθύνσεις είναι 7 bits και μπορούν να κυμαίνονται από 0 έως 127. Ωστόσο, ο κατασκευαστής περιορίζει τις διευθύνσεις στο εύρος από 32 έως 39 (0x20 έως 0x27 στο δεκαεξαδικό σύστημα). [11]. Ακόμα, η συσκευή διαθέτει ένα ποτενσιόμετρο για την ρύθμιση της αντίθεσης της οθόνης, καθώς και έναν διακόπτη (jumper) που παρέχει τροφοδοσία στον οπίσθιο φωτισμό. Τέλος, αντί να έχουμε 12 συνδέσεις από την οθόνη προς τον μικροελεγκτή, έχουμε μόνο 4, όπου οι 2 είναι η τροφοδοσία και οι άλλες 2 είναι για την μεταφορά δεδομένων. Για την διευκόλυνσή μας, θα χρησιμοποιηθεί στον κώδικα η βιβλιοθήκη <LiquidCrystal_I2C.h> από τον Marco Schwartz, όπου θα κάνει όλη αυτή την διαδικασία για εμάς.




Σχήμα 5.7: Σχεδιασμός της συσκευής διεπαφής οθόνης LCD με πρωτόκολλο I2C

### 5.3.2.2 Μετατροπέας επιπέδου τάσης διπλής κατεύθυνσης

Για να επιτύχουμε την παραπάνω σύνδεση της συσκευής διεπαφής οθόνης που λειτουργεί στα 5V με το Arduino Due που λειτουργεί στα 3.3V μέσω του πρωτοκόλλου I2C, χρειαζόμαστε έναν μετατροπέα επιπέδου λογικής τάσης διπλής κατεύθυνσης (Bi-directional logic level shifter). Αυτό μπορούμε να το επιτύχουμε χρησιμοποιώντας ένα διακριτό MOS-FET για κάθε δίαυλο επικοινωνίας με συγκεκριμένα χαρακτηριστικά.




Σχήμα 5.8: Μετατροπέας επιπέδου τάσης διπλής κατεύθυνσης για δίαυλο επικοινωνίας I2C (Bi-directional level shifter for I2C bus)

Στο παραπάνω Σχήμα 5.8 παρατηρούμε ότι, στην αριστερή πλευρά υπάρχουν οι αντιστάσεις έλξης (pull-up resistors) και οι συσκευές συνδεδεμένες στην κατώτερη τάση (3.3V), ενώ στην δεξιά πλευρά υπάρχουν οι αντιστάσεις έλξης (pull-up resistors) και οι συσκευές συνδεδεμένες στην υψηλή τάση (5V). Οι συσκευές έχουν εισόδους/εξόδους οι οποίες λειτουργούν σύμφωνα με την τάση τροφοδοσίας τους και χρησιμοποιούν μια διάταξη εξόδου ανοιχτής εκροής (Open drain output configuration). Ο μετατροπέας τάσης (Level shifter) για κάθε δίαυλο επικοινωνίας είναι πανομοιότυπος και αποτελείται από ένα N-channel enhancement MOS-FET, το T1 για την γραμμή μεταφοράς δεδομένων και το T2 για την γραμμή χρονισμού. Οι πύλες (g) πρέπει να είναι συνδεδεμένες στην χαμηλή τάση τροφοδοσίας, οι πηγές (s) στην πλευρά που είναι ο κατώτερης τάσης δίαυλος επικοινωνίας και οι εκροές (d) στην πλευρά που είναι ο υψηλότερος τάσης δίαυλος επικοινωνίας. Αν στο MOS-FET δεν υπάρχει η εσωτερική δίοδος μεταξύ της πηγής και της εκροής, θα πρέπει να γίνει εξωτερικά η σύνδεσή της[12].

Για την λειτουργία μετατροπής του επίπεδου τάσης θα πρέπει να ληφθούν υπόψη τρείς καταστάσεις[12]:

- Κατάσταση 1. Καμία συσκευή δεν χρησιμοποιεί τον δίαυλο επικοινωνίας, οπότε στην πλευρά με την 'κατώτερη τάση' ο δίαυλος επικοινωνίας λόγω της αντίστασης έλξης (pull-up resistor) στα 3.3V. Η τάση της πύλης (g) και της πηγής (s) είναι η ίδια, στα 3.3V, άρα η τάση της $V_{GS}$ είναι χαμηλότερη από την τάση κατωφλίου ($V_{GS(TH)}$) και το MOS-FET δεν άγει. Αυτό επιτρέπει στην πλευρά με την 'υψηλότερη τάση', τον δίαυλο επικοινωνίας να είναι και αυτός στα 5V εξαιτίας της αντίστασης έλξης (pull-up resistor). Οπότε, οι δίαυλοι επικοινωνίας και στις δύο πλευρές είναι σε επίπεδο ΥΨΗΛΟ (HIGH) αλλά σε διαφορετική τάση.
- Κατάσταση 2. Μια συσκευή 3.3V χρησιμοποιεί τον δίαυλο επικοινωνίας, οπότε η τάση στον δίαυλο επικοινωνίας στην πλευρά της 'κατώτερης τάσης' θα είναι σε ΧΑΜΗΛΟ επίπεδο (LOW). Η τάση στην πηγή (s) του MOS-FET θα είναι και αυτή σε ΧΑΜΗΛΟ επίπεδο (LOW), η τάση στην πύλη (g) θα είναι στα 3.3V. Η τάση $V_{GS}$ θα αυξηθεί, με αποτέλεσμα να ξεπεράσει την τάση κατωφλίου ($V_{GS(TH)}$) και έτσι το MOS-FET θα ξεκινήσει να άγει. Εφόσον το MOS-FET άγει, η τάση στον δίαυλο επικοινωνίας της πλευράς 'υψηλότερης τάσης' θα είναι και

αυτός σε ΧΑΜΗΛΟ επίπεδο (LOW). Οπότε, οι δίαυλοι επικοινωνίας και στις δύο πλευρές είναι σε επίπεδο ΧΑΜΗΛΟ (LOW), με την ίδια τάση.

- Κατάσταση 3. Μια συσκευή 5V χρησιμοποιεί τον δίαυλο επικοινωνίας, οπότε η τάση στον δίαυλο επικοινωνίας στην πλευρά της 'υψηλότερης τάσης' θα είναι σε ΧΑΜΗΛΟ επίπεδο (LOW). Σε αυτήν την περίπτωση, η εσωτερική δίοδος που υπάρχει στο MOS-FET από την εκροή (d) προς το υπόστρωμα (substrate) πολώνεται ορθά. Καθώς η τάση στην πλευρά της 'κατώτερης τάσης' μειώνεται, θα φτάσει στο σημείο όπου η τάση $V_{GS}$ θα είναι μεγαλύτερη από την τάση κατωφλίου ($V_{GS(TH)}$), με αποτέλεσμα το MOS-FET να ξεκινήσει να άγει. Οπότε, ο δίαυλος επικοινωνίας στην πλευρά της 'κατώτερης τάσης' θα έρθει σε επίπεδο ΧΑΜΗΛΟ (LOW), άρα και οι δύο πλευρές θα είναι στο ίδιο επίπεδο και με την ίδια τάση.

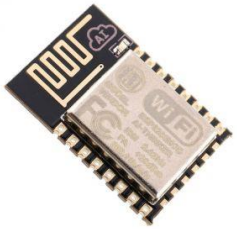
Τα τυπικά χαρακτηριστικά που πρέπει να έχει το MOS-FET για αυτό το κύκλωμα[12] δίνονται στον παρακάτω Πίνακας 5.2, μαζί και τα χαρακτηριστικά του MOS-FET που χρησιμοποιήθηκε, το 2N7000[13].

Πίνακας 5.2: Χαρακτηριστικά επιλογής MOS-FET και επιλεγόμενου MOS-FET

| | Typical MOS-FET | | 2N7000 |
|---|---|---|---|
| Type: | N-channel enhancement mode MOS-FET | | N-channel enhancement mode MOS-FET |
| Gate threshold voltage: | $V_{GS(th)}$ | min. 0.1V max. 2V | Typ. 2.1 V |
| On resistance: | $R_{DS(ON)}$ | max. 100 Ohm @ $I_D$= 3mA, $V_{GS}$= 2.5V | Typ. 2 Ohm @ $I_D$= 500mA, $V_{GS}$= 4.5V |
| Input capacitance: | $C_{iss}$ | max. 100 pF @ $V_{DS}$= 1V, $V_{GS}$ = 0V | Typ. 43 pF @ $V_{DS}$= 25V, $V_{GS}$ = 0V |
| Switching times: | $t_{on}$ $t_{off}$ | max. 50 ns. | Typ. $t_{on}$ = 5 ns, $t_{off}$ = 7 ns |
| Allowed drain current: | $I_D$ | 10 mA or higher. | 350 mA |

### 5.3.3 ESP8266-12E

Για να υπάρχει δυνατότητα ασύρματης σύνδεσης Wi-Fi του συστήματος συναγερμού, επιλέχθηκε το ESP-12E, που είναι μια από τις εκδόσεις του δημοφιλούς μικροελεγκτή ESP8266. Το ESP8266 συνδυάζει την επεξεργαστική ισχύ με την ασύρματη συνδεσιμότητα Wi-Fi. Θα χρησιμοποιηθεί σαν μια πλακέτα επέκτασης για το Arduino Due, ενώ η επικοινωνία τους θα γίνεται σειριακά μέσω της τεχνολογίας UART (Universal Asynchronous Receiver-Transmitter).



Σχήμα 5.9: ESP8266 WiFi Module ESP-12E

### 5.3.3.1 Χαρακτηριστικά ESP-12E

Το ESP12-E διαθέτει συνολικά 22 ακίδες που προσφέρουν πολλαπλές χρήσεις, ενώ η τροφοδοσία του είναι στα 3.3V. Το πρωτόκολλο Wi-Fi που χρησιμοποιεί είναι το 802.11 b/g/n στην συχνότητα 2.4-2.5 GHz. Ακόμα, μπορεί να λειτουργεί ως σημείο πρόσβασης (Access Point-AP) για άλλες συσκευές να συνδεθούν σε αυτό, να λειτουργεί ως σταθμός (Station-STA) και να συνδέεται σε ένα υπάρχον ασύρματο δίκτυο, και τέλος μπορεί να λειτουργεί ταυτόχρονα ως σταθμός και ως σημείο πρόσβασης για άλλες συσκευές[14].

**ESP12E WiFi Module Featues**

| FEATURES | DETAIL |
|---|---|
| WiFi Protocols | 802.11 b/g/n |
| Frequency | 2.4-2.5GHz |
| Security Protocol | WPA/WPA |
| Encryption types | WEP/TKIP/AES |
| Network Protocols | IPv4, TCP/UDP/FTP/HTTP |
| Wireless Network | STA / AP / STA + AP |
| Power Input | 3.0 - 3.3V |
| Operating Temperature Capacity | -40 - 125 |
| GPIO | 11 Channels |
| SPI | 1 Channel |
| I2C | 1 Channel |
| I2S | 1 Channel |
| IR Interface | 1 Channel |
| UART | 1 Channel |
| PWM | 3-pins |
| Serial Debug | Available |
| Ethernet | Not Available |
| WiFi | Available |
| Master Mode | Available |
| Slave Mode | Available |
| Hybrid Mode | Available |
| Bluetooth | Not Available |
| Onboard Antenna | Available |

Σχήμα 5.10: Χαρακτηριστικά ESP12-E

#### 5.3.3.2 Συνδεσμολογία ESP12-E με το Arduino Due

Όπως αναφέρθηκε παραπάνω, το Arduino Due με το ESP12-E επικοινωνούν μέσω της θύρας UART που υπάρχει και στις 2 πλακέτες. Για να επιτευχθεί αυτό θα πρέπει η ακίδα Tx από το Arduino να συνδεθεί στην ακίδα Rx του ESP και η ακίδα Rx του Arduino στην ακίδα Tx του ESP.

Πίνακας 5.3: Συνδεσμολογία ακίδων

| Arduino Due | ESP12-E |
|---|---|
| TX1 (Digital Pin 18) | RXD0 (Digital Pin 21) |
| RX1 (Digital Pin 19) | TXD0 (Digital Pin 22) |

#### 5.3.3.3 Σύνδεση ESP12-E με το δίκτυο και τον διακομιστή

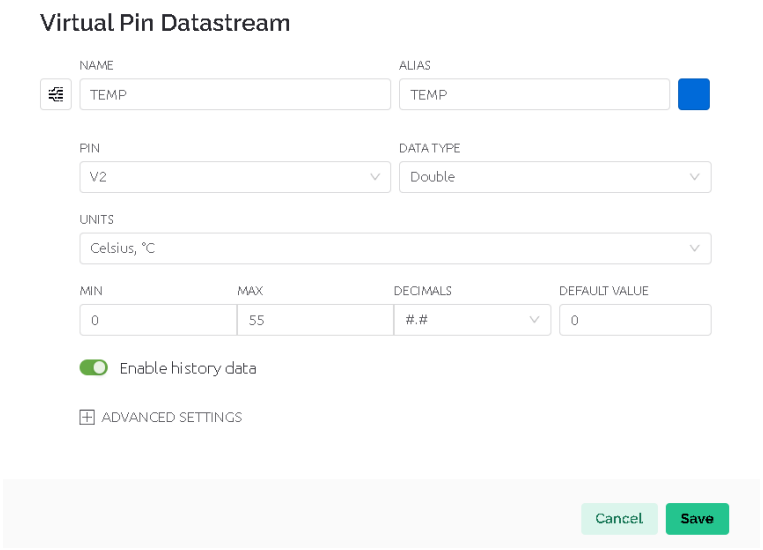
Για την διευκόλυνσή μας να συνδέσουμε το ESP12-E στο δίκτυο θα χρησιμοποιήσουμε την βιβλιοθήκη <ESP8266_Lib.h> και <BlynkSimpleShieldEsp8266.h>. Αρχικά, θα δημιουργήσουμε 2 πίνακες χαρακτήρων, έναν για την ονομασία του δικτύου «char ssid[] = "Όνομα δικτύου";» και έναν για τον κωδικό του δικτύου «char pass[] = "Κωδικός δικτύου";». Έπειτα, πρέπει να δημιουργήσουμε ένα αντικείμενο για την κλάση της βιβλιοθήκης, με την εντολή «ESP8266 wifi(&Serial1);», όπου Serial1 είναι οι ακροδέκτες TX1 και RX1 του Arduino Due. Αφού κάνουμε τα παραπάνω βήματα, στην συνάρτηση «void setup() {}» θα πούμε στο ESP12-E να συνδεθεί με το δίκτυο WiFi και μετά με τον διακομιστή (Server) του Blynk. Με την εντολή «Blynk.config(wifi, auth, "blynk.cloud", 80);» διαμορφώνουμε την σύνδεση του Blynk με το δίκτυο και τον διακομιστή, με την εντολή «Blynk.connectWiFi(ssid, pass)» λέμε στο ESP12-E να συνδεθεί στο δίκτυο και με την εντολή «Blynk.connect();» λέμε στο ESP12-E να συνδεθεί με τον διακομιστή. Τέλος, στην συνάρτηση «void loop() {}» καλούμε την εντολή «Blynk.run();» που διαχειρίζεται την σύνδεση με τον διακομιστή του Blynk, και μας επιτρέπει να λαμβάνουμε δεδομένα από την εφαρμογή, αλλά και να στέλνουμε δεδομένα πίσω στην εφαρμογή.

### 5.3.4 Απομακρυσμένος έλεγχος του Arduino

Για τον απομακρυσμένο έλεγχο του Arduino και την παρακολούθηση των αισθητήρων, χρησιμοποιούμε τη βιβλιοθήκη Blynk, μέσω του ESP12-E που είναι η γέφυρα επικοινωνίας μας. Για την αποστολή και την λήψη πληροφοριών στην εφαρμογή Blynk χρειάστηκε να γράψουμε κάποιες γραμμές κώδικα στο περιβάλλον του Arduino IDE, ενώ για να έχουμε αλληλεπίδραση με την εφαρμογή Blynk δημιουργήσαμε και προσαρμόσαμε κάποια γραφικά στοιχεία (Widgets) εντός της εφαρμογής, τέλος για τις ειδοποιήσεις χρησιμοποιήσαμε την επιλογή συμβάντα (Events).

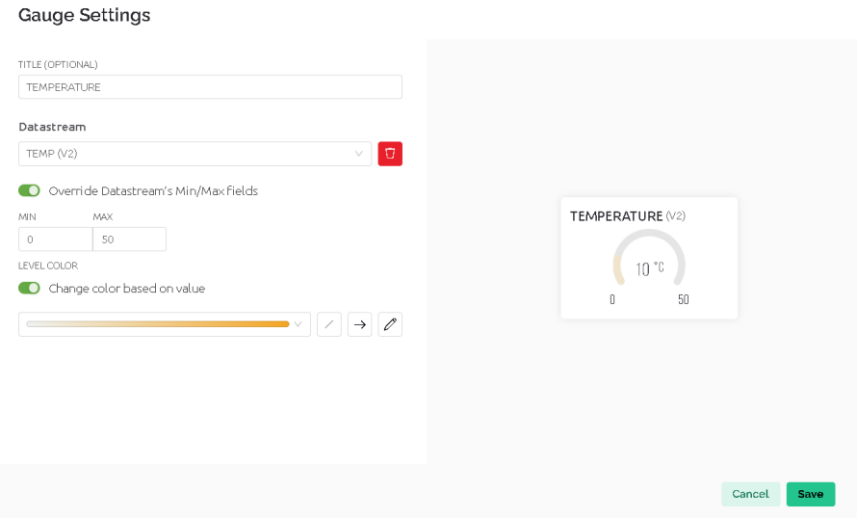
#### 5.3.4.1 Εικονικοί ακροδέκτες

Όλες οι πληροφορίες μεταξύ του Arduino και της εφαρμογής ανταλλάσσονται μέσω των εικονικών ακροδεκτών (Virtual pins) και όχι μέσω των φυσικών ακροδεκτών (Physical GPIO pins). Ο εικονικός ακροδέκτης είναι το κανάλι επικοινωνίας μεταξύ του Arduino και της εφαρμογής. Παράδειγμα για την αποστολή της θερμοκρασίας του αισθητήρα DHT11, δημιουργούμε στην εφαρμογή μια ροή δεδομένων (Datastream) και επιλέγουμε τον τύπο της, στην περίπτωσή μας θα είναι εικονικός ακροδέκτης (Virtual pin). Έπειτα, επιλέγουμε σε ποιόν εικονικό ακροδέκτη θέλουμε να αντιστοιχήσουμε αυτήν την ροή δεδομένων, αλλά και τον τύπο των δεδομένων που θα εισέρχονται, στην περίπτωσή μας θα είναι αριθμητικές τιμές με δεκαδικά ψηφία (Double).




Σχήμα 5.11: Δημιουργία εικονικού ακροδέκτη (Virtual pin)

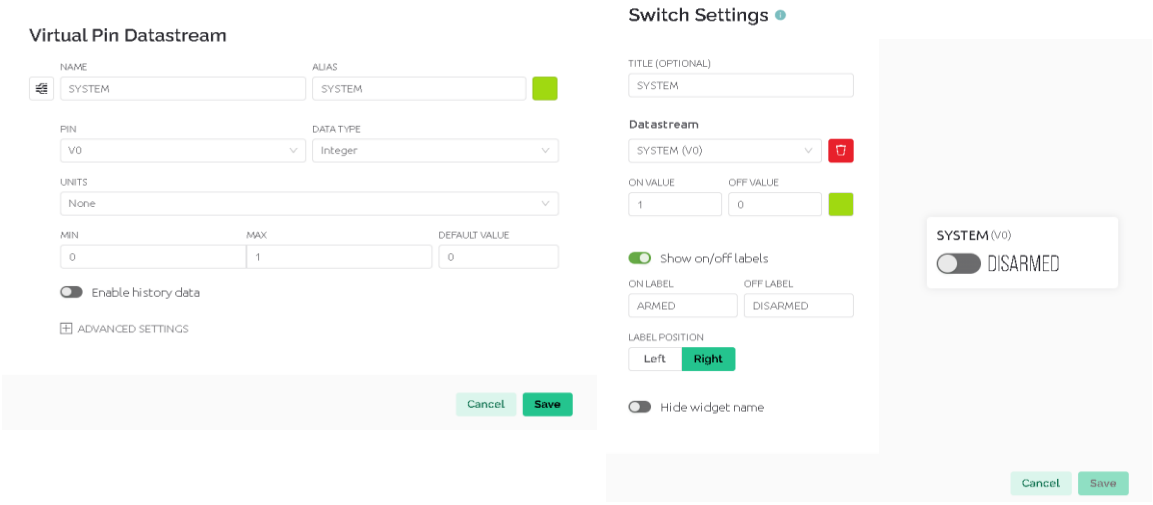
Αφού πλέον δημιουργήσαμε την ροή δεδομένων μας, ήρθε η στιγμή να αποφασίσουμε πως θα απεικονίσουμε αυτά τα δεδομένα. Θα επιλέξουμε να απεικονίσουμε τα δεδομένα με το γραφικό στοιχείο δείκτης (Gauge widget). Κατά την δημιουργία του εικονικού στοιχείου επιλέγουμε τον τύπο της ροής δεδομένων, δηλαδή τον εικονικό ακροδέκτη που φτιάξαμε νωρίτερα.




Σχήμα 5.12: Δημιουργία εικονικού στοιχείου δείκτη (Gauge widget)

Για να μπορέσουμε τώρα να στείλουμε όμως αυτά τα δεδομένα από το Arduino, πρέπει να χρησιμοποιήσουμε κάποιες εντολές από την βιβλιοθήκη Blynk. Η εντολή για να στέλνουμε δεδομένα προς την εφαρμογή είναι η «Blynk.virtualWrite(Vpin, sensorData);», και όπου Vpin είναι το νούμερο του εικονικού ακροδέκτη που δημιουργήσαμε νωρίτερα, ενώ όπου sensorData είναι η μεταβλητή που έχουμε δημιουργήσει στον κώδικά μας για να αποθηκεύουμε τιμές. Στην περίπτωσή μας η εντολή αυτή διαμορφώνεται ως εξής «Blynk.virtualWrite(V2, temperature);».

Όπως αναφέραμε και παραπάνω, μπορούμε να λάβουμε και δεδομένα από την εφαρμογή, με την χρήση έτοιμης συνάρτησης στον κώδικά μας από την βιβλιοθήκη Blynk. Η συνάρτηση αυτή ονομάζεται «BLYNK_WRITE(Vpin){}», και όπου Vpin είναι το νούμερο του εικονικού ακροδέκτη που δημιουργήσαμε. Μέσα στην συνάρτηση μπορούμε να χρησιμοποιήσουμε την μέθοδο «param.asInt();» και να μετατρέψουμε την τιμή του εικονικού ακροδέκτη σε ακέραιο αριθμό και να την αποθηκεύσουμε σε μία μεταβλητή που έχουμε δημιουργήσει ώστε να την επεξεργαστούμε. Παράδειγμα, για την όπλιση και αφόπλιση του συστήματος συναγερμού δημιουργήσαμε μια νέα ροή δεδομένων, τον εικονικό ακροδέκτη V0, και τον τύπο δεδομένων του σαν ακέραιο. Το εικονικό στοιχείο που διαλέξαμε για αυτήν την περίπτωση είναι του διακόπτη (Switch) και του δώσαμε 2 τιμές, την τιμή 0 όταν είναι κλειστός και την τιμή 1 όταν είναι ανοιχτός.

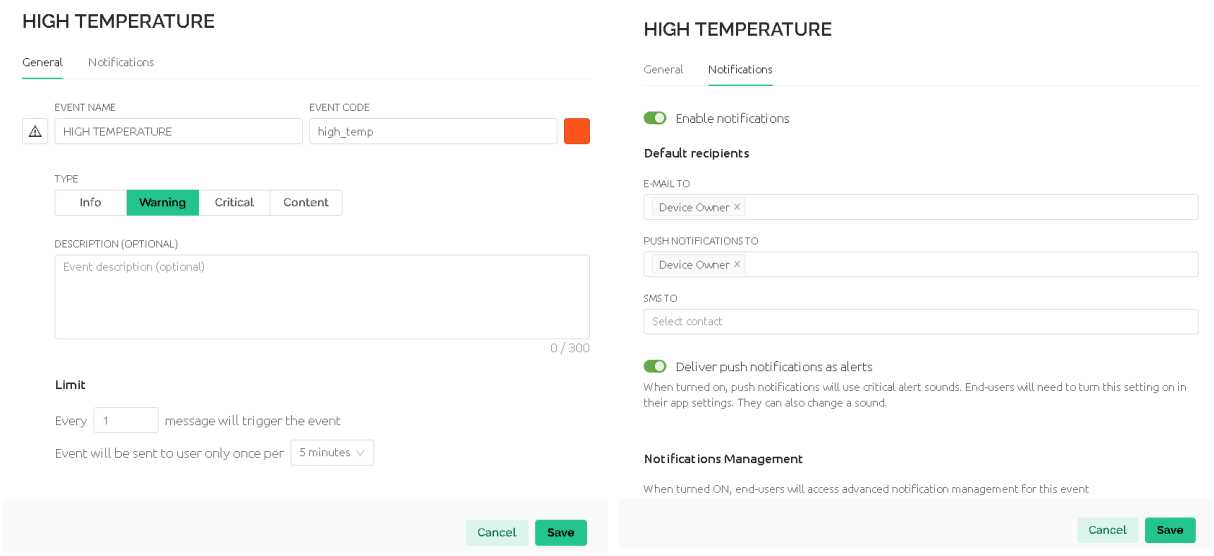



Σχήμα 5.13: Εικονικός διακόπτης και εικονικό στοιχείο για κατάσταση συναγερμού

Στον κώδικά μας καλούμε την έτοιμη συνάρτηση «BLYNK_WRITE(V0){}», και αποθηκεύουμε την τιμή του εικονικού ακροδέκτη στην μεταβλητή με την εντολή «int value = param.asInt();». Έπειτα, ανάλογα την τιμή της μεταβλητής επιλέγουμε είτε να οπλίσουμε είτε να αφοπλίσουμε το σύστημα συναγερμού με τον κατάλληλο κώδικα.

### 5.3.4.2 Συμβάντα

Η εφαρμογή Blynk, πέρα από την αποστολή και λήψη δεδομένων μας επιτρέπει να δημιουργούμε και συμβάντα (Events). Τα συμβάντα χρησιμοποιούνται για να καταγράφουν και να διαχειρίζονται σημαντικά γεγονότα που συμβαίνουν στην συσκευή. Επίσης, τα συμβάντα χρησιμοποιούνται για τις ειδοποιήσεις που μπορούν να σταλούν μέσω email, μέσω push ειδοποιήσεων στο κινητό τηλέφωνο ή να αποσταλούν ως μήνυμα (SMS). Εμείς, θα χρησιμοποιήσουμε αυτήν την λειτουργία για να στέλνουμε email και ειδοποιήσεις push. Αρχικά, πρέπει να δημιουργήσουμε ένα νέο συμβάν στην εφαρμογή και δώσουμε ένα κωδικό όνομα στο συμβάν ώστε όταν θα το καλούμε με την κατάλληλη εντολή στον κώδικά μας, να γνωρίζει η εφαρμογή σε ποιο συμβάν αναφερόμαστε. Παράδειγμα, για την ειδοποίηση αύξησης της θερμοκρασίας έχουμε δημιουργήσει ένα συμβάν με όνομα «HIGH TEMPERATURE» και κωδικό συμβάν «high_temp». Στην δίπλα καρτέλα που είναι οι ειδοποιήσεις, επιλέγουμε να είναι ενεργές και τέλος επιλέγουμε σε ποιόν θα στέλνει email και push ειδοποιήσεις η εφαρμογή.




Σχήμα 5.14: Δημιουργία συμβάν στην εφαρμογή Blynk

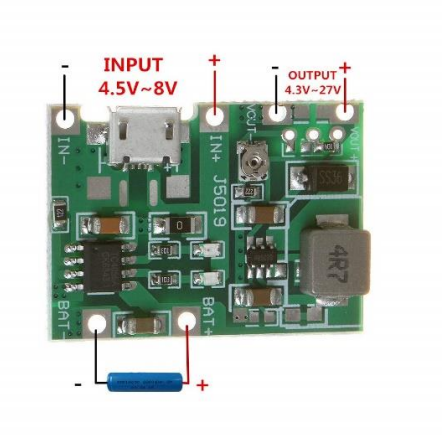
Στον κώδικά μας, χρησιμοποιούμε την εντολή «Blynk.logEvent(event_code);» και όπου «event_code» είναι ο κωδικός συμβάν που έχουμε δώσει στην εφαρμογή. Στην περίπτωση του παραδείγματος με την θερμοκρασία, είναι ο κωδικός «high_temp». Οπότε η εντολή διαμορφώνεται ως εξής «Blynk.logEvent("high_temp", String("High Temperature: ") + temperature);» μαζί με την περιγραφή και την τιμή της μεταβλητής για την θερμοκρασία.

### 5.3.5 Τροφοδοσία

Για την τροφοδοσία του συστήματος συναγερμού, χρησιμοποιούνται δύο πηγές ενέργειας. Η κύρια τροφοδοσία είναι ένα τροφοδοτικό USB 5V 2A, ενώ υπάρχει και μια δευτερεύουσα τροφοδοσία που αποτελείται από μια επαναφορτιζόμενη μπαταρία λιθίου 18650 με τάση 3.6V και χωρητικότητα 2600mAh. Και οι δύο τροφοδοσίες συνδέονται σε ένα Σύστημα Διαχείρισης Μπαταρίας (Battery Management System - BMS), το οποίο τροφοδοτεί το σύστημα συναγερμού είτε από την κύρια είτε από τη δευτερεύουσα πηγή ενέργειας.

#### 5.3.5.1 Σύστημα διαχείρισης μπαταρίας

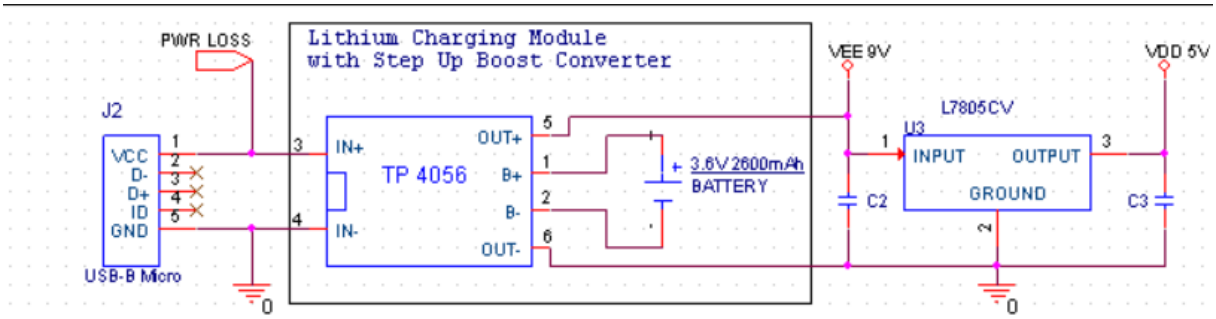
Η πλακέτα που θα χρησιμοποιήσουμε για το σύστημα διαχείρισης μπαταρίας είναι η HW-357, η οποία διαθέτει έναν ενσωματωμένο φορτιστή μπαταρίας TP4056A και έναν μετατροπέα ανύψωσης τάσης (step-up boost converter) στην έξοδο. Ο φορτιστής μπαταρίας TP4056A χρησιμοποιείται για τη φόρτιση μιας μπαταρίας λιθίου, όπως οι μπαταρίες Li-Ion και LiPo, με ασφάλεια και αποτελεσματικότητα. Η τάση φόρτισης της μπαταρίας είναι τα 4.2V, ενώ το ρεύμα είναι έως 1A και διαθέτει προστασία υπερφόρτισης. Ο μετατροπέας ανύψωσης τάσης παρέχει τη δυνατότητα αύξησης της τάσης εξόδου σε σχέση με την τάση εισόδου, επιτρέποντας τη χρήση χαμηλής τάσης εισόδου, όπως USB, για να παρέχει υψηλότερη τάση εξόδου στο κύκλωμα, καθώς το Arduino Due χρειάζεται τάση εισόδου στον ακροδέκτη Vin από 7-12V. Με την βοήθεια του ποτενσιομέτρου που βρίσκεται επάνω στην πλακέτα θα ρυθμίσουμε την τάση εξόδου να είναι στα 9V.




Σχήμα 5.15: Πλακέτα HW-357

### 5.3.5.2 Συνδεσμολογία συστήματος διαχείρισης μπαταρίας

Η συνδεσμολογία της πλακέτας σύμφωνα με το Σχήμα 5.15 και το Σχήμα 5.16 είναι η εξής, η κύρια τροφοδοσία μας που είναι ένα τροφοδοτικό USB 5V 2A θα συνδεθεί στην θύρα USB στο επάνω μέρος της πλακέτας, αυτή θα είναι που θα φορτίζει την μπαταρίας μας και θα παρέχει τροφοδοσία στο σύστημα συναγερμού. Στις ακίδες στο κάτω μέρος της πλακέτας, θα συνδέσουμε την μπαταριοθήκη για να είναι πιο εύκολο να αλλάζουμε την μπαταρία. Η ακίδα OUTPUT (+) θα συνδεθεί στον ακροδέκτη Vin του Arduino Due, αλλά και με έναν εξωτερικό ρυθμιστή τάσης (Voltage regulator) L7805CV-5V-1.5A, ώστε να τροφοδοτούμε με την οθόνη και τον αισθητήρα κίνησης που έχουν τάση λειτουργίας 5V. Η ακίδα OUTPUT (-) θα συνδεθεί με τον ακροδέκτη GND του Arduino Due και με τον ακροδέκτη GND του ρυθμιστή τάσης για να έχουν κοινή τάση αναφοράς. Τέλος, ένα καλώδιο jumper θα συνδεθεί από την ακίδα IN (+) της πλακέτας στον αναλογικό ακροδέκτη A0 του Arduino Due μέσω ενός διαιρέτης τάσης ώστε η τάση στην είσοδο A0 να μην ξεπερνάει τα 3.3V και πάθει βλάβη ο ακροδέκτης. Ο σκοπός του είναι η ανίχνευση της απώλειας της κύριας τροφοδοσίας ώστε μετά με τον κατάλληλο κώδικα να ενημερώνεται ο χρήστης.



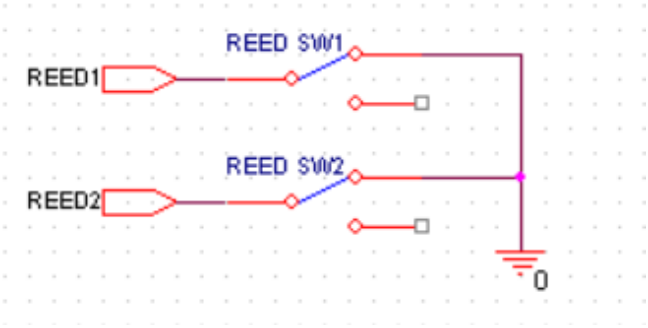

Σχήμα 5.16: Συνδεσμολογία συστήματος διαχείρισης μπαταρίας

### 5.3.6 Μαγνητικές επαφές

Για τον έλεγχο της κεντρικής πόρτας εισόδου και του παραθύρου στο σύστημα συναγερμού θα χρησιμοποιηθούν 2 μαγνητικές επαφές με κωδικό προϊόντος DC-1561, λευκού χρώματος, της εταιρίας ALEPH. Η συνδεσμολογίας τους είναι αρκετά απλή, όπως ένας κοινός διακόπτης, αυτό μας επιτρέπει να ανιχνεύουμε αν η πόρτα ή το παράθυρο, είναι ανοιχτό ή κλειστό.

### 5.3.6.1 Συνδεσμολογία μαγνητικής επαφής

Από την μαγνητική επαφή έχουμε 2 καλώδια, το ένα καλώδιο είναι συνδεμένο στην μία πλευρά του μαγνητικού διακόπτη ενώ το δεύτερο καλώδιο είναι συνδεμένο στην άλλη πλευρά του μαγνητικού διακόπτη. Το ένα καλώδιο το συνδέουμε στον ακροδέκτη της γείωσης (GND) του Arduino, ενώ το δεύτερο καλώδιο το συνδέουμε σε έναν ψηφιακό ακροδέκτη του Arduino που τον έχουμε ρυθμίσει να είναι είσοδος με ενεργοποιημένη την εσωτερική αντίσταση έλξης (Internal pull-up resistor). Έτσι, όταν η μαγνητική επαφή είναι κλειστή ο ψηφιακός ακροδέκτης του Arduino διαβάζει λογικό ΧΑΜΗΛΟ ‘0’ (LOW), και έτσι γνωρίζει ότι καμία παραβίαση δεν έχει γίνει. Αντιθέτως, όταν ο ακροδέκτης διαβάσει λογικό ΥΨΗΛΟ ‘1’ (HIGH) καταλαβαίνει ότι η μαγνητική επαφή έχει ανοίξει και ότι υπάρχει παραβίαση.



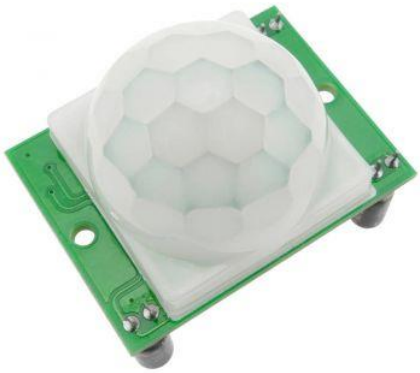

Σχήμα 5.17: Μαγνητικές επαφές

Πίνακας 5.4: Συνδεσμολογία μαγνητικών επαφών

| Καλώδια μαγνητικής επαφής | Ψηφιακοί ακροδέκτες Arduino Due |
|---|---|
| 1ο καλώδιο 1ης μαγνητικής επαφής | Ακροδέκτης 22 |
| 1ο καλώδιο 2ης μαγνητικής επαφής | Ακροδέκτης 24 |
| 2ο καλώδιο 1ης και 2ης μαγνητικής επαφής | Ακροδέκτης Γείωσης (GND) |

### 5.3.7 Ανιχνευτής κίνησης

Για την ανίχνευση κίνησης στο σύστημα συναγερμού θα χρησιμοποιήσουμε έναν παθητικό ανιχνευτή υπέρυθρης ακτινοβολίας (PIR) και συγκεκριμένα τον HC-SR501. Θα τροφοδοτείται μέσω του ρυθμιστή τάσης (Voltage regulator) και η έξοδός του θα συνδέεται σε έναν ψηφιακό ακροδέκτη του Arduino Due.



Σχήμα 5.18: Παθητικός ανιχνευτής υπέρυθρης ακτινοβολίας (PIR) HC-SR501

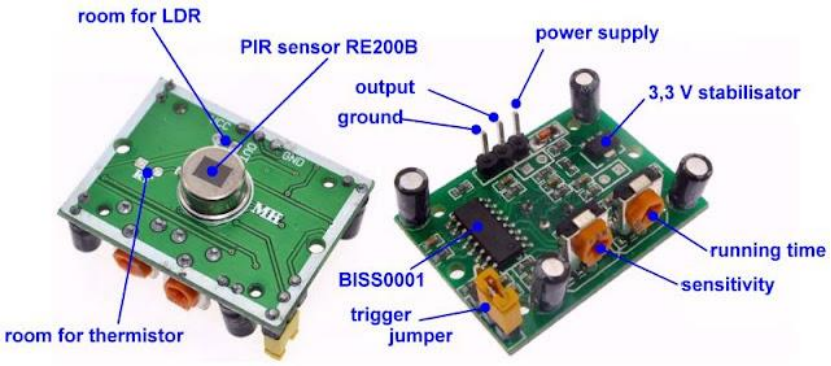
#### 5.3.7.1 Πλακέτα ανιχνευτή κίνησης (PIR)

Στην επάνω πλευρά της πλακέτας του HC-SR501 σύμφωνα με το Σχήμα 5.19, βρίσκεται ο φακός Fresnel που βοηθάει στην αύξηση της εμβέλειας και το πεδίο δράσης του αισθητήρα. Αν τον αφαιρέσουμε, θα δούμε τον παθητικό ανιχνευτή υπέρυθρης ακτινοβολίας τύπου RE200B από την Glolab Corporation, όπου τροφοδοτείται από έναν σταθεροποιητή τάσης στα 3.3V, ενώ αποτελείται από 2 στοιχεία ανίχνευσης συνδεδεμένα με μια διάταξη αντιστοίχισης τάσης (Voltage bucking configuration). Αυτή η διάταξη είναι σχεδιασμένη να αναιρεί τα σήματα που προκαλούνται από αλλαγές θερμοκρασίας, αλλά όταν ένα σώμα περνά μπροστά από τον ανιχνευτή, τότε το πρώτο στοιχείο ενεργοποιείται πρώτα και στην συνέχεια το δεύτερο στοιχείο[15].

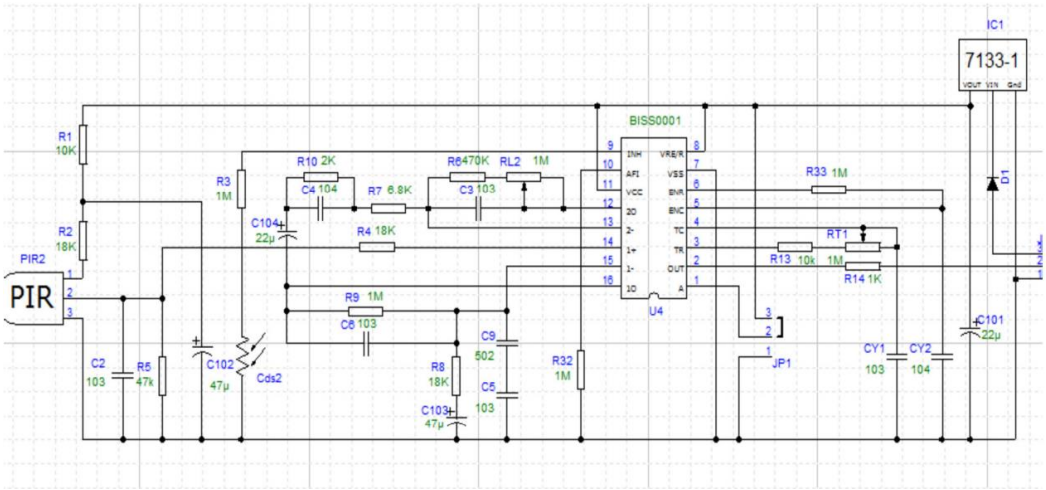
Στην κάτω πλευρά σύμφωνα με το Σχήμα 5.19, υπάρχει το ολοκληρωμένο BISS0001 που είναι ένας ελεγκτής σχεδιασμένος ειδικά για την χρήση του σε παθητικούς ανιχνευτές υπέρυθρης ακτινοβολίας, καθώς χρησιμοποιεί αναλογικές και ψηφιακές τεχνικές σχεδίασης μείωσης θορύβου και κατασκευάζεται με την τεχνολογία CMOS. Αποτελεί έναν από τους πιο σταθερούς ελεγκτές PIR που υπάρχουν[16].

Ακόμα η τροφοδοσία του είναι από 5-20V, καθώς διαθέτει έναν ακριβείας ρυθμιστή τάσης στα 3.3V, ενώ υπάρχει και μια δίοδος προστασίας για τυχόν ανάποδη τροφοδοσία.

Διαθέτει 2 ποτενσιόμετρα στην κάτω πλευρά της πλακέτας σύμφωνα με το Σχήμα 5.19, το αριστερά είναι για την ρύθμιση της απόστασης κάλυψης του ανιχνευτή, το δεξιά είναι για την ρύθμιση του χρόνου που θα παραμένει σε κατάσταση ΥΨΗΛΗ (HIGH) η έξοδος μετά από την ανίχνευση[17].




Σχήμα 5.19: (απο αριστερά) Επάνω πλευρά πλακέτας, Κάτω πλευρά πλακέτας




Σχήμα 5.20: Σχηματικό HC-SR501

Πίνακας 5.5: Χαρακτηριστικά HC-SR501

| Τύπος προϊόντος | HC-SR501 |
|---|---|
| Εύρος τάσης λειτουργίας | 5-20VDC |
| Ρεύμα αδράνειας | <50uA |
| Τάση εξόδου | High 3.3V / Low 0V |
| Χρόνος καθυστέρησης | 5-300S |
| Γωνία ανίχνευσης | <100° γωνιακή κάλυψη |
| Θερμοκρασία λειτουργίας | -15 - +70 κελσίου |
| Μέγεθος φακού αισθητήρα | Διάμετρος: 23mm(προκαθορισμένο) |

#### 5.3.7.2 Συνδεσμολογία ανιχνευτή κίνησης με Arduino

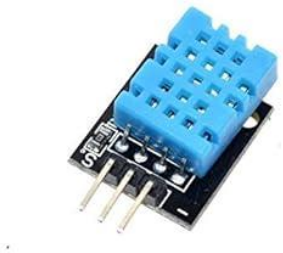
Η συνδεσμολογία του παθητικού ανιχνευτή υπέρυθρης ακτινοβολίας με το Arduino Due σύμφωνα με το Σχήμα 5.20, γίνεται με την σύνδεση της μεσαίας ακίδας του ανιχνευτή στον ψηφιακό ακροδέκτη 26 του Arduino.

Πίνακας 5.6: Συνδεσμολογία με το Arduino Due

| HC-SR501 | Arduino Due |
|---|---|
| Ακίδα 1 | Γείωση (GND) |
| Ακίδα 2 | Ψηφιακός ακροδέκτης 26 |
| Ακίδα 3 | 5V (από τον ρυθμιστή τάσης) |

### 5.3.8 Αισθητήρας Υγρασίας και Θερμοκρασίας

Για την μέτρηση της υγρασίας και της θερμοκρασίας στον χώρο, θα χρησιμοποιηθεί ο αισθητήρας DHT11 που έχει δύο αισθητήρες εσωτερικά, μαζί με έναν 8-bit μικροελεγκτή για την αποστολή των δεδομένων. Ακόμα, είναι ενσωματωμένος σε μια πλακέτα διασύνδεσης (Breakout board) που έχει μια αντίσταση έλξης 4.7ΚΩ συνδεμένη στην έξοδο για την ορθή επικοινωνία με το Arduino.

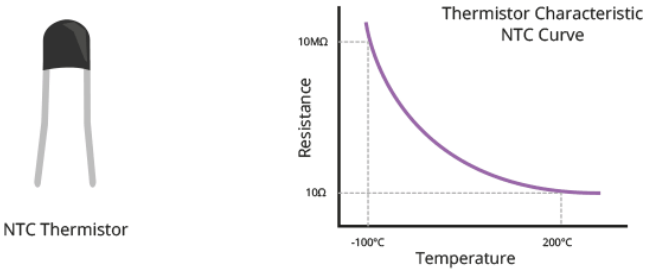


Σχήμα 5.21: Αισθητήρας υγρασίας και θερμοκρασίας DHT11

### 5.3.8.1 Λειτουργία αισθητήρα

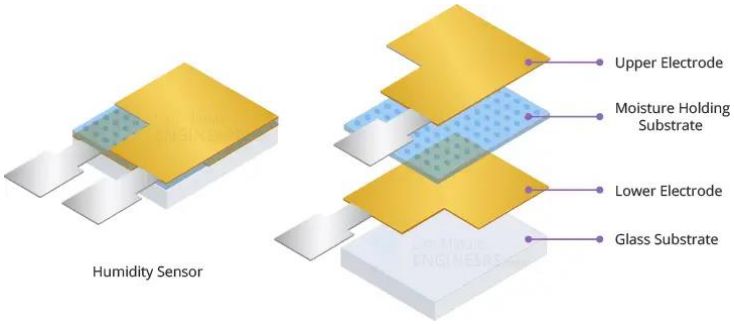
Ο αισθητήρας DHT11, έχει την ικανότητα να μετράει την θερμοκρασία σε ένα εύρος από 0ºC έως 50ºC με ακρίβεια ±2.0ºC, ενώ το εύρος μέτρησης της υγρασίας είναι από 20% έως 80% με ακρίβεια 5%. Ο ρυθμός δειγματοληψίας είναι 1Hz, πράγμα που σημαίνει ότι μπορεί να παρέχει νέα δεδομένα μία φορά κάθε δευτερόλεπτο. Η τάση τροφοδοσίας του κυμαίνεται από 3.0-5.5V[18-19]. Στην περίπτωσή μας θα το τροφοδοτήσουμε με 3.3V από το Arduino Due.

Για την μέτρηση της θερμοκρασίας, έχει μια θερμοαντίσταση με αρνητικό συντελεστή (Negative Temperature Coefficient-NTC) ή αλλιώς θερμίστορ (Thermistor) που είναι στην επιφάνεια της εσωτερικής πλακέτας. Τα θερμίστορ είναι αισθητήρες μεταβλητής αντίστασης και η αντίστασή τους μειώνεται με την αύξηση της περιβαλλοντικής θερμοκρασίας[19].




Σχήμα 5.22: Θερμίστορ και διάγραμμα σχέσης μεταξύ θερμοκρασίας και υγρασίας

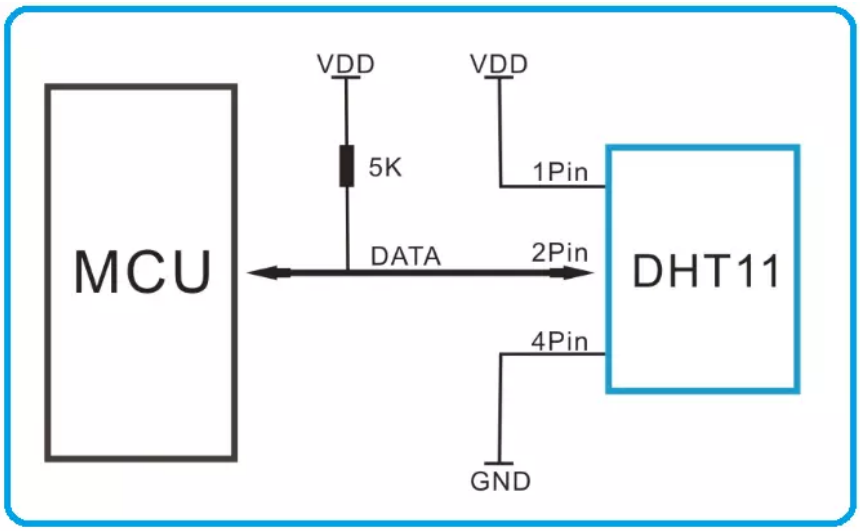
Για την μέτρηση της υγρασίας, έχει έναν χωρητικό αισθητήρα υγρασίας, ο οποίος διαθέτει δυο ηλεκτρόδια και ένα υλικό υπόστρωμα ανάμεσά τους. Το υλικό υπόστρωμα χρησιμοποιείται για να συγκρατεί την υγρασία στην επιφάνεια του. Καθώς η υγρασία αυξάνεται, το υπόστρωμα απορροφά τον ατμό νερού, με αποτέλεσμα την απελευθέρωση ιόντων και μια μείωση της αντίστασης μεταξύ των δύο ηλεκτροδίων. Αυτή η μεταβολή στην αντίσταση των ηλεκτροδίων στη συνέχεια βαθμονομείται χρησιμοποιώντας το συντελεστή υγρασίας (αποθηκευμένο στη μνήμη OTP) και εκδίδεται η τελική τιμή της σχετικής υγρασίας[19].




Σχήμα 5.23: Εσωτερική δομή του αισθητήρα υγρασίας DHT11

Τέλος, ο αισθητήρας όπως αναφέραμε νωρίτερα διαθέτει και έναν μικροελεγκτή για την αποστολή των δεδομένων σε έναν άλλον μικροελεγκτή. Αυτό είναι εφικτό αν συνδέσουμε τον ακροδέκτη εξόδου του αισθητήρα μέσω ενός δίαυλου επικοινωνίας σε έναν ψηφιακό ακροδέκτη του Arduino. Το πρωτόκολλο επικοινωνίας που χρησιμοποιεί είναι το Single Wire Interface (SWI) και χρειάζεται μια αντίσταση έλξης στον δίαυλο επικοινωνίας, όπου στην περίπτωσή μας που ο αισθητήρας είναι ενσωματωμένος στην πλακέτα διασύνδεσης (Breakout board) αυτή είναι ήδη συνδεδεμένη στον ακροδέκτη εξόδου του αισθητήρα, ενώ η επικοινωνία είναι διπλής κατεύθυνσης. Ο αισθητήρας, όταν δεν στέλνει δεδομένα είναι σε λειτουργία χαμηλής κατανάλωσης ενέργειας και ο ακροδέκτης εξόδου είναι σε κατάσταση ΥΨΗΛΗ (HIGH) μέσω της αντίστασης έλξης. Για να λάβει το Arduino δεδομένα από τον αισθητήρα,

θα πρέπει να φέρει τον δίαυλο επικοινωνίας σε κατάσταση ΧΑΜΗΛΗ (LOW) για τουλάχιστον 18μs ώστε να το αντιληφθεί ο αισθητήρας. Αφού ο αισθητήρας καταλάβει ότι το Arduino έφερε τον δίαυλο επικοινωνίας σε κατάσταση ΧΑΜΗΛΗ (LOW), τότε αλλάζει από την κατάσταση χαμηλής κατανάλωσης σε κατάσταση λειτουργίας, και περιμένει να έρθει ο δίαυλος επικοινωνίας σε κατάσταση ΥΨΗΛΗ (HIGH) από το Arduino. Έπειτα, ο αισθητήρας στέλνει σειριακά της βαθμονομημένες τιμές σε 40-bit πληροφορίας, και ξαναεπιστρέφει σε κατάσταση χαμηλής κατανάλωσης περιμένοντας την επόμενη εντολή για δεδομένα από το Arduino. Τέλος, ο χρόνος που πρέπει να περιμένει το Arduino για να λάβει τα δεδομένα από τον αισθητήρα είναι από 20-40μs[18-19].




Σχήμα 5.24: Συνδεσμολογία αισθητήρα με Arduino

Για την ευκολία της χρήσης του αισθητήρα και την λήψη των δεδομένων, θα χρησιμοποιήσουμε την βιβλιοθήκη <DHT.h> από την Adafruit Industries.

### 5.3.8.2 Συνδεσμολογία αισθητήρα με Arduino

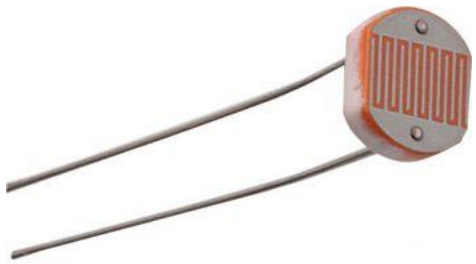
Για την συνδεσμολογία του αισθητήρα με το Arduino, θα συνδέσουμε την ακίδα εξόδου σε έναν ψηφιακό ακροδέκτη του Arduino, ενώ η τροφοδοσία του γίνει από τον ακροδέκτη 3.3V του Arduino.

Πίνακας 5.7: Συνδεσμολογία ακροδεκτών DHT11 με Arduino

| Αισθητήρας DHT11 | Arduino Due |
|---|---|
| Ακίδα εξόδου | Ψηφιακός ακροδέκτης 48 |
| Ακίδα τροφοδοσίας | Τροφοδοσία 3.3V |
| Ακίδα γείωσης (GND) | Γείωση (GND) |

### 5.3.9 Αισθητήρας φωτός

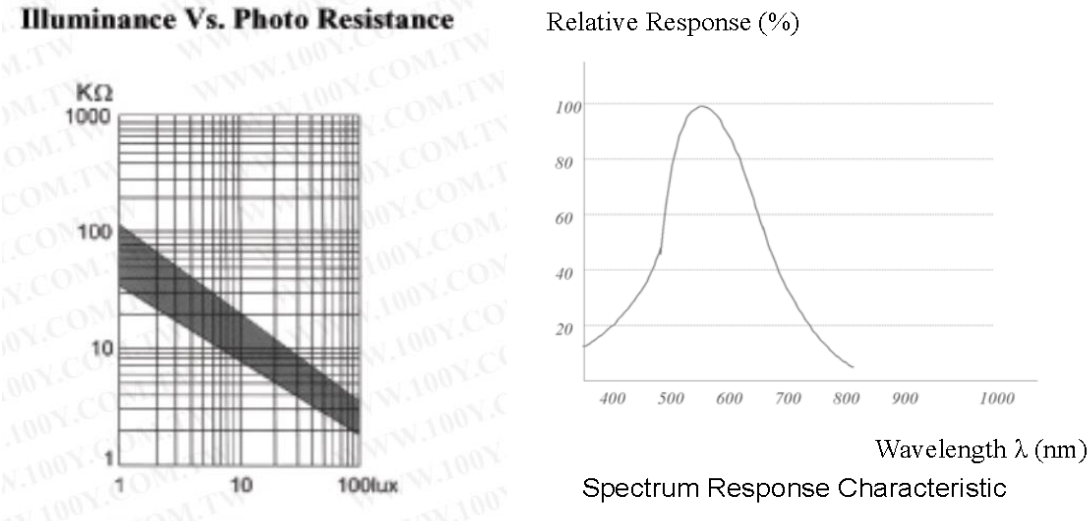
Για την ανίχνευση φωτός στον χώρο θα χρησιμοποιηθεί η φωτοαντίσταση (Light Dependent Resistor-LDR) GL5528, η χρήση της είναι να αντιληφθεί την αλλαγή στη φωτεινότητα που προκαλείται από το άνοιγμα ενός παντζουριού λόγω καιρικών φαινομένων ή άλλων αιτίων. Όταν η φωτοαντίσαση αντιληφθεί αυτήν την αύξηση της φωτεινότητας, θα ειδοποιεί τον χρήστη μέσω της εφαρμογής Blynk. Έτσι, η χρήση της φωτοαντίστασης είναι σημαντική για την αντίδραση σε απρόοπτα γεγονότα όπως η ξαφνική είσοδος φωτός σε ένα χώρο όπου δεν αναμένεται.



Σχήμα 5.25: Φωτοαντίσταση (LDR) GL5528

### 5.3.9.1 Λειτουργία φωτοαντίστασης

Η λειτουργία της φωτοαντίστασης βασίζεται στο φαινόμενο της φωτοαγωγιμότητας (Photoconductivity), όπου η αντίστασή της μεταβάλλεται με την ένταση του φωτός. Η φωτοαντίσταση κατασκευάζεται από ημιαγώγιμο υλικό, και η συνάρτηση της αντίστασή της είναι μη γραμμική σε σχέση με την ένταση του φωτός. Στο σκοτάδι, η αντίσταση της φωτοαντίστασης είναι πολύ μεγάλη της τάξης των ΜΩ, όταν όμως εκτεθεί στο φως τότε η αντίστασή της είναι πολύ μικρή της τάξης των μερικών Ω. Η φωτοαντίσταση GL5528, έχει ευαισθησία στο ορατό φάσμα του φωτός, με κορυφαία ευαισθησία στην περιοχή περίπου 540nm[20-22].



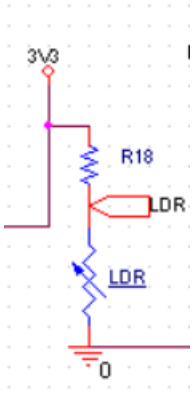

Σχήμα 5.26: (από αριστερά) Χαρακτηριστική καμπύλη φωτός-αντίστασης σε λογαριθμική κλίμακα, Χαρακτηριστική απόκριση φάσματος

### 5.3.9.2 Συνδεσμολογία με το Arduino

Για την σύνδεση της στο Arduino, θα χρησιμοποιήσουμε μια αντίσταση 10ΚΩ σε σειρά με την φωτοαντίσταση ώστε να δημιουργήσουμε έναν διαιρέτη τάσης. Η έξοδος του διαιρέτη τάσης θα είναι από την φωτοαντίσταση και θα συνδεθεί στον αναλογικό ακροδέκτη A1 του Arduino Due. Έπειτα από μετρήσεις που έγιναν, πήραμε τις τιμές αντίστασης της φωτοαντίστασης σε συνθήκες σκοταδιού, ελάχιστου φωτός, μέτριου φωτός, φωτεινού δωματίου και δυνατού φωτός:

- Σκοτάδι: 800KΩ
- Ελάχιστο φως: 35KΩ
- Μέτριο φως: 13KΩ
- Φωτεινού δωματίου: 6KΩ
- Δυνατό Φως: 800Ω

Η αναλογική τιμή που διαβάζει το Arduino είναι τιμή με εύρος από 0 έως 4095, εμείς πρέπει να την επεξεργαστούμε και να την μετατρέψουμε σε Ωμ, για αυτό χρειάστηκε να γράψουμε κάποιες γραμμές κώδικα. Αρχικά, για να διαβάσουμε την τιμή από τον αναλογικό ακροδέκτη πρέπει να δημιουργήσουμε μια μεταβλητή με την εντολή «int ldrValue = analogRead(ldrpin);» , έπειτα θα επεξεργαστούμε την τιμή αυτή ώστε να έχουμε την τιμή σε τάση δημιουργώντας μια νέα μεταβλητή με την εντολή «float voltage = ldrValue * (3.3 / 4095.0);». Τέλος, για να μετατρέψουμε την τιμή της τάσης σε Ωμ θα δημιουργήσουμε μια νέα μεταβλητή με την εντολή «float ldrResistance = (voltage * 10000) / (3.3 - voltage);».




Σχήμα 5.27: Διαιρέτης τάσης

### 5.3.10 Σειρήνα

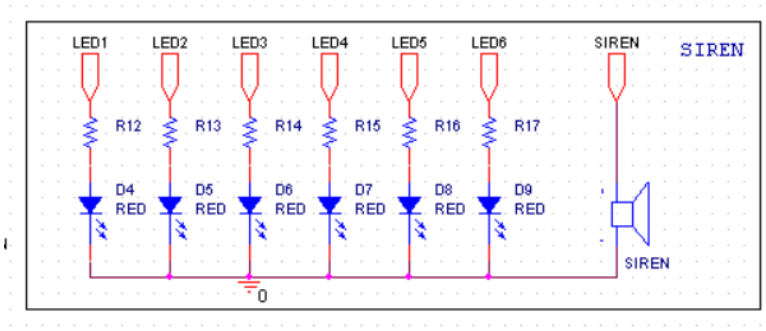
Για την ένδειξη ήχου και οπτικής ειδοποίησης στο σύστημα συναγερμού, θα χρησιμοποιήσουμε 1 παθητικό βομβητή (Passive buzzer) και 6 LEDs κόκκινου χρώματος. Ο βομβητής και το κάθε LED, θα είναι συνδεδεμένο ξεχωριστά σε έναν ψηφιακό ακροδέκτη του Arduino το καθένα. Τα LED θα έχουνε 2 καταστάσεις, η πρώτη είναι σε κατάσταση ηρεμίας που απλά θα αναβοσβήνουν διαδοχικά ανά 3, ενώ σε κατάσταση συναγερμού θα αναβοσβήνουν όλα μαζί και θα προστίθεται και ο ήχος του βομβητή.


Σχήμα 5.28: (από αριστερά) Κόκκινο LED, Παθητικός βομβητής (Buzzer)

### 5.3.10.1 Συνδεσμολογία με το Arduino

Για την σύνδεση του buzzer με το Arduino, θα χρησιμοποιηθεί ο ψηφιακός ακροδέκτης 51. Όταν θα είναι σε κατάσταση συναγερμού το σύστημα, τότε ο ψηφιακός ακροδέκτης του Arduino θα έρχεται σε κατάσταση ΥΨΗΛΗ (HIGH) και το buzzer θα ηχεί μέχρι να αφοπλιστεί το σύστημα.

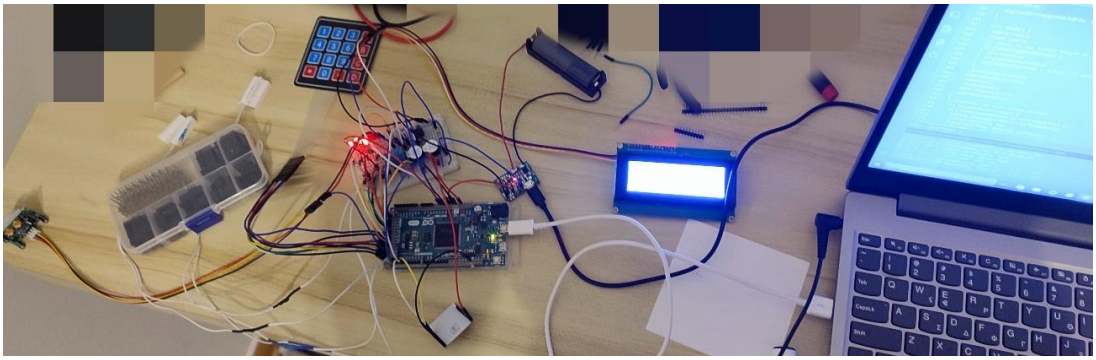
Για τα LED, θα χρησιμοποιηθούν οι ψηφιακοί ακροδέκτες του Arduino 36, 38, 40, 42, 44 και 46. Το κάθε LED θα έχει μια αντίσταση 1ΚΩ σε σειρά, η χρήση της είναι για να περιορίσει το ρεύμα που θα διέρχεται από το LED και να το προστατέψει. Ο υπολογισμός της αντίστασης έγινε με τον νόμο του Ωμ, αφού πρώτα είδαμε τα χαρακτηριστικά του LED. Τα συγκεκριμένα LED έχουν πτώση τάσης 1.8V και μέγιστο ρεύμα 20mA. Η τροφοδοσία από τους ψηφιακούς ακροδέκτες του Arduino Due που χρησιμοποιήσαμε είναι στα 3.3V και το μέγιστο ρεύμα τους είναι στα 15mA. Από τον δεύτερο κανόνα του Κίρχοφ (Kirchhoff) για τις τάσεις, το άθροισμα των τάσεων σε έναν βρόγχο ισούται με 0, οπότε αφού γνωρίζουμε την τάση τροφοδοσίας του ψηφιακού ακροδέκτη και την τάση λειτουργίας του LED, μπορούμε να κάνουμε την πράξη $V_{Dpin}=V_{LED}+V_R$ και να αντικαταστήσουμε τις τιμές των τάσεων, η τελική τιμή της τάσης που θα βρούμε θα είναι 1.5V. Τέλος, για τον υπολογισμό της αντίστασης, δεν θα λάβουμε υπόψιν το μέγιστο ρεύμα του ψηφιακού ακροδέκτη αλλά μια τιμή μικρότερη για να μην ζορίζουμε το Arduino. Η τιμή του ρεύματος που θα λάβουμε υπόψιν είναι στα 1.5mA, και χρησιμοποιώντας τον νόμο του Ωμ $R=V_R/I$ κάνουμε αντικατάσταση τις τιμές και βρίσουμε την τιμή της αντίστασης που είναι 1ΚΩ[23-24].




Σχήμα 5.29: Σχηματικό διάγραμμα σειρήνας

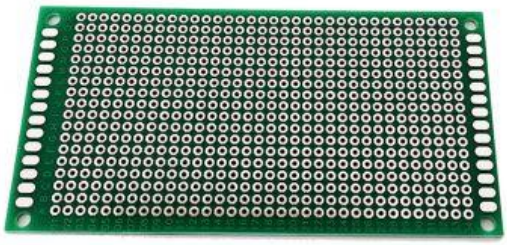
## 5.4 Κατασκευή πλακέτας

Εφόσον έγινε η μελέτη και ο σχεδιασμός του συστήματος συναγερμού, ήρθε η στιγμή που θα συνδέσουμε όλα τα εξαρτήματα αρχικά σε ένα ράστερ για να γίνει η δοκιμή του και έπειτα θα γίνει η κατασκευή της τελικής πλακέτας που θα συνοδεύει το Arduino Due. Όλα τα εξαρτήματα αγοράστηκαν από το ελληνικό κατάστημα GRobotronics, η λίστα προϊόντων βρίσκεται στο Παράρτημα Α.



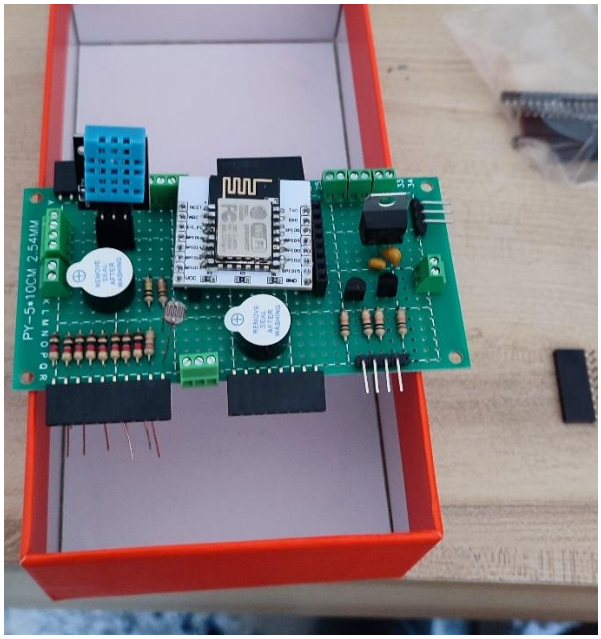
Σχήμα 5.30: Πρόχειρη κατασκευή σε ράστερ

Για την κατασκευή της πλακέτας, αγοράστηκε μια πλακέτα πρωτότυπου μίας όψης, τύπου Perfboard με διαστάσεις 100x50mm, η απόσταση από οπή σε οπή είναι 2.54mm καθώς οι ακίδες από τα περισσότερα εξαρτήματα έχουν αυτήν την απόσταση.



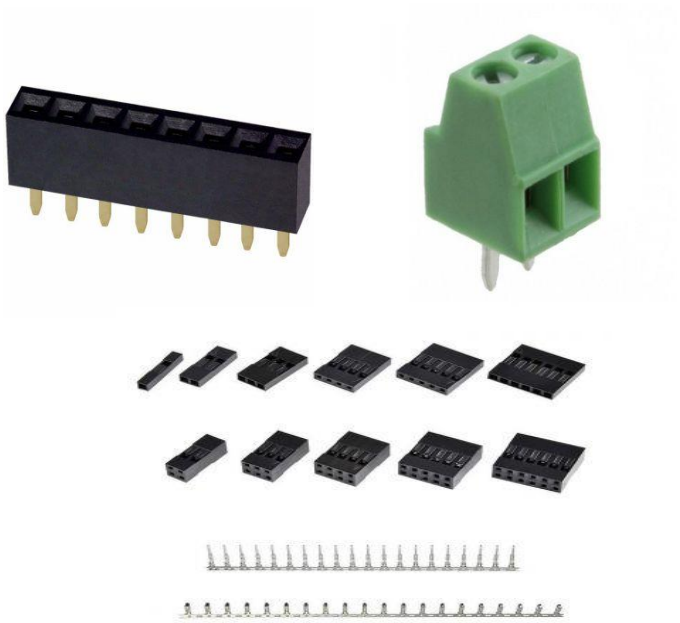
Σχήμα 5.31: Πρωτότυπη πλακέτα μονής όψης

Αρχικά, έγινε η τοποθέτηση των εξαρτημάτων στην πλευρά της πλακέτας που δεν είναι για να γίνουν οι κολλήσεις, αποφασίστηκε η θέση του κάθε εξαρτήματος και έπειτα έγινε η κόλληση τους.



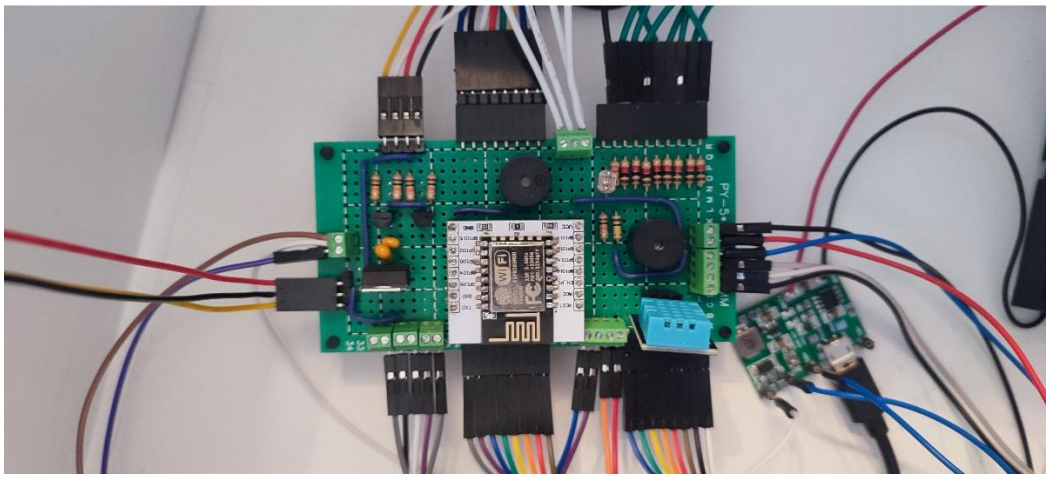
Σχήμα 5.32: Τοποθέτηση εξαρτημάτων στην τελική πλακέτα

Για την εύκολη αντικατάσταση κάποιων εξαρτημάτων, αποφασίστηκε να τοποθετηθούν πρώτα ακροδέκτες που είναι σχεδιασμένοι για να συνδεθούν με αρσενικούς ακροδέκτες και έπειτα να τοποθετηθούν εκεί τα εξαρτήματα. Για την σύνδεση των καλωδίων αποφασίστηκε να τοποθετηθούν κλέμες αλλά και ακροδέκτες τύπου DuPont.



Σχήμα 5.33: (από πάνω αριστερά) Ακροδέκτης θυληκός, Κλέμα, Ακροδέκτες DuPont

Η τελική πλακέτα που τοποθετήθηκε στο σύστημα συναγερμού είναι η παρακάτω.

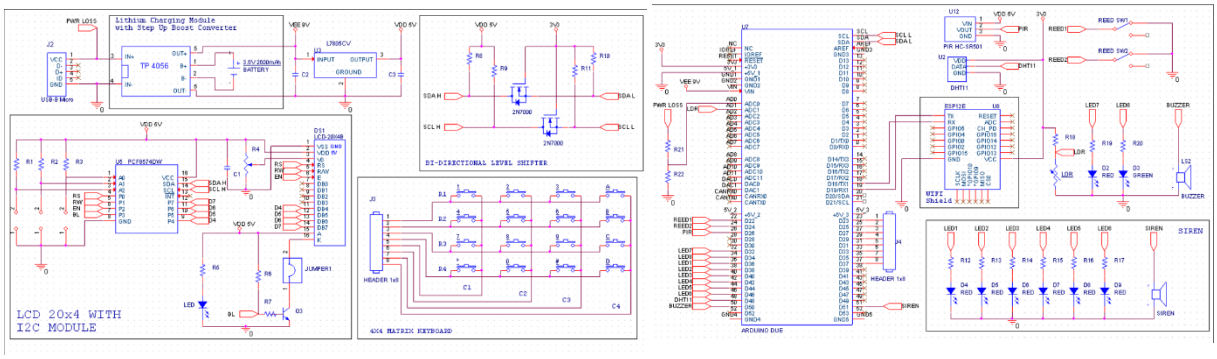


Σχήμα 5.34: Τελική πλακέτα του συστήματος

Σύμφωνα με το Σχήμα 5.34, στην πάνω αριστερά πλευρά συνδέθηκε ο ακροδέκτης DuPont 4-pin που είναι έξοδος της πλακέτας για την οθόνη, δεξιά συνδέθηκε το πληκτρολόγιο με τον ακροδέκτη DuPont 8-pin που είναι είσοδος της πλακέτας. Στην 3-pin κλέμα που βρίσκεται στο επάνω μέρος συνδέθηκαν οι 2 μαγνητικές επαφές, από αριστερά είναι η πρώτη Μ.Ε, αμέσως μετά είναι η δεύτερη Μ.Ε και τέλος είναι η κοινή γείωση από τις μαγνητικές επαφές. Στον επάνω δεξιά ακροδέκτη DuPont 8-pin είναι η έξοδος της πλακέτας και συνδέθηκαν τα LEDs της σειρήνας και της κατάστασης του συναγερμού (οπλισμένος-αφοπλισμένος). Στο κέντρο δεξιά της πλακέτας οι 3 2-pin κλέμες συνδέθηκαν με την σειρά, 3.3V (από Arduino), GND (από Arduino), ακροδέκτης εισόδου από το σύστημα διαχείρισης μπαταρίας για την απώλεια ρεύματος, είσοδος για buzzer 1, είσοδος για buzzer 2, έξοδος DHT11. Κάτω δεξιά στην πλακέτα στον ακροδέκτη DuPont 8-pin είναι η είσοδος της πλακέτας για τα LED. Αμέσως μετά η 4-pin κλέμα είναι με την σειρά από δεξιά, η έξοδος της πλακέτας για την απώλεια ρεύματος, έξοδος της πλακέτας για την φωτοαντίσταση, έξοδος της πλακέτας για Μ.Ε 1, έξοδος της πλακέτας για

Μ.Ε 2. Στο κάτω κεντρικό μέρος της πλακέτας είναι ο ακροδέκτης DuPont 8-pin που είναι έξοδος της πλακέτας από το πληκτρολόγιο. Κάτω αριστερά είναι οι 3 2-pin κλέμες που είναι με την σειρά RX1 είσοδος, TX1 είσοδος, SDA είσοδος, SCL είσοδος, έξοδος PIR. Κέντρο αριστερά της πλακέτας είναι ο ακροδέκτης DuPont 3-pin και είναι η σύνδεση του PIR. Τέλος, στο κέντρο αριστερά είναι η κλέμα για την είσοδο τροφοδοσίας 9V της πλακέτας.

Τα καλώδια που χρησιμοποιήθηκαν για τα LED και το πληκτρολόγιο είναι χειροποίητα με ακροδέκτες DuPont και πολύκλωνο καλώδιο 22AWG. Σε όσα έτοιμα καλώδια δεν υπήρχε τελικός ακροδέκτης DuPont, τοποθετήθηκε για την ευκολία σύνδεσης.

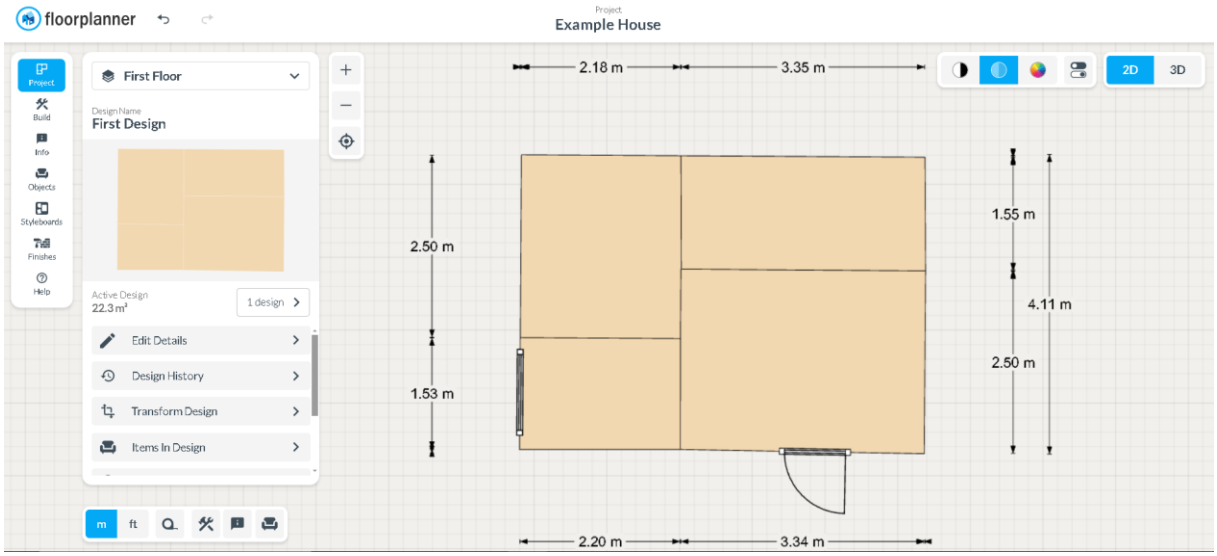



Σχήμα 5.35: Τελικό σχηματικό στο πρόγραμμα PSpice

## 5.5 Κατασκευή μακέτας

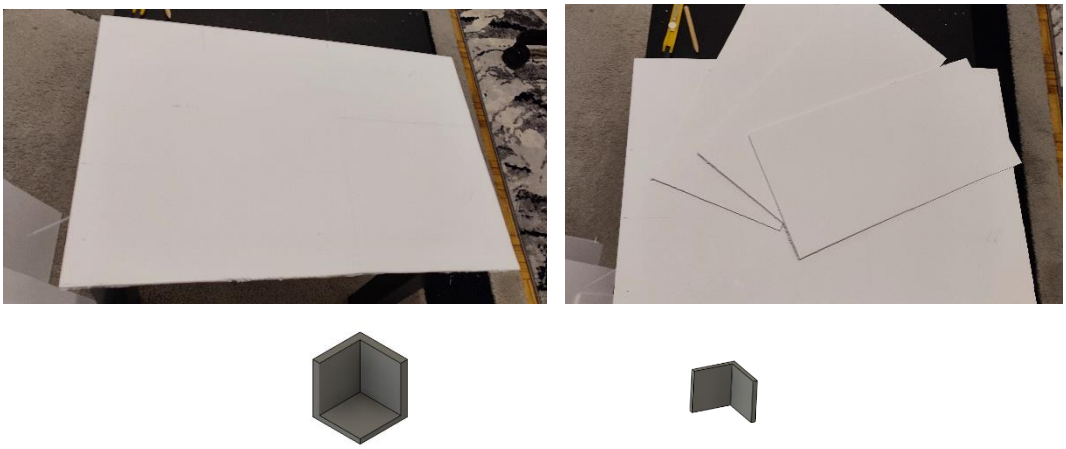
Εφόσον έγινε και η πλακέτα που θα συνοδεύει το Arduino, θα πρέπει να κατασκευάσουμε μια μακέτα ενός πραγματικού σπιτιού για την επίδειξη του συστήματος συναγερμού. Στην μακέτα, θα τοποθετηθούν οι αισθητήρες σε διάφορα σημεία, ενώ θα δημιουργηθεί και ένα δωμάτιο ελέγχου που εκεί θα καταλήγουν όλες οι συνδέσεις.

Για την κατασκευή της μακέτας, αγοράστηκε από το κατάστημα Πλαίσιο χαρτόνι μακέτας σε διαστάσεις 100x70cm και πάχος 5mm. Οι διαστάσεις της μακέτας για την πτυχιακή εργασία, αποφασίστηκαν έπειτα από τον σχεδιασμό της κάτοψης σπιτιού που δημιουργήθηκε στην ιστοσελίδα Floorplanner με αναλογία 1:10, δηλαδή κάθε 1 μέτρο αντιστοιχεί σε 10 εκατοστά.



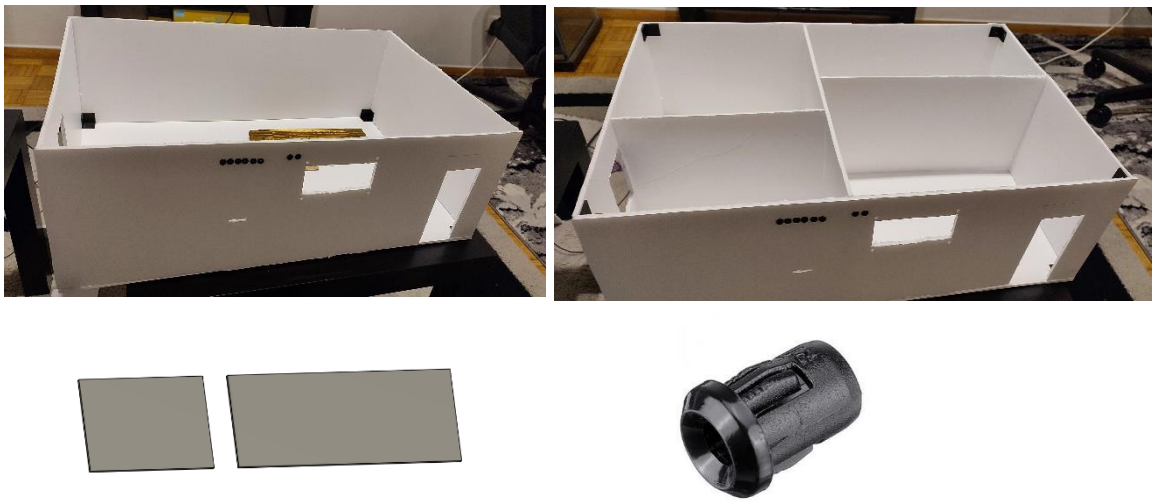

Σχήμα 5.36: Κάτοψη μακέτας με αναλογία 1:10

Στην συνέχεια, κόπηκε το χαρτόνι στις παραπάνω διαστάσεις και κολλήθηκαν με κόλλα στιγμής οι γωνίες που σχεδιάστηκαν στο πρόγραμμα Fusion και μετά εκτυπώθηκαν στο 3D-Printer.



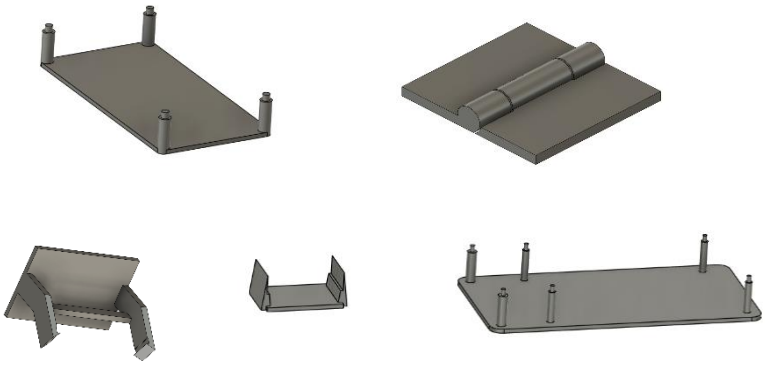
Σχήμα 5.37: (από πάνω αριστερά) Βάση μακέτας για την εργασία, Διαχωριστιά δωματίων, Κάτω γωνίες, Επάνω γωνίες

Έπειτα, δημιουργήθηκαν 8 τρύπες για την προσθήκη των θηκών για LED, αγορασμένες σε κατάλληλο μέγεθος 3mm για τα LED, ενώ δημιουργήθηκαν και τα κατάλληλα κοψίματα/κενά για την προσθήκη της πόρτας, του παραθύρου, της οθόνης και για να περάσουν τα καλώδια.



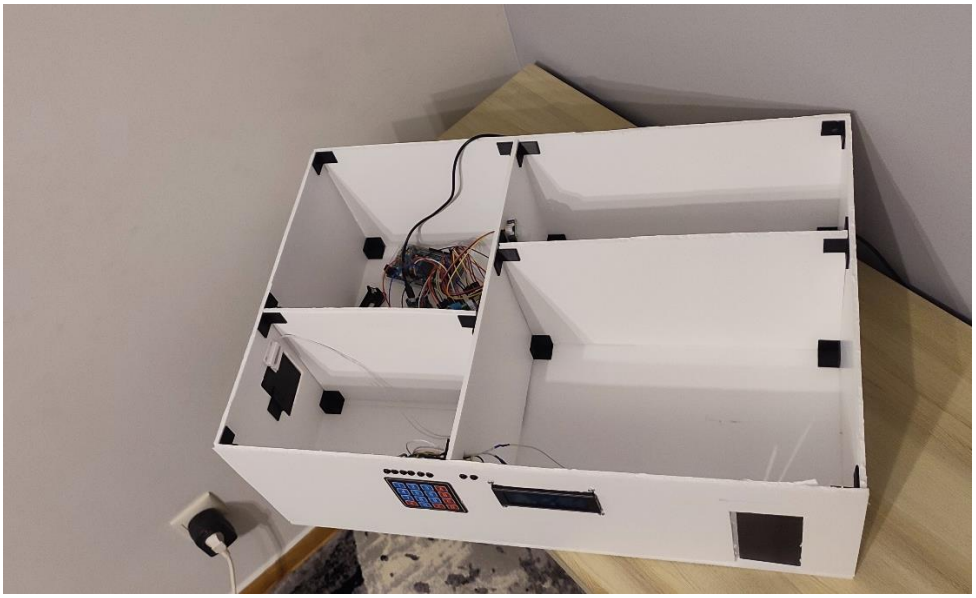
Σχήμα 5.38: (από πάνω αριστερά) Μακέτα με κάτω γωνίες, Μακέτα με πάνω γωνίες και διαχωριστικά, Πόρτα και παράθυρο, Θήκη LED

Ακόμα, μετρήθηκαν οι διαστάσεις και σχεδιάστηκαν στο πρόγραμμα Fusion βάσεις για την πλακέτα, το Arduino, το PIR και το σύστημα διαχείρισης μπαταρίας ώστε να είναι σταθερά. Όπως επίσης και μεντεσέδες για την πόρτα και το παράθυρο. Έπειτα, εκτυπώθηκαν στο 3D-printer.



Σχήμα 5.39: (από πάνω αριστερά) Βάση για την πλακέτα, Μεντεσές, Βάση PIR, Βάση για σύστημα διαχείρισης μπαταρίας, Βάση Arduino

Τέλος, αφού κολλήθηκαν όλα τα κομμάτια της μακέτας, προστέθηκαν στην τελική τους θέση οι αισθητήρες, η οθόνη, το πληκτρολόγιο, τα LED και έγινε η καλωδίωση τους.



Σχήμα 5.40: Τελική μακέτα εργασίας

## 5.6 Επίλογος

Στο κεφάλαιο αυτό περιγράφεται αναλυτικά η ιδέα της πτυχιακής εργασίας, η σχεδίαση του συστήματος συναγερμού για οικιακή χρήση με μπλόκ διάγραμμα, και έπειτα στην έρευνα των υλικών και στην υλοποίησή της. Τέλος, αναφέρεται η κατασκευή της πλακέτας και της μακέτας για την παρουσίαση του τελικού αποτελέσματος.

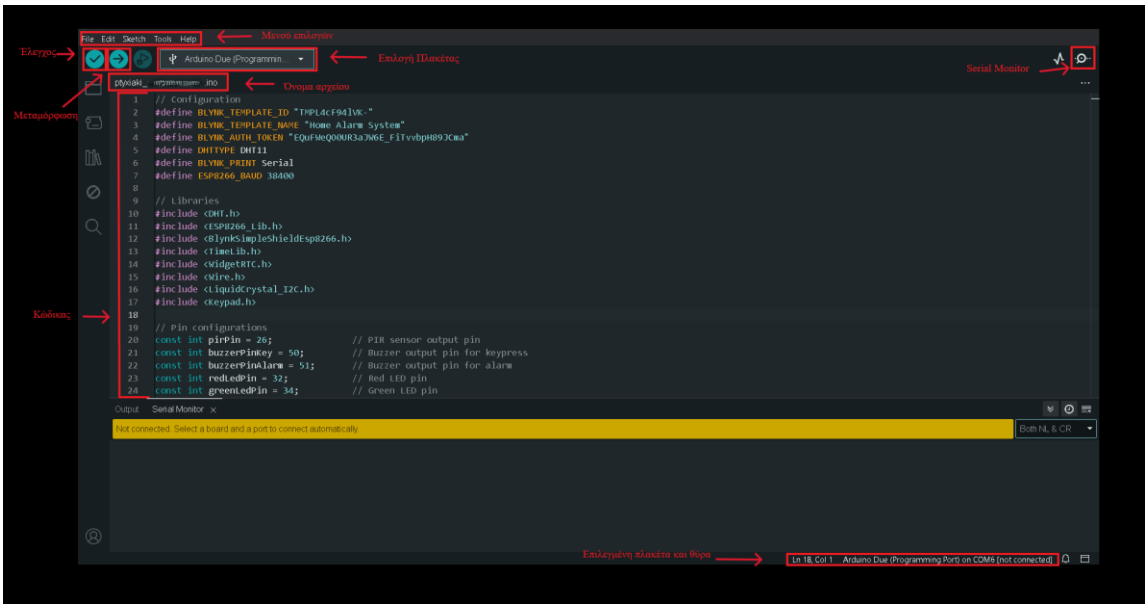
# Κεφάλαιο 6ο: Κώδικας

## 6.1 Εισαγωγή

Το κατασκευαστικό μέρος του συστήματος συναγερμού είναι έτοιμο και το επόμενο βήμα είναι να γραφτεί ο κώδικας. Σε αυτό το κεφάλαιο, θα γίνει ο προγραμματισμός του μικροελεγκτή από το Arduino Due στο περιβάλλον του Arduino IDE, και στην συνέχεια θα τεθεί σε λειτουργία το σύστημα συναγερμού. Τέλος, για να δοκιμαστεί η λειτουργία του θα γίνουν ορισμένα σενάρια ώστε να δούμε πως αντιδρά το σύστημα σε αυτά.

## 6.2 Κώδικας συστήματος συναγερμού

Ο κώδικας του συστήματος συναγερμού θα γραφτεί στο περιβάλλον του Arduino IDE, το οποίο διατίθεται δωρεάν για λήψη. Στο Arduino IDE, πρέπει να δηλώσουμε τον τύπο του Arduino που θα χρησιμοποιήσουμε και την θύρα επικοινωνίας στην οποία είναι συνδεμένο το Arduino.




Σχήμα 6.1: Περιβάλλον Arduino IDE

Όσον αφορά τον κώδικα, σαν πρώτο βήμα αρχικοποιήθηκαν οι απαραίτητες πληροφορίες που θα χρειαστεί η βιβλιοθήκη Blynk για την επικοινωνία με την εφαρμογή, το baud rate για την σειριακή επικοινωνία του Arduino Due με το ESP12-E, καθώς και τον τύπο του αισθητήρα υγρασίας/θερμοκρασίας. Επίσης, δηλώθηκαν όλες οι βιβλιοθήκες που θα χρησιμοποιηθούν στον κώδικα, δηλώθηκαν όλοι οι ακροδέκτες, οι μεταβλητές, οι χρονικές μεταβλητές και τα σημεία ελέγχου (flags), ενώ δημιουργήθηκαν τα απαραίτητα αντικείμενα (objects) για τις κλάσεις των βιβλιοθηκών. Επόμενο βήμα, ήταν να ελεγχθεί αν είναι διαθέσιμο το δηλωμένο ασύρματο δίκτυο για να συνδεθεί το ESP12-E, να εμφανιστούν τα κατάλληλα μηνύματα στην οθόνη LCD και να ορίσουμε την λειτουργία των ακροδεκτών, αν είναι είσοδος ή έξοδος. Έπειτα, γράφτηκε το κύριο πρόγραμμα που ελέγχει την σύνδεση με τον διακομιστή Blynk, την κατάσταση των αισθητήρων, αν πατήθηκε κάποιο πλήκτρο και αν υπάρχει τροφοδοσία από την κύρια πηγή ενέργειας. Τέλος, για να μην είναι μεγάλο το κύριο πρόγραμμα, αποφασίστηκε να δημιουργηθούν συναρτήσεις για κάθε λειτουργία που συμβαίνει στο σύστημα συναγερμού.

Παράδειγμα, δημιουργήθηκε η συνάρτηση που θα οπλίζει το σύστημα. Στην συνάρτηση αυτή αρχικά ελέγχεται αν το σύστημα είναι ήδη οπλισμένο, αν ο κωδικός που εισήγαγε ο χρήστης είναι ο σωστός, και μετά ξεκινάει ο χρόνος όπλισης και τον εμφανίζει στην οθόνη LCD, αφού γίνει αυτή η διαδικασία τότε αλλάζει την μεταβλητή ελέγχου (flag) που έχει οριστεί για την κατάσταση του συστήματος.

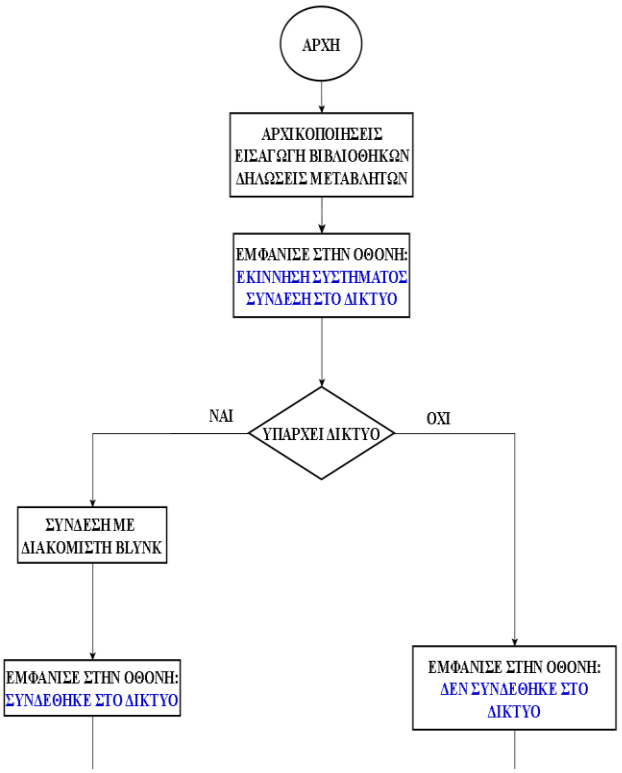
## 6.3 Μεταμόρφωση κώδικα στο Arduino Due

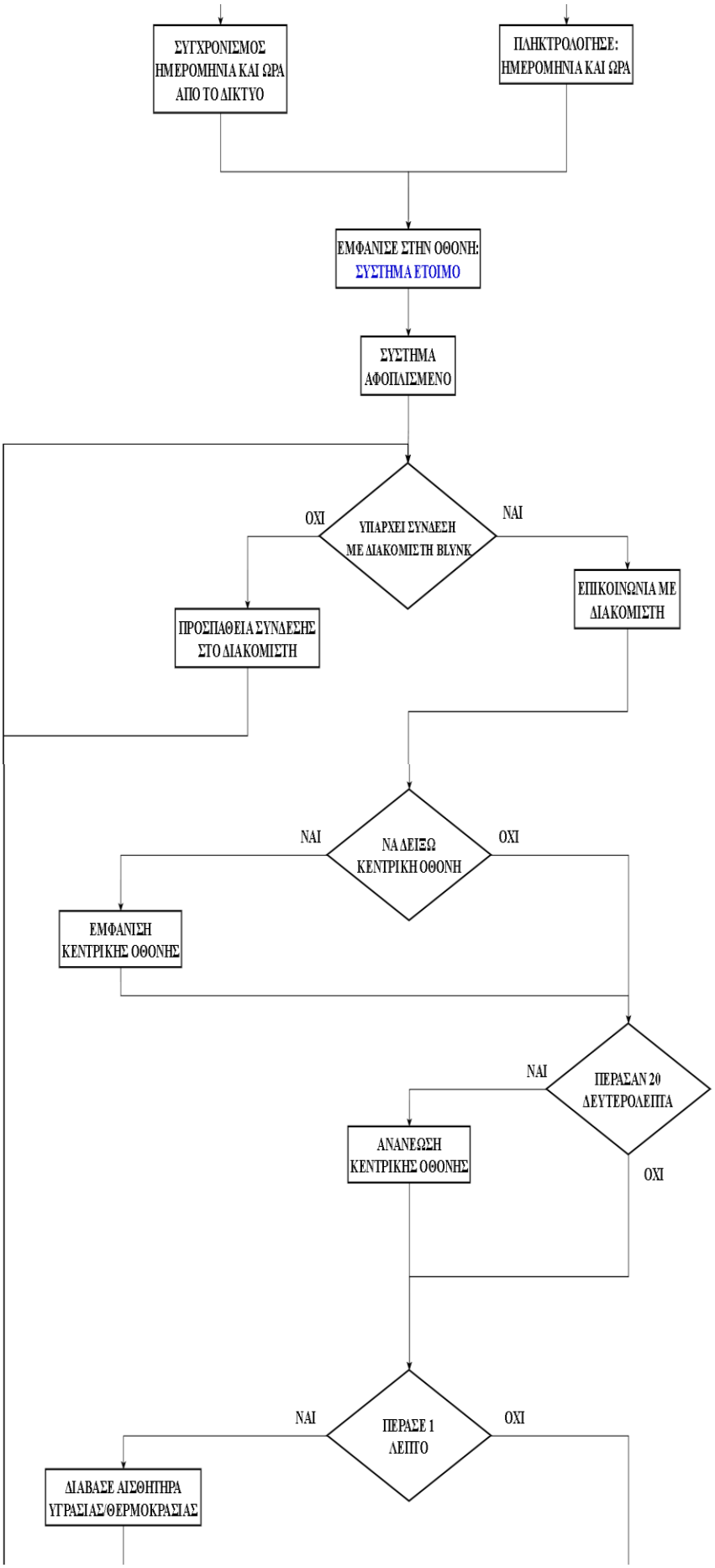
Κατά την διάρκεια της δημιουργίας του κώδικα, το Arduino ήταν συνδεδεμένο με το λάπτοπ και γίνονταν συνεχώς διορθώσεις και βελτιώσεις μέχρι την τελική μεταμόρφωση (upload) του κώδικα στον μικροελεγκτή.

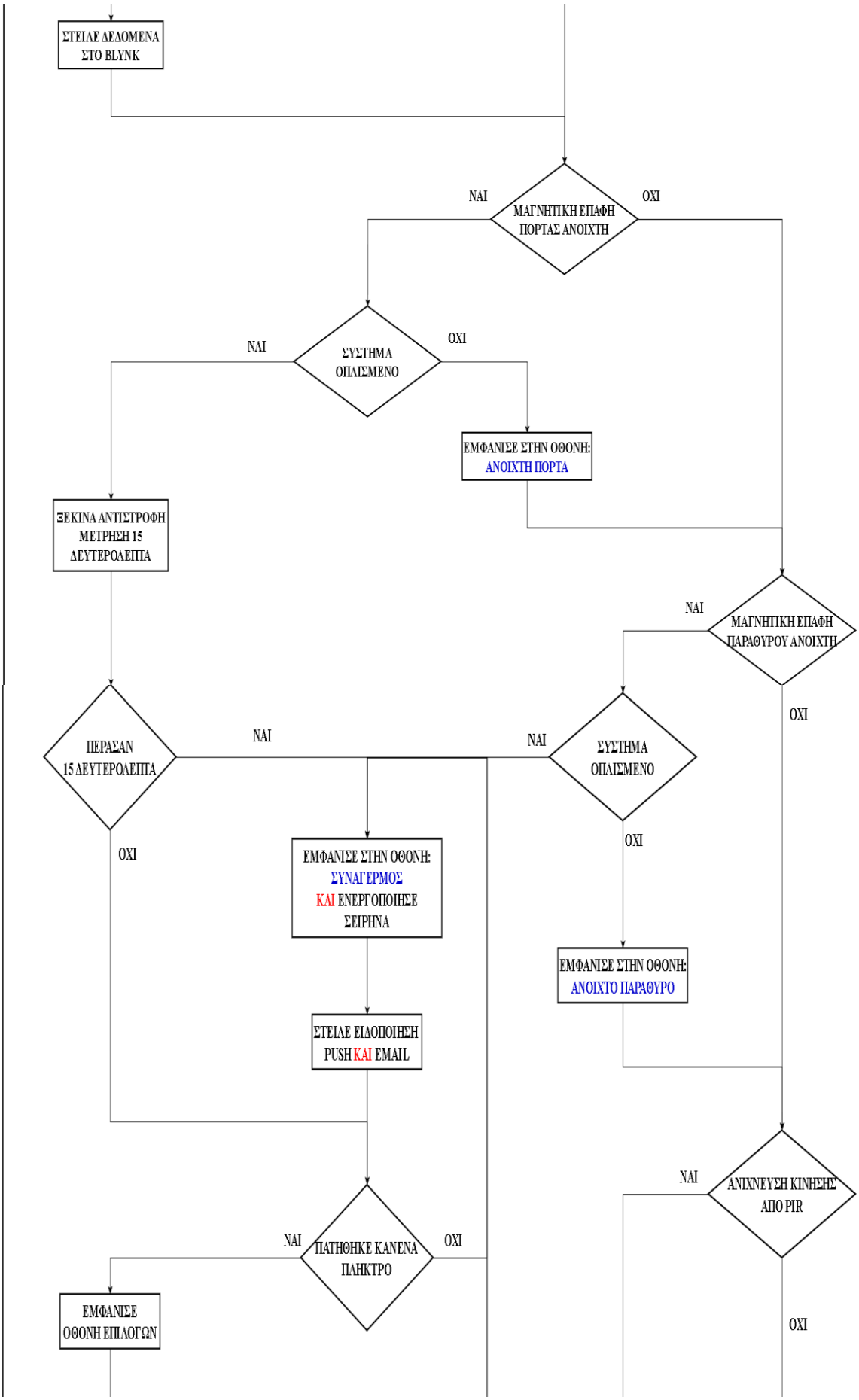
Σχήμα 6.2: Τελική μεταμόρφωση του κωδικα στον Arduino

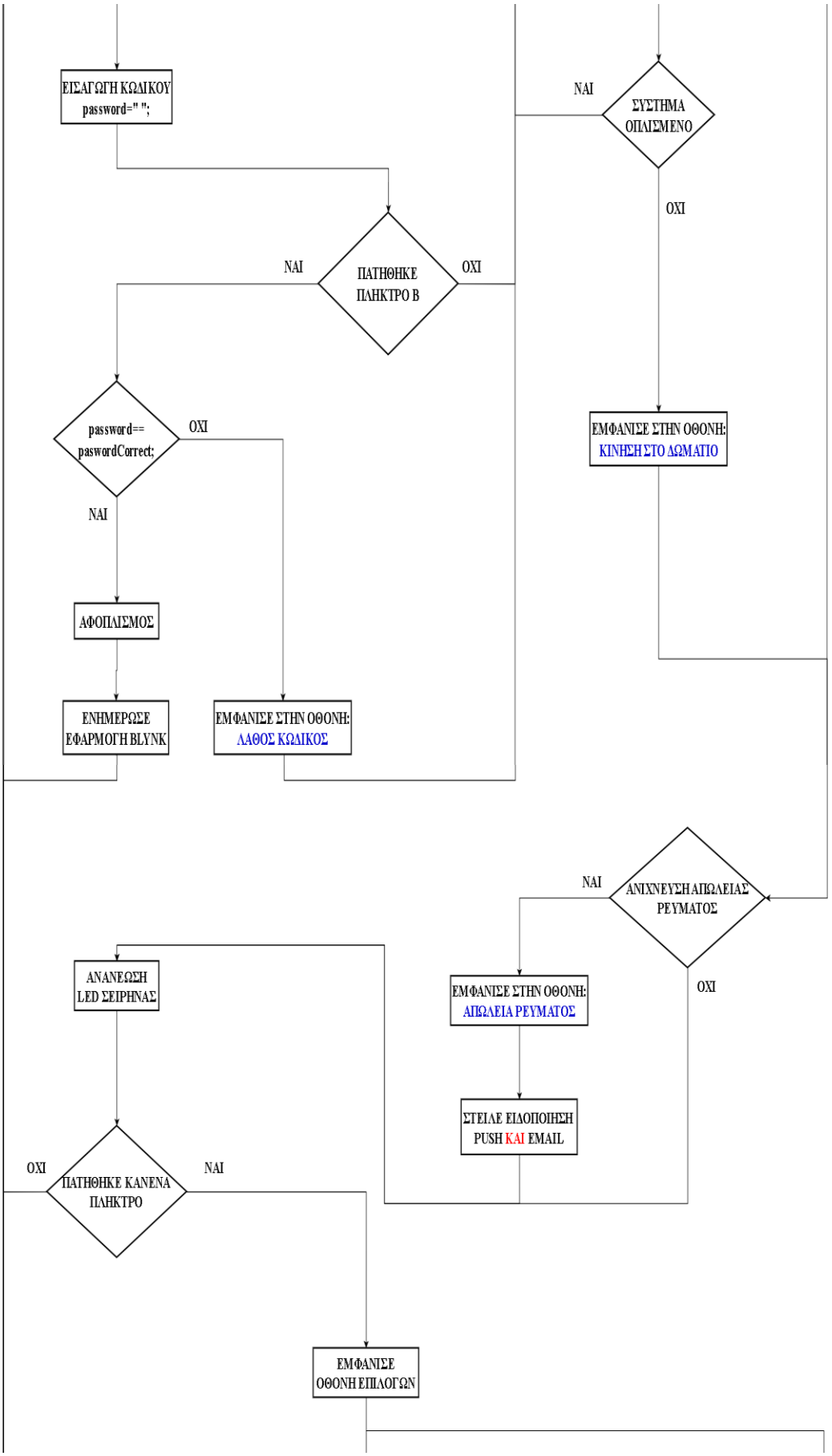
## 6.4 Διάγραμμα ροής

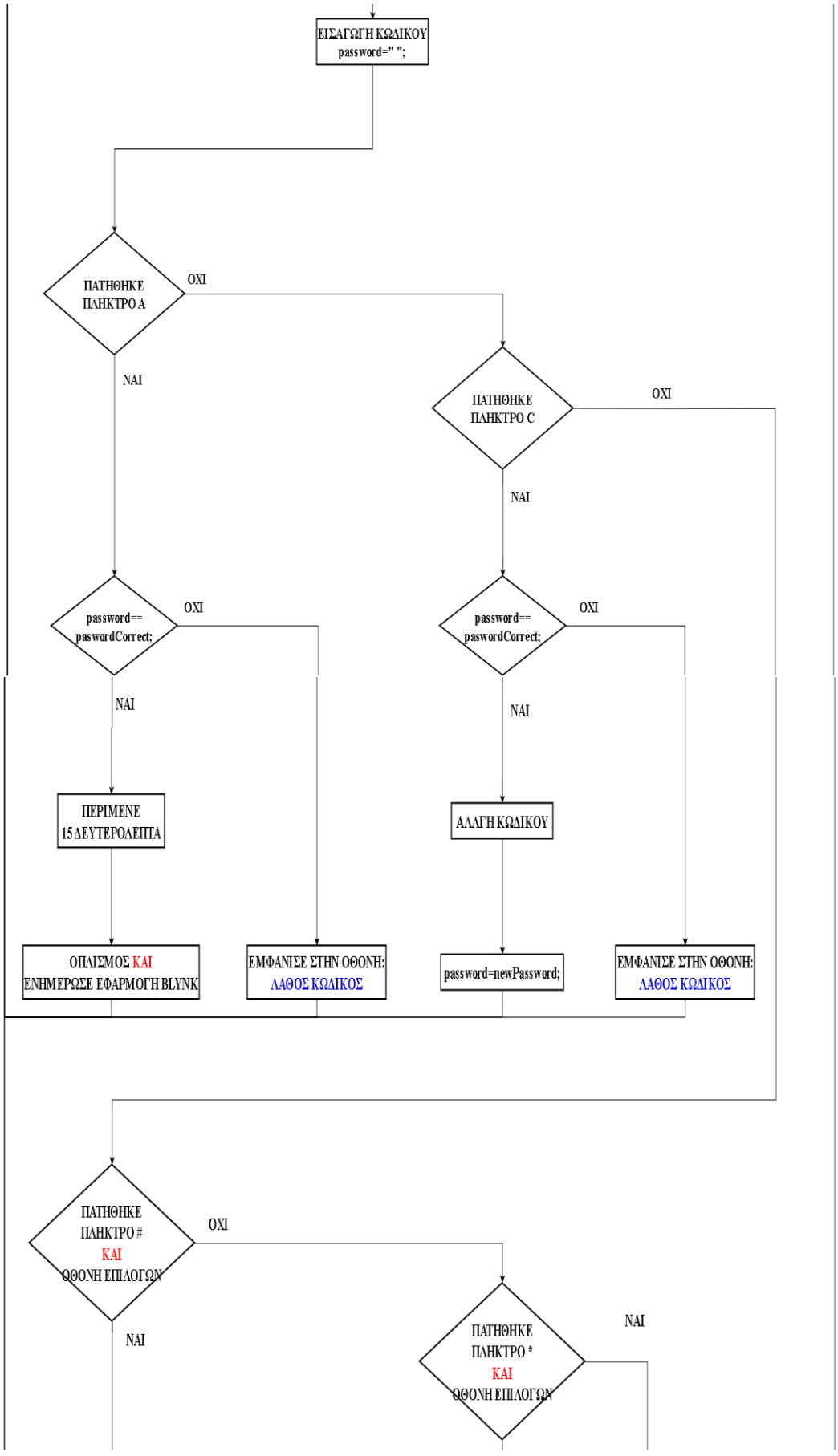
Παρακάτω, είναι το διάγραμμα ροής του κώδικα στην γενική μορφή του.




ΣΥΓΧΡΟΝΙΣΜΟΣ ΗΜΕΡΟΜΗΝΙΑ ΚΑΙ ΩΡΑ ΑΠΟ ΤΟ ΔΙΚΤΥΟ
ΠΛΗΚΤΡΟΛΟΓΗΣΕ: ΗΜΕΡΟΜΗΝΙΑ ΚΑΙ ΩΡΑ
ΕΜΦΑΝΙΣΕ ΣΤΗΝ ΟΘΟΝΗ: ΣΥΣΤΗΜΑ ΕΤΟΙΜΟ
ΣΥΣΤΗΜΑ ΑΦΟΠΛΙΣΜΕΝΟ
ΥΠΑΡΧΕΙ ΣΥΝΔΕΣΗ ΜΕ ΔΙΑΚΟΜΙΣΤΗ BLYNK
ΟΧΙ
ΝΑΙ
ΠΡΟΣΠΑΘΕΙΑ ΣΥΝΔΕΣΗΣ ΣΤΟ ΔΙΑΚΟΜΙΣΤΗ
ΕΠΙΚΟΙΝΩΝΙΑ ΜΕ ΔΙΑΚΟΜΙΣΤΗ
ΝΑ ΔΕΙΞΩ ΚΕΝΤΡΙΚΗ ΟΘΟΝΗ
ΝΑΙ
ΟΧΙ
ΕΜΦΑΝΙΣΗ ΚΕΝΤΡΙΚΗΣ ΟΘΟΝΗΣ
ΠΕΡΑΣΑΝ 20 ΔΕΥΤΕΡΟΛΕΠΤΑ
ΝΑΙ
ΟΧΙ
ΑΝΑΝΕΩΣΗ ΚΕΝΤΡΙΚΗΣ ΟΘΟΝΗΣ
ΠΕΡΑΣΕ 1 ΛΕΠΤΟ
ΝΑΙ
ΟΧΙ
ΔΙΑΒΑΣΕ ΑΙΣΘΗΤΗΡΑ ΥΓΡΑΣΙΑΣ/ΘΕΡΜΟΚΡΑΣΙΑΣ


ΣΤΕΙΛΕ ΔΕΔΟΜΕΝΑ ΣΤΟ BLYNK
ΜΑΓΝΗΤΙΚΗ ΕΠΑΦΗ ΠΟΡΤΑΣ ΑΝΟΙΧΤΗ
ΝΑΙ
ΟΧΙ
ΣΥΣΤΗΜΑ ΟΠΛΙΣΜΕΝΟ
ΝΑΙ
ΟΧΙ
ΕΜΦΑΝΙΣΕ ΣΤΗΝ ΟΘΟΝΗ: ΑΝΟΙΧΤΗ ΠΟΡΤΑ
ΞΕΚΙΝΑ ΑΝΤΙΣΤΡΟΦΗ ΜΕΤΡΗΣΗ 15 ΔΕΥΤΕΡΟΛΕΠΤΑ
ΜΑΓΝΗΤΙΚΗ ΕΠΑΦΗ ΠΑΡΑΘΥΡΟΥ ΑΝΟΙΧΤΗ
ΝΑΙ
ΟΧΙ
ΠΕΡΑΣΑΝ 15 ΔΕΥΤΕΡΟΛΕΠΤΑ
ΝΑΙ
ΟΧΙ
ΣΥΣΤΗΜΑ ΟΠΛΙΣΜΕΝΟ
ΝΑΙ
ΟΧΙ
ΕΜΦΑΝΙΣΕ ΣΤΗΝ ΟΘΟΝΗ: ΣΥΝΑΓΕΡΜΟΣ ΚΑΙ ΕΝΕΡΓΟΠΟΙΗΣΕ ΣΕΙΡΗΝΑ
ΕΜΦΑΝΙΣΕ ΣΤΗΝ ΟΘΟΝΗ: ΑΝΟΙΧΤΟ ΠΑΡΑΘΥΡΟ
ΣΤΕΙΛΕ ΕΙΔΟΠΟΙΗΣΗ PUSH ΚΑΙ EMAIL
ΑΝΙΧΝΕΥΣΗ ΚΙΝΗΣΗΣ ΑΠΟ PIR
ΝΑΙ
ΟΧΙ
ΠΑΤΗΘΗΚΕ ΚΑΝΕΝΑ ΠΛΗΚΤΡΟ
ΝΑΙ
ΟΧΙ
ΕΜΦΑΝΙΣΕ ΟΘΟΝΗ ΕΠΙΛΟΓΩΝ


ΕΙΣΑΓΩΓΗ ΚΩΔΙΚΟΥ password=" ";
ΠΑΤΗΘΗΚΕ ΠΛΗΚΤΡΟ Β
ΝΑΙ
ΟΧΙ
password== paswordCorrect;
ΝΑΙ
ΟΧΙ
ΑΦΟΠΛΙΣΜΟΣ
ΕΝΗΜΕΡΩΣΕ ΕΦΑΡΜΟΓΗ BLYNK
ΕΜΦΑΝΙΣΕ ΣΤΗΝ ΟΘΟΝΗ: ΛΑΘΟΣ ΚΩΔΙΚΟΣ
ΣΥΣΤΗΜΑ ΟΠΛΙΣΜΕΝΟ
ΝΑΙ
ΟΧΙ
ΕΜΦΑΝΙΣΕ ΣΤΗΝ ΟΘΟΝΗ: ΚΙΝΗΣΗ ΣΤΟ ΔΩΜΑΤΙΟ
ΑΝΙΧΝΕΥΣΗ ΑΠΩΛΕΙΑΣ ΡΕΥΜΑΤΟΣ
ΝΑΙ
ΟΧΙ
ΕΜΦΑΝΙΣΕ ΣΤΗΝ ΟΘΟΝΗ: ΑΠΩΛΕΙΑ ΡΕΥΜΑΤΟΣ
ΣΤΕΙΛΕ ΕΙΔΟΠΟΙΗΣΗ PUSH ΚΑΙ EMAIL
ΑΝΑΝΕΩΣΗ LED ΣΕΙΡΗΝΑΣ
ΠΑΤΗΘΗΚΕ ΚΑΝΕΝΑ ΠΛΗΚΤΡΟ
ΟΧΙ
ΝΑΙ
ΕΜΦΑΝΙΣΕ ΟΘΟΝΗ ΕΠΙΛΟΓΩΝ


ΕΙΣΑΓΩΓΗ ΚΩΔΙΚΟΥ
password=" ";
ΠΑΤΗΘΗΚΕ ΠΛΗΚΤΡΟ A
ΟΧΙ
ΝΑΙ
ΠΑΤΗΘΗΚΕ ΠΛΗΚΤΡΟ C
ΟΧΙ
ΝΑΙ
password== paswordCorrect;
ΟΧΙ
ΝΑΙ
password== paswordCorrect;
ΟΧΙ
ΝΑΙ
ΠΕΡΙΜΕΝΕ 15 ΔΕΥΤΕΡΟΛΕΠΤΑ
ΑΛΛΑΓΗ ΚΩΔΙΚΟΥ
ΟΠΛΙΣΜΟΣ ΚΑΙ ΕΝΗΜΕΡΩΣΕ ΕΦΑΡΜΟΓΗ BLYNK
ΕΜΦΑΝΙΣΕ ΣΤΗΝ ΟΘΟΝΗ: ΛΑΘΟΣ ΚΩΔΙΚΟΣ
password=newPassword;
ΕΜΦΑΝΙΣΕ ΣΤΗΝ ΟΘΟΝΗ: ΛΑΘΟΣ ΚΩΔΙΚΟΣ
ΠΑΤΗΘΗΚΕ ΠΛΗΚΤΡΟ # ΚΑΙ ΟΘΟΝΗ ΕΠΙΛΟΓΩΝ
ΟΧΙ
ΝΑΙ
ΠΑΤΗΘΗΚΕ ΠΛΗΚΤΡΟ * ΚΑΙ ΟΘΟΝΗ ΕΠΙΛΟΓΩΝ
ΝΑΙ


Σχήμα 6.3:Γενικό διάγραμμα ροής κώδικα

## 6.5 Λειτουργία συστήματος συναγερμού

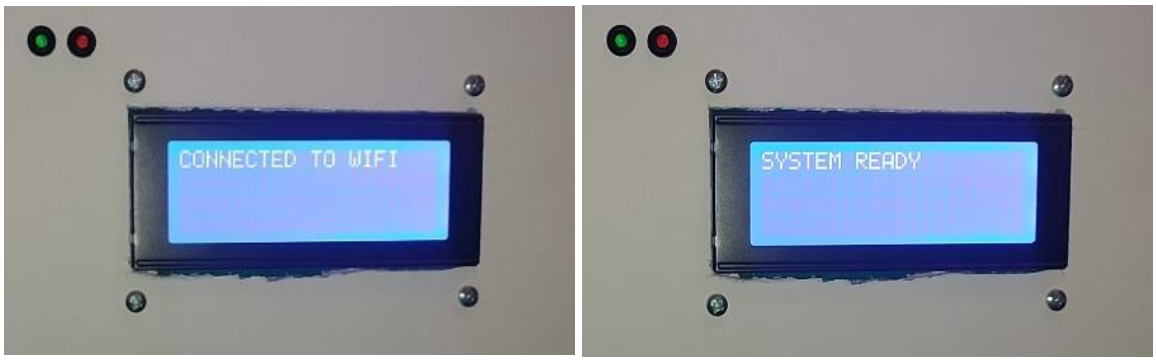
Κατά την εκκίνηση του συστήματος συναγερμού, εμφανίζονται στην οθόνη πληροφορίες από τα βήματα που ακολουθεί το σύστημα μέχρι να ολοκληρωθεί η αρχικοποίηση του.


Σχήμα 6.4: (από αριστερά) Οθόνη εκκίνησης, Αρχικοποίηση οθόνης

Έπειτα, το σύστημα προσπαθεί να συνδεθεί στο δίκτυο και στον διακομιστή Blynk, ενώ στην οθόνη ο χρήστης παρακολουθεί την διαδικασία.


Σχήμα 6.5: Προσπάθεια σύνδεσης στο δίκτυο

Στην περίπτωση που υπάρχει το διαθέσιμο δίκτυο, τότε το σύστημα συνδέεται σε αυτό και ενημερώνει τον χρήστη ότι το σύστημα είναι έτοιμο.




Σχήμα 6.6: (από αριστερά) Σύνδεση στο δίκτυο, Σύστημα έτοιμο

Αφού ολοκληρωθεί η διαδικασία αρχικοποίησης του συστήματος, εμφανίζεται η κεντρική οθόνη του συστήματος με την ώρα και την ημερομηνία από τον διακομιστή, ενώ το κόκκινο LED υποδηλώνει ότι το σύστημα είναι αφοπλισμένο.


Σχήμα 6.7: Κεντρική οθόνη συστήματος

Στην περίπτωση όμως που δεν υπάρχει σύνδεση στο δίκτυο, τότε στην οθόνη εμφανίζεται το μήνυμα που ενημερώνει τον χρήστη για αυτό και ζητάει για την εισαγωγή ώρας και ημερομηνίας χειροκίνητα.


Σχήμα 6.8: (από πάνω αριστερά) Μη σύνδεση στο δίκτυο, Οθόνη εισαγωγής ώρας/ημερομηνίας, Εισαγωγή ώρας, Εισαγωγή ημερομηνίας

Μετά την ολοκλήρωση της χειροκίνητης εισαγωγής ώρας και ημερομηνίας, το σύστημα είναι έτοιμο για χρήση και εμφανίζεται η κεντρική οθόνη του συστήματος. Το κόκκινο LED υποδηλώνει ότι το σύστημα είναι αφοπλισμένο.


Σχήμα 6.9: (από αριστερά) Σύστημα έτοιμο, Κεντρική οθόνη συστήματος

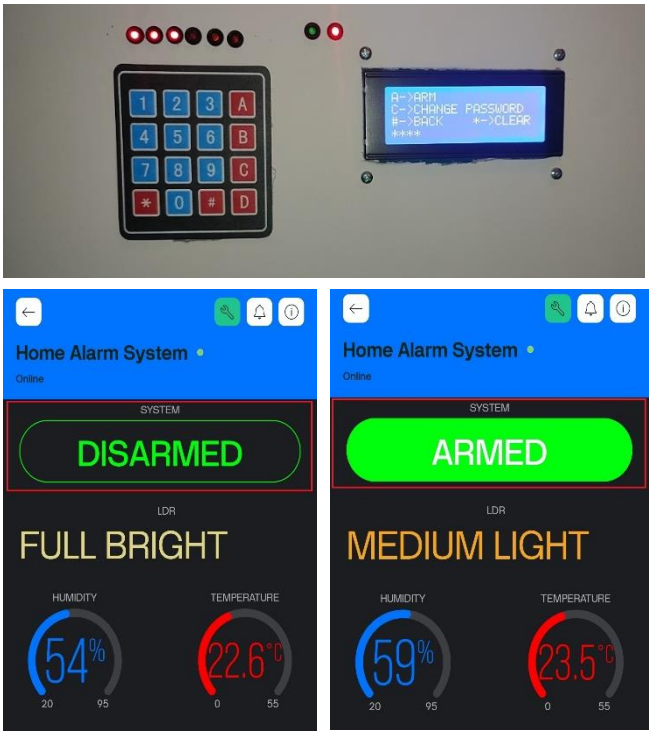
Εφόσον το σύστημα παραμένει εκτός δικτύου, θα προσπαθεί να συνδεθεί στο επιλεγμένο δίκτυο ανά 1 λεπτό. Μετά την επιτυχή σύνδεση του, θα ενημερώνει τον χρήστη μέσω της οθόνης, ενώ ταυτόχρονα θα ενημερώνει την ώρα και την ημερομηνία από τον διακομιστή Blynk.


Σχήμα 6.10: (από αριστερά) Αυτόματη ενημέρωση ώρας/ημερομηνίας, Κεντρική οθόνη συστήματος

Επιπλέον, όταν το σύστημα είναι αφοπλισμένο και κάποια από τις δύο μαγνητικές επαφές είναι ανοιχτή ή ο ανιχνευτής κίνησης διεγερθεί, τότε στην οθόνη εμφανίζεται μήνυμα που ενημερώνει τον χρήστη ότι υπάρχει ανοιχτή ζώνη και ποια είναι αυτή. Αυτό βοηθάει τον χρήστη να καταλάβει αν υπάρχει κάποια ζώνη που δεν λειτουργεί. Παράδειγμα αν το παράθυρο είναι κλειστό και εμφανίζεται το μήνυμα ότι η ζώνη είναι ανοιχτή, αυτό συνεπάγεται ότι υπάρχει κάποια βλάβη στο καλώδιο ή η μαγνητική επαφή θέλει αλλαγή. Επίσης, ο χρήστης δεν μπορεί να οπλίσει το σύστημα αν υπάρχει ανοιχτή ζώνη, και πρέπει να περιμένει να δει στην οθόνη ποια ζώνη είναι ανοιχτή ώστε να την κλείσει και να οπλίσει το σύστημα συναγερμού.


Σχήμα 6.11: (από πάνω αριστερά) Μαγνητική επαφή πόρτας, Μαγνητική επαφή παραθύρου, Ανιχνευτής κίνησης

Ο οπλισμός του συστήματος μπορεί να γίνει με δύο τρόπους. Ο πρώτος τρόπος είναι να πληκτρολογήσουμε τον κωδικό από το πληκτρολόγιο και έπειτα να πιέσουμε το πλήκτρο Α. Ο δεύτερος τρόπος είναι να επιλέξουμε τον οπλισμό του συστήματος από την εφαρμογή Blynk στο κινητό τηλέφωνο μας. Και στις δυο περιπτώσεις, θα υπάρχει 15 δευτερόλεπτα αντίστροφη μέτρηση μέχρι το σύστημα να οπλίσει, αυτό είναι χρήσιμο για να μπορέσουμε να ανοίξουμε την κεντρική πόρτα και να την κλείσουμε χωρίς να διεγείρουμε την ζώνη αυτή. Αφού περάσουν τα 15 δευτερόλεπτα, τότε ανάβει το πράσινο LED που υποδηλώνει ότι το σύστημα είναι οπλισμένο.



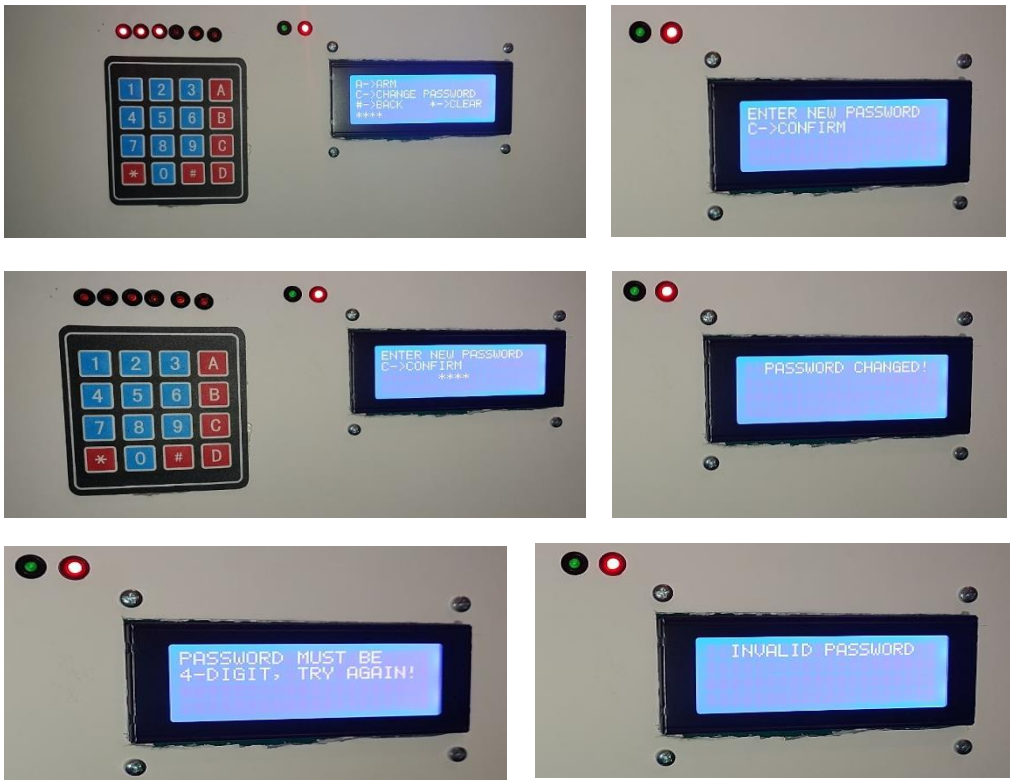

Σχήμα 6.12: (από επάνω και μετά κάτω αριστερά) Οπλισμός απο το πληκτρολόγιο και μενού επιλογών, Οπλισμός από την εφαρμογή, Μετά τον οπλισμό


Σχήμα 6.13: (από αριστερά) Χρόνος εξόδου, Κεντρική οθόνη και σύστημα οπλισμένο

Για να αφοπλίσουμε το σύστημα συναγερμού, υπάρχουν δύο τρόποι και ακολουθούμε παρόμοια διαδικασία. Ο πρώτος τρόπος είναι, αφού εισέλθουμε στον χώρο από την είσοδο που έχουμε δηλώσει ως κεντρική, το σύστημα μας παρέχει 15 δευτερόλεπτα για να πληκτρολογήσουμε τον κωδικό και να πιέσουμε το B για να αφοπλίσουμε. Ο δεύτερος τρόπος, είναι να επιλέξουμε τον αφοπλισμό από την εφαρμογή Blynk στο κινητό τηλέφωνο πριν εισέλθουμε στον χώρο.

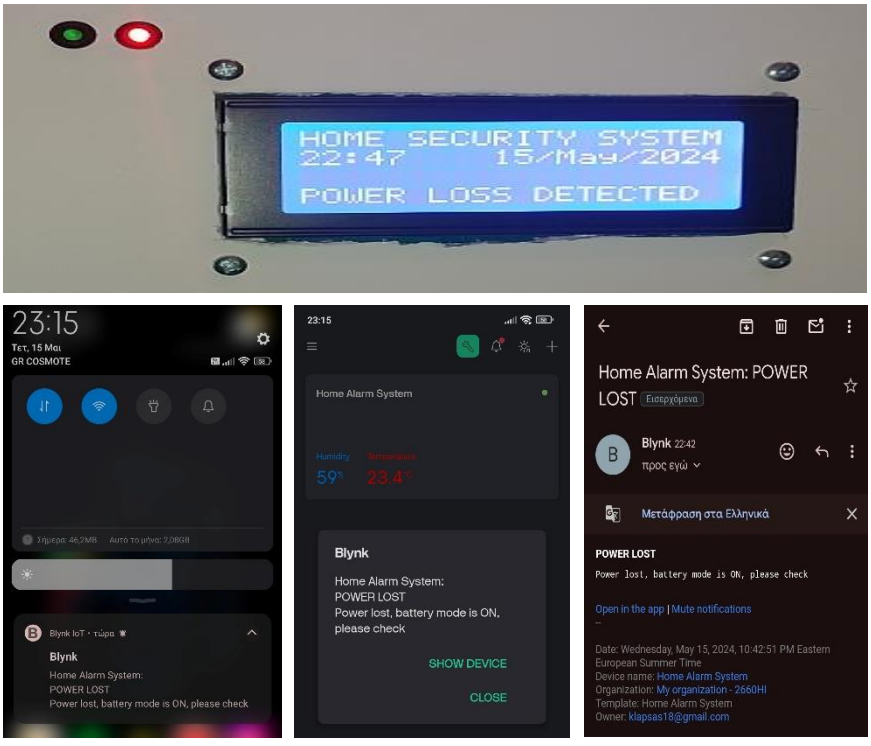

Σχήμα 6.14: (από επάνω αριστερά) Χρόνος εισόδου, Αφοπλισμός από πληκτρολόγιο και μενού επιλογών, Επιλογή αφοπλισμού από εφαρμογή, Μετά τον αφοπλισμό, Μήνυμα αφοπλισμού

Ο χρήστης, έχει την δυνατότητα να κάνει αλλαγή κωδικού εφόσον το σύστημα είναι αφοπλισμένο. Εισάγοντας τον παλιό κωδικό και πιέζοντας το πλήκτρο C, εμφανίζεται στην οθόνη το μήνυμα για πληκτρολόγηση του νέου 4-ψήφιου κωδικού. Αφού ολοκληρωθεί η διαδικασία με επιτυχία, εμφανίζεται στην οθόνη το κατάλληλο μήνυμα. Σε οποιαδήποτε από τις παραπάνω περιπτώσεις που αναφέρθηκαν, αν εισαχθεί λάθος κωδικός τότε εμφανίζεται στην οθόνη το μήνυμα ότι ο κωδικός είναι άκυρος και το σύστημα επιστρέφει στην κεντρική οθόνη. Αν σε περίπτωση ο χρήστης δεν είναι σίγουρος ότι πληκτρολόγησε τον σωστό κωδικό, έχει την δυνατότητα να πιέσει το πλήκτρο * και να εισάγει ξανά τον κωδικό από την αρχή, ενώ με το πλήκτρο # βγαίνει από το μενού επιλογών και επιστρέφει στην κεντρική οθόνη.



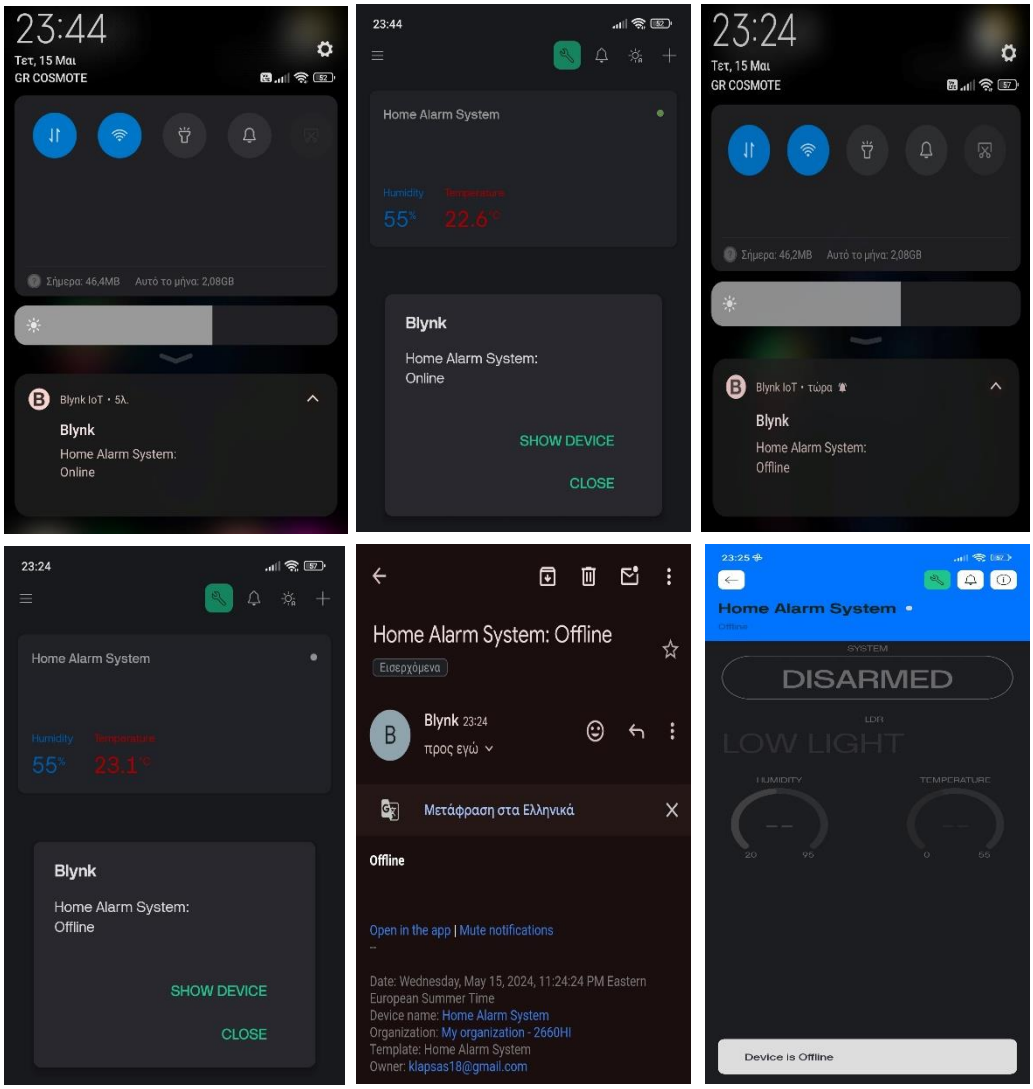

Σχήμα 6.15: (από πάνω αριστερά) Κωδικός και μενού επιλογών, Οθόνη αλλαγής κωδικού, Νέος κωδικός, Επιτυχής αλλαγή κωδικού, Κωδικός μεγαλύτερος από 4 ψηφία, Λάθος κωδικός

Όταν κατά την διάρκεια λειτουργίας του συστήματος συναγερμού η κύρια πηγή ενέργειας είναι εκτός, τότε εμφανίζεται στην κεντρική οθόνη το μήνυμα «ανίχνευση απώλειας ισχύος», ενώ ο χρήστης ενημερώνεται με email και ειδοποίηση push στο κινητό τηλέφωνο από την εφαρμογή Blynk.




Σχήμα 6.16: (από επάνω και μετά κάτω αριστερά) Μήνυμα στην κεντρική οθόνη, Ειδοποίηση push, Ειδοποίηση εντός εφαρμογής, Ειδοποίηση email

Τέλος, ο χρήστης έχει την δυνατότητα να ενημερώνεται αν το σύστημα είναι συνδεμένο ή έχει αποσυνδεθεί από το δίκτυο μέσω της εφαρμογής Blynk, με ειδοποιήσεις push στο κινητό τηλέφωνο του. Αν το σύστημα είναι εκτός δικτύου, τότε στην εφαρμογή δεν έχει την επιλογή να οπλίσει και να αφοπλίσει το σύστημα, όπως επίσης δεν μπορεί να δει την υγρασία/θερμοκρασία και την ποσότητα του φωτός στον χώρο.



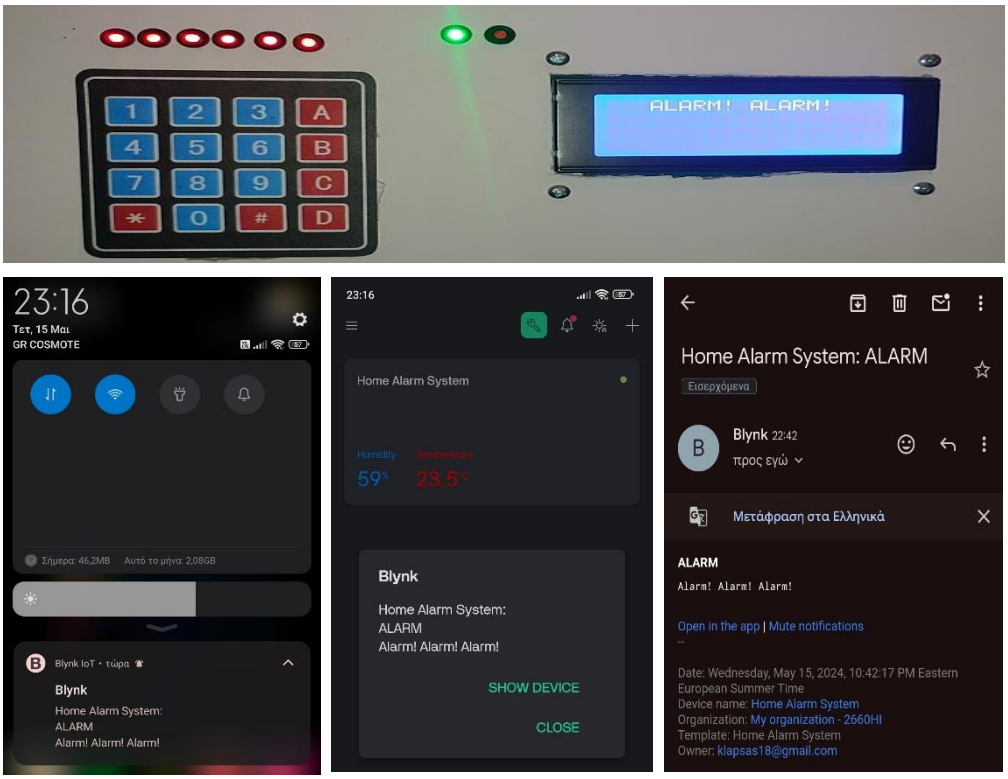

Σχήμα 6.17: (από επάνω αριστερά) Ειδοποίηση push για σύνδεση, Ειδοποίηση εντός εφαρμογής για σύνδεση, Ειδοποίηση push για αποσύνδεση, Ειδοποίηση εντός εφαρμογής για αποσύνδεση, Ειδοποίηση με email για αποσύνδεση, Περιβάλλον εφαρμογής όταν είναι αποσυνδεμένο

## 6.6 Σενάρια

Έχουμε ορίσει ότι η ζώνη της κεντρικής εισόδου θα έχει χρόνο εισόδου πριν χτυπήσει ο συναγερμός, ενώ οι άλλες δυο ζώνες θα ενεργοποιούν κατευθείαν τον συναγερμό. Για την σωστή λειτουργία του συστήματος συναγερμού, θα δοκιμαστούν 3 σενάρια, στο πρώτο σενάριο το σύστημα θα είναι οπλισμένο και θα διεγερθεί η μαγνητική επαφή του παραθύρου, στο δεύτερο σενάριο θα εκτεθεί ο ανιχνευτής φωτός σε αρκετό φως και στο τρίτο σενάριο θα αυξηθεί η υγρασία στον χώρο.

### 6.6.1 Πρώτο σενάριο

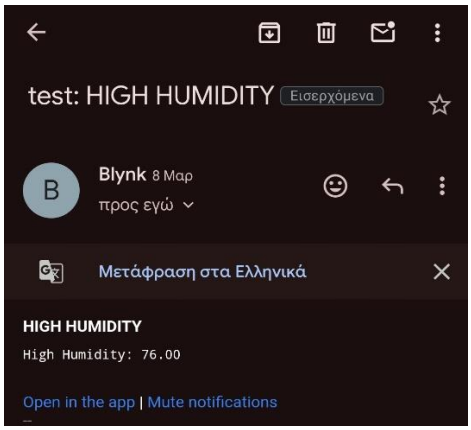
Σε αυτό το σενάριο, ο χρήστης έχει οπλίσει το σύστημα συναγερμού και κάποιος επίδοξος διαρρήκτης ανοίγει το παράθυρο. Τότε ο συναγερμός δεν δίνει 15 δευτερόλεπτα χρόνο εισόδου, αλλά ενεργοποιεί κατευθείαν την σειρήνα και εμφανίζει στην οθόνη το μήνυμα συναγερμός, στέλνει ειδοποίηση push και email. Για την αφόπλιση του συστήματος πρέπει να εισαχθεί ο κωδικός από το πληκτρολόγιο ή να γίνει αφοπλισμός από την εφαρμογή.




Σχήμα 6.18: (από επάνω και μετά κάτω αριστερά) Οθόνη συναγερμού και LED αναβοσβήνουν, Ειδοποίηση push, Ειδοποίηση εντός εφαρμογής, Ειδοποίηση με email

### 6.6.2 Δεύτερο σενάριο

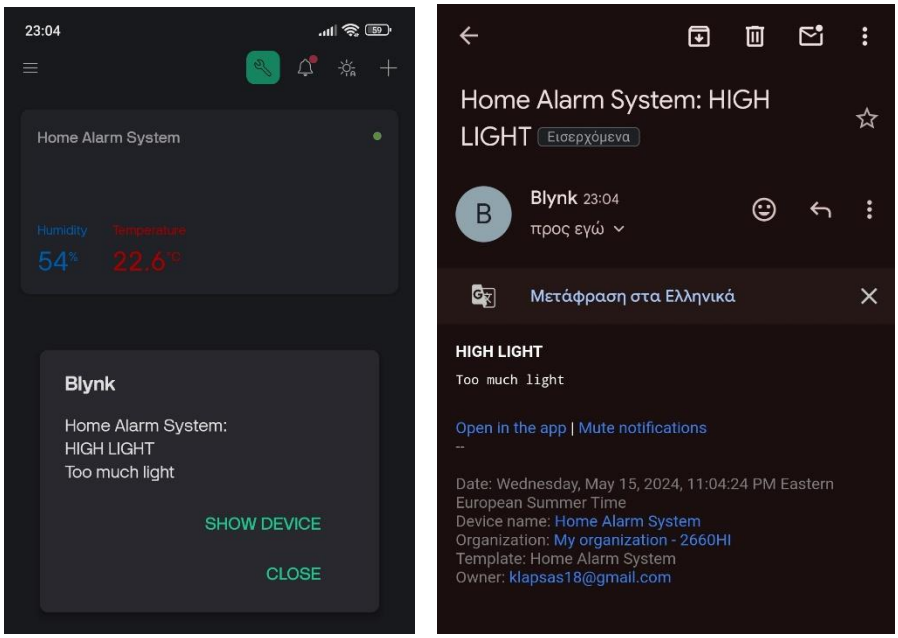
Στο δεύτερο σενάριο, ο χρήστης απουσιάζει από τον χώρο και ξαφνικά η υγρασία ανεβαίνει σε υψηλά επίπεδα και πάνω από το όριο που έχουμε ορίσει. Τότε, το σύστημα θα ενημερώσει τον χρήστη μέσω ειδοποίησης push και email.




Σχήμα 6.19: Ειδοποίηση για υψηλή υγρασία με email

### 6.6.3 Τρίτο σενάριο

Στο τελευταίο σενάριο, ενώ στον χώρο η ένταση του φωτός είναι σε χαμηλά επίπεδα, συμβαίνει μια αλλαγή στην φωτεινότητα του χώρου από ένα σπασμένο παντζούρι που οφείλεται σε πιθανή κακοκαιρία. Ο ανιχνευτής του φωτός το αντιλαμβάνεται και ενημερώνει τον χρήστη με ειδοποίηση push και με email.




Σχήμα 6.20: (από αριστερά) Ειδοποίηση εντός εφαρμογής για αύξηση φωτεινότητας, Ειδοποίηση email για αύξηση φωτεινότητας

## 6.7 Επίλογος

Σε αυτό το κεφάλαιο, αρχικά αναφέρθηκε πως έγινε ο προγραμματισμός του Arduino Due και σε ποιο περιβάλλον. Έπειτα, παρουσιάστηκε το διάγραμμα ροής του κώδικα που έχει σημαντικό ρόλο για την κατανόηση του προγράμματος. Τέλος, τέθηκε σε λειτουργία το σύστημα συναγερμού και έγιναν κάποια σενάρια για την σωστή λειτουργία του.

# Κεφάλαιο 7ο: Συμπεράσματα ή/και προτάσεις βελτίωσης

Για την εκπόνηση της πτυχιακής εργασίας, προηγήθηκαν συζητήσεις με τον επιβλέποντα καθηγητή για την επιλογή του θέματος. Έπειτα, τέθηκαν με την σειρά οι στόχοι που πρέπει να επιτευχθούν για την ολοκλήρωση της. Στη συνέχεια, ακολούθησε έρευνα για την επιλογή του μικροελεγκτή καθώς και για τα περιφερειακά του, τα υλικά κατασκευής της μακέτας και της πλακέτας.

Κατά την διάρκεια της επιλογής των υλικών, το ελληνικό κατάστημα GRobotronics δεν διέθετε οθόνη LCD 20x4 με τάση λειτουργίας στα 3.3V για να μπορεί το Arduino Due να επικοινωνεί μαζί του μέσω I2C, οπότε αποφασίστηκε σαν λύση να δημιουργηθεί μια διάταξη που θα αλλάζει τα επίπεδα τάσης στην μεταξύ τους επικοινωνία, αυτό είχε ως αποτέλεσμα να γίνει έρευνα για αυτήν την διάταξη και τι υλικά θα χρειαστεί για να επιτευχθεί. Ένα ακόμα πρόβλημα που διαπιστώθηκε, είναι στην λειτουργία των LED της σειρήνας, καθώς όταν συμβαίνουν διεργασίες που χρησιμοποιούν την εντολή «delay();» ή εντολές βρόγχου, τότε ο μικροελεγκτής σταματάει την ροή του προγράμματος έως ότου ολοκληρωθούν, αυτό έχει ως αποτέλεσμα να μην αναβοσβήνουν τα LED διαδοχικά σε τριάδες αλλά να μένουν αναμμένα μόνο τα LED της πρώτης ή της δεύτερης τριάδας. Μια λύση, ήταν να γραφτεί εντολή που θα σβήνει εντελώς όλα τα LED της σειρήνας και θα ανάβουν για να συνεχίσουν το μοτίβο τους αφού επανέλθει ο μικροελεγκτής στην κανονική ροή του.

Το κόστος για την κατασκευή της πτυχιακής εργασίας ήταν συνολικά 103,75€. Ακριβότερο υλικό ήταν το Arduino Due, η μπαταρία και η οθόνη LCD, ενώ το κόστος των υπόλοιπων υλικών ήταν μικρότερο από 2€.

Συνεπάγεται το συμπέρασμα, πως με τα παραπάνω κεφάλαια και κυρίως τα κεφάλαια 5 και 6, οι στόχοι της παρούσας πτυχιακής εργασίας επιτεύχθηκαν και αποκτήθηκαν γνώσεις για το εργαλείο Arduino όπως και για ηλεκτρονικές διατάξεις που συμπεριλήφθηκαν στο κύκλωμα. Επιπρόσθετα, διαπιστώνεται ότι το σύστημα συναγερμού που υλοποιήθηκε μπορεί να χρησιμοποιηθεί σαν μια οικονομικότερη επιλογή για ένα σύστημα συναγερμού.

Τέλος, θα μπορούσαν να υπάρξουν μελλοντικές βελτιώσεις σε αυτήν την κατασκευή, όπως για παράδειγμα η σχεδίαση και κατασκευή τυπωμένης πλακέτας για την εξοικονόμηση χώρου. Επιπλέον, θα μπορούσε να προστεθεί και ένα ολοκληρωμένο κύκλωμα GPRS, παράδειγμα το SIM900, που θα δέχεται μια κάρτα SIM και έτσι το σύστημα συναγερμού θα μπορεί να παίρνει τηλέφωνο τον χρήστη για να τον ειδοποιεί όταν υπάρχει συναγερμός σε εξέλιξη, ενώ με αυτόν τον τρόπο ακόμα και τα καλώδια του παρόχου δικτύου στο σπίτι να κοπούν από τον διαρρήκτη δεν θα επηρεάσουν το σύστημα. Ακόμα μια βελτίωση που θα μπορούσε να γίνει, είναι ένα ξεχωριστό κύκλωμα για την σειρήνα με δική της τροφοδοσία, και τα LED να μην ελέγχονται από τον κεντρικό μικροελεγκτή, ώστε να μην επηρεάζεται από τα delay και τους βρόγχους του κώδικα και απλά το Arduino να την ενεργοποιεί. Σημαντική βελτίωση θα ήταν να αλλαχθεί ο μικροελεγκτής με κάποιον που έχει παραπάνω από έναν πυρήνα ή να γραφτεί κώδικας και να προγραμματιστεί το ESP12-E με αυτόν για την καλύτερη διαχείριση την σύνδεσης με τον διακομιστή Blynk ή και ακόμα την διαχείριση των χρόνων εξόδου και εισόδου. Κάτι ακόμα που θα μπορούσε να γίνει, είναι να βελτιωθεί και να εμπλουτισθεί ο κώδικας.

# ΒΙΒΛΙΟΓΡΑΦΙΑ


**Βιβλία**

[1] Μαγκανιάρη Α. Μαρία, *ΣΥΣΤΗΜΑΤΑ ΕΛΕΓΧΟΥ ΚΑΙ ΑΣΦΑΛΕΙΑΣ*. Γ΄ΕΠΑΛ: ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΕΚΔΟΣΕΩΝ «ΔΙΟΦΑΝΤΟΣ»

**Application Note**

[12] Philips Semiconductors, Appl. Note 97055, pp.01-16.

**Data Sheet**

[10] Raystar, “LCD Display 20x4,” RC2004A-BIW-CSX datasheet.

[11] Texas Instruments, “Remote 8-bit I/O expander for I2C-bus,” PCF8574 datasheet, Oct. 2006.

[13] STMicroelectronics, “N-channel Power MOSFET,” 2N7000 datasheet, JUN. 2006.

[15] Globlab Corporation, “Infrared Parts Manual,” RE200B datasheet, Nov. 1999.

[16] Silvan Chip Electronics Tech.Co.,Ltd., “PIR CONTROLLER,” BISS0001 datasheet.

[17] ETC2 [List of Unclassifed Manufacturers], “PIR MOTION DETECTOR,” HC-SR501 datasheet.

[18] ETC [List of Unclassifed Manufacturers], “DHT11 Humidity & Temperature Sensor,” DHT11 datasheet.

[20] ETC2 [List of Unclassifed Manufacturers], “Photoresistor,” GL5528 datasheet.

**Internet Site**

[2] Arduino, “What is Arduino?”, 2022. [Online]. Available: https://docs.arduino.cc/learn/starting-guide/whats-arduino/

[4] Indian Institute of Embedded Systems (IIES), “Role Of Arduino In Real World Applications”, 2024. [Online]. Available: https://iies.in/blog/role-of-arduino-in-real-world-applications/

[5] Arduino, “Due”, 2024. [Online]. Available: https://docs.arduino.cc/hardware/due/#tech-specs

[6] Arduino, “Arduino Due is finally here”, 2024. [Online]. Available: https://blog.arduino.cc/2012/10/22/arduino-due-is-finally-here/

[7] Blynk, “Welcome to Blynk Documentation”, 2024. [Online]. Available: https://docs.blynk.io/en

[8] Embedded, “Keyboard and display multiplexing — the traditional approach”, 2015. [Online]. Available: https://www.embedded.com/keyboard-and-display-multiplexing-the-traditional-approach/

[9] TronicsBench, “Using the Hitachi HD44780 with the Arduino”, 2024. [Online]. Available: https://www.best-microcontroller-projects.com/hitachi-hd44780.html

[14] Microcontrollerslab, “ESP12E WiFi Module Complete Guide,”. [Online]. Available: https://microcontrollerslab.com/esp12e-wifi-module-pinout-arduino-interfacing-examples/

[19] The engineering projects, “Introduction to DHT11,” 2019. [Online]. Available: https://www.theengineeringprojects.com/2019/03/introduction-to-dht11.html

[21] Electronics Tutorials, “Light Sensors”. [Online]. Available: https://www.electronics-tutorials.ws/io/io_4.html

[22] EEPower, “What are Photoresistors?”. [Online]. Available: https://eepower.com/resistor-guide/resistor-types/photo-resistor/#

[23] Wikipedia, “Νόμος του Ωμ”. [Online]. Available: https://el.wikipedia.org/wiki/

[24] Wikipedia, “Κανόνες του Κίρχοφ”. [Online]. Available: https://el.wikipedia.org/wiki/

**Paper in Conference Proceedings**

[3] Yusuf Abdullahi Badamasi, “The working principle of an Arduino,” presented at 11th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, Nigeria, 2014.

# ΠΑΡΑΡΤΗΜΑ Α : ΚΑΤΑΛΟΓΟΣ ΥΛΙΚΩΝ

| Προϊόν | Τιμή | Ποσότητα | Υποσύνολο |
|---|---|---|---|
| Οθόνη | 6,37 € | 1 | 6,37 € |
| I2C module | 0,97 € | 1 | 0,97 € |
| Buzzer | 0,32 € | 2 | 0,65 € |
| Arduino Due | 42,66 € | 1 | 42,66 € |
| Πληκτρολόγιο | 1,21 € | 1 | 1,21 € |
| Jumper wires | 2,90 € | 2 | 5,80 € |
| Κόκκινο LED | 0,06 € | 7 | 0,45 € |
| Πράσινο LED | 0,06 € | 1 | 0,06 € |
| LED Holder | 0,06 € | 8 | 0,52 € |
| Σύστημα διαχείρισης μπαταρίας | 2,34 € | 1 | 2,34 € |
| Μπαταρία λιθίου 18650 | 5,48 € | 1 | 5,48 € |
| Μπαταριοθήκη | 0,97 € | 1 | 0,97 € |
| PIR | 2,26 € | 1 | 2,26 € |
| Αντιστάσεις 1ΚΩ | 0,02 € | 8 | 0,13 € |
| Αντιστάσεις 10ΚΩ | 0,02 € | 5 | 0,08 € |
| Αντιστάσεις 22ΚΩ | 0,02 € | 1 | 0,02 € |
| Αντιστάσεις 100ΚΩ | 0,02 € | 1 | 0,02 € |
| Πλακέτα δοκιμών | 2,58 € | 1 | 2,58 € |
| Dupont ακροδέκτες kit | 3,87 € | 1 | 3,87 € |
| ESP12-E | 2,90 € | 1 | 2,90 € |
| ESP Breakout | 0,48 € | 1 | 0,48 € |
| JST XH wire | 0,24 € | 2 | 0,48 € |
| Pin header 1x8 | 0,24 € | 6 | 1,44 € |
| Κλέμες | 0,32 € | 6 | 1,94 € |
| Voltage regulator 5V | 0,32 € | 1 | 0,32 € |
| DHT11 | 1,94 € | 1 | 1,94 € |
| Πυκνωτές 100nF | 0,16 € | 1 | 0,16 € |
| Πυκνωτές 330nF | 0,20 € | 1 | 0,20 € |
| LDR | 0,16 € | 1 | 0,16 € |
| Πλακέτα διάτρητη 50χ100mm | 0,97 € | 1 | 0,97 € |
| Καλώδια πολύκλωνα 22AWG 6 χρώματα | 11,94 € | 1 | 11,94 € |
| Καλάι 0,7mm | 6,29 € | 1 | 6,29 € |
| Mosfet 2N7000 | 0,12 € | 2 | 0,24 € |
| Χαρτόνι μακέτας 70x100cm 5mm | 6,45 € | 2 | 12,90 € |
| | | Σύνολο: | 118,80 € |
| | | Σύνολο με ΦΠΑ: | 147,31 € |

# ΠΑΡΑΡΤΗΜΑ Β : ΚΩΔΙΚΑΣ

```
// Configuration
#define BLYNK_TEMPLATE_ID "TMPL4cF94lVK-"
#define BLYNK_TEMPLATE_NAME "Home Alarm System"
#define BLYNK_AUTH_TOKEN "EQuFWeQO0UR3aJW6E_FiTvvbpH89JCma"
#define DHTTYPE DHT11
#define BLYNK_PRINT Serial
```

```
#define ESP8266_BAUD 38400

// Libraries
#include <DHT.h>
#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>
#include <TimeLib.h>
#include <WidgetRTC.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>

// Pin configurations
const int pirPin = 26;              // PIR sensor output pin
const int buzzerPinKey = 50;        // Buzzer output pin for keypress
const int buzzerPinAlarm = 51;      // Buzzer output pin for alarm
const int redLedPin = 32;           // Red LED pin
const int greenLedPin = 34;         // Green LED pin
const int ledPin1 = 36;             // Siren Led1
const int ledPin2 = 38;             // Siren Led2
const int ledPin3 = 40;             // Siren Led3
const int ledPin4 = 42;             // Siren Led4
const int ledPin5 = 44;             // Siren Led5
const int ledPin6 = 46;             // Siren Led6
const int powerDetectPin = A0;      // Analog pin connected to the voltage divider
const float voltageThreshold = 0.8; // Voltage threshold for power loss detection in volts
const int reedSwitchPin1 = 22;      // Pin for the first reed switch
const int reedSwitchPin2 = 24;      // Pin for the second reed switch
const int dhtpin = 48;              // Pin for temperature and humidity sensor
const int ldrpin = A1;              // Analog pin for light sensor

// Flags
bool powerLost = false;             // Flag indicating whether a power loss has been detected
bool systemArm = false;             // Flag indicating whether the system is armed
bool alarmTriggered = false;        // Flag indicating whether the alarm has been triggered
bool offlineAlarmTriggered = false; // Flag indicating whether the alarm is triggered while offline
bool Pir = false;                   // Flag indicating whether motion is detected by the PIR sensor
bool Reed1 = false;                 // Flag indicating whether the main door is open (reed switch 1)
bool Reed2 = false;                 // Flag indicating whether the window is open (reed switch 2)
bool displayPirMessage = false;     // Flag for display the triggered message of Pir sensor
bool displayReed1Message = false;   // Flag for display the triggered message of first Reed sensor
bool displayReed2Message = false;   // Flag for display the triggered message of second Reed sensor
bool isArming = false;              // Flag indicating whether the system is Arming or not
bool inPasswordScreen = false;      // Flag indicating whether the user is in password screen
bool shouldUpdateScreen = true;     // Flag to track if the screen should be updated
bool lastKeyPress = false;          // Flag for tracking if the user is pressed key after the first time
bool firstKeyPressed = false;       // Flag for tracking if the user is pressed key for first time
bool connectWifiFisrtTime = false;  // Flag indicating whether the system is connected or not in wifi
after the initialization
```

```
// Initialize DHT
DHT dht(dhtpin, DHTTYPE);  // Object for managing the sensor library

// Initialize ESP8266 and Blynk
char ssid[] = "test";
char pass[] = "123456789";
char auth[] = "EQuFWeQO0UR3aJW6E_FiTvvbpH89JCma";  // Auth key for blynk server
char server[] = "blynk.cloud";
int port = 80;
ESP8266 wifi(&Serial1);  // Object for ESP8266_Lib and with this i can manage the EPS8266
int ReCnctFlag;          // Reconnection Flag
int ReCnctCount = 0;     // Reconnection counter

// Objects for blynk library
BlynkTimer timer;  // Object that can manage the timer
WidgetRTC rtc;     // Object that can manage the rtc from blynk server

// Initialize LCD
LiquidCrystal_I2C lcd(0x27, 20, 4);  // Object for managing the lcd library

// Initialize Keypad
const byte numRows = 4;
const byte numCols = 4;
char keypressed;
char keymap[numRows][numCols] = {
  { '1', '2', '3', 'A' },
  { '4', '5', '6', 'B' },
  { '7', '8', '9', 'C' },
  { '*', '0', '#', 'D' }
};
byte rowPins[numRows] = { 23, 25, 27, 29 };                    // Row pinouts of the keypad
byte colPins[numCols] = { 31, 33, 35, 37 };                    // Column pinouts of the keypad
Keypad myKeypad = Keypad(makeKeymap(keymap), rowPins, colPins, numRows, numCols);   // Object for keypad libraty that can manage the pressed keys from user

// Password to arm and disarm the system
String key;              // Variable for pressed key
String pad;              // Variable for entered password
String padSecret;        // Variable for printing "*" instead of key pressed
String password = "1234";  // Password

// Timers and Intervals
unsigned long startPirTime = 0;              // Timer for tracking how much time is showing the message of Pir in lcd
unsigned long startReed1Time = 0;            // Timer for tracking how much time is showing the message of first Reed in lcd
unsigned long startReed2Time = 0;            // Timer for tracking how much time is showing the message of second Reed in lcd
```

```
unsigned long lastKeyPressTime = 0;          // Timer for tracking how much time passed since the user pressed the last key
unsigned long StarDisplayTime = 0;           // Timer for tracking how much time passed since the message of the two Reeds is showed in lcd
unsigned long previousMillis = 0;            // Timer for tracking how much time passed since the siren leds change status
const long wait = 500;                       // Interval in milliseconds for changing the status of siren leds
unsigned long lastDisplayTime = 0;           // Timer for tracking how much time passed since the lcd is refreshed
unsigned long lastSensorReadTime = 0;        // Timer for tracking how much time passed since the DHT11 sensor refresh its values
const unsigned long displayInterval = 20000;     // Interval in milliseconds for refreshing the lcd
const unsigned long sensorReadInterval = 60000;  // Interval in milliseconds for refreshing the DHT11 sensor values

// Variable to track the LED sequence step
int step = 0;          // Regular blink step counter
int alarmBlinkStep = 0;  // Alarm-triggered blink step counter

// Initialization for SetTime function when system start without Wifi ON
int hr = 0;
int min = 0;
int sec = 0;
int currentDay = 1;
int mnth = 1;
int yr = 2024;

// Void setup
void setup() {
  Serial.begin(115200);
  Serial1.begin(ESP8266_BAUD);
  delay(10);
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.print("SYSTEM STARTING");
  delay(2000);
  lcd.clear();
  lcd.print("LCD INITIALIZED");
  delay(2000);
  lcd.clear();
  lcd.print("CONNECTING TO WIFI..");
  dht.begin();
  myKeypad.setDebounceTime(40);
  if (wifi.joinAP(ssid, pass)) {  // If WIFI is available then connect
    connectWifiFisrtTime = true;
    Blynk.config(wifi, auth, "blynk.cloud", 80);  // Blynk configuration function
    Blynk.connectWiFi(ssid, pass);
    Blynk.connect();  // Blynk function for connect to the server
```

```
  lcd.clear();
  lcd.print("CONNECTED TO WIFI");
  delay(2000);
 } else {  // If WIFI is not available
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("NOT CONNECTED");
  lcd.setCursor(0, 1);
  lcd.print("TO WIFI");
  delay(1000);
  wifi.joinAP(ssid, pass);                    // Give the ESP8266 the
  Blynk.config(wifi, auth, "blynk.cloud", 80);  //  Blynk.config
  Blynk.connectWiFi(ssid, pass);
  lcd.clear();
  // Set initial time and date
  setTime(hr, min, sec, currentDay, mnth, yr);
  // Prompt user for time
  lcd.setCursor(0, 0);
  lcd.print("SET TIME: HH:MM");

  hr = getUserInput(10, 0);
  min = getUserInput(13, 0);

  // Move cursor to date position
  lcd.setCursor(0, 1);

  lcd.print("SET DATE: DD/MM/YY");

  currentDay = getUserInput(10, 1);
  mnth = getUserInput(13, 1);
  yr = getUserInput(16, 1);

  // Set the updated time and date
  setTime(hr, min, sec, currentDay, mnth, yr);
 }
 setSyncInterval(10 * 60);  // Sync interval in seconds (10 minutes)
 analogReadResolution(12);
 // Declare pins
 pinMode(pirPin, INPUT);
 pinMode(buzzerPinKey, OUTPUT);
 pinMode(buzzerPinAlarm, OUTPUT);
 pinMode(redLedPin, OUTPUT);
 pinMode(greenLedPin, OUTPUT);
 pinMode(reedSwitchPin1, INPUT_PULLUP);
 pinMode(reedSwitchPin2, INPUT_PULLUP);
 pinMode(ledPin1, OUTPUT);
 pinMode(ledPin2, OUTPUT);
 pinMode(ledPin3, OUTPUT);
 pinMode(ledPin4, OUTPUT);
```

```
  pinMode(ledPin5, OUTPUT);
  pinMode(ledPin6, OUTPUT);
  // Initialize LCD Message
  Blynk.virtualWrite(V0, 0);
  lcd.clear();
  lcd.print("SYSTEM READY");
  delay(2000);
  detectPowerLoss();
  updateLEDs(true, false);
}

BLYNK_CONNECTED() {
  // Synchronize time on connection
  ReCnctCount = 0;
  rtc.begin();
  if (systemArm) {
    Blynk.virtualWrite(V0, 1);
  } else {
    Blynk.virtualWrite(V0, 0);
  }
  if (!alarmTriggered && !inPasswordScreen && !isArming && !connectWifiFisrtTime) {
    lcd.clear();
    lcd.print("TIME WILL BE SYNC");
    lcd.setCursor(0, 1);
    lcd.print("WITH SERVER TIME");
    delay(2000);
    connectWifiFisrtTime = true;
  }
  if (alarmTriggered) {
    Blynk.logEvent("alarm", String("Alarm! Alarm! Alarm!"));
  }
  if (offlineAlarmTriggered && !alarmTriggered) {
    Blynk.logEvent("alarm", String("Alarm! Alarm! Alarm!"));
    offlineAlarmTriggered = false;
  }
}

void loop() {
  timer.run();
  if (Blynk.connected() && !inPasswordScreen && !isArming) {  // If connected run as normal
    Blynk.run();
  } else if (ReCnctFlag == 0 && !inPasswordScreen && !isArming) {  // If NOT connected and not
already trying to reconnect, set timer to try to reconnect in 30 seconds
    ReCnctFlag = 1;                                    // Set reconnection Flag
    Serial.println("Starting reconnection timer in 45 seconds...");
    timer.setTimeout(45000L, []() {  // Lambda Reconnection Timer Function
      if (!inPasswordScreen && !isArming) {
        ledsoff();
        ReCnctFlag = 0;  // Reset reconnection Flag
```

```
      ReCnctCount++;   // Increment reconnection Counter
      Serial.print("Attempting reconnection #");
      Serial.println(ReCnctCount);
      wifi.joinAP(ssid, pass);  // Corrected WiFi begin
      Blynk.config(wifi, auth, "blynk.cloud", 80);
      Blynk.connectWiFi(ssid, pass);
      Blynk.connect();  // Try to reconnect to the server
    } else {
      ReCnctFlag = 0;
    }
  });  // END Timer Function
}
unsigned long currentTime = millis();

// Check if it's time to display the main screen
if (currentTime - lastDisplayTime >= displayInterval) {
  displayMainScreen();
  lastDisplayTime = currentTime;
}

// Check if it's time to read sensor data
if (currentTime - lastSensorReadTime >= sensorReadInterval) {
  readSensorData();
  lastSensorReadTime = currentTime;
}
// Check if the screen should be updated
if (shouldUpdateScreen) {
  lcd.clear();
  displayMainScreen();
  StarDisplayTime = millis();
  shouldUpdateScreen = false;  // Reset the flag after updating the screen
}
checkReedSwitch1();
checkReedSwitch2();
readPIR();
readKeypad();
detectPowerLoss();
updateSirenLEDs();
}
// Function for main screen message
void displayMainScreen() {
 if ((!inPasswordScreen || !lastKeyPress) && !alarmTriggered) {
   String displayString = clockDisplay();
   lcd.setCursor(0, 0);
   lcd.print("HOME SECURITY SYSTEM");
   lcd.setCursor(0, 1);
   lcd.print(displayString);
   // Check if power loss has been detected
   if (powerLost) {
```

```
      lcd.setCursor(0, 3);
      lcd.print("POWER LOSS DETECTED");
    } else {
      // If no power loss, print name in the 4th row
      lcd.setCursor(0, 3);
      lcd.print("GIORGOS  KLAPSINAKIS");
    }
    return;
  } else {
    return;
  }
}
// Function for Password actions screen message
void displayPasswordScreen(bool systemArm) {
  if (!firstKeyPressed) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(systemArm ? "B->DISARM" : "A->ARM");
    if (!systemArm) {
      lcd.setCursor(0, 1);
      lcd.print("C->CHANGE PASSWORD");
      lcd.setCursor(0, 2);
      lcd.print("#->BACK    *->CLEAR");
      lcd.setCursor(0, 3);
      lcd.print(padSecret);
    } else {
      lcd.setCursor(0, 1);
      lcd.print("#->BACK    *->CLEAR");
      lcd.setCursor(0, 2);
      lcd.print(padSecret);
    }
    return;
  } else {
    lcd.setCursor(0, 0);
    lcd.print(systemArm ? "B->DISARM" : "A->ARM");
    if (!systemArm) {
      lcd.setCursor(0, 1);
      lcd.print("C->CHANGE PASSWORD");
      lcd.setCursor(0, 2);
      lcd.print("#->BACK    *->CLEAR");
      lcd.setCursor(0, 3);
      lcd.print(padSecret);
    } else {
      lcd.setCursor(0, 1);
      lcd.print("#->BACK    *->CLEAR");
      lcd.setCursor(0, 2);
      lcd.print(padSecret);
    }
    return;
```

```
  }
}
// Function for reading and processing keypad input from the user for system control
void readKeypad() {
  keypressed = myKeypad.getKey();
  if (keypressed != NO_KEY && keypressed != 'A' && keypressed != 'B' && keypressed != 'C' &&
keypressed != '*' && keypressed != '#') {
    inPasswordScreen = true;  // Enter password screen mode
    lastKeyPressTime = millis();
    lastKeyPress = true;
    digitalWrite(buzzerPinKey, HIGH);  // Make a beep sound for
    delay(50);                         // the key pressed
    digitalWrite(buzzerPinKey, LOW);
    delay(10);  // Delay for better visual separation of asterisks on LCD
    key = String(keypressed);
    pad += key;
    padSecret += '*';  // Add '*' to the printed password
    displayPasswordScreen(systemArm);
    firstKeyPressed = true;
  }
  if (millis() - lastKeyPressTime >= 10000 && lastKeyPress) {
    // Code to exit the password screen
    pad = "";
    padSecret = "";
    key = "";
    shouldUpdateScreen = true;
    lastKeyPress = false;
    firstKeyPressed = false;
    inPasswordScreen = false;
    return;  // Exit the function early
  }
  if (keypressed == 'A') {
    armSystem();
    inPasswordScreen = false;  // Exit password screen mode
  } else if (keypressed == 'B') {
    disarmSystem();
    inPasswordScreen = false;  // Exit password screen mode
  } else if (keypressed == 'C' && !systemArm) {
    changePassword();
    inPasswordScreen = false;                    // Exit password screen mode
  } else if (keypressed == '*' && inPasswordScreen) {  // Clear the entered digits
    pad = "";
    padSecret = "";
    key = "";
    firstKeyPressed = false;
    displayPasswordScreen(systemArm);
    return;
  } else if (keypressed == '#' && inPasswordScreen && !alarmTriggered) {
    pad = "";
```

```
    padSecret = "";
    key = "";
    inPasswordScreen = false;  // Exit password screen mode
    shouldUpdateScreen = true;
    lastKeyPress = false;
    firstKeyPressed = false;
    return;
  } else if (keypressed == '#' && inPasswordScreen && alarmTriggered) {
    lcd.clear();
    lcd.setCursor(3, 0);
    lcd.print("ALARM! ALARM!");
    pad = "";
    padSecret = "";
    key = "";
    inPasswordScreen = false;  // Exit password screen mode
    lastKeyPress = false;
    firstKeyPressed = false;
    return;
  }
}
// Function for arming the system
void armSystem() {
  if (systemArm) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("SYSTEM ALREADY :");
    lcd.setCursor(0, 1);
    lcd.print("ARMED");
    delay(2000);
    if (alarmTriggered) {
      lcd.clear();
      lcd.setCursor(3, 0);
      lcd.print("ALARM! ALARM!");
    } else {
      shouldUpdateScreen = true;
    }
    pad = "";  // Reset the variables to their initial states
    padSecret = "";
    key = "";
    lastKeyPress = false;
    firstKeyPressed = false;
    return;  // Exit the function without re-arming
  }
  bool passwordCorrect = checkPassword(pad);  // Variable for checking if the password is correct
  int state2 = digitalRead(reedSwitchPin2);
  int pirValue = digitalRead(pirPin);
  if (passwordCorrect && state2 == LOW && pirValue == LOW) {
    lcd.clear();
    isArming = true;
```

```
  ledsoff();
  // Start of the arming countdown with sound
  for (int i = 15; i > 0; --i) {
    if (i > 7) {
      lcd.setCursor(0, 2);
      lcd.print("ARMING IN " + String(i) + " SECONDS ");
      digitalWrite(buzzerPinKey, HIGH);
      digitalWrite(redLedPin, HIGH);
      delay(100);
      digitalWrite(buzzerPinKey, LOW);
      digitalWrite(redLedPin, LOW);
      delay(900);
    } else if (i <= 7 && i > 3) {
      lcd.setCursor(0, 2);
      lcd.print("ARMING IN " + String(i) + " SECONDS ");
      digitalWrite(buzzerPinKey, HIGH);
      digitalWrite(redLedPin, HIGH);
      delay(400);
      digitalWrite(buzzerPinKey, LOW);
      digitalWrite(redLedPin, LOW);
      delay(600);
    } else {
      lcd.setCursor(0, 2);
      lcd.print("ARMING IN " + String(i) + " SECONDS ");
      digitalWrite(buzzerPinKey, HIGH);
      digitalWrite(redLedPin, HIGH);
      delay(1000);
      digitalWrite(buzzerPinKey, LOW);
      digitalWrite(redLedPin, LOW);
    }
  }
  lcd.setCursor(0, 2);
  lcd.print("                ");
  isArming = false;
  systemArm = true;  // Update the system variable
  lastKeyPress = false;
  firstKeyPressed = false;
  shouldUpdateScreen = true;
  updateLEDs(false, true);  // Turn off red LED, turn on green LED
  pad = "";                 // Reset the variables to their initial states
  padSecret = "";
  key = "";
  delay(10);
  Blynk.virtualWrite(V0, 1);
  return;
} else if (!passwordCorrect) {
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("INVALID PASSWORD");
```

```
    delay(2000);
    pad = "";  // Reset the variables to their initial states
    padSecret = "";
    key = "";
    shouldUpdateScreen = true;
    firstKeyPressed = false;
    lastKeyPress = false;
    return;
  } else {
    pad = "";  // Reset the variables to their initial states
    padSecret = "";
    key = "";
    shouldUpdateScreen = true;
    lastKeyPress = false;
    firstKeyPressed = false;
    Blynk.virtualWrite(V0, 0);
    return;  // Exit the function without re-arming
  }
}
// Function for disarming the system
void disarmSystem() {
  if (!systemArm) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("SYSTEM ALREADY :");
    lcd.setCursor(0, 1);
    lcd.print("DISARMED");
    delay(2000);
    pad = "";  // Reset the variables to their initial states
    padSecret = "";
    key = "";
    shouldUpdateScreen = true;
    lastKeyPress = false;
    firstKeyPressed = false;
    return;  // Exit the function without re-arming
  }
  bool passwordCorrect = checkPassword(pad);  // Variable for checking if the password is correct
  if (passwordCorrect) {
    updateLEDs(true, false);          // Turn on red LED, turn off green LED
    digitalWrite(buzzerPinAlarm, LOW);  // Turn off the Alarm Buzzer from the alarmTrigger Function
    pad = "";                         // Reset the variables to their initial states
    padSecret = "";
    key = "";
    systemArm = false;  // Update the system variable
    lcd.clear();
    lcd.setCursor(2, 0);
    lcd.print("SYSTEM DISARMED");
    delay(2000);
    shouldUpdateScreen = true;
```

```
    lastKeyPress = false;
    firstKeyPressed = false;
    alarmTriggered = false;
    Reed1 = false;
    Reed2 = false;
    Blynk.virtualWrite(V0, 0);
    return;
  } else if (!passwordCorrect) {
    lcd.clear();
    lcd.setCursor(2, 0);
    lcd.print("INVALID PASSWORD");
    delay(2000);
    if (alarmTriggered) {
      lcd.clear();
      lcd.setCursor(3, 0);
      lcd.print("ALARM! ALARM!");
    } else {
      shouldUpdateScreen = true;
    }
    pad = "";  // Reset the variables to their initial states
    padSecret = "";
    key = "";
    lastKeyPress = false;
    firstKeyPressed = false;
    return;
  } else {
    lcd.clear();
    lcd.setCursor(3, 0);
    lcd.print("ALARM! ALARM!");
    pad = "";  // Reset the variables to their initial states
    padSecret = "";
    key = "";
    lastKeyPress = false;
    firstKeyPressed = false;
    return;  // Exit the function without re-arming
  }
}
// Function for changing the system password
void changePassword() {
  bool passwordCorrect = checkPassword(pad);  // Variable for checking if the password is correct
  if (passwordCorrect && !systemArm) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("ENTER NEW PASSWORD");
    lcd.setCursor(0, 1);
    lcd.print("C->CONFIRM");
    pad = "";  // Reset the variables to their initial states
    padSecret = "";
    key = "";
```

```
    String newPassword = "";  // Variable for the new password
    ledsoff();
    while (newPassword.length() <= 4) {
      keypressed = myKeypad.getKey();
      if (keypressed != NO_KEY && keypressed != 'A' && keypressed != 'B' && keypressed != 'C' &&
keypressed != '*' && keypressed != 'D' && keypressed != '#') {
        digitalWrite(buzzerPinKey, HIGH);  // Make a beep sound for
        delay(50);                         // the key pressed
        digitalWrite(buzzerPinKey, LOW);
        delay(10);  // Delay for better visual separation of asterisks on LCD
        lcd.setCursor(7, 2);
        padSecret += '*';
        lcd.print(padSecret);
        key = String(keypressed);
        newPassword += key;
      } else if (keypressed == 'C') {
        break;
      }
    }
    if (newPassword.length() == 4) {
      password = newPassword;  // Update the password
      pad = "";                // Reset the variables to their initial states
      padSecret = "";
      key = "";
      lcd.clear();
      lcd.setCursor(2, 0);
      lcd.print("PASSWORD CHANGED!");
      delay(2000);
      shouldUpdateScreen = true;
      lastKeyPress = false;
      firstKeyPressed = false;
      return;
    }
  } else {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("PASSWORD MUST BE");
    lcd.setCursor(0, 1);
    lcd.print("4-DIGIT, TRY AGAIN!");
    delay(2000);
    pad = "";  // Reset the variables to their initial states
    padSecret = "";
    key = "";
    shouldUpdateScreen = true;
    lastKeyPress = false;
    firstKeyPressed = false;
    return;
  }
}
```

```
// Function for the PIR sensor
void readPIR() {
  int pirValue = digitalRead(pirPin);
  if (pirValue == HIGH && !Pir) {
    Pir = true;
    if (systemArm && !alarmTriggered && !Reed1) {
      triggerAlarmInst();
    } else if (!systemArm && !alarmTriggered && !inPasswordScreen && !isArming) {
      lcd.clear();
      lcd.setCursor(2, 0);
      lcd.print("MOTION DETECTED");
      lcd.setCursor(0, 1);
      lcd.print("IN THE SECOND ROOM!");
      startPirTime = millis();   // Store the current time
      displayPirMessage = true;  // Set the flag to display the message
    }
  } else if (pirValue == LOW && Pir) {
    Pir = false;
  }
  if (displayPirMessage && millis() - startPirTime >= 2000 && !inPasswordScreen) {
    displayPirMessage = false;  // Reset the flag
    shouldUpdateScreen = true;
  }
}
// Function for the Main door reed switch
void checkReedSwitch1() {
  int state1 = digitalRead(reedSwitchPin1);
  if (state1 == HIGH && !Reed1) {
    if (systemArm && !alarmTriggered) {
      Reed1 = true;
      triggerAlarm();
    } else if (!systemArm && !alarmTriggered && !inPasswordScreen && !isArming && millis() -
StarDisplayTime >= 5000 && !shouldUpdateScreen) {
      lcd.clear();
      lcd.setCursor(2, 0);
      lcd.print("MOTION DETECTED");
      lcd.setCursor(1, 1);
      lcd.print("MAIN DOOR IS OPEN!");
      startReed1Time = millis();   // Store the current time
      displayReed1Message = true;  // Set the flag to display the message
      Reed1 = true;
    }
  } else if (state1 == LOW && Reed1) {
    Reed1 = false;
  }
  if (displayReed1Message && millis() - startReed1Time >= 2000 && !inPasswordScreen) {
    displayReed1Message = false;  // Reset the flag
    shouldUpdateScreen = true;
    Reed1 = false;
```

```
  }
}
// Function for the window reed switch
void checkReedSwitch2() {
  int state2 = digitalRead(reedSwitchPin2);
  if (state2 == HIGH && !Reed2) {
    if (systemArm && !alarmTriggered) {
      Reed2 = true;
      triggerAlarmInst();
    } else if (!systemArm && !alarmTriggered && !inPasswordScreen && !isArming && millis() -
StarDisplayTime >= 5000 && !shouldUpdateScreen) {
      lcd.clear();
      lcd.setCursor(2, 0);
      lcd.print("MOTION DETECTED");
      lcd.setCursor(2, 1);
      lcd.print("WINDOW IS OPEN!");
      startReed2Time = millis();   // Store the current time
      displayReed2Message = true;  // Set the flag to display the message
      Reed2 = true;
    }
  } else if (state2 == LOW && Reed2) {
    Reed2 = false;
  }
  if (displayReed2Message && millis() - startReed2Time >= 2000 && !inPasswordScreen) {
    displayReed2Message = false;  // Reset the flag
    shouldUpdateScreen = true;
    Reed2 = false;
  }
}
// Function for the main door triggerAlarm with 15 second countdown duration
void triggerAlarm() {
  lcd.clear();
  lcd.setCursor(2, 0);
  lcd.print("MOTION DETECTED");
  lcd.setCursor(1, 1);
  lcd.print("MAIN DOOR IS OPEN!");
  unsigned long countdownStartTime = millis();
  const unsigned long countdownDuration = 15000;  // 15 seconds
  bool countdownExpired = false;
  ledsoff();
  // Allow 15 seconds for the user to enter the password
  while (!countdownExpired) {
    unsigned long currentTime = millis();
    if (currentTime - countdownStartTime >= countdownDuration || !systemArm) {
      countdownExpired = true;
      break;
    } else {
      lcd.setCursor(0, 3);
```

```
      lcd.print("TIME LEFT:" + String((countdownDuration - (currentTime - countdownStartTime)) /
1000) + " SECONDS");
      readKeypad();  // Check for keypad input during countdown
    }
  }
  // Check if the system is still armed before activating the alarm
  if (systemArm && !alarmTriggered) {
    lcd.clear();
    lcd.setCursor(3, 0);
    lcd.print("ALARM! ALARM!");
    digitalWrite(buzzerPinAlarm, HIGH);
    alarmTriggered = true;
    offlineAlarmTriggered = true;
    Blynk.logEvent("alarm", String("Alarm! Alarm! Alarm!"));
  }
}
// Function for Instant alarmTrigger without countdown duration
void triggerAlarmInst() {
  // Check if the system is still armed before activating the alarm
  if (systemArm && !alarmTriggered) {
    lcd.clear();
    lcd.setCursor(3, 0);
    lcd.print("ALARM! ALARM!");
    digitalWrite(buzzerPinAlarm, HIGH);
    alarmTriggered = true;
    offlineAlarmTriggered = true;
    Blynk.logEvent("alarm", String("Alarm! Alarm! Alarm!"));
  }
}
// Function for password check
bool checkPassword(String enteredPassword) {
  return enteredPassword == password;
}
// Function for update the LED's based on system state
void updateLEDs(bool redLedStatus, bool greenLedStatus) {
  digitalWrite(redLedPin, redLedStatus ? HIGH : LOW);
  digitalWrite(greenLedPin, greenLedStatus ? HIGH : LOW);
}
// Function for detecting the power loss of the main power supply
void detectPowerLoss() {
  int sensorValue = analogRead(powerDetectPin);
  float voltage = sensorValue * (4.5 / 4095.0);  // Convert analog value to voltage (assuming 4.5V UPS
output)

  if (voltage < voltageThreshold && !powerLost && !inPasswordScreen && !alarmTriggered) {
    // Power loss detected
    powerLost = true;  // Update power status flag
    shouldUpdateScreen = true;
    Blynk.logEvent("power_lost", String("Power lost, battery mode is ON, please check"));
```

```
    return;
  } else if (voltage > voltageThreshold && powerLost && !inPasswordScreen && !alarmTriggered) {
    // Power restored
    powerLost = false;  // Update power status flag
    shouldUpdateScreen = true;
    Blynk.logEvent("power_lost", String("Power recover, battery mode is OFF, please check"));
    return;
  }
}

void updateSirenLEDs() {
  unsigned long currentMillis = millis();
  // Check if it's time to change the LED state
  if (currentMillis - previousMillis >= wait) {
    previousMillis = currentMillis;  // Save the last time the LED state was toggled
    // If alarm is not triggered, do regular blinking
    if (!alarmTriggered) {
      // Toggle LED states based on the current step
      switch (step) {
        case 0:
          digitalWrite(ledPin1, HIGH);
          digitalWrite(ledPin2, HIGH);
          digitalWrite(ledPin3, HIGH);
          digitalWrite(ledPin4, LOW);
          digitalWrite(ledPin5, LOW);
          digitalWrite(ledPin6, LOW);
          break;
        case 1:
          digitalWrite(ledPin1, LOW);
          digitalWrite(ledPin2, LOW);
          digitalWrite(ledPin3, LOW);
          digitalWrite(ledPin4, HIGH);
          digitalWrite(ledPin5, HIGH);
          digitalWrite(ledPin6, HIGH);
          break;
      }
      // Increment the step counter and reset if needed
      step++;
      if (step > 1) {
        step = 0;
      }
    } else {
      // If alarm is triggered, blink all LEDs together
      switch (alarmBlinkStep) {
        case 0:
          digitalWrite(ledPin1, HIGH);
          digitalWrite(ledPin2, HIGH);
          digitalWrite(ledPin3, HIGH);
          digitalWrite(ledPin4, HIGH);
```

```
        digitalWrite(ledPin5, HIGH);
        digitalWrite(ledPin6, HIGH);
        break;
      case 1:
        digitalWrite(ledPin1, LOW);
        digitalWrite(ledPin2, LOW);
        digitalWrite(ledPin3, LOW);
        digitalWrite(ledPin4, LOW);
        digitalWrite(ledPin5, LOW);
        digitalWrite(ledPin6, LOW);
        break;
    }
    // Increment the alarm blink step counter and reset if needed
    alarmBlinkStep++;
    if (alarmBlinkStep > 1) {
      alarmBlinkStep = 0;
    }
    // Reset the regular step counter when alarm is triggered
    step = 0;
   }
  }
}

String clockDisplay() {
  time_t t = now();
  String currentHour = (hour(t) < 10 ? "0" : "") + String(hour(t));
  String currentMinute = (minute(t) < 10 ? "0" : "") + String(minute(t));
  String currentTime = currentHour + ":" + currentMinute;
  String currentMonth = monthShortStr(month(t));
  String currentDay = (day(t) < 10 ? "0" : "") + String(day(t));
  String currentDate = currentDay + "/" + currentMonth + "/" + year(t);
  return currentTime + "    " + currentDate;
}

void readSensorData() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  // Read LDR value
  int ldrValue = analogRead(ldrpin);
   float voltage = ldrValue * (3.3 / 4095.0);                    // Convert analog reading to voltage (12-bit
resolution)
  float ldrResistance = (voltage * 10000) / (3.3 - voltage);  // Calculate LDR resistance (assuming a 10k
resistor)
  //Send LDR value to the App
  Blynk.virtualWrite(V1, humidity);
  Blynk.virtualWrite(V2, temperature);
  //Blynk.virtualWrite(V5, ldrResistance);
  if (humidity > 70) {
    Blynk.logEvent("high_hum", String("High Humidity: ") + humidity);
```

```
  }
  if (temperature > 42) {
    Blynk.logEvent("high_temp", String("High Temperature: ") + temperature);
  }
  if (ldrResistance <= 1200) {
    Blynk.logEvent("high_light", String("Too much light"));
    Blynk.virtualWrite(V3, 4);
    Blynk.setProperty(V3, "color", "#E0D68B");
  }
  if (ldrResistance > 1200 && ldrResistance <= 7000) {
    Blynk.virtualWrite(V3, 3);
    Blynk.setProperty(V3, "color", "#F8E71C");
  }
  if (ldrResistance > 7000 && ldrResistance <= 20000) {
    Blynk.virtualWrite(V3, 2);
    Blynk.setProperty(V3, "color", "#F5A623");
  }
  if (ldrResistance > 20000 && ldrResistance <= 50000) {
    Blynk.virtualWrite(V3, 1);
    Blynk.setProperty(V3, "color", "#BD10E0");
  }
  if (ldrResistance > 50000) {
    Blynk.virtualWrite(V3, 0);
    Blynk.setProperty(V3, "color", "#000000");
  }
}

BLYNK_WRITE(V0) {
  int value = param.asInt();  // Get value as integer
  if (value == 0) {
    pad = "1234";
    disarmSystem();
    inPasswordScreen = false;  // Exit password screen mode
  } else if (value == 1) {
    pad = "1234";
    armSystem();
    inPasswordScreen = false;  // Exit password screen mode
  }
}
void ledsoff() {
  digitalWrite(ledPin1, LOW);
  digitalWrite(ledPin2, LOW);
  digitalWrite(ledPin3, LOW);
  digitalWrite(ledPin4, LOW);
  digitalWrite(ledPin5, LOW);
  digitalWrite(ledPin6, LOW);
  return;
}
```

```
int getUserInput(int col, int row) {
  String input = "";

  lcd.setCursor(col, row);

  while (input.length() < 2) {
    keypressed = myKeypad.getKey();
    if (keypressed && isDigit(keypressed)) {
      input += keypressed;
      lcd.print(keypressed);
      delay(50);  // Debounce delay
    }
  }

  return input.toInt();
}
```