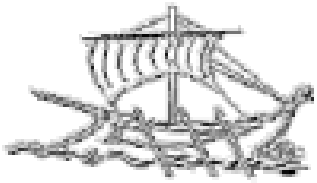


ΑΕΙ ΠΕΙΡΑΙΑ Τ.Τ.
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

΄ΕΞΥΠΝΟ ΣΥΣΤΗΜΑ ΕΛΕΓΧΟΥ ΠΡΟΣΒΑΣΗΣ

Ιωάννης Ατλαμάζογλου

Εισηγητής: κ. Σπυρίδων Ματιάτος, Καθηγητής Εφαρμογών

ΑΘΗΝΑ
ΔΕΚΕΜΒΡΙΟΣ 2017

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΈΞΥΠΝΟ ΣΥΣΤΗΜΑ ΕΛΕΓΧΟΥ ΠΡΟΣΒΑΣΗΣ

Ιωάννης Ατλαμάζογλου

Α.Μ. 43825

Εισηγητής:

Εισηγητής: κ. Σπυρίδων Ματιάτος, Καθηγητής Εφαρμογών

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Ιωάννης Ατλαμάζογλου, του Αποστόλου, με αριθμό μητρώου 43825 φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό του προγραμματισμού .

Την προσπάθειά μου αυτή υποστήριξε ο επιβλέπων καθηγητής μου, τον οποίο θα ήθελα να ευχαριστήσω.

ΠΕΡΙΛΗΨΗ

Το θέμα της Πτυχιακής είναι η κατασκευή ενός έξυπνου συστήματος ελέγχου πρόσβασης εσωτερικής ή εξωτερικής πύλης με βάση την πλατφόρμα Arduino IDE (Intergrated Development Environment ή Ολοκληρωμένο περιβάλλον ανάπτυξης) και το υλικό Arduino Uno. Στόχος είναι ο καλύτερος και έξυπνος έλεγχος του μηχανισμού της κλειδαριάς της πύλης που θέλουμε να ελέγξουμε και η προστασία του χώρου από παραβιάσεις.

Τα χαρακτηριστικά λειτουργίας του είναι τα εξής:

- Διάλογος μεταξύ χρήστη - συστήματος μέσω οθόνης
- Έλεγχος πρόσβασης ή με πληκτρολόγηση κωδικού ή ανέπαφα με Κάρτα Πρόσβασης
- Δυνατότητα αλλαγής κωδικού πρόσβασης
- Δυνατότητα προσθήκης ή κατάργησης Καρτών
- Δυνατότητα αδρανοποίησης μετά από συνεχόμενες λανθασμένες εισαγωγές του κωδικού πρόσβασης
- Ενημέρωση διαχειριστή μέσω SMS σε περιπτώσεις παραβίασης της πύλης
- Δυνατότητα αδρανοποίησης σε συγκεκριμένες ώρες της ημέρας

Λέξεις κλειδιά: Arduino, οθόνη, κωδικός, ανέπαφα, SMS, αδρανοποίηση, πρόσβαση, εξουσιοδότηση, απομακρυσμένος έλεγχος

ABSTRACT

The theme of the diploma thesis is the construction of an intelligent access control system for an internal or external gateway based on the Arduino IDE (Intergrated Development Environment) platform and the Arduino Uno hardware. The aim is to better and cleverly control the mechanism of the gate lock that we want to control and to protect the area from tampering.

The specification of this system are as follows:

- Graphical user interface
- Credentials through PIN or contactlessly
- PIN change ability
- Store or delete contactless cards
- After three failed attempts, system is disabled
- In case of infringement, an SMS message is sent to the administrator
- Disable on not working hours

Keywords: Arduino, SMS, GUI, password, contactless, disable, access, authorization, remote control, PIN

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Η κάτωθι πτυχιακή αφορά την μελέτη και την κατασκευή έξυπνης ηλεκτρονικής κλειδαριάς.

Παρουσιάζονται οι λειτουργίες των μικροελεγκών, τα πλεονεκτήματα, τα μειονεκτήματα και οι διαφορές τους με τους μικροεπεξεργαστές. Επίσης γίνεται αναφορά στα είδη των μικροελεγκτών σε βιομηχανικό και εμπορικό επίπεδο.

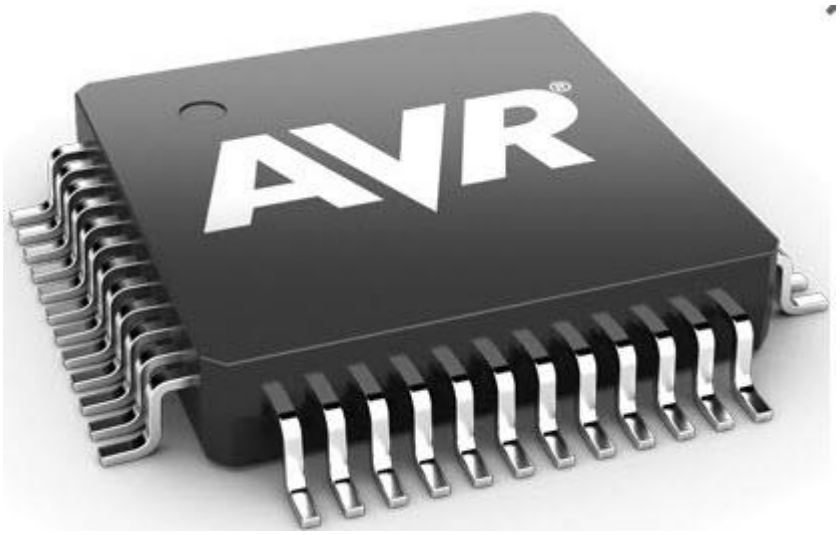
Επίσης γίνεται αναφορά στην πλατφόρμα Arduino IDE (Intergrated Development Environment ή Ολοκληρωμένο περιβάλλον ανάπτυξης), η οποία παρέχει στο υλικό Arduino Uno τον προγραμματισμό του, καθώς επίσης και στην γλώσσα με την οποία γράφονται τα προγράμματα.

Τέλος παρουσιάζονται αναλυτικά όλα τα εξαρτήματα που χρησιμοποιήθηκαν και ο τρόπος λειτουργίας του συστήματος. Επίσης παρουσιάζονται το διάγραμμα συνδεσμολογίας των εξαρτημάτων καθώς και ο τρόπος που επικοινωνούν μεταξύ τους. Ακόμα γίνεται παρουσίαση του διαγράμματος ροής του προγράμματος σε συνοπτικό και αναλυτικό επίπεδο.

1 ΟΙ ΜΙΚΡΟΕΛΕΓΚΤΕΣ

1.1 Τι είναι οι Μικροελεγκτές

Ένας μικροελεγκτής είναι ένα μικρό υπολογιστικό κύκλωμα, σχεδιασμένο σε ένα και μόνο ολοκληρωμένο κύκλωμα. Όπως κάθε υπολογιστικό κύκλωμα, περιέχει κεντρική μονάδα επεξεργασίας, έναν αριθμό καταχωρητών, κυκλώματα μνήμης και κυκλώματα ελέγχου περιφερειακών συσκευών. Κάθε μικροελεγκτής είναι λοιπόν ικανός να ανταλλάξει σήματα με το εξωτερικό περιβάλλον, να εκτελέσει πράξεις ανάμεσα σε μεταβλητές και να καταχωρήσει κάποιες τιμές στη μνήμη τυχαίας προσπέλασης που διαθέτει.



Εικόνα 1.1 – Εξωτερική όψη μικροελεγκτή

1.2 Χαρακτηριστικά Μικροελεγκτών

Κάθε μικροελεγκτής περιέχει μέσα σε ένα και μοναδικό ολοκληρωμένο κύκλωμα τα παρακάτω στοιχεία:

- έναν αριθμό από καταχωρητές ειδικού σκοπού (συσσωρευτή, καταχωρητή κατάστασης, μετρητή προγράμματος, καταχωρητή εντολών, καταχωρητή δείκτη).
- εσωτερικούς χρονιστές - απαριθμητές.

- αριθμητική και λογική μονάδα (ALU)
- μονάδα αποκωδικοποίησης εντολών.

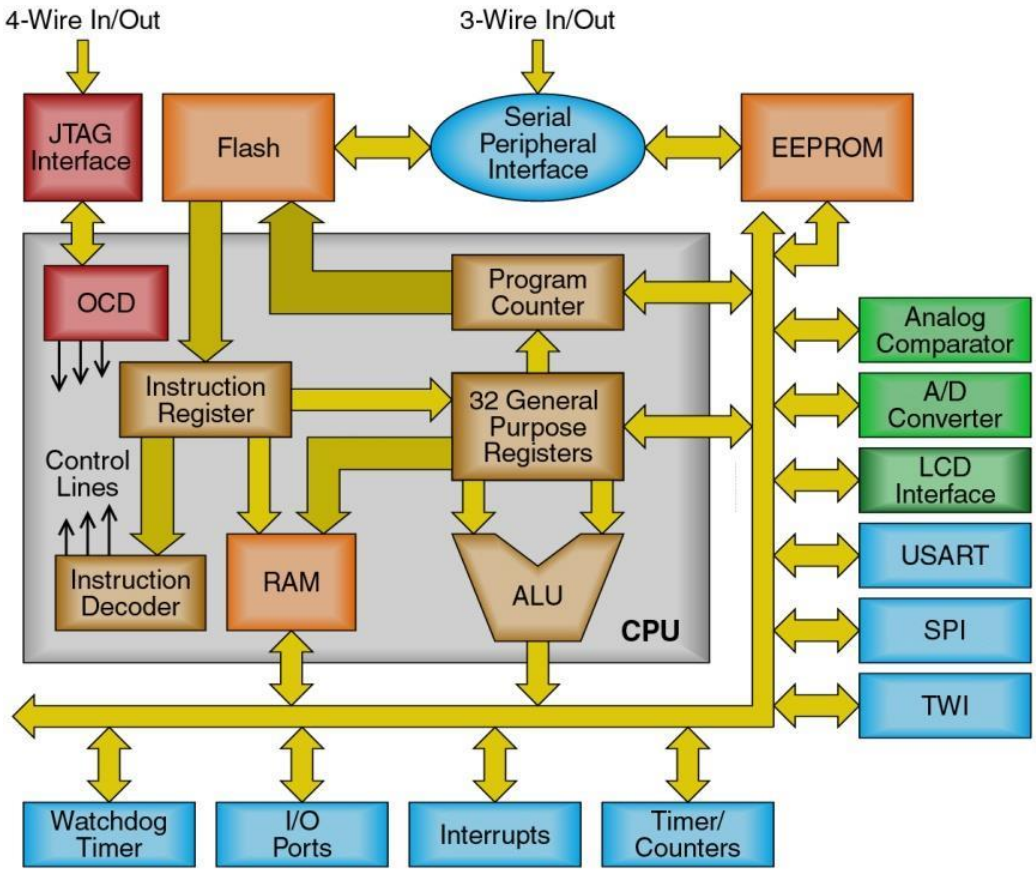
Βασικά στοιχεία ενός μικροελεγκτή αποτελούν:

- η μνήμη προγράμματος (ROM ή EPROM) και
- η μνήμη καταχωρητών / μεταβλητών (RAM).

Στους μικροελεγκτές διακρίνουμε επίσης τα κυκλώματα χρονισμού και ελέγχου.

Τέλος, βασικά μέρη ενός μικροελεγκτή είναι:

- παράλληλες θύρες εισόδου/εξόδου
- άλλα περιφερειακά κυκλώματα (A/D μετατροπείς κλπ.)



Εικόνα 1.2 – Εσωτερική αρχιτεκτονική μικροελεγκτή

1.3 Πρόσθετες λειτουργίες

Μέσα από τις θύρες I/O ένας μικροελεγκτής μπορεί να δέχεται σήματα εισόδου με τη μορφή λογικών ψηφιακών καταστάσεων, χαρακτήρες ή bytes δεδομένων με την τεχνική της ασύγχρονης ή της σύγχρονης σειριακής επικοινωνίας, σήματα διακοπών, ή σε ορισμένες περιπτώσεις και αναλογικά σήματα, τα οποία στη συνέχεια μετατρέπονται σε ψηφιακά. Επίσης μπορεί να αποστέλλει σήματα σε άλλες συσκευές μέσα από θύρες εξόδου, να οδηγεί ηλεκτρονόμους, διόδους LED και άλλα κατάλληλα κυκλώματα, που συνήθως περιλαμβάνονται σε κάθε μορφής αυτοματισμό.

1.4 Πλεονεκτήματα μικροελεγκτών

Στους σύγχρονους μικροεπεξεργαστές για μη ενσωματωμένα συστήματα (π.χ. τους μικροεπεξεργαστές των προσωπικών υπολογιστών), δίνεται έμφαση στην υπολογιστική ισχύ. Η ευελιξία ανάπτυξης διαφορετικών εφαρμογών είναι μεγάλη, καθώς η λειτουργικότητα του τελικού συστήματος καθορίζεται από τα εξωτερικά περιφερειακά τα οποία διασυνδέονται με την κεντρική μονάδα (μικροεπεξεργαστή), η οποία δεν είναι εξειδικευμένη. Αντίθετα, στους μικροεπεξεργαστές για ενσωματωμένα συστήματα (μικροελεγκτές), οι οποίοι έχουν μικρότερες ή και μηδαμινές δυνατότητες συνεργασίας με εξωτερικά περιφερειακά, αυτού του είδους, η ευελιξία είναι περιορισμένη, καθώς και η υπολογιστική ισχύς. Οι μικροελεγκτές δίνουν έμφαση στο μικρό αριθμό ολοκληρωμένων κυκλωμάτων που απαιτείται για τη λειτουργία μιας συσκευής, το χαμηλό κόστος και την εξειδίκευση.

Αναλυτικά, τα πλεονεκτήματα των μικροελεγκτών σε σχέση με τους μικροεπεξεργαστές είναι:

- Αυτονομία, μέσω της ενσωμάτωσης σύνθετων περιφερειακών υποσυστημάτων όπως μνήμες και θύρες επικοινωνίας. Έτσι πολλοί μικροελεγκτές δεν χρειάζονται κανένα άλλο ολοκληρωμένο κύκλωμα για να λειτουργήσουν.

- Η ενσωμάτωση περιφερειακών σημαίνει ευκολότερη υλοποίηση εφαρμογών λόγω των απλούστερων διασυνδέσεων. Επίσης, οδηγεί σε χαμηλότερη κατανάλωση ισχύος, μεγιστοποιώντας τη φορητότητα και ελαχιστοποιεί το κόστος της συσκευής στην οποία ενσωματώνεται ο μικροελεγκτής.
- Χαμηλό κόστος.
- Μεγαλύτερη αξιοπιστία, και πάλι λόγω των λιγότερων διασυνδέσεων.
- Μειωμένες εκπομπές ηλεκτρομαγνητικών παρεμβολών και μειωμένη ευαισθησία σε αντίστοιχες παρεμβολές από άλλες ηλεκτρικές και ηλεκτρονικές συσκευές. Το πλεονέκτημα αυτό προκύπτει από το μικρότερο αριθμό και μήκος εξωτερικών διασυνδέσεων καθώς και τις χαμηλότερες ταχύτητες λειτουργίας.
- Περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους-εξόδους (για δεδομένο μέγεθος ολοκληρωμένου κυκλώματος), λόγω της μη δέσμευσής τους για τη σύνδεση εξωτερικών περιφερειακών.
- Μικρό μέγεθος συνολικού υπολογιστικού συστήματος.[9]

1.5 Διαφορές από τον μικροεπεξεργαστή

Σε τι διαφέρει ένας μικροελεγκτής από έναν συνηθισμένο μικροεπεξεργαστή;

Ο μικροελεγκτής είναι ένα μικρό αυτόνομο υπολογιστικό σύστημα, προγραμματισμένο να εκτελεί μία συγκεκριμένη λογική ακολουθία εντολών, οι οποίες έχουν καταχωρηθεί στην προγραμματιζόμενη μόνιμη μνήμη του. Κάθε φορά που θα επανεκκινείται ο μικροελεγκτής, θα εκτελεί την ίδια λογική. Θα ανακαλεί τα δεδομένα, θα τα επεξεργάζεται και με βάση τα αποτελέσματα της επεξεργασίας θα ελέγχει το περιβάλλον του. Πρόκειται, δηλαδή, για σύστημα ειδικού σκοπού, αφιερωμένο στον έλεγχο και την εξυπηρέτηση ενός συγκεκριμένου αυτοματισμού. Αντίθετα, ένας μικροεπεξεργαστής μετά την εκκίνησή του δεν είναι από μόνος του σε θέση να εκτελέσει κάποια λογική ακολουθία. Αν και μπορεί να συνδεθεί με μνήμες RAM και ROM, αυτές αποτελούν

ξεχωριστές μονάδες, που συνήθως δεν ολοκληρώνονται μέσα στον ίδιο τον μικροεπεξεργαστή.[9]

1.6 Διαδεδομένες κατηγορίες μικροελεγκτών

Λόγω του ισχυρότατου ανταγωνισμού αλλά και της τάσης ενσωμάτωσης των μικροελεγκτών σε κάθε ηλεκτρική και ηλεκτρονική συσκευή, η βιομηχανία μικροελεγκτών έχει καταλήξει στην παραγωγή ανταγωνιστικών μοντέλων μαζικής παραγωγής καθώς και μικροελεγκτών για πιο εξειδικευμένες εφαρμογές. Έτσι διακρίνονται κυρίως οι εξής κατηγορίες:

- Μικροελεγκτές (καμμιά φορά 4-bit αλλά συνήθως 8-bit) πολύ χαμηλού κόστους, γενικής χρήσης, με πολύ μικρό αριθμό ακροδεκτών (ακόμη και λιγότερους από 8). Σχεδιάζονται με έμφαση στη χαμηλή κατανάλωση ισχύος και την αυτάρκεια, ώστε να χρειάζονται ελάχιστα ή και καθόλου εξωτερικά εξαρτήματα και να μη μπορεί να αντιγραφεί εύκολα το εσωτερικό λογισμικό τους. Απουσιάζει η δυνατότητα επέκτασης της μνήμης τους. Μερικά μοντέλα είναι ευρέως γνωστά στους ερασιτέχνες ηλεκτρονικούς, όπως πχ οι περισσότεροι μικροελεγκτές των σειρών PIC (Microchip), AVR (Atmel) και 8051 (Intel, Atmel, Dallas κ.α.).
- Μικροελεγκτές (συνήθως 8-bit αλλά και 16 ή 32-bit) χαμηλού κόστους, γενικής χρήσης, με μέτριο έως σχετικά μεγάλο αριθμό ακροδεκτών. Διαθέτουν μεγάλο αριθμό κοινών περιφερειακών, όπως θύρες UART (universal asynchronous receiver/transmitter) ή (γενικός ασύγχρονος δέκτης / πομπός), I^2C (Inter-Integrated Circuit) ή (Διασυνδεδεμένο Κύκλωμα), SPI (Serial Peripheral Interface Bus) ή (Σειριακός περιφερειακός δίαυλος διασύνδεσης) , μετατροπείς αναλογικού σε ψηφιακό και ψηφιακού σε αναλογικό. Στους κατασκευαστές της Άπω Ανατολής (Ιαπωνία, Κορέα), συνηθίζεται η ενσωμάτωση ελεγκτών οθόνης υγρών κρυστάλλων και πληκτρολογίου. Μερικές φορές παρέχουν δυνατότητα εξωτερικής επέκτασης της μνήμης τους.

- Μικροελεγκτές (κυρίως 32-bit) μέσου κόστους, γενικής χρήσης, με μεγάλο αριθμό ακροδεκτών. Χαρακτηρίζονται από έμφαση στην ταχύτητα εκτέλεσης εντολών, υψηλή αυτάρκεια περιφερειακών και μεγάλες δυνατότητες εσωτερικής ή εξωτερικής μνήμης προγράμματος (FLASH) και RAM. Στο χώρο αυτό έχουν ισχυρή παρουσία οι αρχιτεκτονικές με υψηλή μεταφερσιμότητα λογισμικού από τον ένα στον άλλο κατασκευαστή. Π.χ. μεταξύ των μικροελεγκτών τύπου ARM (Advanced RISC Machine) και τα αρχικά σημαίνουν προχωρημένη μηχανή RISC ή MIPS architecture, το σύνολο των βασικών εντολών που αναγνωρίζει η ALU είναι ακριβώς το ίδιο, μειώνοντας έτσι τις μεγάλες αλλαγές στο λογισμικό, όταν στο μέλλον ο πελάτης υιοθετήσει ένα μικροελεγκτή άλλου κατασκευαστή (αρκεί, φυσικά, να υποστηρίζει κι αυτός το σύνολο εντολών ARM ή MIPS, αντίστοιχα).
- Μικροελεγκτές εξειδικευμένων εφαρμογών, οι οποίοι ενσωματώνουν συνήθως κάποιο εξειδικευμένο πρωτόκολλο επικοινωνίας το οποίο υλοποιείται πάντοτε σε hardware. Τέτοιοι μικροελεγκτές χρησιμοποιούνται σε τηλεπικοινωνιακές συσκευές όπως τα μόντεμ.

Η μεγάλη μερίδα πωλήσεων των μικροελεγκτών εξακολουθεί να αφορά αυτούς των 8-bit, καθώς είναι η κατηγορία με το χαμηλότερο κόστος και το μικρότερο μέγεθος λογισμικού για το ίδιο αποτέλεσμα, ιδίως επειδή οι σύγχρονες οικογένειες μικροελεγκτών 8-bit έχουν πολύ βελτιωμένες επιδόσεις σε σχέση με το παρελθόν.[8]

1.7 Γλώσσες προγραμματισμού και εργαλεία ανάπτυξης

Οι μικροελεγκτές χαρακτηρίζονται από ένα περιορισμένο ρεπερτόριο εντολών, οι οποίες μπορούν να γραφούν σε συμβολική μορφή (assembly), με τη βοήθεια μνημονικών ονομάτων. Στους μικροελεγκτές μεσαίας τάξης (midrange), το μήκος της εντολής σε γλώσσα μηχανής είναι 14 bits, τα οποία καταχωρούνται στη μνήμη προγράμματος, τύπου EEPROM (electrically erasable programmable read-only memory ή ηλεκτρικά απαλείψιμη προγραμματίσιμη μνήμη μόνο για ανάγνωση).

Η επιτυχία μιας οικογένειας μικροελεγκτών καθορίζεται σε μεγάλο βαθμό από τη διαθεσιμότητα και την ευχρηστία των σχετικών εργαλείων ανάπτυξης, όπως

μεταφραστές από γλώσσες υψηλού επιπέδου σε γλώσσα κατανοητή από τον μικροελεγκτή (assembly), προγραμματιστές της εσωτερικής μνήμης και εργαλεία εκσφαλμάτωσης (debuggers). Στους μικροελεγκτές, τα εργαλεία αυτά δεν αποτελούνται μόνον από λογισμικό, καθώς δεν υπάρχει τυποποιημένος τρόπος επικοινωνίας με αυτούς. Έτσι διατίθενται προγραμματιστές της εσωτερικής μνήμης (συνήθως μέσω θύρας JTAG (Joint Test Action Group) ή (Κοινή ομάδα δράσης δοκιμής) ή USB) καθώς και έτοιμες πλακέτες (evaluation boards) με ψηφιακές ή και αναλογικές εξόδους. Οι πλακέτες αυτές έχουν ως κεντρική τους μονάδα τον εκάστοτε μικροελεγκτή, δυνατότητα εύκολου προγραμματισμού και συνήθως συνοδεύονται από λογισμικό ανάπτυξης εφαρμογών με έτοιμα παραδείγματα. Ετσι ο χρήστης μπορεί να δοκιμάσει τις περισσότερες δυνατότητες του μικροελεγκτή πριν καν σχεδιάσει τη δική του πλακέτα. Στον τομέα των εργαλείων ανάπτυξης, δραστηριοποιούνται όχι μόνο οι ίδιοι οι κατασκευαστές μικροελεγκτών αλλά και εξειδικευμένες εταιρείες.

Η πιο διαδεδομένη γλώσσα προγραμματισμού των μικροελεγκτών είναι η C, η C++ και οι παραλλαγές τους. Σε τμήματα του λογισμικού όπου απαιτείται ταχύτητα ή μικρό μέγεθος χρησιμοποιούμενης μνήμης, μπορεί να χρησιμοποιείται η Assembly. Όμως οι μεγαλύτερες απαιτήσεις σε λειτουργικότητα και η ευκολία προγραμματισμού της C έναντι της assembly, σε συνδυασμό με την επάρκεια μνήμης των σύγχρονων μικροελεγκτών, έχουν γενικά εκτοπίσει την Assembly από τις περισσότερες εφαρμογές.[9]

1.8 Παραδείγματα εφαρμογών με μικροελεγκτές

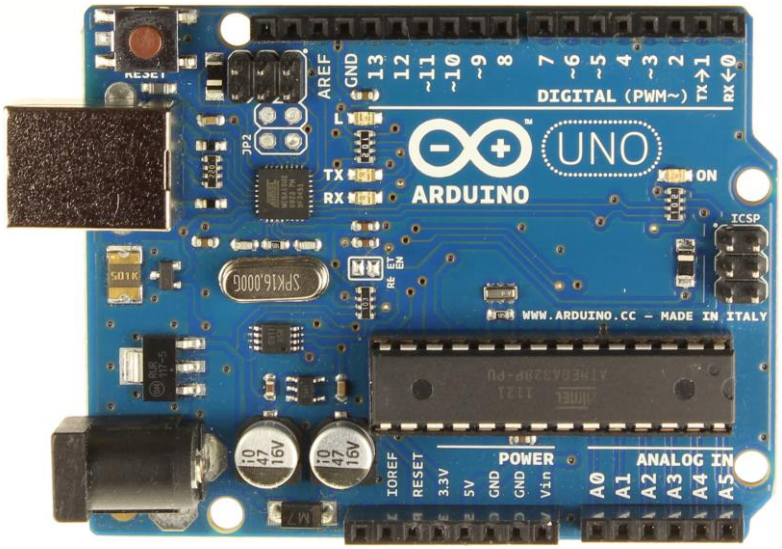
Ο μικροελεγκτής λειτουργεί ως ψηφιακός ελεγκτής συλλέγοντας δεδομένα από τους αισθητήρες και ενεργοποιώντας εξωτερικά κυκλώματα, όπως κινητήρες, ηλεκτρονόμους, κυκλώματα ισχύος κ.λ.π. Αντίστοιχα με όσα περιγράψαμε, ο μικροελεγκτής λαμβάνει ως είσοδο την τρέχουσα κατάσταση του συστήματος που ελέγχει, συνήθως με τη μορφή ενός αναλογικού σήματος, που προέρχεται από έναν ή περισσότερους αισθητήρες. Το σήμα αυτό ψηφιοποιείται, με τη βοήθεια ενός μετατροπέα αναλογικού σήματος σε ψηφιακό και συγκρίνεται με μια τιμή αναφοράς, που περιγράφει την επιθυμητή κατάσταση του συστήματος. Με βάση την απόκλιση της τρέχουσας κατάστασης από την επιθυμητή κατάσταση αναφοράς, λαμβάνεται μια απόφαση και παράγεται ένα σήμα «οδήγησης», που

επιδρά πάνω στο ελεγχόμενο σύστημα και μεταβάλλει την κατάστασή του. Για το σκοπό αυτό, ο μικροελεγκτής διαθέτει κατάλληλες θύρες εισόδου/εξόδου, με τις οποίες διασυνδέεται με τον εξωτερικό κόσμο. Επίσης διαθέτει περιφεριεακά, όπως κυκλώματα παραγωγής παλμών μεταβαλλόμενου εύρους (PWM) ή (Pulse-width modulation) ή (Διαμόρφωση εύρους παλμού) για οδήγηση κινητήρων, μετατροπής αναλογικού σήματος σε ψηφιακό (ADC), θύρες επικοινωνίας κ.ά. Ο ψηφιακός ελεγκτής είναι συνήθως σύστημα πραγματικού χρόνου.[6]

2 ΤΟ ARDUINO UNO

2.1 Τι είναι το Arduino uno

Το Arduino uno είναι ένας μικροελεγκτής μονής πλακέτας, δηλαδή μια απλή μητρική πλακέτα ανοικτού κώδικα με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για τη γλώσσα προγραμματισμού C++ και ένα σύνολο από βιβλιοθήκες, υλοποιημένες επίσης στην C++). Οι περισσότερες εκδόσεις του Arduino μπορούν να αγοραστούν προ-συναρμολογημένες ενώ το διάγραμμα και οι πληροφορίες για το υλικό είναι ελεύθερα διαθέσιμα για αυτούς που θέλουν να συναρμολογήσουν το Arduino μόνοι τους.[1]



Εικόνα 2.1 – Φωτογραφία του υλικού Arduino Uno

2.2 Ανάλυση Υλικού

Μία πλακέτα Arduino uno αποτελείται από ένα μικροελεγκτή Atmel AVR (ATmega328 και ATmega168 στις νεότερες εκδόσεις, ATmega8 στις παλαιότερες) και συμπληρωματικά εξαρτήματα για την διευκόλυνση του χρήστη στον

προγραμματισμό και την ενσωμάτωσή του σε άλλα κυκλώματα. Όλες οι πλακέτες περιλαμβάνουν ένα γραμμικό ρυθμιστή τάσης 5V και έναν κρυσταλλικό ταλαντωτή 16MHz (ή κεραμικό αντηχητή σε κάποιες παραλλαγές). Ο μικροελεγκτής είναι από κατασκευής προγραμματισμένος με τον *bootloader (φορτωτής του πυρήνα)*, έτσι ώστε να είναι μπορεί να προγραμματιστεί .

Σε εννοιολογικό επίπεδο, στην χρήση του Arduino software stack, όλα τα boards (πλακέτες) προγραμματίζονται με μία RS-232 σειριακή σύνδεση, αλλά ο τρόπος που επιτυγχάνεται αυτό διαφέρει σε κάθε έκδοση υλικού. Οι σειριακές πλάκες Arduino περιέχουν ένα απλό level shifter (μετατοπιστής επιπέδου) κύκλωμα για να μετατρέπει μεταξύ σήματος επιπέδου RS-232 και TTL (Transistor–transistor logic). Τα σύγχρονα Arduino προγραμματίζονται μέσω USB και αυτό καθίσταται δυνατό μέσω της εφαρμογής προσαρμοστικών chip USB-to-Serial. Κάποιες παραλλαγές, όπως το Arduino mini και το ανεπίσημο Boarduino, χρησιμοποιούν ένα αφαιρούμενο USB-to-Serial καλώδιο ή board, Bluetooth ή άλλες μεθόδους. (Όταν χρησιμοποιείται με παραδοσιακά εργαλεία μικροελεγκτής αντί για το Arduino IDE (Ολοκληρωμένο περιβάλλον ανάπτυξης ή Intergrated Development Environment) , χρησιμοποιείται πρότυπος προγραμματισμός AVR ISP).

Ο πίνακας Arduino εκθέτει τα περισσότερα microcontroller I/O (εισόδων και εξόδων) ακροδέκτες για χρήση από άλλα κυκλώματα. Τα Diecimila, Duemilanove και το τρέχον Uno παρέχουν 14 ψηφιακούς I/O ακροδέκτες, έξι από τους οποίους μπορούν να παράγουν διαμορφωμένα σήματα ως προς το πλάτος του παλμού, και έξι αναλογικά δεδομένα. Αυτοί οι ακροδέκτες βρίσκονται στην μπροστινή μεριά του υλικού. [2]

2.3 Λογισμικό Πλατφόρμας

Το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) του Arduino είναι μία εφαρμογή γραμμένη σε Java, που λειτουργεί σε πολλές πλατφόρμες και προέρχεται από το IDE για τη γλώσσα προγραμματισμού Processing και το σχέδιο Wiring. Έχει σχεδιαστεί για να μυήσει τους καλλιτέχνες στον προγραμματισμό, καθώς και τους νέους που δεν είναι εξοικειωμένοι με την ανάπτυξη λογισμικού. Περιλαμβάνει ένα πρόγραμμα επεξεργασίας κώδικα με χαρακτηριστικά όπως είναι η επισήμανση σύνταξης και ο συνδυασμός αγκύλων και είναι επίσης σε θέση να μεταγλωττίζει

και να φορτώνει προγράμματα στην πλακέτα με ένα μόνο κλικ. Ένα πρόγραμμα ή κώδικας που γράφτηκε για Arduino ονομάζεται *σκίτσο* (sketch).

Εικόνα 2.2 – Φωτογραφία από το Arduino IDE

Τα Arduino προγράμματα είναι γραμμένα σε C ή C++. Το Arduino IDE έρχεται με μια βιβλιοθήκη λογισμικού που ονομάζεται "Wiring", γεγονός που καθιστά πολλές κοινές λειτουργίες εισόδου/εξόδου πολύ πιο εύκολες. Οι χρήστες πρέπει μόνο να ορίσουν δύο λειτουργίες για να κάνουν ένα πρόγραμμα κυκλικής εκτέλεσης:

- setup() :μία συνάρτηση που τρέχει μία φορά στην αρχή του προγράμματος η οποία αρχικοποιεί τις ρυθμίσεις
- loop() :μία συνάρτηση που καλείται συνέχεια μέχρι η πλακέτα να απενεργοποιηθεί

Έτσι, η βασική λειτουργία του Arduino είναι ότι τρέχει η συνάρτηση setup() μία φορά στην αρχή και ακολούθως η loop() ξανά και ξανά μέχρι να το κλείσουμε (να μην τροφοδοτείται με ρεύμα) ή να πατήσουμε το πλήκτρο reset. Στην περίπτωση του Reset ξανατρέχει η συνάρτηση setup() μία φορά και ακολούθως η loop() ξανά και ξανά, όπως δηλαδή ακριβώς και όταν αρχικά ενεργοποιείται με ρεύμα ο μικροελεγκτής. Ένα χαρακτηριστικό των περισσότερων πλακετών Arduino είναι ότι έχουν ένα LED και μία αντίσταση φορτίου που συνδέονται μεταξύ του ακροδέκτη 13 και της γείωσης, ένα βολικό χαρακτηριστικό για πολλά απλά τεστ.[3]

2.4 Περιφερειακά

Η κύρια λειτουργία του μικροελεγκτή βασίζεται στο να ελέγχει τις θύρες που διαθέτει και είτε να δίνει ρεύμα είτε να παίρνει ρεύμα από αυτές. Στην αρχικοποίηση κάθε προγράμματος (μέσα στη συνάρτηση setup) θα χρειαστεί να χαρακτηρίσουμε τους ακροδέκτες που χρησιμοποιούμε ως είσοδο ή ως έξοδο.

Η συνάρτηση pinMode(Pin, Mode) χρησιμοποιείται με το όνομά της και ορίσματα όπως τον αριθμό ακροδέκτη και την κατάσταση λειτουργίας που χαρακτηρίζεται με τη λέξη INPUT (είσοδος) ή OUTPUT(έξοδος).

Και τα 14 pins του Arduino μπορούν να λειτουργούν ως ψηφιακές έξοδοι, δηλαδή δίνουν έξοδο 0 ή 5V. Αυτό γίνεται με χρήση της συνάρτησης «digitalWrite(Pin, Value)», όπου το όρισμα «Pin» αναφέρεται στον ακροδέκτη για τον οποίο θα δώσουμε τάση εξόδου, ενώ η τάση εξόδου μπορεί να είναι 0 V ή 5 V, οι οποίες αναπαρίστανται με προκαθορισμένες τιμές στην παράμετρο «value»:

- LOW : θα δώσει 0 V στην έξοδο
- HIGH : θα δώσει 5 V στην έξοδο

Επίσης και τα 14 ψηφιακά pins του Arduino χρησιμοποιούνται ως ψηφιακές είσοδοι, δηλαδή να “διαβάσουν” ως είσοδο τάση με τιμή είτε 0 είτε 5V. Αυτό γίνεται με χρήση της συνάρτησης «digitalRead(Pin)», όπου το όρισμα Pin αναφέρεται στον ακροδέκτη για τον οποίο θα πάρουμε είσοδο, ενώ η συνάρτηση επιστρέφει με το όνομά της την τιμή εισόδου. Η τάση εισόδου μπορεί να είναι 0V ή 5V, οι οποίες αναπαρίστανται με προκαθορισμένες τιμές στην τιμή που διαβάζουμε:

- LOW : όταν λάβει τάση 0 V στην είσοδο
- HIGH : όταν λάβει τάση 5 V στην είσοδο

Στο πρόγραμμά μας μπορούμε να θέσουμε τον μικροελεγκτή σε κατάσταση αναμονής για ένα συγκριμένο χρονικό διάστημα. Αυτό το επιτυγχάνουμε με χρήση της συνάρτησης delay(time) όπου στη θέση time δίνουμε το χρόνο σε ms (1/1000 sec). Η εντολή «delay(time)» σημαίνει ότι σταματά στο σημείο αυτό η εκτέλεση του προγράμματός μας για το χρόνο time και ύστερα συνεχίζει να εκτελείται.

Ένα τυπικό πρώτο πρόγραμμα για έναν μικροελεγκτή είναι να αναβοσβήνει απλά ένα LED. Στο περιβάλλον του Arduino IDE, ο χρήστης μπορεί να γράψει ένα πρόγραμμα σαν αυτό: [3]

```
#define LED_PIN 13

void setup () {
 pinMode (LED_PIN, OUTPUT); // enable pin 13 for digital output
}

void loop () {
 digitalWrite (LED_PIN, HIGH); // turn on the LED
 delay (1000); // wait one second (1000 milliseconds)
 digitalWrite (LED_PIN, LOW); // turn off the LED
 delay (1000); // wait one second
}
```

3 ΤΟ ΕΞΥΠΝΟ ΣΥΣΤΗΜΑ ΕΛΕΓΧΟΥ ΠΡΟΣΒΑΣΗΣ

3.1 Περιγραφή

Παρακάτω γίνεται ανάλυση της λειτουργίας του συστήματος. Αρχικά, μόλις ξεκινήσει για πρώτη φορά, το σύστημα εκετελεί την διαδικασία με την οποία αρχικοποιούνται όλοι οι παράμετροι, οι εξωτερικές συσκευές και για πρώτη φορά γίνεται διαμόρφωση της ημερομηνίας και ώρας αλλά και του ωραρίου λειτουργίας.

΄Υστερα το σύστημα περνάει στην δεύτερη φάση κατά την οποία λειτουργεί σε κατάσταση αναμονής εισόδου χρήστη εμφανίζοντας την αρχική οθόνη. Ο απλός χρήστης και ταυτόχρονα διαχειριστής έχει τις εξείς δυνατότητες:

- Να εισάγει το κωδικό πρόσβασης ώστε να ανοίξει η πύλη

Η εισαγωγή του κωδικού γίνεται από το πληκτρολόγιο. Με την εισαγωγή κάθε αριθμού , ο χρήστης ανατροφοδοτείται όταν ακούσει την ηλεκτρονική και μικρής διάρκειας ηχητική ειδοποίηση. Επίσης με κάθε εισαγωγή αριθμού στην οθόνη εμφανίζεται ο χαρακτήρας της δίεσης, ο οποίος δηλώνει ότι ο αριθμός που έχει εισάγει ο χρήστης έχει επεξεργαστεί και αποθηκευτεί στην μνήμη. Ο χρήστης μπορεί ανά πάσα στιγμή να γνωρίζει πόσα ψηφία έχει εισάγει καθώς όπως αναφέρθηκε προηγουμένως με κάθε ψηφίο που εισάγει ο χρήστης, το σύμβολο της δίεσης εμφανίζεται στην οθόνη. Επίσης πολύ σηματικό να αναφερθεί είναι ότι το σύστημα παρέχει μια ασφαλιστική δικλείδα που εξασφαλίζει ότι αν ο κωδικός πρόσβασης έχει εισαχθεί τρείς συνεχόμενες φορές λανθασμένος, τότε το σύστημα τίθεται σε κατάσταση αδράνειας για ένα λεπτό. Αυτό έχει ως στόχο να δυσκολέψει την παραβίαση του συστήματος, με την εισαγωγή τυχαίων ψηφίων.

- Να εισάγει την κάρτα πρόσβασης ώστε να ανοίξει η πύλη

Ο χρήστης χρειάζεται να ‘’περάσει’’ ανέπαφα την κάρτα του από τον αναγνώστη αποφεύγοντας τις απότομες κινήσεις της κάρτας πάνω από τον αναγνώστη για την αποφυγή σφάλματος.

Μόλις γίνει η ταυτοποίηση, τότε σε περίπτωση που η κάρτα την οποία εισήγαγε ο χρήστης είναι αποθηκευμένη στη μνήμη του συστήματος ή ο κωδικός που

εισήγαγε ταυτίζετα με τον κωδικό που έχει οριστεί, η πύλη ξεκλειδώνει και μπορεί να ανοίξει. Αν ο χρήστης δεν ανοίξει την πύλη, τότε μετά από πέντε δευτερόλεπτα η πύλη ξανακλειδώνει. Αν ο χρήστης ανοίξει την πύλη τότε το σύστημα τίθεται σε κατάσταση αδράνειας μέχρι η πύλη να κλείσει.

Όταν το σύστημα ανιχνεύσει ότι η πύλη είναι ανοιχτή αλλά χωρίς να έχει δοθεί εξουσιοδότηση από το ίδιο το σύστημα τότε πρέπει να ενημερώσει τον έναν εκ των διαχειριστών ότι η πύλη έχει παραβιαστεί. Για τον σκοπό αυτό, το σύστημα στέλνει μήνημα μέσω SMS κάθε δέκα δευτερόλεπτα. Παράλληλα στην περίπτωση που η πύλη κλείσει τότε εμφανίζεται η αρχική οθόνη και το σύστημα συνεχίζει την λειτουργία του κανονικά.

Πέρα από τις απλές λειτουργίες ο χρήστης έχει τα παρακάτω διακαιώματα:

- Μπορεί να προσθέσει ή να διαγράψει νέες κάρτες

Ο χρήστης αφού πατήσει το πλήκτρο «A» ή το πλήκτρο «D» αντίστοιχα, τότε πρέπει να περάσει την κάρτα από τον αναγνώστη και να περιμένει να εμφανιστεί μήνυμα στην οθόνη. Σε περίπτωση επιτυχίας, το σύστημα θα ειδοποιήσει τον χρήστη ότι η κάρτα αποθηκεύτηκε ή διαγράφηκε αντίστοιχα. Σε περίπτωση αποτυχίας θα εμφανιστεί η αρχική οθόνη.

- Μπορεί να αλλάξει τον κωδικό πρόσβασης

Ο χρήστης αφού πατήσει το πλήκτρο «C» τότε το σύστημα ζητάει την εισαγωγή του κωδικού πρόσβασης. Έτσι η αλλαγή του κωδικού έχει ενα επίπεδο ασφαλείας. Ο χρήστης για να θέσει νέο κωδικό το σύστημα απαιτεί την εισαγωγή του τρέχοντος κωδικού για να συνεχίσει, αλλιώς θα εμφανιστεί η αρχική οθόνη. Σε περίπτωση επιτυχούς αλλαγής του κωδικού τότε το σύστημα θα ειδοποιήσει τον χρήστη ότι ο κωδικός άλλαξε και θα επιστρέψει στην αρχική οθόνη.

- Μπορεί να τροποποιήσει την ημερομηνία και το ωράριο λειτουργίας

Η αλλαγή της ημερομηνίας, της ώρα και του ωραρίου λειτουργίας γίνεται αν ο χρήστης πατήσει το πλήκτρο «B». Τότε το σύστημα πραγματοποιεί διάλογο με το χρήστη για την εισαγωγή της ώρας, των λεπτών, της μέρας , του μήνα , του έτους καθώς και του χρονικού εύρους μέσα στην μέρα στο οποίο το σύστημα θα είναι σε λειτουργία. Το σύστημα ελέγχει συνεχώς αν βρίσκεται εκτός ωραρίου λειτουργίας.

Σε περίπτωση όπου το σύστημα είναι εκτός ωραρίου, τότε τίθεται σε κατάσταση πλήρους αδράνειας, μέχρι να εισέλθει ξανά στον ωράριο.

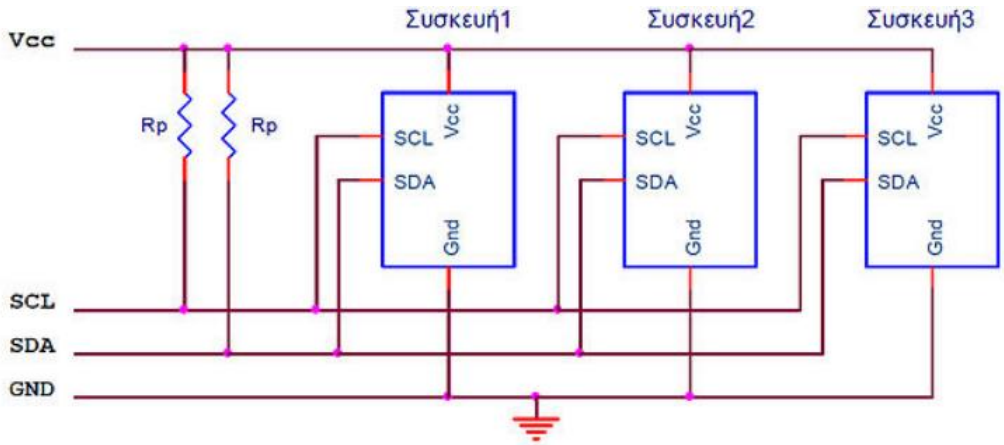
3.2 Ανάλυση των Εξαρτημάτων

Για την κατασκευή χρησιμοποιήθηκαν τα παρακάτω εξαρτήματα:

- Οθόνη είκοσι χαρακτήρων και τεσσάρων γραμμών

Εικόνα 3.1 – Οθόνη που χρησιμοποιήθηκε

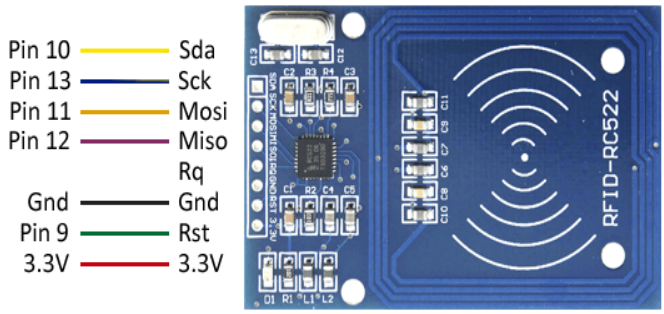
Για την γραφική απεικόνιση χρησιμοποιήθηκε μία οθόνη 20x4 όπου 20 ο αριθμός των στηλών και 4 αντιπροσωπεύει τον αριθμό των σειρών. Με τον μικροελεγκτή επικοινωνεί μέσω του διαύλου I2C (Inter-Integrated Circuit) ή (Διασυνδεδεμένο Κύκλωμα) ο οποίος είναι ένας σειριακός δίαυλος που δημιουργήθηκε από τη Philips και χρησιμοποιείται για την σύνδεση περιφερειακών μικρής ταχύτητας σε μητικές πλακέτες, embedded systems (ενσωματωμένα συστήματα), κινητά τηλέφωνα ή άλλες ηλεκτρονικές συσκευές. Ο δίαυλος I2C δεν χρησιμοποιείται μόνο για την επικοινωνία συσκευών που βρίσκονται πάνω σε ένα τυπωμένο κύκλωμα, αλλά και για την επικοινωνία συσκευών που συνδέονται με καλώδια.



Εικόνα 3.2 – Εξωτερικές συσκευές συνδεδεμένες στον δίαυλο I2C

Στην εικόνα φαίνεται ένα παράδειγμα διαύλου I2C. Όπως φαίνεται για τη μεταφορά των δεδομένων (0 ή 1) χρησιμοποιεί μόνο δύο καλώδια (τα οποία είναι ημιαμφίδρομης κατεύθυνσης): Τα SCL και SDA. Η γραμμή SCL είναι η γραμμή ρολογιού, ενώ η SDA είναι η γραμμή δεδομένων. Οι γραμμές αυτές συνδέονται σε όλες τις συσκευές, που υπάρχουν πάνω στο δίαυλο. Προφανώς εκτός από τα παραπάνω καλώδια που μεταφέρουν δεδομένα, απαιτείται και ένα τρίτο καλώδιο το οποίο είναι η γείωση (GND) ή 0 V. Επίσης μπορεί να υπάρχει και ένα τέταρτο καλώδιο το οποίο είναι η γραμμή τροφοδοσίας (VCC ή VDD), με την οποία τροφοδοτούνται με ισχύ οι διάφορες συσκευές που συνδέονται στο δίαυλο.

- Αναγνώστης ανέπαφων καρτών



Εικόνα 3.3 – Ο αναγνώστης καρτών που χρησιμοποιήθηκε

Η ανέπαφη εξακρίβωση του χρήστη βασίζεται στον αναγνώστη RFID (radio-frequency identification) ή (ταυτοποίηση ραδιοσυχνοτήτων). Η τεχνολογία RFID αναφέρεται σε μια τεχνολογία μέσω της οποίας τα ψηφιακά δεδομένα που κωδικοποιούνται σε ετικέτες RFID ή έξυπνες κάρτες διαβάζονται από έναν αναγνώστη μέσω ραδιοκυμάτων.

Μία κάρτα RFID έχει ενσωματωμένο πομπό και δέκτη. Το πραγματικό στοιχείο RFID που περιέχεται σε μια κάρτα έχει δύο μέρη: ένα ολοκληρωμένο κύκλωμα για την αποθήκευση και την επεξεργασία πληροφοριών, και μια κεραία για τη λήψη και τη μετάδοση ενός σήματος. Η κάρτα RFID διαθέτει μη πτητική αποθήκευση μνήμης και μπορεί να περιλαμβάνει σταθερή ή προγραμματιζόμενη λογική για επεξεργασία δεδομένων μετάδοσης και αισθητήρων. Οι κάρτες μπορούν να είναι παθητικές, ενεργές ή βοηθητικές παθητικές μπαταρίες.

Εικόνα 3.4 – Μία από τις κάρτες που χρησιμοποιήθηκε

Μια παθητική κάρτα είναι η φθηνότερη επιλογή και δεν διαθέτει μπαταρία. Η κάρτα χρησιμοποιεί ενέργεια ραδιοσυχνοτήτων που μεταδίδεται από τον αναγνώστη. Μια ενεργή κάρτα διαθέτει ενσωματωμένη μπαταρία, μεταδίδοντας περιοδικά τα διαπιστευτήρια της.

Επιπλέον, μια κάρτα μπορεί να είναι μόνο για ανάγνωση ή ανάγνωση / εγγραφή. Μια κάρτα μόνο για ανάγνωση έχει έναν σειριακό αριθμό που έχει οριστεί εργοστασιακά και χρησιμοποιείται για την αναγνώριση σε μια βάση δεδομένων,

ενώ μια κάρτα ανάγνωσης / εγγραφής μπορεί να έχει συγκεκριμένα προσαρμοσμένα δεδομένα γραμμένα στην ετικέτα από το χρήστη.

Ο αναγνώστης RFID διαθέτει έναν αμφίδρομο ραδιοπομπό-μεταδότη (transciever). Ο πομποδέκτης μεταδίδει κωδικοποιημένο ραδιοφωνικό σήμα για να διερευνήσει την ετικέτα. Το ραδιοφωνικό σήμα ουσιαστικά ενεργοποιεί την κάρτα. Με τη σειρά του, ο αναμεταδότης σήματος μετατρέπει το ραδιοφωνικό σήμα σε χρησιμοποιήσιμη ισχύ και ανταποκρίνεται στον αναγνώστη.

Η ποσότητα των πληροφοριών που είναι αποθηκευμένες σε μια κάρτα RFID ποικίλλει. Για παράδειγμα, μια παθητική κάρτα μπορεί να αποθηκεύει μόνο έως και 1024 bytes πληροφοριών - δηλαδή μόνο ένα kilobyte (KB), αρκετά για να αποθηκευτεί ένα πλήρες όνομα, αριθμός ταυτότητας, γενέθλια, πληροφορίες πιστωτικών καρτών και πολλά άλλα.

- ρελέ 5V

Εικόνα 3.5 – Το ρελέ που χρησιμοποιήθηκε

Για την οδήγηση του μηχανισμού της κλειδαριάς χρησιμοποιείται ρελέ. Ένα από τα πιο χρήσιμα πράγματα που μπορεί να κάνει ένα Arduino είναι να ελέγχει συσκευές υψηλότερης τάσης (120-240V) όπως ανεμιστήρες, φώτα, θερμάστρες

και άλλες οικιακές συσκευές. Δεδομένου ότι το Arduino λειτουργεί σε 5V, δεν μπορεί να ελέγξει απευθείας αυτές τις συσκευές υψηλότερης τάσης, αλλά μπορεί να χρησιμοποιηθεί ένα ρελέ 5V και να χρησιμοποιηθεί το Arduino για να ελέγξει το ρελέ.

- Πλακέτα κινητής τηλεφωνίας

Εικόνα 3.6 – Πλακέτα κινητής τηλεφωνίας που χρησιμοποιήθηκε

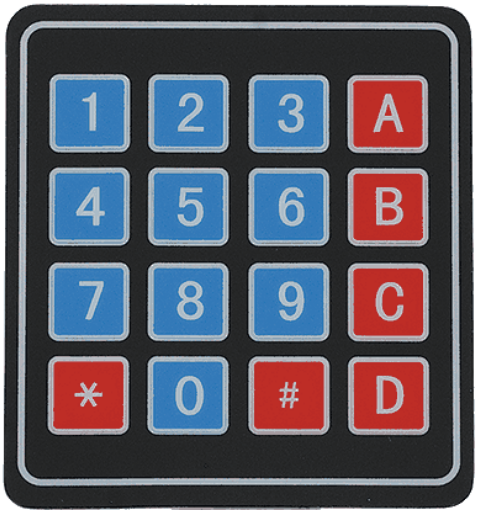
Για την ικανότητα κινητής σύνδεσης, χρησιμοποιείται η πλακέτα A6 GSM (Global System for Mobile communications) ή (Παγκόσμιο Σύστημα Κινητών Επικοινωνιών). Ο μικροελεγκτής επικοινωνεί με την πλακέτα μέσω του σειριακού καναλιού tx/rx και οι εντολές που αντιλαμβάνεται είναι τύπου AT μία γλώσσα που καταλαβαίνουν οι περισσότερες, αν όχι όλες οι πλακέτες τέτοιου είδους.

- Ηχητικός βομβητής

Εικόνα 3.7 – Ο βομβητής που χρησιμοποιήθηκε

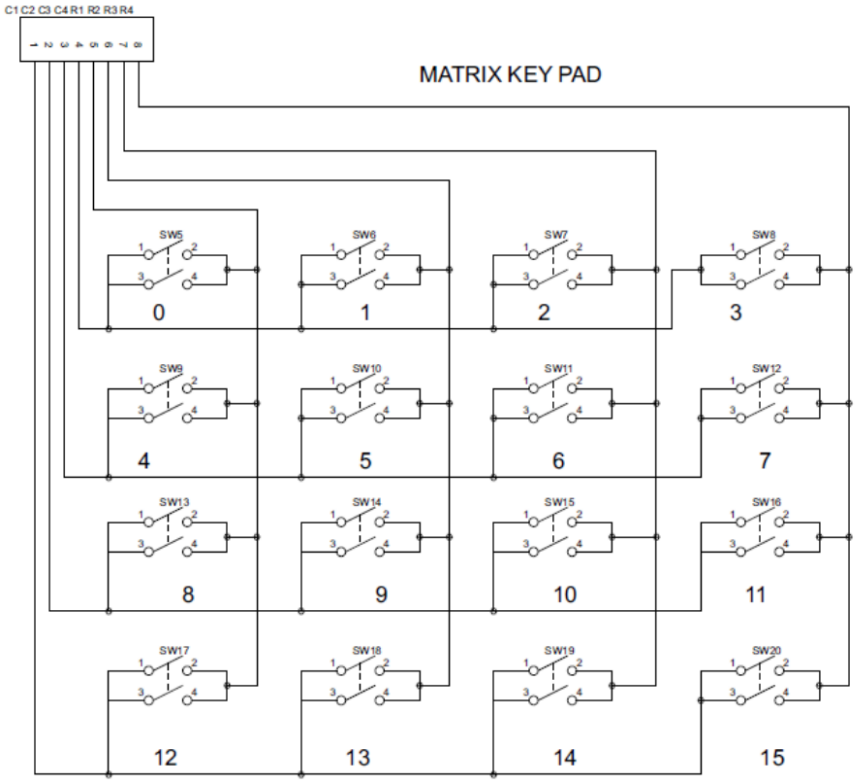
Για την ηχητική ειδοποίηση χρησιμοποιείται ενεργός βομβητής. Είναι ενα πιεζοηλεκτρικό στοιχείο το οποίο μπορεί να κινείται από ένα ταλαντευόμενο ηλεκτρονικό κύκλωμα ή άλλη πηγή ακουστικού σήματος, που οδηγείται με έναν πιεζοηλεκτρικό ενισχυτή ήχου. Οι ήχοι που χρησιμοποιούνται συνήθως για να υποδείξουν ότι ένα πλήκτρο έχει πατηθεί.

- Πληκτρολόγιο δεκαέξι πλήκτρων



Εικόνα 3.8 – Το πληκτρολόγιο που χρησιμοποιήθηκε

Παρακάτω εξηγείται η λογική όπου ένα πληκτρολόγιο 4x4 επικοινωνεί με τον μικροελεγκτή. Η ίδια λογική ισχύει για οποιοδήποτε πληκτρολόγιο NxM, όπου N σειρές και M στήλες. Για να διαβάσει μια ψηφιακή είσοδο ο ελεγκτής, απαιτείται ένας ακροδέκτης. Όταν υπάρχουν πολλές ψηφιακές είσοδοι που πρέπει να διαβαστούν, δεν είναι εφικτό να διαθέσουμε έναν ακροδέκτη για κάθε μία από αυτές. Εάν το πληκτρολόγιο είναι 2x2, θα χρειαστεί 2 ακίδες για τις σειρές και 2 ακίδες για τις στήλες. Σε μια τέτοια περίπτωση δεν υπάρχει διαφορά στο κόστος ανάγνωσης. Αλλά αν θεωρήσουμε ένα πληκτρολόγιο 10x10 χρειάζονται μόνο 20 ακροδέκτες (10 για τις σειρές και 10 για τις στήλες) για να διαβάστούν 100 ψηφιακές είσοδοι.



Εικόνα 3.9 – Αρχιτεκτονική πληκτρολογίου 16 πλήκτρων

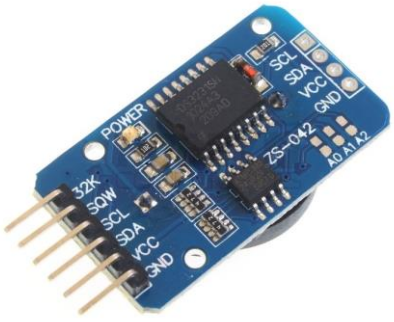
Αρχικά όλοι οι διακόπτες θεωρείται ότι είναι ανοιχτοί. Επομένως δεν υπάρχει σύνδεση μεταξύ των σειρών και των στηλών. Όταν πατηθεί οποιοσδήποτε από τα πλήκτρα, η αντίστοιχη σειρά και η στήλη είναι συνδεδεμένη. Αυτό θα οδηγήσει τον ακροδέκτη της στήλης από την κατάσταση αρχικά υψηλής τάσης (λογικό ‘1’) σε

κατάσταση χαμηλής τάσης (λογικό ‘0’). Χρησιμοποιώντας αυτή τη λογική, μπορεί να ανιχνευθεί το πάτημα του πλήκτρου.

- Arduino Uno

Εικόνα 3.10 – Η πλακέτα Arduino Uno που χρησιμοποιήθηκε

- Ψηφιακό ρολόϊ πραγματικού χρόνου



Εικόνα 3.11 – Το ρολόΐ που χρησιμοποιήθηκε

.

Για την ώρα και την ημερομηνία χρησιμοποιείται ένα ειδικό κύκλωμα (RTC). Ένα ρολόι πραγματικού χρόνου (συνήθως με τη μορφή ολοκληρωμένου κυκλώματος) που παρακολουθεί την τρέχουσα ώρα. Με τον μικροελεγκτή επικοινωνεί επίσης μέσω του διαύλου I2C (Inter-Integrated Circuit) ή (Διασυνδεδεμένο Κύκλωμα).

- Μαγνητικοί διακόπτες

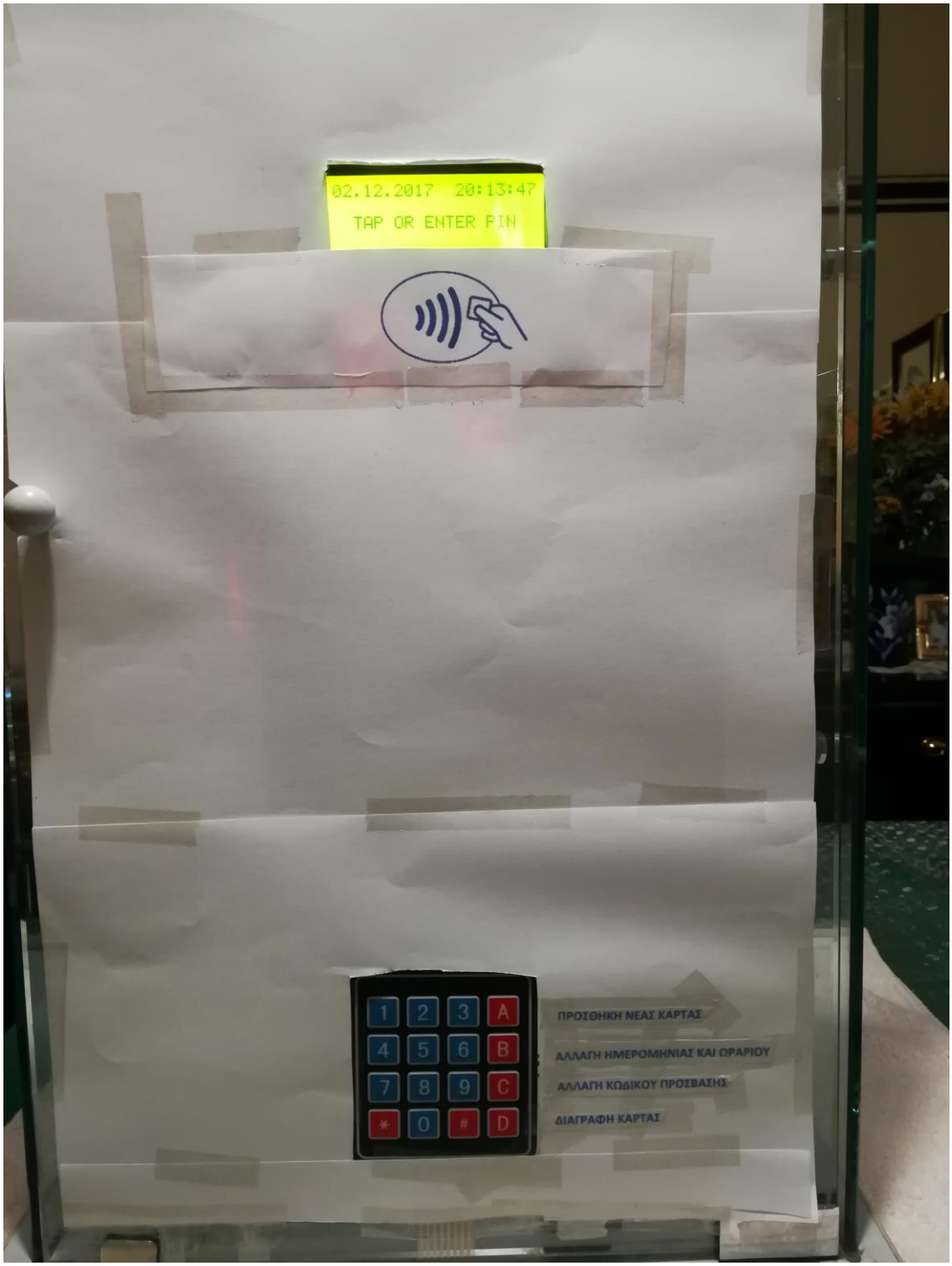
Εικόνα 3.12 – Οι διακόπτες που χρησιμοποιήθηκαν

Τέλος για την ανίχνευση εάν η πόρτα θα είναι κλειστή ή ανοιχτή χρησιμοποιείται μαγνητικός διακόπτης όπως εκείνοι που χρησιμοποιούνται στους συναγερμούς

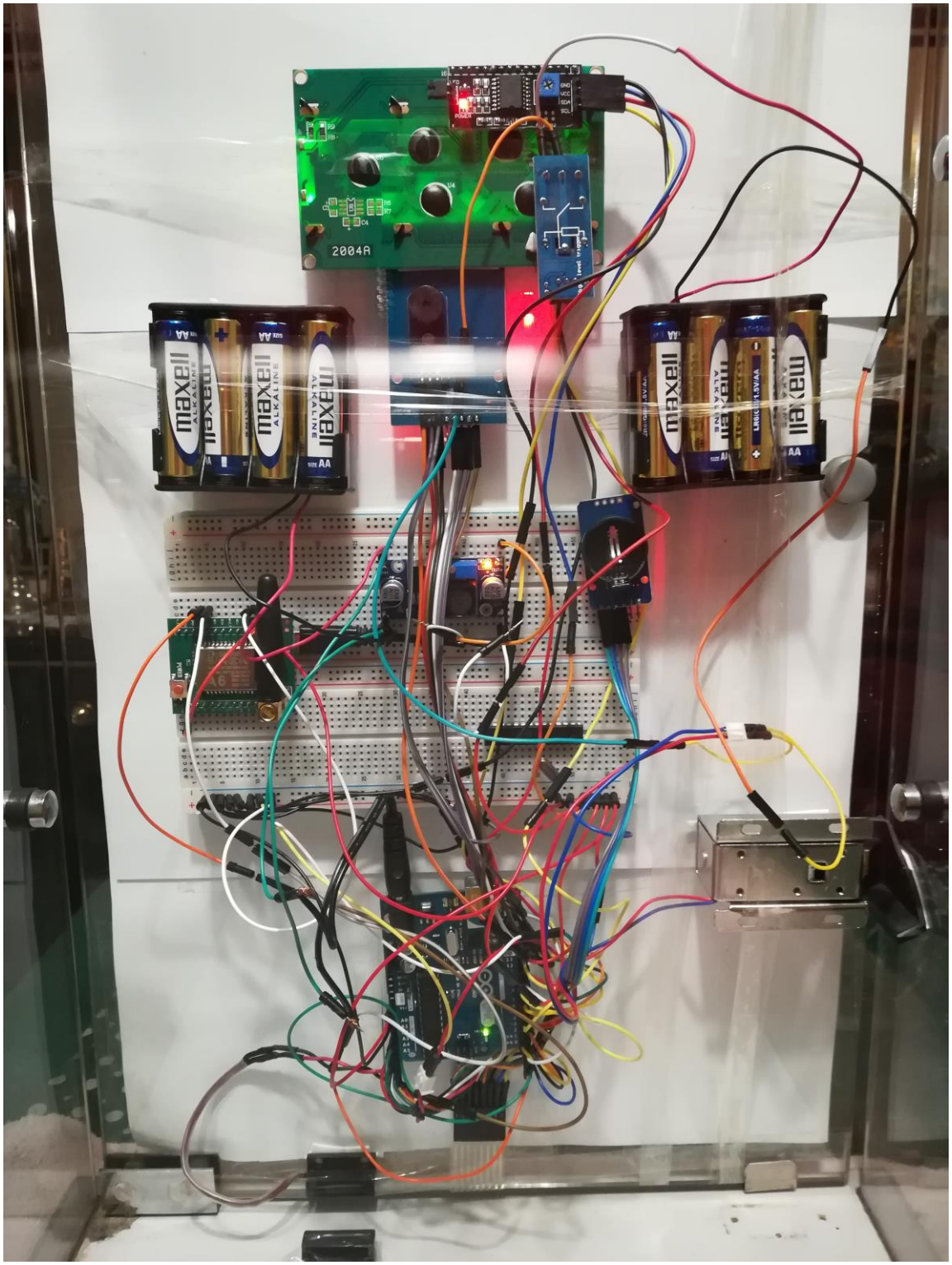
- Ηλεκτρονική κλειδαριά

Εικόνα 3.13 – Η κλειδαριά που χρησιμοποιήθηκε

Η παραπάνω κλειδαριά λειτουργεί με βάση την αρχή του ηλεκτρομαγνητισμού, όπου ένα ρεύμα συνεχούς ρεύματος δημιουργεί ένα μαγνητικό πεδίο κινεί τον πύρο και ως συνέπεια η πύλη ξεκλειδώνει. Για τον έλεγχο της χρησιμοποιείται το ρελέ καθώς το Arduino uno δεν είναι ικανό να παρέχει την απαιτούμενη τροφοδοσία.



Εικόνα 3.14 – Η τελική κατασκευή από την εξωτερική πλευρά

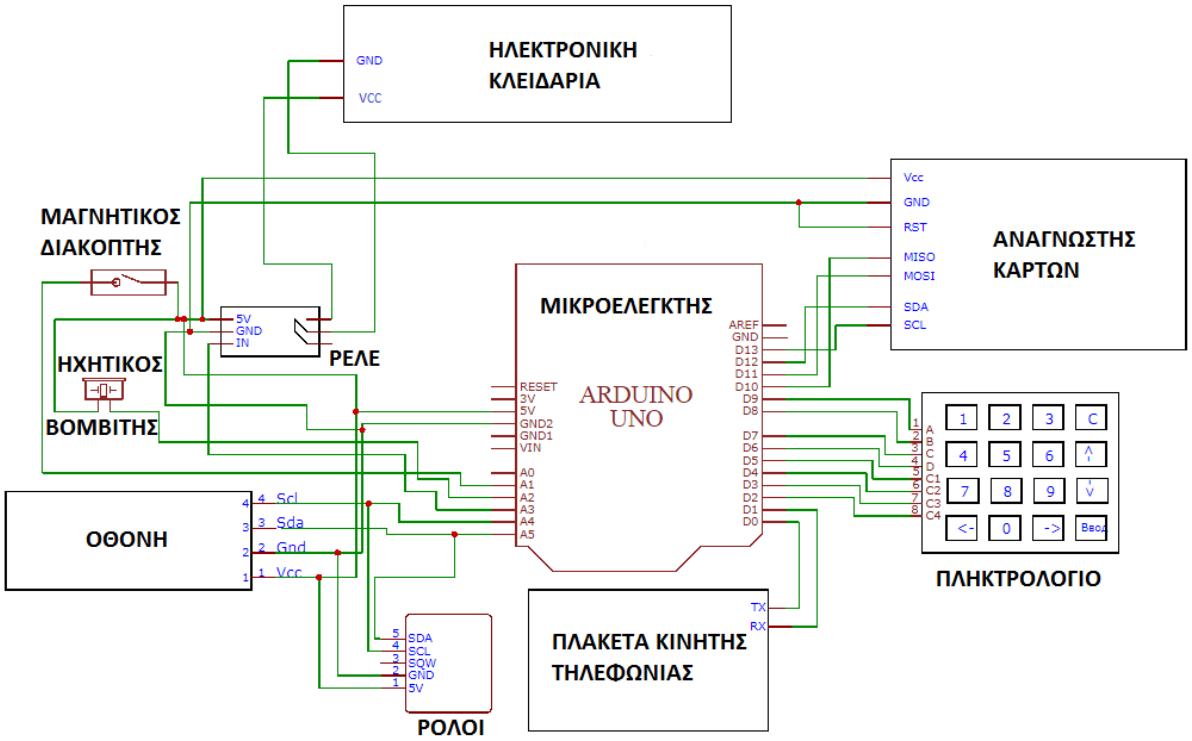


Εικόνα 3.15 – Η τελική κατασκευή από την εσωτερική πλευρά

.

3.3 Block διάγραμμα

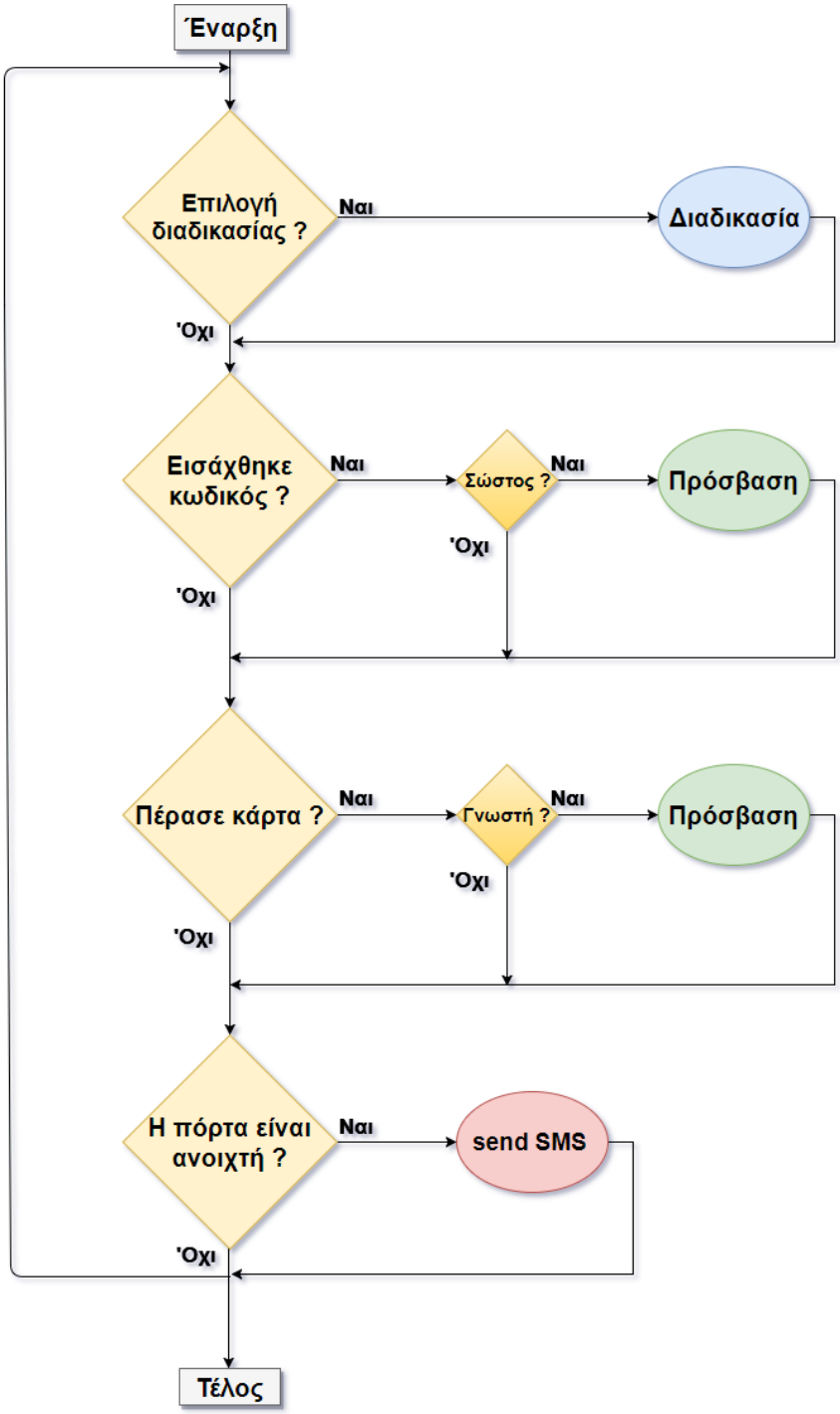
Το παρακάτω διάγραμμα έχει δημιουργηθεί με το πρόγραμμα Eagle και αποτελεί το διάγραμμα συνδεσμολογίας των εξωτερικών μονάδων σε σχέση με το Arduino Uno.



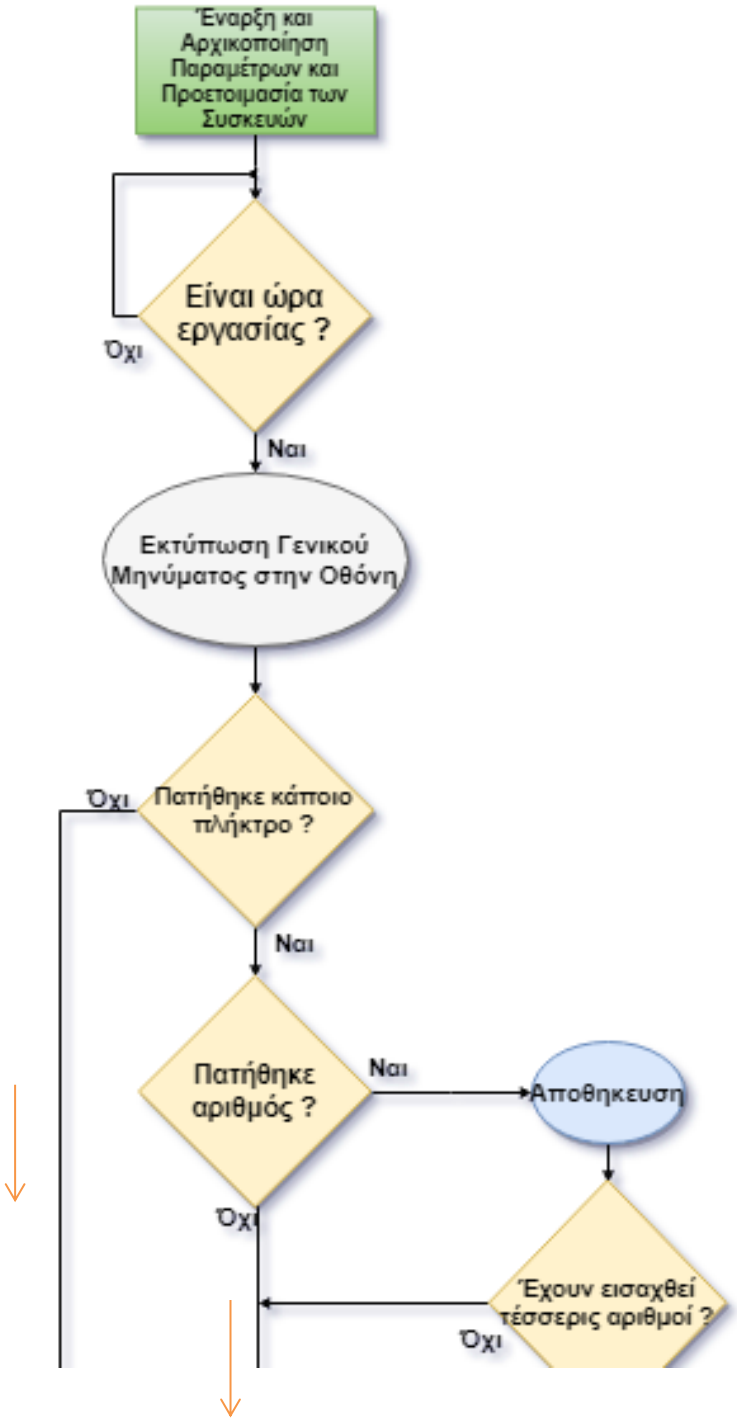
Ο μικροελεγκτής περιμένει για είσοδο χρήστη από το πληκτρολόγιο ή τον αναγνώστη καρτών. Η οθόνη και ο ηχητικός βομβιτής έχουν τον ρόλο της εξόδου για τον μικρολεγκτή. Το ρολόι εμφανίζει τις πληροφορίες της ώρας στην οθόνη. Η πλακέτα κινητής τηλεφωνίας ενεργοποιείται σε περίπτωση όπου ο μαγνητικός διακόπτης είναι ανοιχτός χωρίς να έχει προηγηθεί είσοδος από τον χρήστη. Το ρελέ ενεργοποιείται ύστερα από εντολή εξουσιοδότησης του μικροελεγκτή και αυτό με την σειρά του ενεργοποιεί την ηλεκτρονική κλειδαριά.

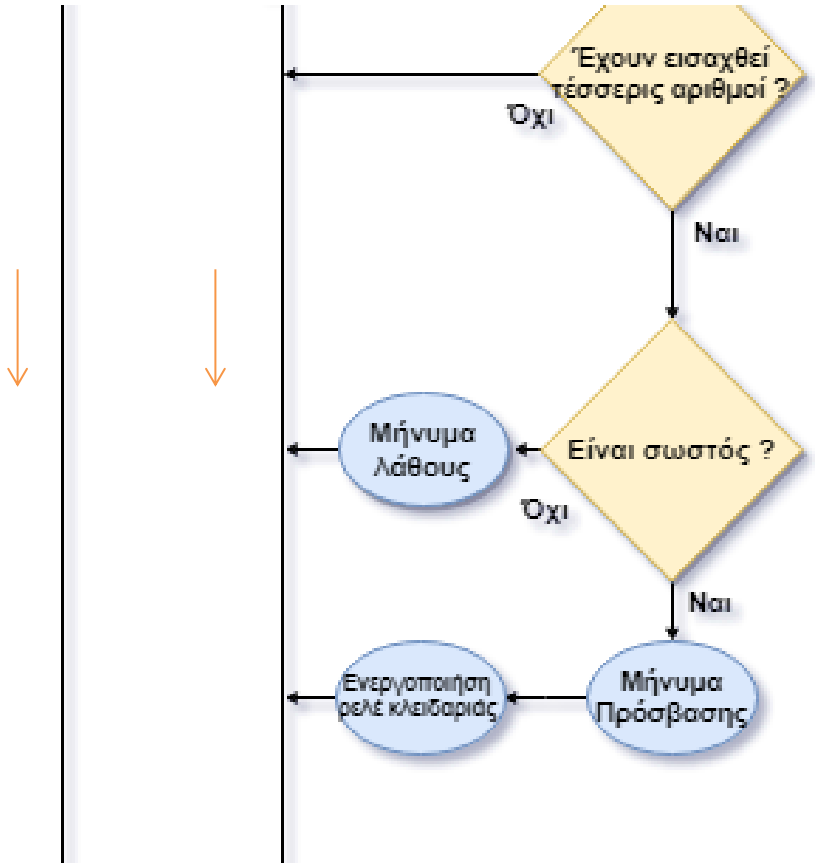
3.4 Διάγραμμα Ροής (Συνοπτικό)

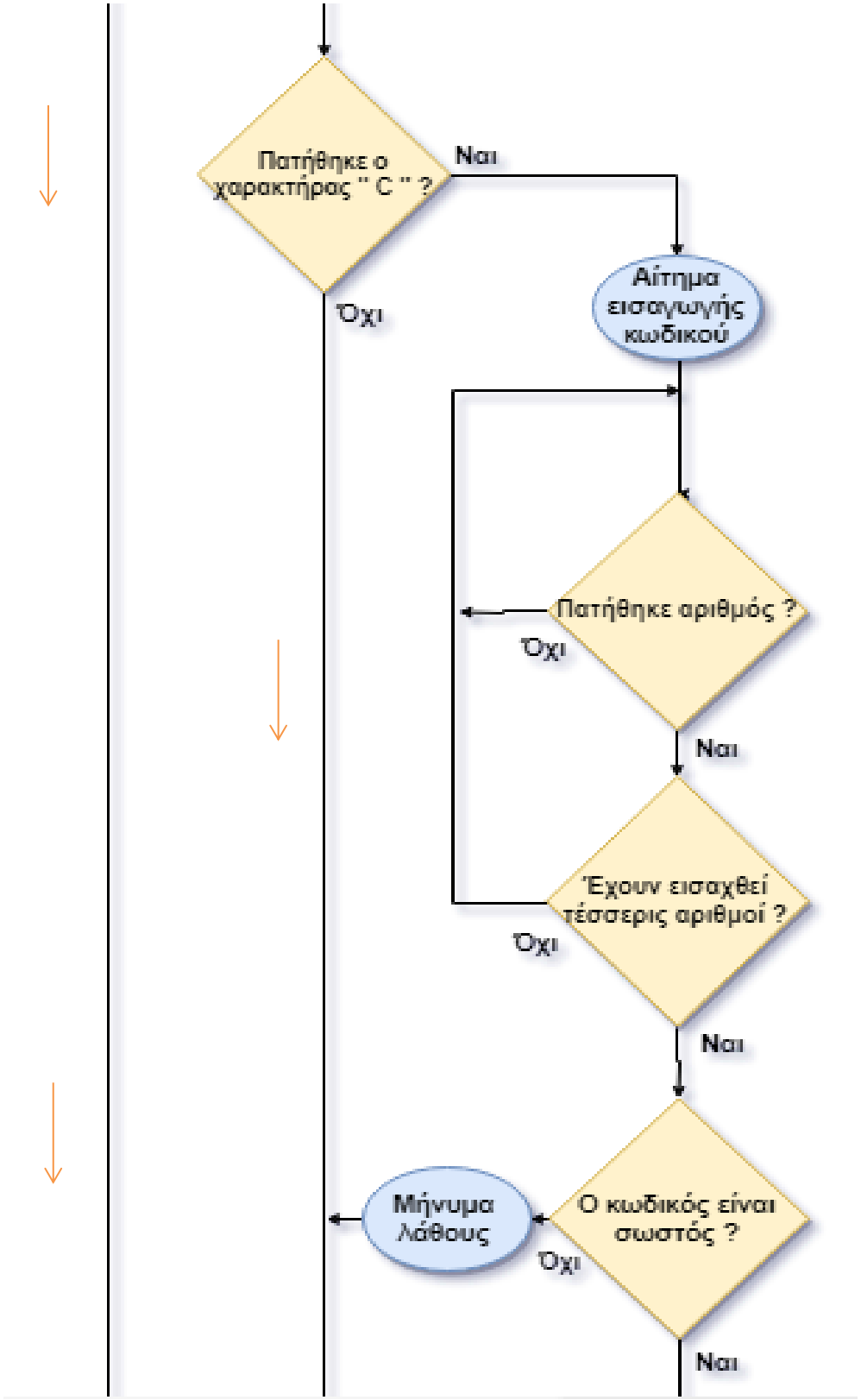
Το παρακάτω διάγραμμα ροής έχει δημιουργηθεί μέσω της διαδικτυακής εφαρμογής www.draw.io και αποτυπώνει συνοπτικά τον τρόπο λειτουργίας του συστήματος.

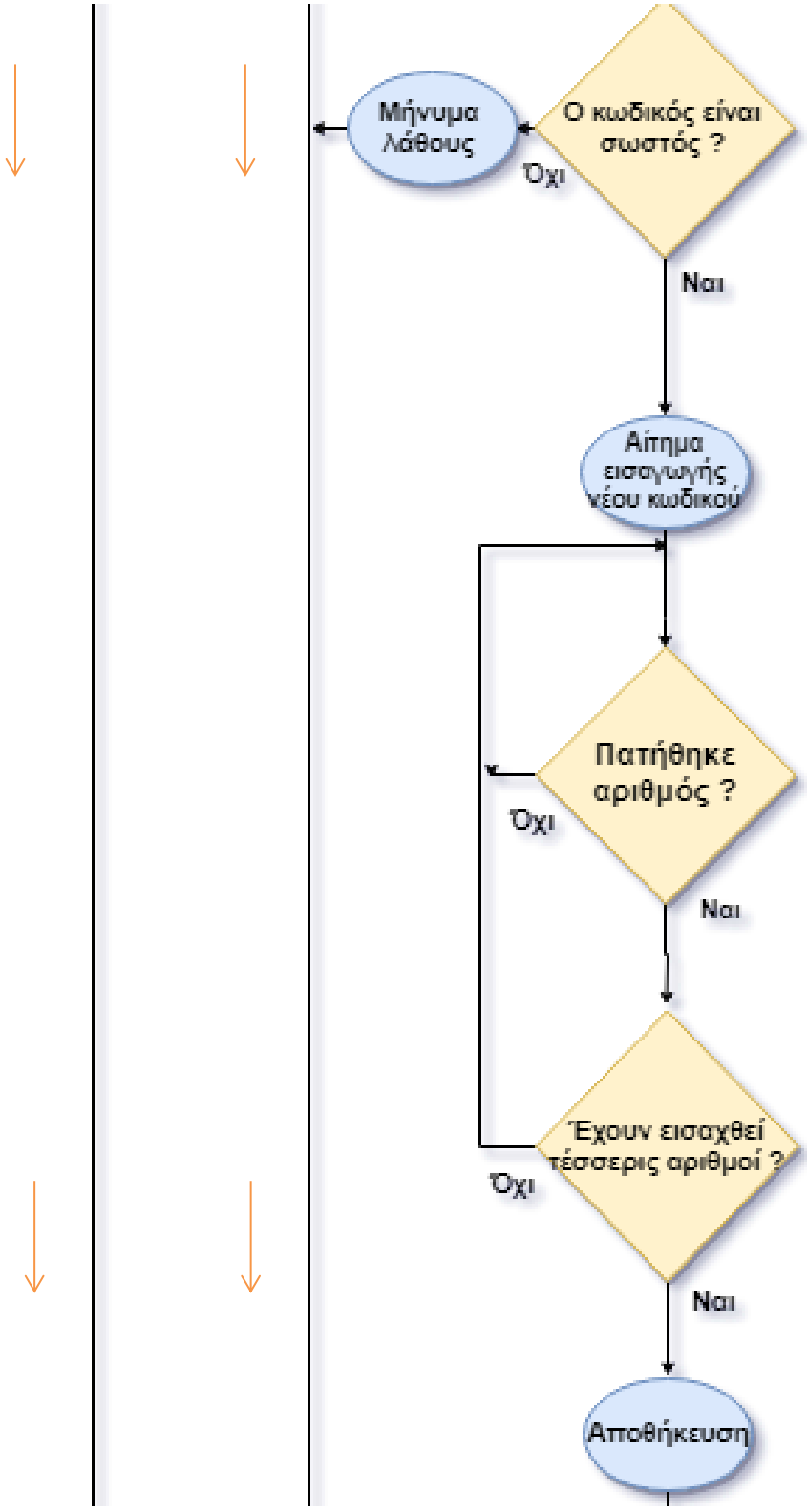


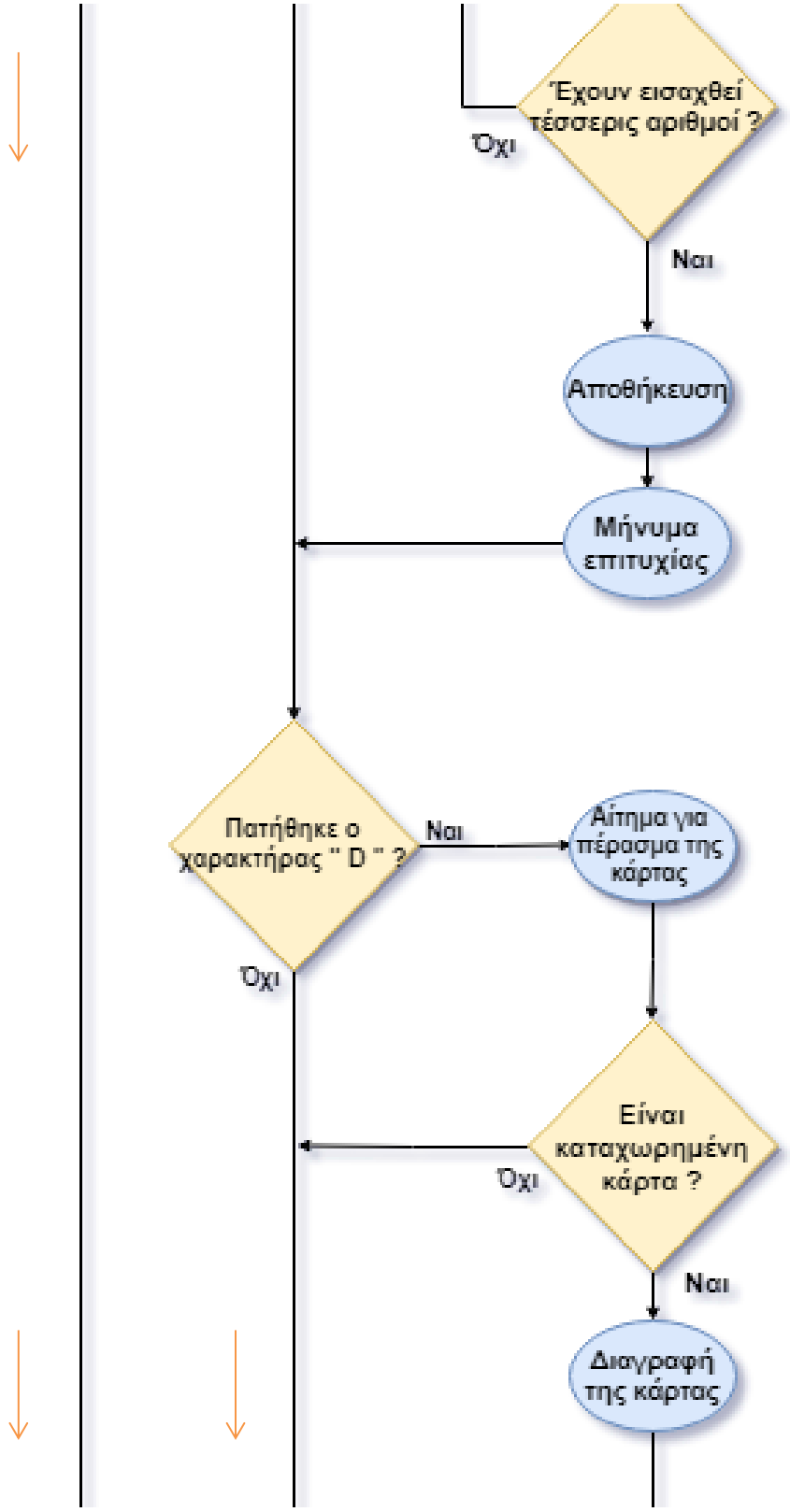
3.5 Πλήρες Διάγραμμα Ροής

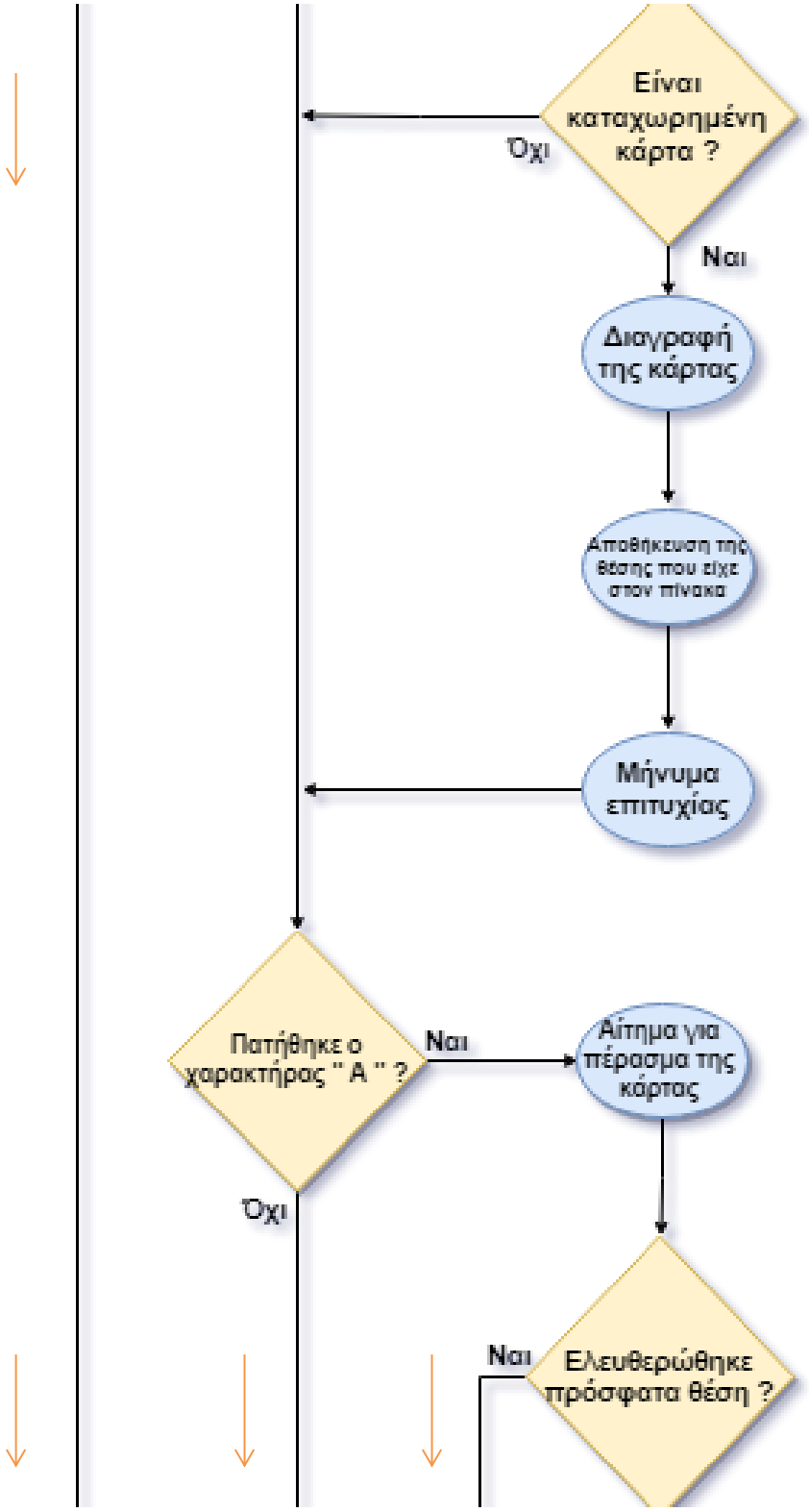


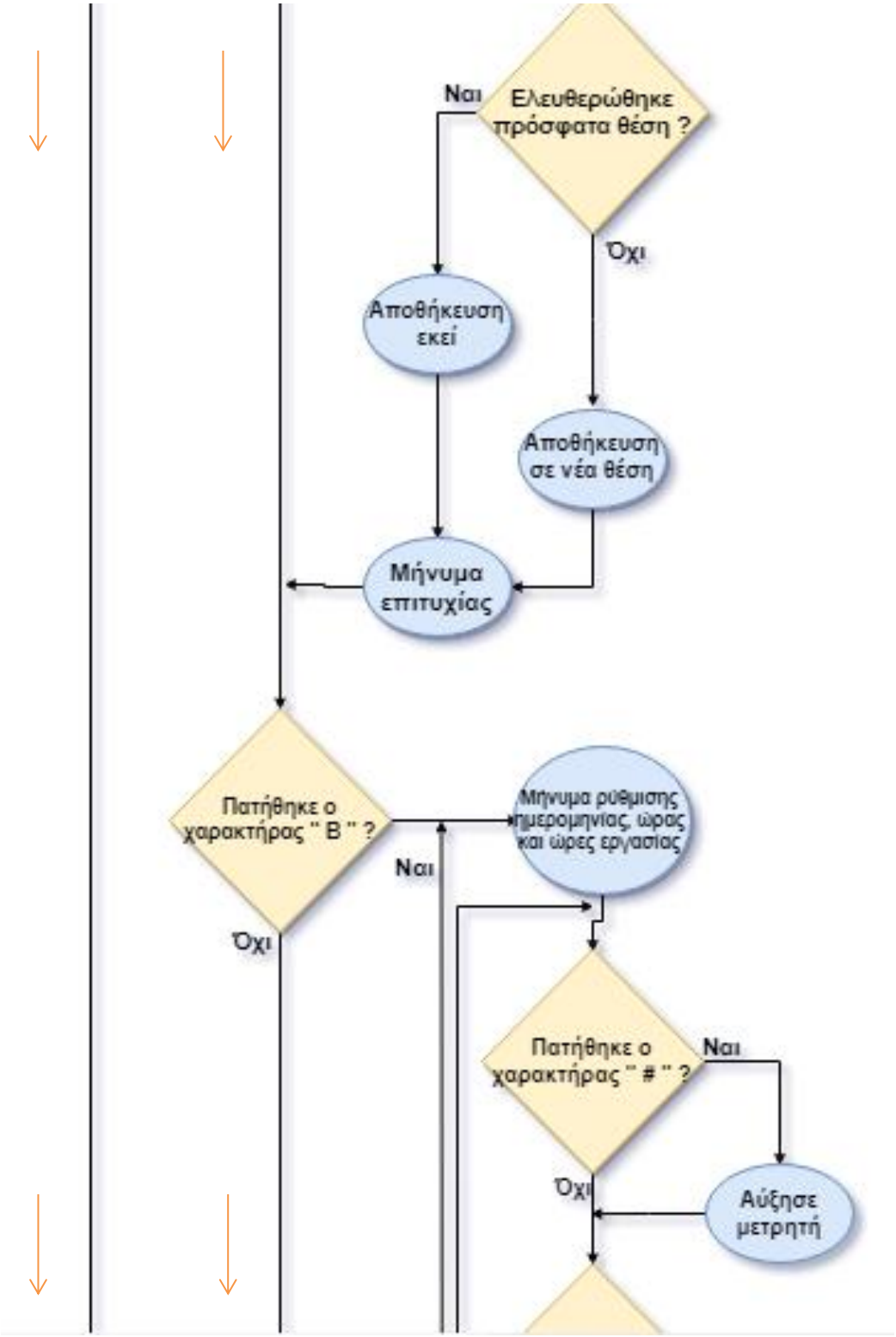

Έχουν εισαχθεί τέσσερις αριθμοί ?
Όχι
Ναι
Είναι σωστός ?
Μήνυμα Λάθους
Όχι
Ναι
Μήνυμα Πρόσβασης
Ενεργοποίηση ρελέ κλειδαριάς

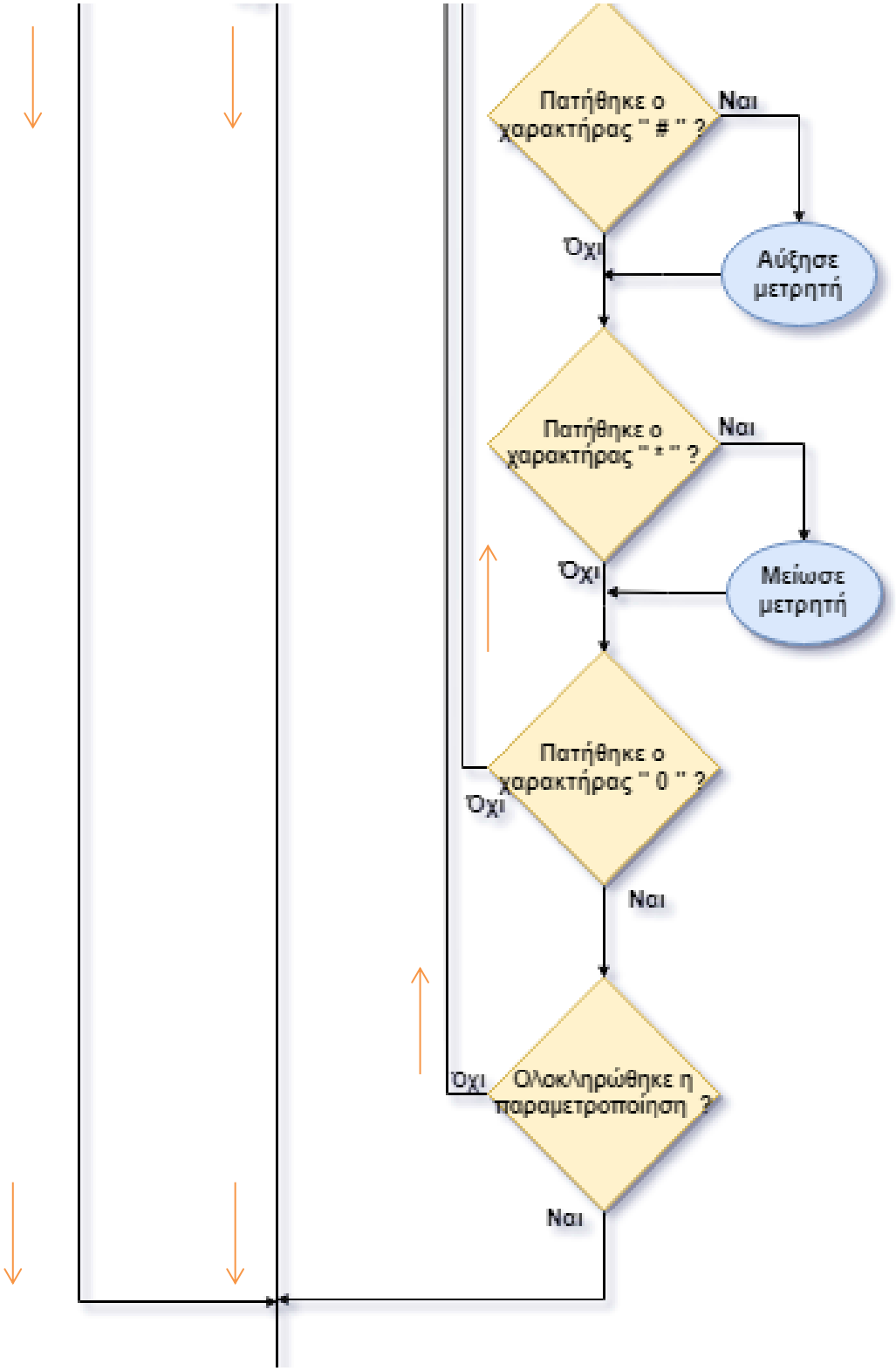

Πατήθηκε ο χαρακτήρας " C " ?
Ναι
Όχι
Αίτημα εισαγωγής κωδικού
Πατήθηκε αριθμός ?
Όχι
Ναι
Έχουν εισαχθεί τέσσερις αριθμοί ?
Όχι
Ναι
Ο κωδικός είναι σωστός ?
Μήνυμα λάθους
Όχι
Ναι

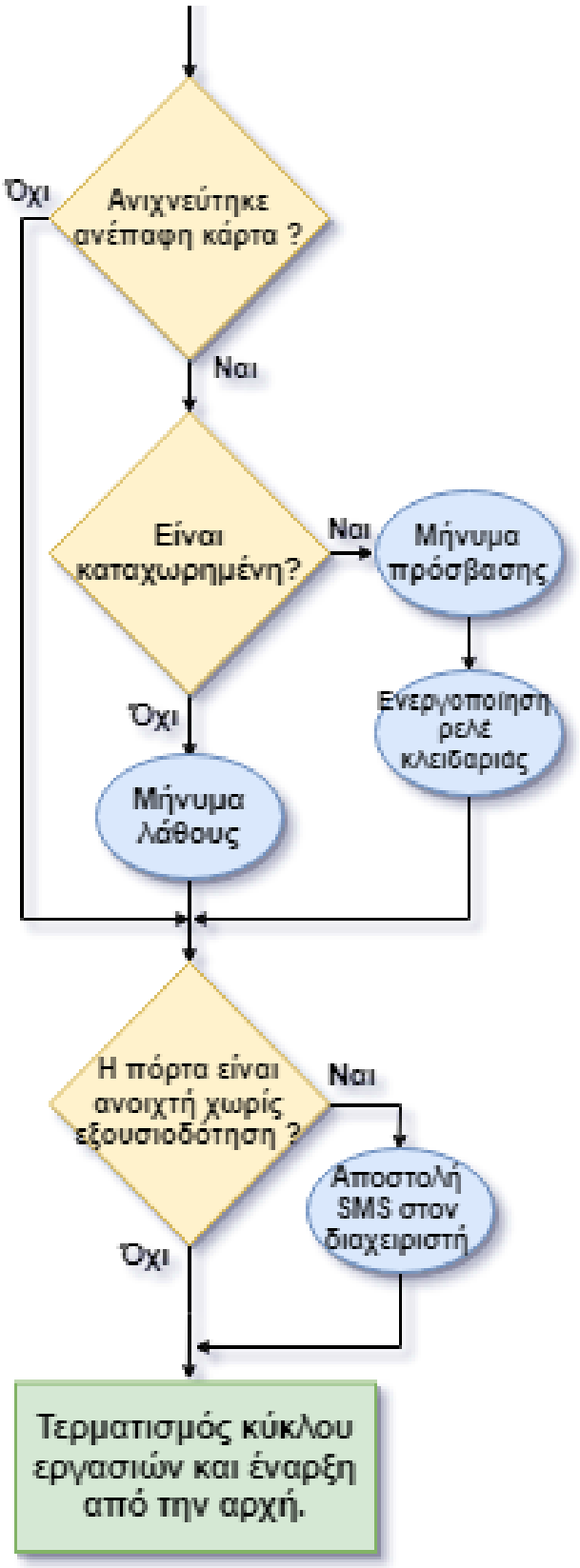

Μήνυμα λάθους
Ο κωδικός είναι σωστός ?
Όχι
Ναι
Αίτημα εισαγωγής νέου κωδικού
Πατήθηκε αριθμός ?
Όχι
Ναι
Έχουν εισαχθεί τέσσερις αριθμοί ?
Όχι
Ναι
Αποθήκευση


Έχουν εισαχθεί τέσσερις αριθμοί ?
Όχι
Ναι
Αποθήκευση
Μήνυμα επιτυχίας
Πατήθηκε ο χαρακτήρας " D " ?
Ναι
Όχι
Αίτημα για πέρασμα της κάρτας
Είναι καταχωρημένη κάρτα ?
Όχι
Ναι
Διαγραφή της κάρτας


Είναι καταχωρημένη κάρτα ?
Όχι
Ναι
Διαγραφή της κάρτας
Αποθήκευση της θέσης που είχε στον πίνακα
Μήνυμα επιτυχίας
Πατήθηκε ο χαρακτήρας " A " ?
Ναι
Όχι
Αίτημα για πέρασμα της κάρτας
Ναι
Ελευθερώθηκε πρόσφατα θέση ?


Ναι
Ελευθερώθηκε πρόσφατα θέση ?
Όχι
Αποθήκευση εκεί
Αποθήκευση σε νέα θέση
Μήνυμα επιτυχίας
Πατήθηκε ο χαρακτήρας " B " ?
Ναι
Μήνυμα ρύθμισης ημερομηνίας, ώρας και ώρες εργασίας
Όχι
Πατήθηκε ο χαρακτήρας " # " ?
Ναι
Όχι
Αύξησε μετρητή


Πατήθηκε ο χαρακτήρας " # " ?
Ναι
Όχι
Αύξησε μετρητή
Πατήθηκε ο χαρακτήρας " * " ?
Ναι
Όχι
Μείωσε μετρητή
Πατήθηκε ο χαρακτήρας " 0 " ?
Όχι
Ναι
Ολοκληρώθηκε η παραμετροποίηση ?
Όχι
Ναι


Ανιχνεύτηκε ανέπαφη κάρτα ?
Όχι
Ναι
Είναι καταχωρημένη?
Ναι
Μήνυμα πρόσβασης
Ενεργοποίηση ρελέ κλειδαριάς
Όχι
Μήνυμα λάθους
Η πόρτα είναι ανοιχτή χωρίς εξουσιοδότηση ?
Ναι
Αποστολή SMS στον διαχειριστή
Όχι
Τερματισμός κύκλου εργασιών και έναρξη από την αρχή.

3.6 Επίλυση Προβλημάτων

Τα προβλήματα που παρουσιάστηκαν ήταν περισσότερο τεχνικής φύσεως από άποψη υλικού. Υπήρξαν και προβληματισμοί σχετικά με το προγραμματιστικό κομμάτι, αλλά και το κατασκευαστικό.

Το πρώτο πρόβλημα που παρουσιάστηκε αφορούσε την οθόνη η οποία δεν επικοινωνούσε σωστά με τον μικροελεγκτή με αποτέλεσμα να μην εμφανίζεται κανένας χαρακτήρας.

Επειδή η οθόνη επικοινωνεί πρώτα με το I2C module και αυτό με τον μικροελεγκτή στην αρχή ήταν δύσκολο να γίνει αντιληπτό. Επειδή όμως διαπιστώθηκε ότι η οθόνη υπάκουε σε μία και μόνο από τις πολλές εντολές που χρησιμοποιήθηκαν, τότε το συμπέρασμα που βγήκε ήταν ότι το πρόβλημα βρισκόταν στην πλακέτα I2C. Το πρόβλημα επιλύθηκε άμμεσα με την αντικατάσταση της.

Ο επόμενος προβληματισμός ήταν πώς θα δημιουργούσαμε τον αλγόριθμο που αλλάζει τον κωδικό πρόσβασης. Έτσι η λύση ήταν να αποθηκεύουμε τον νέο κωδικό σε μία προσωρινή μεταβλητή και ύστερα να αποδώσουμε την τιμή της προσωρινής μεταβλητής στην μόνιμη μεταβλητή όπου χρησιμοποείται για τον κωδικό. Το μειονέκτημα είναι ότι αν γίνει επανεκίννηση της συσκευής, ο κωδικός θα χαθεί και θα γίνει αρχικοποιήση της τιμής του με τον αρχικό που έχει οριστεί ως προεπιλεγμένος.

Το επόμενο πρόβλημα και δυσκολότερο από τα προηγούμενα ήταν η δημιουργία αλγόριθμου όπου θα αποθήκευε / διέγραφε πλαστικές κάρτες με όριο τις είκοσι. Η λύση ήταν η δημιουργία ενός πίνακα είκοσι σειρών και έξι στηλών όπου ο αριθμός έξι αποτελεί τα εξι bytes μοναδικής πληροφορίας ανά κάρτα. Και ο αριθμός των σειρών αποτελεί την χωρητικότητα του συστήματος σε κάρτες. Η διαδικασία της διαγραφής ήταν έξυπνη καθώς όταν μία κάρτα έπρεπε να διαγραφεί, τότε την θέση της θα έπαιρνε η επόμενη κάρτα που επρόκειτο να προστεθεί στον πίνακα. Επίσης το ίδιο μειονέκτημα ισχύει και εδώ καθώς αν γίνει επανεκίννηση της συσκευής, ο πίνακας θα αδειάσει.

3.7 Μελοντικές προσθήκες

Μελλοντικά μπορούμε να προσθέσουμε την ικανότητα σύνδεσης με έναν κεντρικό server (εξυπηρετητή) ώστε η διαχείριση των χρηστών να γίνεται από αντίστοιχη εφαρμογή με χρήση βάσης δεδομένων. Επίσης θα μπορούσαμε να χρησιμοποιούμε το Arduino Uno ως μονάδα εισαγωγής δεδομένων αποκλειστικά και όλη η επεξργασία να γίνεται στο κεντρικό σύστημα. Έτσι σε περιπτώσεις περισσότερων της μίας πύλης, οι αλλαγές θα γίνονται σε ένα σύστημα.

Ακόμα θα μπορούσε να προστεθεί και ένας mail server ώστε οι διαχειριστές να ενημερώνονται μέσω της ηλεκτρονικής αλληλογραφίας.

Θα μπορούσε να συνδεθεί και μία κάμερα ώστε οι διαχειριστές να έχουν οπτική επαφή εξ αποστάσεως.

Επίσης θα μπορούσε να βελτιωθεί η απόκριση του συστήματος και να ελεγχθεί η κατανάλωση ενέργειας.

ΠΑΡΑΡΤΗΜΑ : ΚΩΔΙΚΑΣ

```
#include <Keypad.h>
#include <MFRC522.h> // Include of the RC522 Library
#include <SPI.h> // Used for communication via SPI with the Module
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DS3231.h>

LiquidCrystal_I2C lcd(0x3F,20,4);  // set the LCD address to 0x3F for a
20 chars and 4 line display

DS3231  rtc(SDA, SCL);
int set_hour = 00;
int set_min = 00;
int set_day = 01;
int set_month = 01;
int set_year = 2017;
Time  t;
int working_hour_from = 00;
int working_hour_to = 00;

#define Password_Lenght 5 // Give enough room for 4 chars + NULL char
char Data[Password_Lenght]; // 4 is the number of chars it can hold +
the null char = 5
char Master[Password_Lenght] = "0000";
byte data_count = 0, master_count = 0;


char customKey;
const byte ROWS = 4; //four rows
const byte COLS = 4; //three columns
char hexaKeys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {4, 2, 5, 6 }; //connect to the row pinouts of the
keypad
```

```
byte colPins[COLS] = {A3, A2, A1, A0 }; //connect to the column pinouts
of the keypad

//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins,
ROWS, COLS);


#define SDAPIN 10 // RFID Module SDA Pin is connected to the UNO 10 Pin
#define RESETPIN 8 // RFID Module RST Pin is connected to the UNO 8 Pin

#define Mag_Switch 9
#define Relay 7
#define Buzzer 3 // Pin 3 connected to + pin of the Buzzer

byte FoundTag; // Variable used to check if Tag was found
byte ReadTag; // Variable used to store anti-collision value to read Tag
information
byte TagData[MAX_LEN]; // Variable used to store Full Tag Data
byte TagSerialNumber[5]; // Variable used to store only Tag Serial
Number
byte GoodTagSerialNumber[5][5]; // The Tag Serial number we are looking
for
String GoodTag="False"; // Variable used to confirm good Tag Detected

MFRC522 nfc(SDAPIN, RESETPIN); // Init of the library using the UNO pins
declared above

int lock_counter_flag=0;
int tag_counter_store=0;
int tag_last_free=-1;


/*********************************************************************
*******************************/



void setup() {

lcd.init();// initialize the lcd
lcd.backlight();
lcd.setCursor(0,0);
lcd.print("SMART ACCESS CONTROL");
```

```
lcd.setCursor(2,1);
lcd.print("ARDUINO BASED");
lcd.setCursor(2,2);
lcd.print("COPYRIGHTS 2017");
lcd.setCursor(2,3);
lcd.print("Power By IOANNIS");
delay(2000);
pinMode(Mag_Switch, INPUT_PULLUP);
pinMode(Relay, OUTPUT);
pinMode(Buzzer, OUTPUT); // Set buzzer pin to an Output pin
digitalWrite(Buzzer, HIGH); // Buzzer Off at startup
SPI.begin();
rtc.begin();
Serial.begin(9600);
// Start to find an RFID Module
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Looking for RFID...");
delay(1000);
nfc.begin();
byte version = nfc.getFirmwareVersion(); // Variable to store Firmware
version of the Module
// If can't find an RFID Module
if (! version) {
lcd.setCursor(0,1);
lcd.print("Not found RFID !");
while(1); //Wait until a RFID Module is found
}
lcd.setCursor(0,1);
// If found, print the information about the RFID Module
lcd.print("Found chip RC522 !");
lcd.setCursor(0,2);
lcd.print("Firmware version:");
lcd.setCursor(0,3);
lcd.print("0x");
lcd.print(version, HEX);
delay(1000);
lcd.clear();
set_Time();
}
```

```
/*********************************************************************
*******************************/
/*********************************************************************
*******************************/
/*********************************************************************
*******************************/




void loop()
{

    enter_Credentials_msg();

    customKey = customKeypad.getKey();
    if (customKey) // makes sure a key is actually pressed, equal to
(customKey != NO_KEY)
    {
          if(numEntered())
          {
            store_data();
          }
          if(customKey == 'C')
          {
              password_Change();
          }
          if(customKey == 'D')
          {
              card_Delete();
          }
          if(customKey == 'A')
          {
              card_Add();
          }
          if(customKey == 'B')
          {
              set_Time();
          }

    }

    check_if_Password_entered();
```

```
    FoundTag = nfc.requestTag(MF1_REQIDL, TagData);

    if (FoundTag == MI_OK)
    {
        card_Scan();

        card_Check();
    }

    if(digitalRead(Mag_Switch))
    {
      smsSend();
    }

}




/*********************************************************************
*******************************/
/*********************************************************************
*******************************/
/*********************************************************************
*******************************/


void set_Time()
{

    lcd.clear();

    while(1)
    {
            customKey = customKeypad.getKey();
            if(customKey == '#')
            {
                  set_hour ++;
            }
            if(customKey == '*')
            {
                  set_hour --;
            }
            if(customKey == '0')
```

```
		{
			break;
		}
		delay(100);
		lcd.clear();
		lcd.setCursor(4,0);
		lcd.print("Set the Time");
		lcd.setCursor(4,2);
		lcd.print("Hour");
		lcd.setCursor(5,3);
		lcd.print(set_hour);
	}

	lcd.clear();

	while(1)
	{
		customKey = customKeypad.getKey();
		if(customKey == '#')
		{
			set_min ++;
		}
		if(customKey == '*')
		{
			set_min --;
		}
		if(customKey == '0')
		{
			break;
		}
		delay(100);
		lcd.clear();
		lcd.setCursor(4,0);
		lcd.print("Set the Time");
		lcd.setCursor(8,3);
		lcd.print(":");
		lcd.setCursor(10,2);
		lcd.print("Minute");
		lcd.setCursor(10,3);
		lcd.print(set_min);
	}

	rtc.setTime(set_hour, set_min, 00);
```

```
lcd.clear();

while(1)
{
        customKey = customKeypad.getKey();
        if(customKey == '#')
        {
              set_day ++;
        }
        if(customKey == '*')
        {
              set_day --;
        }
        if(customKey == '0')
        {
              break;
        }
        delay(100);
        lcd.clear();
        lcd.setCursor(4,0);
        lcd.print("Set the Date");
        lcd.setCursor(2,2);
        lcd.print("Day");
        lcd.setCursor(3,3);
        lcd.print(set_day);
}

lcd.clear();

while(1)
{
        customKey = customKeypad.getKey();
        if(customKey == '#')
        {
              set_month ++;
        }
        if(customKey == '*')
        {
              set_month --;
        }
        if(customKey == '0')
        {
              break;
        }
```

```
            delay(100);
            lcd.clear();
            lcd.setCursor(4,0);
            lcd.print("Set the Date");
            lcd.setCursor(6,3);
            lcd.print("/");
            lcd.setCursor(7,2);
            lcd.print("Month");
            lcd.setCursor(9,3);
            lcd.print(set_month);
    }

    lcd.clear();

    while(1)
    {
            customKey = customKeypad.getKey();
            if(customKey == '#')
            {
                  set_year ++;
            }
            if(customKey == '*')
            {
                  set_year --;
            }
            if(customKey == '0')
            {
                  break;
            }
            delay(100);
            lcd.clear();
            lcd.setCursor(4,0);
            lcd.print("Set the Date");
            lcd.setCursor(12,3);
            lcd.print("/");
            lcd.setCursor(14,2);
            lcd.print("Year");
            lcd.setCursor(14,3);
            lcd.print(set_year);

    }

    rtc.setDate(set_day, set_month, set_year);
```

```
    lcd.clear();

    while(1)
    {
            customKey = customKeypad.getKey();
            if(customKey == '#')
            {
                  working_hour_from ++;
            }
            if(customKey == '*')
            {
                  working_hour_from --;
            }
            if(customKey == '0')
            {
                  break;
            }
            delay(100);
            lcd.clear();
            lcd.setCursor(2,0);
            lcd.print("Set Working Hours");
            lcd.setCursor(4,2);
            lcd.print("from");
            lcd.setCursor(5,3);
            lcd.print(working_hour_from);
    }

    lcd.clear();

    while(1)
    {
            customKey = customKeypad.getKey();
            if(customKey == '#')
            {
                  working_hour_to ++;
            }
            if(customKey == '*')
            {
                  working_hour_to --;
            }
            if(customKey == '0')
            {
                  break;
            }
```

```
            delay(100);
            lcd.clear();
            lcd.setCursor(2,0);
            lcd.print("Set Working Hours");
            lcd.setCursor(12,2);
            lcd.print("to");
            lcd.setCursor(9,3);
            lcd.print("-");
            lcd.setCursor(13,3);
            lcd.print(working_hour_to);
    }

    lcd.clear();

}



/*********************************************************************
*******************************/


void card_Add()
{
    lcd.clear();
    lcd.setCursor(2,1);
    lcd.print("TAP AND WAIT");
    delay(3000);
    FoundTag = nfc.requestTag(MF1_REQIDL, TagData);
    if(FoundTag == MI_OK)
    {
        lcd.setCursor(1,3);
        lcd.print("Detected !");
        delay(3000);
        card_Scan();
        int i,j;
        if(tag_last_free == -1)
        {
          j = tag_counter_store;
        }
        else
        {
          j = tag_last_free;
        }
```

```
        for(i=0; i < 4; i++)
        {
            GoodTagSerialNumber[j][i] = TagSerialNumber[i];
        }
        if(tag_last_free == -1)
        {
          tag_counter_store++;
        }
        else
        {
          tag_last_free = -1;
        }
        if(tag_counter_store >= 5)
        {
          tag_counter_store=0;
        }
        lcd.clear();
        lcd.setCursor(1,3);
        lcd.print("Saved  !");
        delay(1000);
     }
     lcd.clear();
}


/*********************************************************************
*******************************/


void card_Delete()
{

    lcd.clear();
    lcd.setCursor(2,1);
    lcd.print("TAP AND WAIT");
    delay(3000);
    FoundTag = nfc.requestTag(MF1_REQIDL, TagData);
    if(FoundTag == MI_OK)
    {
        lcd.setCursor(1,3);
        lcd.print("Detected !");
        delay(3000);
        card_Scan();
        int i,j;
```

```
        for(j=0; j < 5; j++)
        {
        for(i=0; i < 4; i++)
        {
            if (GoodTagSerialNumber[j][i] != TagSerialNumber[i])
            {
                 break;     // if not equal, then break out of the "for"
loop
            }

            if (i == 3)
            {

                for(i=0; i < 4; i++)
                {
                    GoodTagSerialNumber[j][i] = 0x00;
                }

                tag_last_free=j;

                lcd.clear();
                lcd.setCursor(1,3);
                lcd.print("Deleted !");
                delay(1000);
                break;
            }
        } }
    }
    lcd.clear();
}

/*********************************************************************
*****************************/



void card_Check()
{
    // Check if detected Tag has the right Serial number we are looking
for
            for(int j=0; j < 5; j++)
            {

    for(int i=0; i < 4; i++)
```

```
    {
        if (GoodTagSerialNumber[j][i] != TagSerialNumber[i])
        {
             break;     // if not equal, then break out of the "for"
loop
        }
        if (i == 3)
        {
            GoodTag="TRUE";     // if we made it to 4 loops then the Tag
Serial numbers are matching
        }
    }}
    if (GoodTag == "TRUE")
    {
       access_Approved_msg();
    }
    else
    {
         access_Decline_msg();
    }
}




/*********************************************************************
*******************************/




void card_Scan()
{
  clearData();
  delay(200);
  // Get anti-collision value to properly read information from the Tag
  ReadTag = nfc.antiCollision(TagData);
  memcpy(TagSerialNumber, TagData, 4); // Write the Tag information in
the TagSerialNumber variable
  /*lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Tag detected.");
  lcd.setCursor(0,1);
  lcd.print("Serial Number: ");
```

```
  lcd.setCursor(0,2);
  for (int i = 0; i < 4; i++)
  { // Loop to print serial number to serial monitor
      lcd.print(TagSerialNumber[i], HEX);
      lcd.print(", ");
  } */
  delay(1000);
}


/*****************************************************************
*******************************/

void lock_device()
{
    for(int i=59; i>=0; i--)
    {
      lcd.clear();
      lcd.setCursor(1,0);
      lcd.print("SYSTEM IS DISABLED");
      lcd.setCursor(0,2);
      lcd.print("Please Try Again in");
      lcd.setCursor(9,3);
      lcd.print(i);
      delay(1000);
    }
    lock_counter_flag=0;
    lcd.clear();
}


/*****************************************************************
*******************************/

void enter_Credentials_msg()
{
    t = rtc.getTime();
    while( (t.hour >= working_hour_to) || (t.hour <= working_hour_from)
)
    {
        lcd.setCursor(3,1);
        lcd.print("No Entry Mode");
    }
    lcd.setCursor(12,0);
```

```
    lcd.print(rtc.getTimeStr());
    lcd.setCursor(0,0);
    lcd.print(rtc.getDateStr());
    lcd.setCursor(2,2);
    lcd.print("TAP OR ENTER PIN");
}


/*********************************************************************
*******************************/



void access_Decline_msg()
{
    lcd.clear();
    lcd.setCursor(2,1);
    lcd.print("Access Declined");
    delay(1500);
    lock_counter_flag++;
    if(lock_counter_flag==3)
    {
      lock_device();
    }
    lcd.clear();
}


/*********************************************************************
*******************************/



void access_Approved_msg()
{
      lcd.clear();
      lcd.setCursor(2,1);
      lcd.print("Access Approved");
      digitalWrite (Buzzer, LOW) ;// Buzzer Off
      delay (400) ;// delay 1ms
      digitalWrite(Buzzer, HIGH); // Buzzer Off at startup
      digitalWrite(Relay, HIGH);
      delay(5000);
      while(digitalRead(Mag_Switch) == HIGH ){}
```

```
      delay(1000);
      digitalWrite(Relay, LOW);
      lock_counter_flag=0;
      lcd.clear();

}


/******************************************************************
*******************************/



void check_if_Password_entered()
{
      if(data_count == Password_Lenght-1) // if the array index is equal
to the number of expected chars, compare data to master
      {
            if(!strcmp(Data, Master)) // equal to (strcmp(Data, Master)
== 0)
            {
                access_Approved_msg();
            }
            else
            {
                access_Decline_msg();
            }
            clearData();
      }
}



/******************************************************************
*******************************/



void password_Change()
{
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("CHANGE YOUR PASSWORD");
        lcd.setCursor(1,2);
```

```
        lcd.print("Enter Old Password");
         while(1)
          {
             customKey = customKeypad.getKey();
             if (customKey) // makes sure a key is actually pressed,
equal to (customKey != NO_KEY)
             {
                  if(numEntered())
                  {
                    store_data();
                  }
             }

             if(data_count == Password_Lenght-1) // if the array index
is equal to the number of expected chars, compare data to master
              {

                   if(!strcmp(Data, Master)) // equal to (strcmp(Data,
Master) == 0)
                   {
                       lcd.clear();
                       lcd.setCursor(2,1);
                       lcd.print("PASSWORD ok");
                       delay(2000);
                       lcd.clear();
                       lcd.setCursor(0,0);
                       lcd.print("CHANGE YOUR PASSWORD");
                       lcd.setCursor(1,2);
                       lcd.print("Enter New Password");
                       data_count=0;
                       while(1)
                       {
                            customKey = customKeypad.getKey();
                           if (customKey) // makes sure a key is
actually pressed, equal to (customKey != NO_KEY)
                           {
                               if(numEntered())
                               {
                                 store_data();
                               }
                           }
```

```
                            if(data_count == Password_Lenght-1) // if
the array index is equal to the number of expected chars, compare data
to master
                            {
                                    strcpy(Master,Data);
                                    lcd.clear();
                                    lcd.setCursor(2,1);
                                    lcd.print("PASSWORD CHANGED");
                                    delay(1500);
                                    clearData();
                                    lcd.clear();
                                    break;
                            }
                        }
                        break;

                    }
                    else
                    {
                        lcd.clear();
                        lcd.setCursor(2,1);
                        lcd.print("WRONG PASSWORD !");
                        delay(1500);
                        lcd.clear();
                    }
                    clearData();
                    break;
                }
            }
}


/*****************************************************************
*******************************/




int numEntered()
{
      if (customKey == '0' || customKey == '1' || customKey == '2' ||
customKey == '3' ||
          customKey == '4' || customKey == '5' || customKey == '6' ||
customKey == '7' ||
```

```
          customKey == '8' || customKey == '9' )
      {
          return 1;
      }
      else
      {
          return 0;
      }
}




/*********************************************************************
*******************************/



void store_data()
{
    Data[data_count] = customKey; // store char into data array
    lcd.setCursor(8+data_count,3);
    lcd.print("*");
    data_count++; // increment data array by 1 to store new char, also
keep track of the number of chars entered
    digitalWrite (Buzzer, LOW) ;// Buzzer Off
    delay (10) ;// delay 1ms
    digitalWrite(Buzzer, HIGH); // Buzzer Off at startup
}



/*********************************************************************
*******************************/



void clearData()
{
  while(data_count !=0)
  {   // This can be used for any array size,
```

```
    Data[data_count--] = 0; //clear array for new data
  }
  GoodTag="False";
  return;
}


/*********************************************************************
*******************************/



void smsSend()
{

    char phone_no[]="6981468242";
    Serial.begin(9600);
    delay(300);
    Serial.println("AT+CMGF=1");
    delay(2000);
    Serial.print("AT+CMGS=\"");
    Serial.print(phone_no);
    Serial.write(0x22);
    Serial.write(0x0D);  // hex equivalent of Carraige return
    Serial.write(0x0A);  // hex equivalent of newline
    delay(2000);
    Serial.print("Door has opened ubnormally.");
    delay(500);
    Serial.println (char(26));//the ASCII code of the ctrl+z is 26
    delay(10000);

}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] "Arduino - Introduction". arduino.cc.

[2] Justin Lahart (27 November 2009). "Taking an Open-Source Approach to Hardware". The Wall Street Journal. Retrieved 2014-09-07.

[3] "How many Arduinos are "in the wild?" About 300,000". Adafruit Industries. May 15, 2011. Retrieved 2013-05-26.

[4] Arduino Uno official webpage (arduino.cc)

[5] Augarten, Stan (1983). The Most Widely Used Computer on a Chip: The TMS 1000. State of the Art: A Photographic History of the Integrated Circuit. New Haven and New York: Ticknor & Fields. ISBN 0-89919-195-9. Retrieved 2009-12-23

[6] "Oral History Panel on the Development and Promotion of the Intel 8048 Microcontroller" (PDF). Computer History Museum Oral History, 2008. p. 4. Retrieved 2016-04-04.

[7] "Atmel's Self-Programming Flash Microcontrollers" (PDF). 2012-01-24. Retrieved 2008-10-25. by Odd Jostein Svendsli 2003

[8] Edwards, Robert (1987). "Optimizing the Zilog Z8 Forth Microcontroller for Rapid Prototyping" (PDF). Martin Marietta: 3. Retrieved 9 December 2012.

[9] "8052-Basic Microcontrollers" by Jan Axelson 1994