



ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ
ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

{RFID Σύστημα Διαχείρισης Αντικειμένων}



ΚΩΝΣΤΑΝΤΙΝΟΣ ΤΣΑΓΚΑΡΑΚΗΣ | ΑΜ: 2017/032
ΜΑΪΟΣ - ΧΡΗΣΤΟΣ ΙΟΡΔΑΝΙΔΗΣ | ΑΜ: 2017/136

ΕΠΙΒΛΕΠΩΝ
ΑΝΑΠΛΗΡΩΤΗΣ ΚΑΘΗΓΗΤΗΣ
ΔΗΜΗΤΡΙΟΣ ΜΠΕΧΤΣΗΣ
ΘΕΣΣΑΛΟΝΙΚΗ {2023}

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Διεθνούς Πανεπιστημίου Ελλάδος.

Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν.1256/1982, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον.

Ονοματεπώνυμο φοιτητών

ΚΩΣΤΑΝΤΙΝΟΣ ΤΣΑΓΚΑΡΑΚΗΣ,

ΜΑΪΟΣ - ΧΡΗΣΤΟΣ ΙΟΡΔΑΝΙΔΗΣ

(Υπογραφές)

Ευχαριστίες

Πρωτίστως θα θέλαμε να ευχαριστήσουμε θερμά τον υπεύθυνο καθηγητή κ. Δημήτριο Μπεχτσή για την καθοδήγηση και συμβολή του στην εκπόνηση και ολοκλήρωση της διπλωματικής μας εργασίας, και για την εμπιστοσύνη που μας έδειξε κατά τη διάρκεια αυτής. Επίσης στα υπόλοιπα μέλη της εξεταστικής επιτροπής για την προσεκτική ανάγνωση της εργασίας.

Πάνω απ' όλα θα θέλαμε να ευχαριστήσουμε από καρδιάς καταρχάς τους γονείς μας για την στήριξη και την εμπιστοσύνη που μας έδειξαν κατά τη διάρκεια εκπόνησης της παρούσας διπλωματικής εργασίας όπως και για την αγάπη τους όλα αυτά τα χρόνια.

Περίληψη

Ζούμε σε μια κοινωνία ανάπτυξης και τεχνολογίας, μια κοινωνία στην οποία η ανάπτυξη των τεχνολογικών μέσων εξελίσσεται ραγδαία. Νέα τεχνολογικά ευρήματα έρχονται στο προσκήνιο καθιστώντας την καθημερινότητά μας, άρρηκτα συνδεδεμένη με αυτά.

Ένα παράδειγμα νέων τεχνολογιών είναι και το RFID (Radio Frequency Identification) ή αλλιώς «ταυτοποίηση μέσω ραδιοκυμάτων», η οποία αποτελεί μία εφαρμογή που μπορεί να χρησιμοποιηθεί σε πλήθος τομέων όπως στην υγεία, στη βιομηχανία, στην εφοδιαστική αλυσίδα αλλά και στην σε εφαρμογές της καθημερινότητας όπως στην πληρωμή διοδίων, στα διαβατήρια, στις πιστωτικές κάρτες, για έλεγχο πρόσβασης σε κτίρια, στα κλειδιά αυτοκινήτων και άλλα, με σκοπό την αύξηση της ποιότητας των παρεχόμενων υπηρεσιών με ταυτόχρονη μείωση του κόστους. Η τεχνολογία RFID είναι μία μέθοδος αυτόματου εντοπισμού και αναγνώρισης αντικειμένων, που βασίζεται στην αποθήκευση και στην ασύρματη ανάκτηση δεδομένων, τα οποία αποθηκεύονται σε ειδικές μικροσκοπικές συσκευές που ονομάζονται ετικέτες RFID (RFID tags) ή αναμεταδότες (transponders).

Μέσω της παρούσας διπλωματικής εργασίας με τίτλο «RFID Σύστημα Διαχείρισης Αντικειμένων» δίνετε μια ολοκληρωμένη εικόνα για την λειτουργία ενός RFID συστήματος. Τα RFID συστήματα αποτελούνται από δυο μέρη: Το υλικό (hardware) και το λογισμικό (software) κομμάτι. Στα επόμενα κεφάλαια θα παρουσιαστεί ο τρόπος σύνδεσης των επιμέρους εξαρτημάτων μεταξύ τους αλλά και η επικοινωνία με τον ηλεκτρονικό υπολογιστή, η ανάπτυξη μιας εφαρμογής για την απεικόνιση των αποτελεσμάτων καθώς και οι κώδικες που χρησιμοποιήθηκαν για την ολοκλήρωση της εργασίας.

Η διπλωματική παρουσίαση χωρίζεται σε πέντε κεφάλαια τα οποία καλύπτουν το μεγαλύτερο εύρος της τεχνολογίας RFID και τον τρόπο λειτουργίας του υπό εξέταση συστήματος. Τα κεφάλαια που απαρτίζουν την εργασία είναι τα εξής:

Το πρώτο κεφάλαιο στο οποίο γίνεται η περιγραφή ενός συστήματος RFID, ο τρόπος εφαρμογής στην καθημερινή ζωή καθώς και μια ιστορική αναφορά στην τεχνολογία που χρησιμοποιήθηκε. Ακολουθεί το δεύτερο κεφάλαιο το οποίο περιλαμβάνει μια αναλυτική παρουσίαση των εξαρτημάτων, του τρόπου σύνδεσης μεταξύ τους και την επικοινωνία με τον ηλεκτρονικό υπολογιστή. Το τρίτο κεφάλαιο εμβαθύνει στο λογισμικό (Software) του συστήματος. Γλώσσες προγραμματισμού που χρησιμοποιήθηκαν, παρουσίαση και εγκατάσταση των προγραμμάτων καθώς και manuals χρήσης της εφαρμογής είναι κάποια από τα σημαντικότερα υποκεφάλαια της ενότητας αυτής.

Στο τέταρτο κεφάλαιο θα γίνει αναλυτικότερη παρουσίαση της εφαρμογής, μια περιγραφή του γραφικού περιβάλλοντος, του τρόπου λειτουργίας της και της σύνδεσης του προγράμματος με την βάση δεδομένων. Τέλος στο πέμπτο κεφάλαιο γίνεται μια σύγκριση των τεχνολογικών μέσων της παρούσας εργασίας σε σχέση με αντίστοιχα της αγοράς και εξετάζονται τα περιθώρια βελτίωσης του εξοπλισμού.

Abstract

We live in a society of development and technology, a society in which the development of technological means is developing rapidly. New technological discoveries are coming to the fore, making our everyday life inextricably linked with them.

A prime example of such technologies is that of RFID (Radio Frequency Identification) technology, or otherwise "identification through radio waves", which is an application that can be used in a number of sectors such as health, industry, the supply chain and also in applications of daily life such as paying tolls, passports, credit cards, to control access to buildings, car keys, etc. with the aim of increasing the quality of the services provided while reducing costs. RFID technology is a method of automatic location and identification of objects, based on the storage and wireless retrieval of data, which are stored in special microscopic devices called RFID tags, or transponders.

Through this thesis entitled "RFID Asset Management System" a comprehensive picture of the operation of an RFID system is given. RFID systems consist of two parts: the hardware and the software part. The next chapters will show how to connect the individual components to each other, as well as their communication with the computer, the development of an application to display the results as well as the codes that were needed for the proper operation of the project.

The diplomatic presentation is divided into five chapters which cover the wider range of RFID technology and how the system under review works. The chapters that make up the work are the following:

The first chapter in which the description of an RFID system is made, the way of application in daily life as well as a historical reference to the technology used. The second chapter follows, which includes a detailed presentation of the components, how they are connected to each other and their communication with the computer. The third chapter delves into the software of the system. Programming languages, presentation and installation of important programs as well as user manuals for the application are some categories of this chapter.

In the fourth chapter there will be a more detailed presentation of the application, a description of the graphical environment, how it functions and the connection of the program to the database. Finally, in the fifth chapter a comparison of the technological means used in writing this paper is made in relation to market counterparts, while the margins for equipment improvement are examined.

ΠΕΡΙΕΧΟΜΕΝΑ	Σελ.
Κατάλογος Εικόνων	8
Κατάλογος Πινάκων	10
Κεφάλαιο 1: Περιγραφή της εφαρμογής	11
1.1 Γενικά	11
1.2 Ιστορική αναφορά στην τεχνολογία που χρησιμοποιήθηκε	11
1.3 Τρόπος λειτουργίας	13
1.3.1 Τύποι συστημάτων RFID	18
1.4 Πλεονεκτήματα και μειονεκτήματα	20
1.5 Πρακτικές εφαρμογές	21
Κεφάλαιο 2: Εξοπλισμός, Σύνδεση περιφερειακών, Επικοινωνία με Υ/Η	25
2.1 Παρουσίαση των εξαρτημάτων του συστήματος	25
2.1.1 Πλακέτα μικροελεγκτή Arduino	25
2.1.2 Sparkfun Rfid Reader M6E-NANO	26
2.1.3 Sparkfun UHF RFID Antenna	27
2.1.4 Ετικέτες RFID (Tags)	28
2.2 Επικοινωνία του συστήματος με τον υπολογιστή	29
2.2.1 Ενσύρματη επικοινωνία	29
2.2.2 Προσπάθεια ενσωμάτωσης πλακέτας Wi-Fi Shield	29
Κεφάλαιο 3: Προγράμματα (Software) και manual σύνδεσης της εφαρμογής	30
3.1 Ιστορική αναφορά σε γλώσσες προγραμματισμού	30
3.2 Γλώσσα προγραμματισμού Python	32
3.2.1 Γιατί επιλέχθηκε η συγκεκριμένη γλώσσα	32
3.2.2 PyCharm	33
3.3 Βάση δεδομένων SQL	34
3.3.1 MySQL	34
3.4 Arduino IDE	35
3.5 Εγχειρίδιο χρήσης εφαρμογής (User Manual)	37
3.5.1 Σύνδεση Python με MySQL database	37
3.5.2 Σύνδεση Arduino με Python	38
3.5.3 Δημιουργία πινάκων SQL	38
3.5.4 Δημιουργία κωδικού πρόσβασης	40
Κεφάλαιο 4: Παρουσίαση εργασίας- Περιγραφή εφαρμογής	41
4.1 Σύντομη περιγραφή της εφαρμογής	41
4.2 Γραφικό περιβάλλον και λειτουργίες	42

Κεφάλαιο 5: Έρευνα αγοράς, περιθώρια βελτίωσης	45
5.1 Εισαγωγή στις εταιρίες RFID	45
5.2 Έρευνα αγοράς	46
5.2.1 Κεραίες RFID	47
5.2.2 Αναγνώστες RFID	49
5.2.3 Ετικέτες RFID	51
5.3 Προτάσεις βελτίωσης του υπάρχοντος εξοπλισμού	54
Βιβλιογραφία	55
Διαδικτυακές Πηγές	56
Παράρτημα Α	59
Παράρτημα Β	92

Κατάλογος Εικόνων

Εικόνα 1.....	Εξέλιξη RFID
Εικόνα 2.....	Ετικέτα RFID
Εικόνα 3.....	Παθητικές ετικέτες RFID
Εικόνα 4.....	Ημιπαθητική ετικέτα RFID
Εικόνα 5.....	Ενεργητική ετικέτα RFID
Εικόνα 6.....	Αναγνώσιμη ετικέτα RFID
Εικόνα 7.....	Υλικά κατασκευής ετικετών RFID
Εικόνα 8.....	Διάφορες Μορφές Ετικετών RFID
Εικόνα 9.....	Αναγνώστες RFID
Εικόνα 10.....	Κεραία RFID
Εικόνα 11.....	Τρόπος λειτουργίας Κεραίας RFID
Εικόνα 12.....	Middleware
Εικόνα 13.....	Τρόπος λειτουργίας συστήματος RFID
Εικόνα 14.....	Περιοχές συχνοτήτων UHF RFID
Εικόνα 15.....	Σύζευξη backscatter (οπισθοσκέδαση)
Εικόνα 16.....	Πρακτικές εφαρμογές συστημάτων RFID
Εικόνα 17.....	Εφαρμογή συστημάτων RFID στα Logistics
Εικόνα 18.....	Στοιχεία Έξυπνων Πόλεων
Εικόνα 19.....	Εφαρμογές RFID στην υγεία
Εικόνα 20.....	Πλακέτα Arduino
Εικόνα 21.....	Αναγνώστης (Reader) M6E-NANO
Εικόνα 22.....	Κεραία (Antenna) Sparkfun UHF RFID
Εικόνα 23.....	Παθητικές ετικέτες (tags)
Εικόνα 24.....	Αναλυτική Μηχανή
Εικόνα 25.....	ENIAC - Ο πρώτος Η/Υ
Εικόνα 26.....	Γλώσσες Προγραμματισμού
Εικόνα 27.....	Απεικόνιση serial plotter με δεδομένα
Εικόνα 28.....	Παράδειγμα συμπληρωμένου πίνακα com_port_mapping.
Εικόνα 29.....	Οι πίνακες της βάσης
Εικόνα 30.....	Παράθυρο εισαγωγής κωδικού
Εικόνα 31.....	Γραφικό περιβάλλον εφαρμογής

Εικόνα 32.....Παράθυρο λειτουργίας της εφαρμογής με
καταχωρημένα δεδομένα

Εικόνα 33.....Χρωματική απεικόνιση των τιμών R_{ssi}

Κατάλογος Πινάκων

Πίνακας 1.....	Χαρακτηριστικά τύπων συστημάτων RFID
Πίνακας 2.....	Στοιχεία πλακετών Arduino
Πίνακας 3.....	Datasheet της πλακέτας Sparkfun
Πίνακας 4.....	Datasheet της κεραίας Sparkfun

Κεφάλαιο 1: Περιγραφή της εφαρμογής

1.1 Γενικά

Σκοπός της παρούσας εργασίας είναι η ανάγνωση και καταγραφή διάφορων tags ανάλογα με την τοποθεσία τους στο χώρο. Στο πλαίσιο αυτό αναπτύχθηκε μια εφαρμογή μέσω της οποίας ο αναγνώστης είναι ικανός να εντοπίσει εάν ένα αντικείμενο βρίσκεται εντός ή εκτός ενός δωματίου, όπως επί παραδείγματι αν τα προϊόντα ενός καταστήματος έχουν μεταφερθεί έξω από αυτό.

Ξεκινώντας την διπλωματική παρουσίαση της εν λόγω εργασίας κρίθηκε σκόπιμο να γίνει μια εισαγωγική αναφορά σε κάποια γενικά αντικείμενα σχετικά με αυτήν, ενώ στη συνέχεια θα ακολουθήσει μία αναλυτική παρουσίαση των επιμέρους τμημάτων που απαρτίζουν το σύνολο της εργασίας.

Το υλικό (hardware), το λογισμικό (software), ο τρόπος λειτουργίας της εφαρμογής και τα περιθώρια βελτίωσης είναι κάποια από τα κεφάλαια που θα αναπτυχθούν παρακάτω.

Ορισμένες διατάξεις στις οποίες μπορεί να χρησιμοποιηθεί και να βρει εφαρμογή το υπόψη σύστημα είναι τα συστήματα ασφαλείας και τα συστήματα παρακολούθησης ή καταγραφής.

1.2 Ιστορική αναφορά στην τεχνολογία που χρησιμοποιήθηκε

Όπως γίνεται αντιληπτό από το θέμα της εργασίας που τιτλοφορείται «RFID Σύστημα Διαχείρισης Αντικειμένων» βασικό αντικείμενο ενασχόλησης θα αποτελέσει η RFID τεχνολογία. Η συντομογραφία RFID είναι τα αρχικά των λέξεων **Radio Frequency IDentification**, με την ελληνική μετάφραση να αποδίδεται ως «Ταυτοποίηση μέσω ραδιοσυχνοτήτων».

Το RFID είναι η επόμενη γενιά τεχνολογίας barcoding. Αντί για μια σειρά τυπωμένων γραμμών που απλά τυπώνονται σε μια ετικέτα, μια ετικέτα RFID αποτελεί ένα μικροσκοπικό ηλεκτρονικό κύκλωμα. Η πιο απλή μορφή ετικέτας RFID μοιάζει πολύ με μια ετικέτα γραμμικού κώδικα. Αλλά αντί να σαρωθεί από ένα λείζερ που πρέπει να "βλέπει" τον γραμμικό κώδικα, η ετικέτα RFID πρέπει απλώς να περάσει κοντά από έναν ειδικά εξοπλισμένο πομποδέκτη RFID. Ο πομποδέκτης βομβαρδίζει την ετικέτα με αόρατα ραδιοκύματα, ενεργοποιώντας έτσι το κύκλωμα της ετικέτας RFID, το οποίο στέλνει τις πληροφορίες του πίσω σε αυτόν. Η ετικέτα είναι εντελώς παθητική και δεν χρειάζεται να περιέχει πηγή ενέργειας. Οι ετικέτες RFID συνήθως αποθηκεύουν έναν μοναδικό κωδικό 64-bit. Ένας τυπικός εκτυπωμένος γραμμικός κώδικας, όπως ένας γενικός κωδικός προϊόντος, μπορεί να αποθηκεύσει μόνο 11 ψηφία, αλλά τα 64 bit, περίπου ισοδύναμα με έναν αριθμό 19 ψηφίων, επιτρέπουν αρκετά περισσότερους συνδυασμούς.¹

¹ The Kimball Group Reader: *Relentlessly Practical Tools for Data Warehousing and Business Intelligence*, 2nd Edition John Wiley & Sons, Indianapolis USA, 2016

Η τεχνολογία RFID εμφανίζεται για πρώτη φορά κατά την διάρκεια του Β' Παγκοσμίου Πολέμου (1939-1945) στην πολεμική αεροπορία της Αγγλίας (RAF) και έπαιξε σημαντικό ρόλο στο διαχωρισμό φίλων και εχθρικών αεροσκαφών. Τα αεροσκάφη της Βρετανικής Αεροπορίας έφεραν το σύστημα IFF (Identify Friendly Foe), ένα σύστημα πομπού - δέκτη για το διαχωρισμό των συμμαχικών από τα αντίπαλα αεροπλάνα.

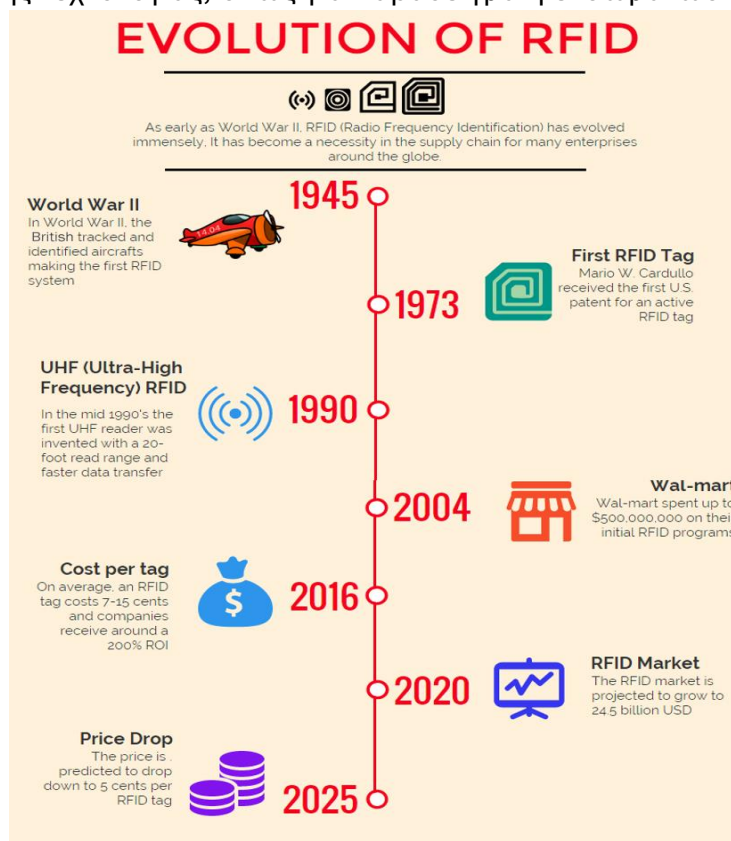
Οι έρευνες ανάπτυξης και εξέλιξης της RFID τεχνολογίας συνεχίστηκαν μετά τον Β' Παγκόσμιο Πόλεμο ενώ στην δεκαετία του 1970 εμφανίστηκαν τα πρώτα παθητικά συστήματα τα οποία με απλές κεραίες και αναμεταδότες παρακολουθούσαν διαφορά αντικείμενα. Η ανάπτυξη αυτών το συστημάτων δεν πέρασε απαρατήρητη από την βιομηχανία και έτσι την περίοδο 1980-1990 αναπτύσσονται οι πρώτες εμπορικές εφαρμογές σε συστήματα ασφάλειας και ελέγχου ενώ παράλληλα τα ραδιοκύματα χαμηλής και μεσαίας συχνότητας εξελίσσονται σε εξαιρετικά υψηλής (Ultra- High Frequency).

Από το 2000 και ειδικότερα από το 2010 μέχρι και σήμερα έχουν γίνει σημαντικές αλλαγές στον τομέα της συγκεκριμένης τεχνολογίας, όπως για παράδειγμα η ενσωμάτωση της Rfid τεχνολογίας στο IoT (Internet of Things) ανοίγοντας έτσι τους δρόμους για μια παγκόσμια πλέον παρακολούθηση. Εκτός αυτού, μια εφαρμογή του Rfid γίνεται ιδιαίτερα δημοφιλής στο ευρύ κοινό για την ευχρηστία του και την απλοποίηση ανέπαφων συναλλαγών. Ο λόγος γίνεται για την NFC (Near Field Communication) τεχνολογία η οποία βασίζεται στην επαφή ή στην προσέγγιση δυο συσκευών.

Οι ετικέτες RFID εκμεταλλεύονται τον μεγαλύτερο χώρο αποθήκευσης κώδικα για να επιτρέπουν την παρακολούθηση της ύπαρξης κάθε προϊόντος. Καθώς η ετικέτα RFID πρέπει απλώς να περάσει κοντά από έναν πομποδέκτη RFID, κάθε πόρτα μπορεί να εξοπλιστεί για να

ανιχνεύει τη διέλευση μιας ετικέτας RFID σε ένα αντικείμενο. Οι εταιρείες που ασχολούνται με τη διαχείριση της εφοδιαστικής αλυσίδας παρακολουθούν ήδη τη μετακίνηση μεμονωμένων προϊόντων από τη γραμμή συναρμολόγησης, σε μια παλέτα, στον όροφο της αίθουσας αποστολής, σε ένα φορτηγό και από το φορτηγό σε μια απομακρυσμένη τοποθεσία παράδοσης. Κάθε μία από αυτές τις φυσικές τοποθεσίες είναι εξοπλισμένη με έναν πομποδέκτη RFID και φυσικά, αυτός ο όγκος δεδομένων πηγαίνει σε μια βάση δεδομένων.

Στον τομέα του λιανεμπορίου, η τεχνολογία RFID θα επέτρεπε σε έναν αγοραστή να περάσει απλώς με το καλάθι αγορών του από μια πόρτα με δυνατότητα RFID, όπου θα



Εικόνα 1. Εξέλιξη RFID

ανιχνευόταν ολόκληρο το καλάθι και θα καταγραφόταν το χρηματικό σύνολο των προϊόντων. Σε περίπτωση που ο καταναλωτής θα διέθετε μια πιστωτική κάρτα με σύστημα RFID, θα είχε τη δυνατότητα να εγκρίνει σε ένα αντίστοιχο μηχάνημα την συναλλαγή και να αποχωρήσει από το κατάστημα αγορών.²

Σήμερα τα συστήματα που βασίζονται στην τεχνολογία RFID κατακλύζουν τον κόσμο. Καθημερινά όλο και περισσότεροι κλάδοι τείνουν να τα ενσωματώνουν στο εργασιακό τους περιβάλλον με την τελευταία εξέλιξη να είναι η συνένωση RFID και τεχνητής νοημοσύνης - Artificial Intelligence (AI).

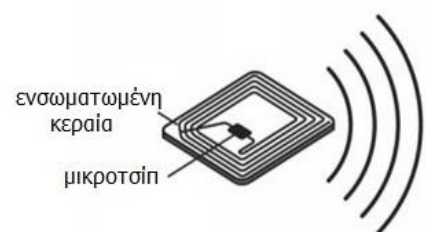
1.3 Τρόπος λειτουργίας

Στην ουσία, ένα σύστημα RFID είναι απλώς ένας αναγνώστης και μια ετικέτα που επικοινωνούν μέσω μιας συγκεκριμένης συχνότητας, όπως κάθε άλλη ραδιοεπικοινωνία. Οι αναγνώστες, οι κεραίες, οι ετικέτες και η συχνότητα αποτελούν τα βασικά στοιχεία ενός συστήματος RFID³ και στις ακόλουθες ενότητες θα παρουσιαστεί μια επισκόπηση του τρόπου λειτουργίας τους.

Η τεχνολογία αναγνώρισης μέσω ραδιοσυχνοτήτων (RFID) βασίζεται στην απλή ιδέα ότι υπάρχει ένα ηλεκτρονικό κύκλωμα σε μια μη-τροφοδοτούμενη («παθητική») ετικέτα και δεν απαιτεί την παροχή ενέργειας ή κάποια συντήρηση. Το κύκλωμα αυτό μπορεί να τροφοδοτείται περιστασιακά εξ' αποστάσεως από μία διάταξη (ή συσκευή) ανάγνωσης, μέσω εκπομπής ενέργειας προς αυτό. Δεδομένου του τρόπου τροφοδότησης, η ετικέτα ανταλλάσσει πληροφορίες με τη συσκευή ανάγνωσης. Η ετικέτα συνίσταται από ένα απλό πηνίο κεραίας μέσα σε μια θήκη από γυαλί ή πλαστικό, συγκολλημένο στο ολοκληρωμένο κύκλωμα. Η διασύνδεση είναι ασύρματη και βασίζεται στα ραδιοκύματα τα οποία μεταδίδονται στον αέρα. Παράλληλα η αναγνώριση αντικειμένων δεν απαιτεί οπτική επαφή (σε αντίθεση με τον γραμμωτό κώδικα (barcode) που έχει μέσο διασύνδεσης τις υπέρυθρες και απαιτεί οπτική επαφή).

Σε ένα βασικό σύστημα RFID, απαιτούνται τέσσερα βασικά συστατικά για να επιτευχθεί η μετάδοση των δεδομένων:

- Ο transponder (που καλείται και απλά tag - ετικέτα) το οποίο προγραμματίζεται με πληροφορία που το αναγνωρίζει μοναδικά, καθορίζοντας έτσι το concept του “automatic identification”. Ο πομποδέκτης είναι τόσο μικρός που χωράει σε μια μικρή αυτοκόλλητη ετικέτα (ταμπελάκι), σε μια κάρτα ή σε ένα μικρό μπρελόκ και αποτελείται όπως φαίνεται στην εικόνα 2, από ένα μικροτσίπ με ενσωματωμένη κεραία για να στέλνει τα δεδομένα στη συσκευή ανάγνωσης.



Εικόνα 2. Ετικέτα RFID

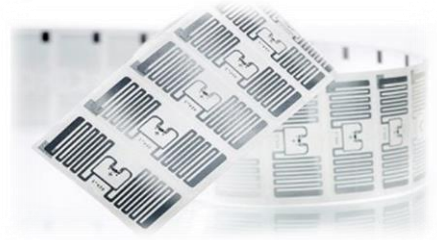
² The Kimball Group Reader: *Relentlessly Practical Tools for Data Warehousing and Business Intelligence*, 2nd Edition John Wiley & Sons, Indianapolis USA, 2016

³ Sweeney P. J., “*RFID for Dummies*”, 1st Edition, Wiley Publishing Inc., Indiana USA, 2005

Οι ετικέτες γενικά κατηγοριοποιούνται:

➤ Ανάλογα με την πηγή της ενεργειάς τους σε παθητικές (passive), ημιπαθητικές-ημιενεργητικές (Semi-passive or semi active) και σε ενεργητικές (active).

Οι παθητικές ετικέτες δεν τροφοδοτούνται από εσωτερική πηγή ενέργειας (μπαταρία). Η κεραία τους είναι κατά τέτοιο τρόπο σχεδιασμένη, ώστε να παίρνει ενέργεια από το σήμα που λαμβάνει και απαντά στέλνοντας σήμα το οποίο λαμβάνεται από τον αναγνώστη. Η τιμή των παθητικών ετικετών, για ποσότητες 10 εκατομμυρίων τεμαχίων, είναι περίπου 5 cents ανά τεμάχιο.⁴



Εικόνα 3. Παθητικές Ετικέτες RFID



Οι ημιπαθητικές ετικέτες σε αντίθεση με τις παθητικές, έχουν δική τους πηγή ενέργειας, αλλά η μπαταρία τους χρησιμοποιείται μόνο για να τροφοδοτεί με ενέργεια το μικροτσίπ και όχι για να εκπέμπει σήματα. Τα ραδιοκύματα αντανακλώνονται πίσω και λαμβάνονται από τον αναγνώστη, όπως συμβαίνει με τις παθητικές ετικέτες.

Εικόνα 4. Ημιπαθητική Ετικέτα RFID

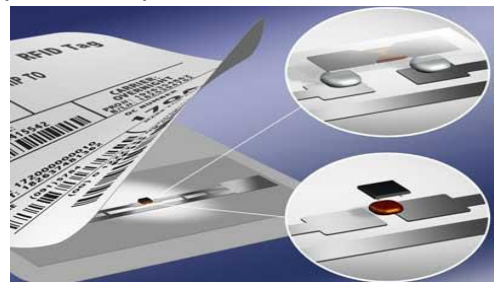
Τέλος, οι ενεργητικές ετικέτες, όπως και οι ημιπαθητικές, έχουν δική τους πηγή ενέργειας, με τη διαφορά ότι αυτή χρησιμοποιείται και για να τροφοδοτεί το ολοκληρωμένο κύκλωμα, αλλά και να εκπέμπει το σήμα στον αναγνώστη (reader). Πολλές ενεργές ετικέτες έχουν δραστηκή εμβέλεια περίπου 100 μέτρων και η διάρκεια ζωής της μπαταρίας τους είναι περίπου 10 χρόνια.



Εικόνα 5. Ενεργητική Ετικέτα RFID

➤ Ανάλογα με τη δυνατότητα επανεγγραφής διακρίνονται σε αναγνώσιμες (Read only), μίας εγγραφής-πολλών αναγνώσεων (Write Once Read Many) και επανεγγράψιμες (Read - Write).

Στις αναγνώσιμες (read only) ετικέτες η αναγνώριση της ταυτότητας κωδικοποιείται στο στάδιο της παραγωγής της και δεν υπάρχει δυνατότητα επανεγγραφής, ενώ η αποθήκευση των δεδομένων ασφαλείας ακολουθείται από έναν μοναδικό σειριακό αριθμό. Η διαφορά των επανεγγράψιμων (Read - Write) από τις αναγνώσιμες (Read only) ετικέτες είναι ότι οι πρώτες διαθέτουν τσιπ ανάγνωσης-εγγραφής, όπου μπορεί κάποιος να προσθέσει πληροφορίες στην ετικέτα ή να γράψει πάνω από υπάρχουσες πληροφορίες όταν η ετικέτα βρίσκεται εντός εμβέλειας ενός αναγνώστη.⁵



Εικόνα 6. Αναγνώσιμη ετικέτα RFID

⁴ <https://www.pemptousia.gr/2021/06/rfid-radio-frequency-identification/>

⁵ <https://www.rfidjournal.com/faq/whats-the-difference-between-read-only-and-read-write-rfid-tags>

➤ Ανάλογα με το υλικό κατασκευής τους διακρίνονται σε πλαστικές, γυάλινες, υφασμάτινες, χάρτινες, αυτοκόλλητες.⁶



Εικόνα 7. Υλικά κατασκευής ετικετών RFID

➤ Ανάλογα με την μορφή τους διακρίνονται σε κάρτα, εμφύτευμα, μπρελόκ, ετικέτα, χάρτινο εισιτήριο, ταινία βραχιόλι.



Εικόνα 8. Διάφορες Μορφές Ετικετών RFID

• Ο transceiver (που καλείται κυρίως και reader - αναγνώστης) είναι μια συσκευή που χρησιμοποιείται για την ανάγνωση πληροφοριών από και πιθανώς επίσης την εγγραφή πληροφοριών σε ετικέτες RFID. Ένας αναγνώστης είναι συνήθως συνδεδεμένος με μια βάση δεδομένων back-end για την κωδικοποίηση πληροφοριών σε αυτήν τη βάση

δεδομένων για περαιτέρω επεξεργασία. Η μονάδα ελέγχου ενός αναγνώστη έχει τρεις βασικές λειτουργίες που ελέγχουν την επικοινωνία μεταξύ αυτού και των ετικετών RFID, την κωδικοποίηση και την αποκωδικοποίηση σημάτων και την επικοινωνία με τον διακομιστή υποστήριξης για την αποστολή πληροφοριών στη βάση δεδομένων



Εικόνα 9. Αναγνώστες RFID

⁶ <https://gaorfid.com/el/rfid-tags-market-report/>

υποστήριξης ή εκτέλεση των εντολών από τον διακομιστή υποστήριξης. Η μονάδα ελέγχου έχει τη δυνατότητα να εκτελέσει περισσότερες λειτουργίες στην περίπτωση πολύπλοκων συστημάτων, όπως η εκτέλεση αλγορίθμων για την επικοινωνία με πολλαπλές ετικέτες, η κρυπτογράφηση αιτημάτων που αποστέλλονται από τον αναγνώστη και η αποκρυπτογράφηση των απαντήσεων που λαμβάνονται από ετικέτες και η εκτέλεση του ελέγχου ταυτότητας μεταξύ συσκευών ανάγνωσης και ετικετών.

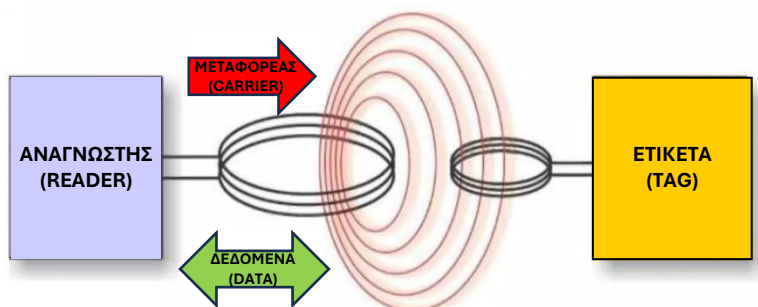
Οι συσκευές ανάγνωσης RFID μπορούν να εκτελούν σάρωση ετικετών υψηλής ταχύτητας. Εκατοντάδες αντικείμενα μπορούν να αντιμετωπιστούν από έναν μόνο αναγνώστη μέσα σε ελάχιστο χρόνο, επομένως η χρήση τους μπορεί να επεκταθεί σε εφαρμογές όπως η διαχείριση μιας εφοδιαστικής αλυσίδας, όπου συχνά υπάρχει ανάγκη ελέγχου ενός μεγάλου αριθμού αντικειμένων. Οι αναγνώστες είναι αρκετό να τοποθετούνται μόνο σε κάθε είσοδο και έξοδο. Όταν τα προϊόντα εισέρχονται ή εξέρχονται από αυτές, οι αναγνώστες μπορούν να τα αναγνωρίσουν αμέσως και να αποστείλουν τις απαραίτητες πληροφορίες στη βάση δεδομένων για περαιτέρω επεξεργασία.⁷

- Η κεραία ανάγνωσης μεταδίδει την ηλεκτρομαγνητική ενέργεια για να ενεργοποιήσει την ετικέτα, πραγματοποιεί τη μεταφορά δεδομένων και στέλνει τις οδηγίες στην ετικέτα, λαμβάνοντας παράλληλα πληροφορίες από την ετικέτα. Επειδή γενικά, η θέση ή ο προσανατολισμός του αναγνωρισμένου αντικειμένου είναι τυχαίος και ο τρόπος τοποθέτησης της ετικέτας σε αυτό δεν είναι σταθερός, η κεραία ανάγνωσης θα πρέπει να είναι μια κυκλικά πολωμένη κεραία, προκειμένου να αποφευχθεί η απώλεια πόλωσης όταν αλλάζει ο προσανατολισμός του αναγνωρισμένου αντικειμένου. Στο παθητικό σύστημα RFID, η ενέργεια για τη διατήρηση της λειτουργίας της ετικέτας προέρχεται από το ηλεκτρομαγνητικό κύμα που μεταδίδεται από την κεραία ανάγνωσης. Εδώ το παθητικό σύστημα συζητείται κυρίως για να δείξει την επίδραση των παραμέτρων της κεραίας στην απόδοση του συστήματος.⁸



Εικόνα 10. Κεραία RFID

Η κεραία του αναγνώστη όπως φαίνεται στην *εικόνα 11*, εκπέμπει συγκεκριμένες ραδιοσυχνότητες έτσι ώστε να ενεργοποιεί, να διεγείρει την ετικέτα (tag) και στη συνέχεια η ετικέτα μεταδίδει τα δεδομένα πίσω στην κεραία.



Εικόνα 11. Τρόπος λειτουργίας κεραίας

⁷ Vacca R. John, *Computer and Information Security Handbook*, Morgan Kaufmann Publishers, Burlington USA, 2009

⁸ Keskilammi, Sydanheimo & Kivikoski, *Passive RFID Systems and the Effects of Antenna Parameters on Operational Distance in Radio Frequency Technology for Automated Manufacturing and Logistics Control*, 2003



Εικόνα 12. Middleware

- Το ενδιάμεσο λογισμικό (reader interface layer), ή αλλιώς middleware, το οποίο συμπιέζει χιλιάδες σήματα ετικετών σε μια συγκεκριμένη αναγνώριση και επίσης δρα σαν κανάλι μεταφοράς μεταξύ των στοιχείων RFID (hardware) και των συστημάτων (software) της εφαρμογής του πελάτη, όπως το απόθεμα, η παραλαβή και τα logistics.⁹

Το ενδιάμεσο λογισμικό RFID είναι ένα επίπεδο λογισμικού που δημιουργείται μεταξύ των αναγνώστων και των επιχειρηματικών εφαρμογών, χρησιμοποιείται για την επεξεργασία όλων των πληροφοριών και ροών από τους αναγνώστες. Στην πραγματική εφαρμογή, το ενδιάμεσο λογισμικό RFID ολοκληρώνει την επεξεργασία δεδομένων, τη μετάδοση και τη διαχείριση παραμέτρων των αναγνώστων, παρακολουθώντας την κατάσταση λειτουργίας της συσκευής, διαχειρίζεται και επεξεργάζεται τη ροή δεδομένων μεταξύ της ετικέτας και του αναγνώστη και παρέχει διεπαφές προγράμματος εφαρμογής (API) μεταξύ των συσκευών RFID και της επιχειρηματικής εφαρμογής. Είναι καλή και γρήγορη λύση για ολοκληρωτές συστημάτων που δεν είναι εξοικειωμένοι με την ανάπτυξη RFID¹⁰.

Η εικόνα 13 δείχνει πως λειτουργεί ένα απλό σύστημα RFID και τα τέσσερα κύρια συστατικά του συστήματος αυτού.



Εικόνα 13. Τρόπος λειτουργίας συστήματος RFID

Η λειτουργία του όλου συστήματος είναι πολύ απλή και βασίζεται κατά κύριο λόγο στην αμφίδρομη επικοινωνία μεταξύ της ετικέτας και του αναγνώστη. Για να γίνει αυτό πρέπει η ετικέτα να βρεθεί κοντά στη συσκευή ανάγνωσης. Στην περίπτωση αυτή τα ραδιοκύματα από την ετικέτα (πομποδέκτη) αποστέλλονται μέσω της κεραίας και διαβάζονται από τον αναγνώστη. Ο αναγνώστης στη συνέχεια εκπέμπει συνεχόμενα ραδιοκύματα με αποτέλεσμα και πάλι μέσω της κεραίας να ενεργοποιεί την ετικέτα όταν αυτή βρίσκεται εντός εμβελείας. Το ηλεκτρομαγνητικό πεδίο που δημιουργείται θέτει σε λειτουργία το μικροτσίπ της ετικέτας. Με την ενεργοποίηση μεταδίδει τις πληροφορίες πίσω στον αναγνώστη και το εκάστοτε λογισμικό επεξεργάζεται τα δεδομένα που λήφθηκαν.

⁹ <https://www.graphicarts.gr/enimerosi/arthra-themata/tecnologia-rfid>

¹⁰ <https://www.linkedin.com/pulse/rfid-middleware-introduction-bella-gao-1c>

1.3.1 Τύποι συστημάτων RFID

Υπάρχουν τρεις κύριοι τύποι συστημάτων RFID: χαμηλής συχνότητας (LF), υψηλής συχνότητας (HF) και εξαιρετικά υψηλής συχνότητας (UHF). Διατίθεται επίσης RFID με λειτουργία μικροκυμάτων. Κύρια διαφορά των παραπάνω συστημάτων μεταξύ τους είναι η συχνότητες μετάδοσης οι οποίες ποικίλλουν πολύ ανά χώρα και περιοχή και η απόσταση ανάγνωσης της ετικέτας.

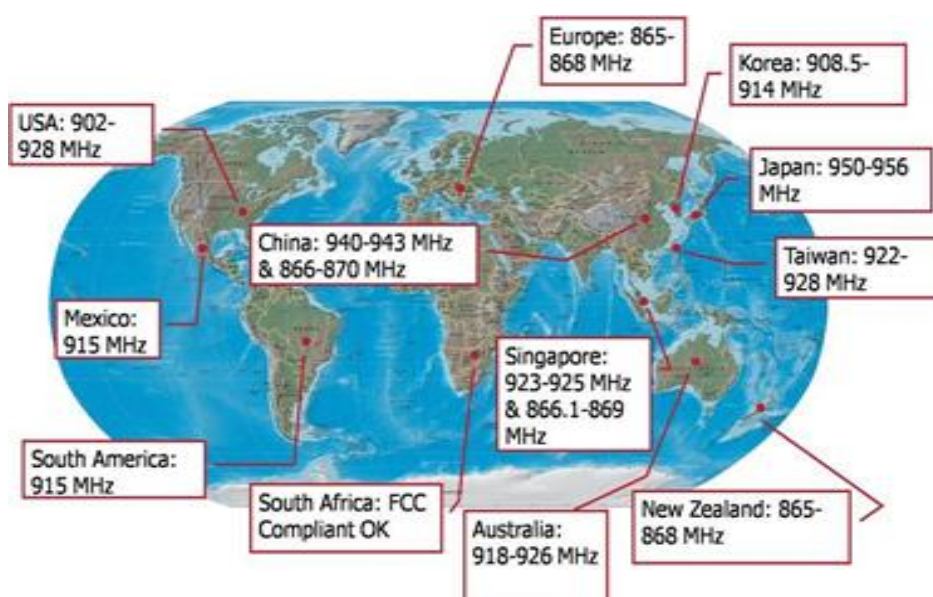
Οι βασικοί τύποι συστημάτων είναι τρεις:

α. **Συστήματα RFID χαμηλής συχνότητας.** Το εύρος των συχνοτήτων λειτουργίας στα συστήματα χαμηλής συχνότητας κυμαίνεται από 30 έως 500 KHz με συνήθεις συχνότητες λειτουργίας από 125 έως 135 KHz. Τα LF συστήματα RFID έχουν μικρές περιοχές μετάδοσης σχεδόν οπουδήποτε, ενώ οι αποστάσεις που μπορούν να καλύψουν είναι από μερικά εκατοστά μέχρι λίγο λιγότερο από 2 μέτρα.

β. **Συστήματα RFID υψηλής συχνότητας.** Τα συστήματα υψηλής συχνότητας έχουν εύρος συχνοτήτων λειτουργίας από 3 έως 30 MHz με συνήθη συχνότητα λειτουργίας τα 13,56 MHz. Τα HF συστήματα RFID έχουν περιοχές μετάδοσης οπουδήποτε και οι αποστάσεις κάλυψης είναι από μερικά εκατοστά μέχρι λιγότερο από 2 μέτρα.

γ. **Συστήματα RFID εξαιρετικά υψηλής συχνότητας.** Τα συστήματα εξαιρετικά υψηλής συχνότητας έχουν εύρος συχνοτήτων λειτουργίας από 300 MHz έως 960 MHz, με την τυπική συχνότητα τα 433 MHz και μπορούν γενικά να διαβαστούν από 25 και πλέον πόδια μακριά. 3 έως 30 MHz με συνήθη συχνότητα λειτουργίας τα 13,56 MHz. Τα UHF συστήματα RFID έχουν περιοχές μετάδοσης οπουδήποτε και οι αποστάσεις κάλυψης είναι από μερικά εκατοστά μέχρι λιγότερο από 2 μέτρα¹¹.

Για μεταδόσεις UHF RFID κάθε χώρα έχει τις δικές της περιοχές συχνοτήτων όπως φαίνεται στο παρακάτω σχήμα.

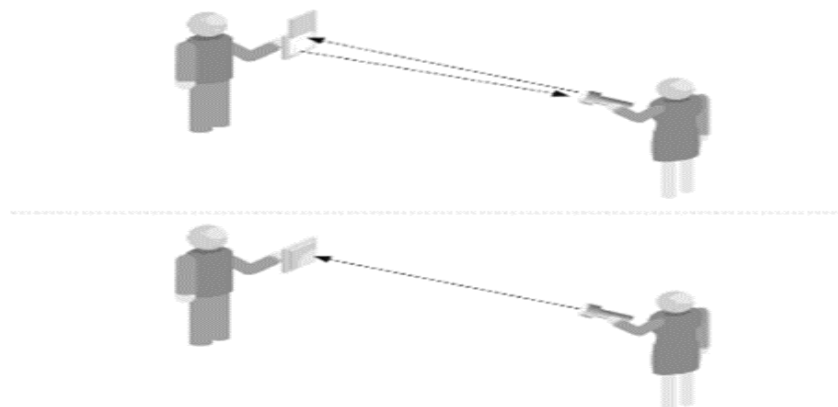


Εικόνα 14. Περιοχές συχνοτήτων UHF RFID

¹¹ <https://www.techtarget.com/iotagenda/definition/RFID-radio-frequency-identification>

δ. **Συστήματα RFID μικροκυμάτων.** Τα συστήματα RFID που λειτουργούν σε συχνότητες μικροκυμάτων στα 2,45 GHz και 5,8 GHz είναι γνωστά ως συστήματα RFID SHF (Super High Frequency). Ωστόσο, αυτές οι ζώνες συχνοτήτων χρησιμοποιούνται ευρέως και μπορεί να είναι επιρρεπείς σε παρεμβολές, καθώς αποτελούν συχνότητες λειτουργίας για πολλές συσκευές όπως ασύρματα τηλέφωνα και φούρνους μικροκυμάτων. Τα συστήματα μικροκυμάτων που λειτουργούν σε συχνότητα 2,45 GHz και μπορούν να διαβαστούν από αποστάσεις μεγαλύτερες των 10 μέτρων¹². Τα συστήματα RFID μικροκυμάτων χρησιμοποιούν έναν συνδυασμό παθητικών ετικετών, με υποβοήθηση μπαταρίας και ενεργών ετικετών με βάση αυτό που απαιτεί η εφαρμογή. Αυτές οι ετικέτες έχουν επίσης μεγάλο εύρος ανάγνωσης. Το εύρος ανάγνωσης για ενεργές ετικέτες είναι σημαντικά υψηλότερο σε σύγκριση με τις παθητικές ετικέτες SHF και μπορεί να φτάνει πάνω από 100 μέτρα.

Ο μηχανισμός σύζευξης μιας ετικέτας καθορίζει τον τρόπο με τον οποίο ένα κύκλωμα στην ετικέτα και ένα κύκλωμα στον αναγνώστη επηρεάζουν το ένα το άλλο για να στείλουν και να λάβουν πληροφορίες ή ισχύ¹³. Τα συστήματα SHF RFID βασίζονται στις αρχές σύζευξης ακτινοβολίας μακρού πεδίου ή σύζευξης οπισθοσκέδασης (backscatter) και έχουν μικρά μήκη κύματος. Ο όρος "backscatter", περιγράφει τον τρόπο με τον οποίο τα κύματα RF που μεταδίδονται από τον αναγνώστη διασκορπίζονται πίσω από την ετικέτα. Δηλαδή, τα κύματα αντανακλώνται πίσω στην πηγή για να στείλουν ένα σήμα. Φανταστείτε τον αναγνώστη ως φακό και την ετικέτα ως καθρέφτη σηματοδότησης με κάλυμμα, όπως φαίνεται στο παρακάτω σχήμα.



Εικόνα 15. Σύζευξη backscatter (οπισθοσκέδαση)

Στον παρακάτω πίνακα φαίνεται συγκεντρωτικά τα χαρακτηριστικά των τύπων συστημάτων RFID:

Διαμόρφωση	Συνήθης Συχνότητα	Τρόπος Σύζευξης	Εμβέλεια Επικοινωνίας Τυπική Μέγιστη		Ρυθμός Μετάδοσης Δεδομένων	Εξέλιξη	Κόστος Αναγνώστη
LF	125 to 135kHz	Inductive	20 cm	100 cm	Low	Very Mature	Low
HF	13.56 Mhz	Inductive	10 cm	≤2 m	High	Established	Medium
UHF	300 to 960 MHz	Backscatter	3 m	≥9	Medium	New	Very High
Microwave	2.45 GHz	Backscatter	3 m	≥10	Medium	In Development	Very High
	5.8 GHz	Backscatter	3 m	?	Medium	Future Development	Very High

Πίνακας 1. Χαρακτηριστικά τύπων συστημάτων RFID

¹² <https://www.everythingrf.com/community/microwave-frequency-shf-rfid-tags-systems>

¹³ Bill Glover, Himanshu Bhatt, *RFID Essentials* O'Reilly Media, Inc., California, USA, 2006

1.4 Πλεονεκτήματα και μειονεκτήματα

Η τεχνολογία RFID όπως και κάθε τεχνολογική εφαρμογή έχει τα θετικά και τα αρνητικά χαρακτηριστικά της. Για να θεωρηθεί ένα τεχνολογικό εύρημα επιτυχημένο θα πρέπει τα πλεονεκτήματα να υπερτερούν των μειονεκτημάτων.

Πλεονεκτήματα RFID :

- Αποδοτικότητα: Καθώς τα RFID συστήματα είναι αρκετά αποδοτικότερα από τον άνθρωπο επιτρέπουν σε διάφορες εφαρμογές/διαδικασίες να επιτυγχάνονται με μεγαλύτερη ταχύτητα όπως η συλλογή δεδομένων με αποτέλεσμα να αυξάνεται η αποτελεσματικότητα και να αναδένει το κέρδος.
- Ακρίβεια: Η συμμετοχή του ανθρώπινου παράγοντα μειώνεται με αποτέλεσμα να μειώνεται και η πιθανότητα λάθους που ενδεχομένως να προκαλούσε.
- Παρακολούθηση: Τα συστήματα RFID παρέχουν την δυνατότητα ελέγχου προϊόντων και εξοπλισμού σε βιομηχανίες καθώς προσφέρουν και παρακολούθηση περιουσιακών στοιχείων.
- Αυτοματοποίηση: Όλα τα προηγούμενα συμβάλουν στην αυτοματοποίηση γραμμών παραγωγής. Ένα βασικό κέντρο ελέγχου διαχειρίζεται όλο τον όγκο των πληροφοριών που λαμβάνονται από το RFID μειώνοντας το κόστος εργασίας και βελτιώνοντας τη λειτουργική απόδοση.
- Η σάρωση μπορεί να γίνει από απόσταση.
- Τα δεδομένα λαμβάνονται σε πραγματικό χρόνο.

Μειονεκτήματα RFID :

- Κόστος: Η αρχική εγκατάσταση ενός συστήματος RFID μπορεί σε πολλές περιπτώσεις να είναι αρκετά ακριβή.
- Παρεμβολές: Συσκευές εκπομπής ιδίων συχνοτήτων ακόμα και μεταλλικά αντικείμενα ενδέχεται να επηρεάζουν το σήμα.
- Ασφάλεια: Τα δεδομένα είναι εκτεθειμένα. Οι ετικέτες μπορούν να διαβαστούν οποιαδήποτε στιγμή από οποιόν κατέχει έναν αναγνώστη. Προσωπικά και ιατρικά δεδομένα, οικονομικές συναλλαγές, στρατιωτικά απόρρητα είναι κάποια από τα επικινδυνότερα στοιχεία κλοπής, πράγμα που εγείρει ανησυχίες περί προστασίας προσωπικών δεδομένων και πολιτικού απορρήτου.

1.5 Πρακτικές εφαρμογές

Τα RFID συστήματα βρίσκουν δεκάδες εφαρμογές στην καθημερινή ζωή. Από την απλή ανέπαφη συναλλαγή μέχρι την διαχείριση εφοδιαστικής αλυσίδας και την παρακολούθηση ιατρικών δεδομένων. Αρκετές εταιρείες συνεισφέρουν πολύ στην τεχνολογία με αποτέλεσμα αυτή να εξελίσσεται ταχέως καθώς οι υιοθετούντες γνωρίζουν πώς να την χρησιμοποιούν με μεγαλύτερη αποδοτικότητα. Η τεχνολογία RFID είναι άρρηκτα συνδεδεμένη με βιομηχανίες οι οποίες δραστηριοποιούνται στους τομείς της ένδυσης, των οχημάτων, των φαρμακευτικών προϊόντων και πολλών άλλων. Η αναγνώριση με ραδιοσυχνότητες μπορεί να λύσει πραγματικά επιχειρηματικά προβλήματα. Κάποιες από την βασικότερες εφαρμογές είναι:



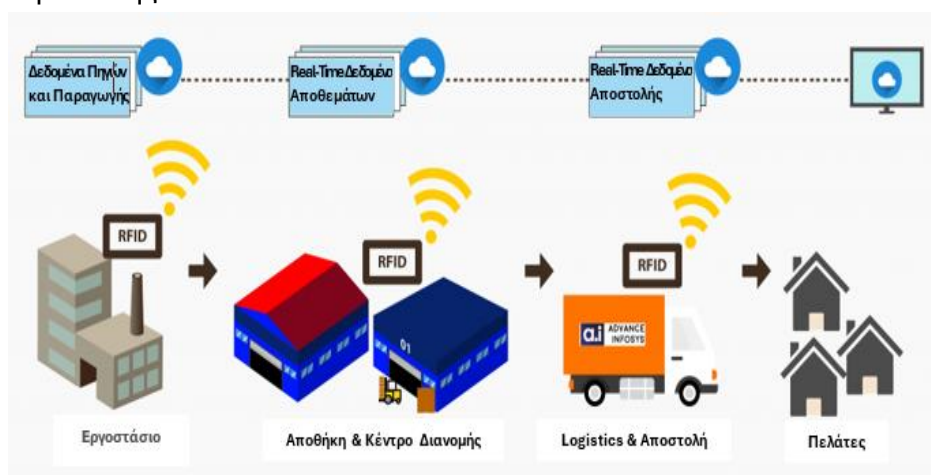
Εικόνα 16. Πρακτικές εφαρμογές συστημάτων RFID

➤ **Κτηνοτροφία:** Μικροτσιπ εμφυτεύονται στα ζώα, ακόμη και οικόσιτα, για την ταυτοποίηση και την παρακολούθηση τους. Ετικέτες παρέχουν στο ιδιοκτήτη τις απαραίτητες πληροφορίες που χρειάζεται.



➤ **Διαχείριση αποθεμάτων:** Ένα σύστημα είναι ικανό να ενημερώνεται σε πραγματικό χρόνο για τη κατάσταση των αποθεμάτων μιας αποθήκης, ή ενός καταστήματος λιανικής. Ως εκ τούτου θα αποφεύγεται το έλλειμα και το πλεόνασμα προϊόντων.

➤ **Logistics:** Τα προϊόντα παρακολουθούνται σε όλη την πορεία τους μειώνοντας πιθανά λάθη και αυξάνοντας την αξιοπιστία και την αποτελεσματικότητα της αποστολής και παράδοσης.



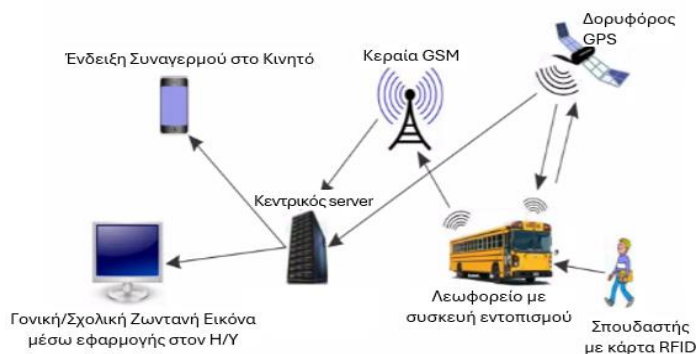
Εικόνα 17. Εφαρμογή συστημάτων RFID στα Logistics

➤ **Συστήματα ασφαλείας:** Ένα σύστημα ελέγχου πρόσβασης RFID είναι ένας τύπος συστήματος κάρτας κλειδιού (γνωστό και ως μπρελόκ ή key fob) που χρησιμοποιείται για την επαλήθευση της ταυτότητας κάποιου ατόμου. Τα συστήματα πρόσβασης αυτά χρησιμοποιούνται σε διαμερίσματα κατοικιών, βιβλιοθήκες και βιομηχανικές εγκαταστάσεις, ενώ τα είναι πολύ διαδεδομένα και τα συστήματα κλειδαριάς πόρτας για συγκροτήματα γραφείων και οικιών.



➤ **Ταυτότητες και διαβατήρια:** Για την ευκολότερη ταυτοποίηση των ανθρώπων τα βιομετρικά διαβατήρια και ταυτότητες, περιέχουν ένα μικροτσίπ και μια κεραία που χρησιμοποιεί ετικέτες RFID για τη μετάδοση των δεδομένων σε έναν αναγνώστη.

➤ **Σύστημα παρακολούθησης σχολικών λεωφορείων:** Μέσω συστήματος RFID μπορεί να γίνεται παρακολούθηση των σχολικών λεωφορείων και να υπάρχει αυτοματοποιημένη και ζωντανή προβολή της συμπεριφοράς των μαθητών μέσα σε αυτά. Στη συνέχεια σε περιπτώσεις έκτακτης ανάγκης μπορεί να γίνεται αποστολή ειδοποιήσεων τόσο στη διεύθυνση του σχολείου όσο και στους γονείς. Επίσης μπορεί να υπάρχει καταγραφή του πλήθους κατά την επιβίβαση και την αποβίβαση των μαθητών από το σχολικό λεωφορείο¹⁴.



➤ **Πληρωμές:** Ανέπαφες συναλλαγές επιτρέπονται με κάρτες και κινητά τηλέφωνα. Η τεχνολογία ανέπαφων πληρωμών επιτρέπει συναλλαγές χωρίς να απαιτείται φυσική επαφή μεταξύ της κάρτας και του τερματικού. Οι κάρτες NFC δεν χρησιμοποιούν ένα γενικό πρωτόκολλο για συναλλαγές πληρωμών. Αντίθετα, κάθε επωνυμία κάρτας ορίζει το δικό της αποκλειστικό πρωτόκολλο με βάση τις αρχές EMV, δηλαδή τα αντίστοιχα πρωτόκολλα ασφαλείας της εκάστοτε κάρτας. Αυτό σημαίνει ότι ένας αναγνώστης MasterCard PayPass δεν μπορεί να επεξεργαστεί συναλλαγές για κάρτες Visa payWave και το αντίστροφο. Ωστόσο, λόγω της αύξησης της χρήσης των καρτών ανέπαφων πληρωμών, έχουν αναπτυχθεί υβριδικοί αναγνώστες καρτών που αναγνωρίζουν όλες τις επωνυμίες.¹⁵



➤ **Βιβλιοθήκες:** Αυτόματοι δανεισμοί βιβλίων και διαχείριση αποθεμάτων στα ράφια καθώς και καθολικός έλεγχος ολόκληρης της βιβλιοθήκης μέσω ενός υπολογιστή.

¹⁴ https://www.slideshare.net/ashtopustech/rfid-technology-next-generation-application-solutions-49590687?from_action=save

¹⁵ <https://www.trendmicro.com/vinfo/pl/security/news/security-technology/next-gen-payment-processing-tech-rfid-credit-cards>

➤ **Έξυπνες πόλεις:** Αυτές οι πόλεις αξιοποιούν διάφορες τεχνολογικές εξελίξεις για να βελτιώσουν την ποιότητα ζωής των κατοίκων τους. Η τεχνολογία RFID προσφέρει μια πληθώρα εφαρμογών που μπορούν να βελτιώσουν την αστική ζωή, συμπεριλαμβανομένης της διαχείρισης στάθμευσης, των μέσων μαζικής μεταφοράς και των υπηρεσιών κοινής ωφέλειας. Άλλες εφαρμογές των έξυπνων πόλεων που χρησιμοποιούν την τεχνολογία RFID είναι:



Εικόνα 18. Στοιχεία Έξυπνων

❖ Αυτοματοποιημένη είσπραξη διοδίων: Οι ετικέτες RFID που είναι προσαρτημένες στα οχήματα επιτρέπουν την απρόσκοπτη είσπραξη διοδίων χωρίς την ανάγκη φυσικής πληρωμής, μειώνοντας την κυκλοφοριακή συμφόρηση στα δρόμα και βελτιώνοντας τη συνολική ροή της κυκλοφορίας.

❖ Έξυπνη διαχείριση στάθμευσης: Τα συστήματα στάθμευσης με δυνατότητα RFID μπορούν να παρακολουθούν και να διαχειρίζονται με ακρίβεια τις διαθέσιμες θέσεις parking, παρέχοντας πληροφορίες σε πραγματικό χρόνο στους οδηγούς και μειώνοντας τον χρόνο αναζήτησης θέσεων στάθμευσης. Έτσι μειώνεται η κυκλοφοριακή συμφόρηση, η ταλαιπωρία των οδηγών, η φθορά των οχημάτων και οι εκπομπές ρύπων.

❖ Έκδοση εισιτηρίων στα μέσα μαζικής μεταφοράς: Οι ανέπαφες έξυπνες κάρτες ή οι εφαρμογές για κινητά μπορούν να απλοποιήσουν την έκδοση εισιτηρίων για λεωφορεία, τρένα και μετρό. Με αυτόν τον τρόπο οι επιβάτες μπορούν να πληρώσουν γρήγορα και εύκολα τους ναύλους τους, εξαλείφοντας την ανάγκη για φυσικά εισιτήρια ή συναλλαγές σε μετρητά.

❖ Βελτιστοποίηση της Διαχείρισης Απορριμμάτων: Οι έξυπνοι κάδοι απορριμμάτων μπορούν να παρακολουθούν το επίπεδο πλήρωσης σε πραγματικό χρόνο, συντελώντας στη μείωση των δρομολογίων των απορριματοφόρων, τη βελτιστοποίηση των χρονοδιαγραμμάτων συλλογής, την εξοικονόμηση πόρων και μια πιο καθαρή πόλη. Επιπλέον οι ετικέτες RFID μπορούν να χρησιμοποιηθούν για την παρακολούθηση και την ταξινόμηση ανακυκλώσιμων υλικών, επιτρέποντας αποτελεσματικές διαδικασίες ανακύκλωσης και μείωση της ποσότητας των απορριμμάτων που αποστέλλονται σε χώρους υγειονομικής ταφής.

❖ Αποτελεσματική διαχείριση της εφοδιαστικής αλυσίδας: Οι ετικέτες RFID μπορούν να προσαρτηθούν σε προϊόντα, επιτρέποντας την παρακολούθηση σε πραγματικό χρόνο σε όλη την αλυσίδα εφοδιασμού. Αυτό επιτρέπει στις επιχειρήσεις να βελτιστοποιούν τα επίπεδα αποθέματος, να μειώνουν τα αποθέματα και να βελτιώνουν τη συνολική αποτελεσματικότητα της αλυσίδας εφοδιασμού.

❖ Διαχείριση περιουσιακών στοιχείων της πόλης: Οι δήμοι μπορούν να χρησιμοποιούν την τεχνολογία αυτή για την παρακολούθηση και τη διαχείριση περιουσιακών στοιχείων που ανήκουν στην πόλη, όπως οχήματα, εξοπλισμό και υποδομές. Αυτό εξασφαλίζει καλύτερη χρήση των πόρων, μειώνει την κλοπή ή την απώλεια και επιτρέπει την προληπτική συντήρηση.¹⁶

¹⁶ <https://fastercapital.com/content/RFID-in-Smart-Cities--Enabling-Seamless-Urban-Living.html>

➤ **Ιατρική – Φαρμακευτική:** Οι εφαρμογές RFID περιλαμβάνουν ταυτοποίηση και παρακολούθηση ασθενών, φαρμάκων, μετάγγιση αίματος, παρακολούθηση εξοπλισμού και περιουσιακών στοιχείων, όπως και συλλογή δεδομένων που προέρχονται από αισθητήρες. Για παράδειγμα, μπορεί να προτείνουν τη χρήση εμφυτεύσιμης τεχνολογίας RFID στην ιατρική που εξυπηρετεί τη λειτουργία ενός φορητού ιατρικού φακέλου.

Επιπλέον, η δυνατότητα απομακρυσμένης σύνδεσης δεδομένων RFID εφαρμόστηκε σε έναν οίκο φροντίδας ηλικιωμένων για να μεταδώσει εξ αποστάσεως κινήσεις και φυσιολογικά δεδομένα ασθενών με αναπηρία ή κλινήρεις.



Εικόνα 19. Εφαρμογές RFID στην υγεία

Κεφάλαιο 2: Εξοπλισμός, Σύνδεση περιφερειακών, Επικοινωνία με Υ/Η

2.1 Παρουσίαση των εξαρτημάτων του συστήματος

Η παρακάτω ενότητα θα επικεντρωθεί στο Hardware της εργασίας. Θα ακολουθήσει μια περιγραφή των αντικειμένων και των τεχνικών χαρακτηριστικών τους καθώς και του σωστού τρόπου λειτουργίας τους. Τέλος θα παρουσιαστεί ο τρόπος σύνδεσης με τον υπολογιστή και η προσπάθεια ενσωμάτωσης πλακέτας Wi-Fi για την ασύρματη μεταφορά δεδομένων.

2.1.1 Πλακέτα μικροελεγκτή Arduino



Εικόνα 20. Πλακέτα Arduino

Βασικό εξάρτημα του συστήματος είναι η πλακέτα Arduino και συγκεκριμένα το μοντέλο UNO. Φέρει τον επεξεργαστή ATmega328, και αποτελεί ένα από τα πιο γνωστά μοντέλα μικροεπεξεργαστών. Πρόκειται για μια μονάδα επεξεργασίας δεδομένων εισόδων και εξόδων ανοιχτού κώδικα που πρωτοπαρουσιάστηκε το 2003 στην Ιταλία με το Arduino UNO να κυκλοφορεί το 2010 .

Η πλακέτα αποτελείται από τον επεξεργαστή, από 14 ψηφιακές εισόδους/εξόδους εκ των οποίων 6 μπορούν να χρησιμοποιηθούν ως Pulse Width Modulation (PWM), 6 αναλογικές εισόδους/εξόδους, 1 θύρα USB-B για την τροφοδοσία αλλά και τον προγραμματισμό της πλακέτας, μια δεύτερη θύρα καθαρά τροφοδοσίας με τάση που παρέχεται είτε από πρίζα είτε από απλή μπαταρία. Την πλακέτα απαρτίζουν ακόμη, ένας κρύσταλλος 16MHz και οι μνήμες SRAM (2KB), EEPROM (1KB) και FLASH MEMORY (32KB). Για την τροφοδοσία μέσω θύρας usb είναι απαραίτητη τάση λειτουργίας 5V ενώ μέσω της θύρας τροφοδοσίας η τάση κυμαίνεται από 6-20 Volt αν και συνιστάται το εύρος 7-12 Volt.

Board Name & Part#	Board Size Group	Board Communication	MCU Part# & Pins	MCU I/O Voltage	MCU Core	MCU Clock	MCU Flash	MCU SRAM	MCU EEPROM	MCU USART & UART	MCU SPI	MCU I ² C	MCU Other Bus Peripherals	MCU Timers 32/24/16/8 /WD/RT/RC	MCU ADC & DAC	MCU Engines
Uno R3, ^[10] A000066, ^[9] Uno R3 SMD, ^[19] A000073 ^[20]	Uno	USB-B	ATmega328P, ^[12] 28 pin DIP, 32 pin SMD	5V (1.8-5.5V)	8bit AVR	16 MHz*	32 KB	2 KB	1 KB	1, 0	1	1	None	0, 0, 1, 2, WD	10bit, None	None

Πίνακας 2. Στοιχεία πλακετών Arduino

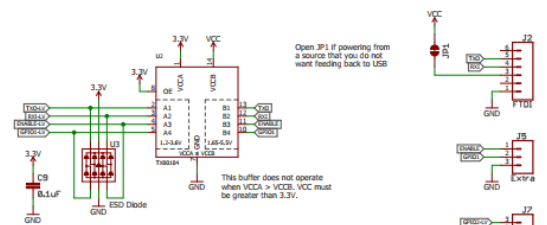
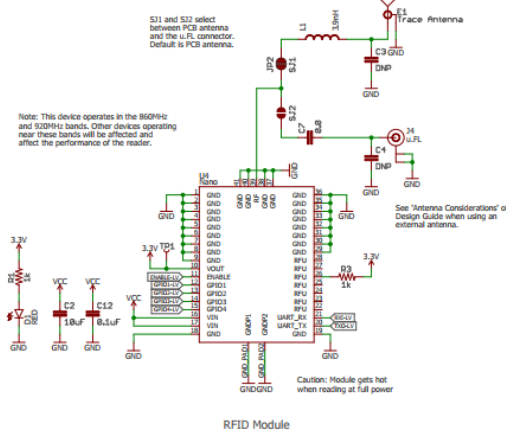
2.1.2 Sparkfun Rfid Reader M6E-NANO

Το δεύτερο βασικότερο εξάρτημα είναι ο αναγνώστης (reader) - πλακέτα Sparkfun Simultaneous RFID Reader - M6E Nano. Πρόκειται για μια πλακέτα ειδικά σχεδιασμένη για την ανάγνωση παθητικών ετικετών (Rfid tags). Φέρει τον μικροεπεξεργαστή SAMD21 και με τον τρόπο κατασκευής shield επιτυγχάνεται η επικοινωνία με την πλακέτα Arduino μέσω σειριακής θύρας.

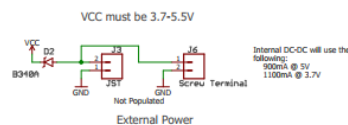
Όσον αφορά το τσιπάκι M6-NANO της ThingMagic, αυτό αποτελεί την μονάδα ανάγνωσης των παθητικών ετικετών ECP Gen 2 που λειτουργεί σε διαμόρφωση UHF (Ultra-High Frequency) με ικανότητα ανάγνωσης έως και 150 ετικετών ταυτόχρονα και ονομαστική τάση λειτουργίας τα 3,3V. Η εμβέλεια της ποικίλει ανάλογα με την ισχύ εξόδου που θα καθοριστεί από τον χρήστη, και κυμαίνεται από 0dBm έως 27dBm, ισχύς που της επιτρέπει, με την κατάλληλη κεραία, να διαβάζει ετικέτες που βρίσκονται έως και περίπου 5 μέτρα μακριά. Το Reader δεν μπορεί να λειτουργήσει μόνο του και χρειάζεται εξωτερική κεραία για την λήψη σημάτων.



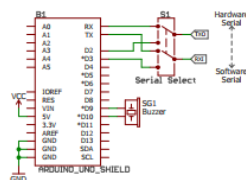
Εικόνα 21. Αναγνώστης (Reader) M6E-NANO



Interface and Logic Conversion



External Power



Shield and Serial Select

Changes on v10:
Decreased size of RF pad to avoid GND pad on module
Finalized trace antenna design and matching network
Added thermals to user solderable GND connections
Opened up heatsink area on bottom of PCB



Released under the Creative Commons
Attribution Share-Alike 4.0 License
<https://creativecommons.org/licenses/by-sa/4.0/>
TITLE: Simultaneous_RFID_Tag_Reader
Design by: N. Seidle
Date: 2/20/2017 9:37 AM
Sheet: 1/1

Datasheet

Πίνακας 3. Datasheet της πλακέτας
Sparkfun

2.1.3 Sparkfun UHF RFID Antenna



Τελευταίο εξάρτημα του συστήματος είναι η κεραία της Sparkfun η οποία συνδέεται με την πλακέτα – αναγνώστη (reader) που περιεγράφηκε παραπάνω. Είναι κεραία UHF με εύρος λειτουργίας από 860MHz έως 960MHz και απολαβή 6dBi. Είναι απαραίτητη η σύνδεση της μέσω καλωδίου όπως φαίνεται στην εικόνα 22. Κάθε κεραία UHF RFID διαθέτει ένα καλώδιο 30 cm που τερματίζεται με θηλυκό βύσμα RP-TNC

**Εικόνα 22. Κεραία (Antenna)
Sparkfun UHF RFID**

GSM-34-900-58-30CM-1814

860-960MHz PANEL 6dBi+RG58(30CM)+TNC FEMALE RP

Datasheet

**Πίνακας 4. Datasheet της κεραίας
Sparkfun**

2.1.4 Ετικέτες RFID (Tags)

Στο υπό εξέταση σύστημα χρησιμοποιήθηκαν παθητικές ετικέτες. Οι συγκεκριμένες ετικέτες δεν διαθέτουν μπαταρία ή άλλη εσωτερική πηγή ενέργειας, αλλά βασίζονται στην ενέργεια που λαμβάνεται από τη συσκευή ανάγνωσης RFID και την κεραία της για τροφοδοσία. Έτσι δεν υπήρξε ο περιορισμός της εξάρτησης των ετικετών από την ισχύ και το χρόνο ζωής κάποιου είδους μπαταρίας καθώς χρειάζονται πολύ μικρή ποσότητα ενέργειας για να τροφοδοτήσουν το εσωτερικό τους τσιπ RFID και να στείλουν μια απάντηση με τις προγραμματισμένες πληροφορίες. Επιπλέον, οι παθητικές ετικέτες RFID έχουν πολύ χαμηλό κόστος, κάτι που δεν θα ήταν δυνατό αν περιλάμβαναν μπαταρία.



Εικόνα 23. Παθητικές ετικέτες (tags)

2.2 Επικοινωνία του συστήματος με τον υπολογιστή

2.2.1 Ενσύρματη επικοινωνία

Στη συγκεκριμένη εργασία η επικοινωνία επιτυγχάνεται μέσω σειριακής θύρας του Arduino και ενός καλωδίου USB 2.0 - USB B που συνδέει τον υπολογιστή με την πλακέτα για την επιτυχημένη μεταφορά δεδομένων.

2.2.2 Προσπάθεια ενσωμάτωσης πλακέτας Wi-Fi Shield

Στα πρώτα στάδια της εργασίας έγινε μια προσπάθεια ενσωμάτωσης Wi-Fi Shield πλακέτας. Στόχος ήταν η επίτευξη ασύρματης επικοινωνίας του υπολογιστή με το Arduino για την ευκολότερη ανταλλαγή δεδομένων σε δύσβατες τοποθεσίες καθώς και για την αποφυγή περιττών καλωδίων. Έγιναν αρκετές προσπάθειες με διαφορετικές πλακέτες διάφορων εταιρειών, παρόλα αυτά καμία δεν απέδωσε. Μια από τις πλακέτες που χρησιμοποιήθηκαν ήταν η ESP8266 ESP-12E UART WIFI Wireless Shield. Ένα από τα σημαντικότερα προβλήματα που εμφανίστηκε ήταν το γεγονός ότι και τα δυο εξαρτήματα, η ESP8266 και η RFID tag reader πλακέτα της Sparkfun, επικοινωνούσαν με το Arduino μέσω των ίδιων θυρών. Το πρόβλημα προκύπτει από την σύνδεση των ακίδων RX(P0) και TX(P1) του ESP8266 με τους ακροδέκτες του Arduino D0 (RX) και D1 (TX) αντίστοιχα. Οι δύο ακίδες αυτές αντιστοιχούν σε σειριακές θύρες οι οποίες είναι υπεύθυνες για τη επικοινωνία με την πλακέτα, επομένως παραμένουν απασχολημένες κατά την διάρκεια ανταλλαγής δεδομένων με το Arduino. Συμπεραίνεται ότι είναι αδύνατη η συνύπαρξη και των δύο πλακετών αφού το shield της Sparkfun επικοινωνεί συνεχώς με τον μικροεπεξεργαστή.

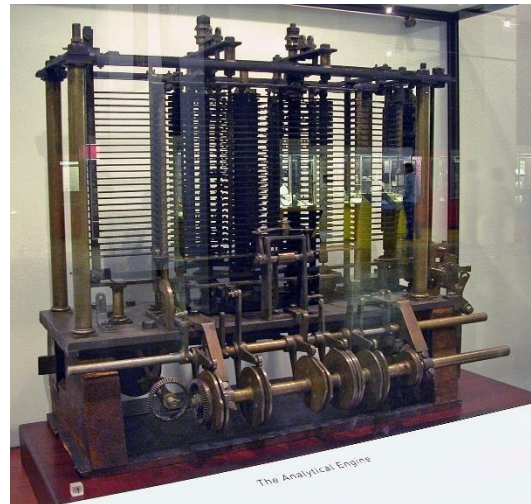


Παρόλα αυτά έγινε δοκιμή της ESP8266 μόνη της για να επιβεβαιωθεί η λειτουργικότητά της. Αρχικά αναβαθμίστηκε το firmware της πλακέτας μέσω εγκατάστασης προγράμματος driver. Αυτό ήταν απαραίτητο για να μπορέσει η πλακέτα να αναγνωριστεί στην εφαρμογή Arduino IDE και να χρησιμοποιηθούν οι εντολές AT. Έτσι επιτεύχθηκε η επικοινωνία με web server ώστε να γίνει σύνδεση σε τοπικό Wi-Fi δίκτυο. Η διαδικασία έγινε αρκετά δύσκολη με αποτέλεσμα να μην υλοποιηθεί η αρχική ιδέα της ενσωμάτωσης Wi-Fi πλακέτας στο RFID σύστημά.

Κεφάλαιο 3: Προγράμματα (Software) και manual σύνδεσης της εφαρμογής

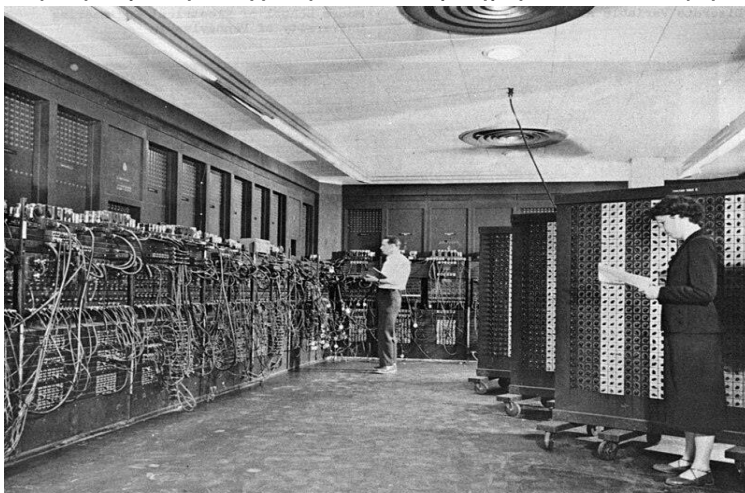
3.1 Ιστορική αναφορά σε γλώσσες προγραμματισμού

Ο προγραμματισμός σαν ιδέα δεν είναι σύγχρονη ανακάλυψη, καθώς έχει τις ρίζες του στα έτη 1837-1849 με την εφεύρεση της Αναλυτικής Μηχανής, ενός μηχανήματος το οποίο χρησιμοποιούνταν για μαθηματικές πράξεις όπως για παράδειγμα ο υπολογισμός των αριθμών Bernoulli. Η Αναλυτική Μηχανή ενσωμάτωσε μια αριθμητική μονάδα, τη ροή ελέγχου με τη μορφή διακλαδώσεων υπό ορούς και βρόγχους καθώς και ενσωματωμένη μνήμη. Θεωρείται από πολλούς ως το πρώτο δημοσιευμένο πρόγραμμα υπολογιστή στον κόσμο.



Εικόνα 24. Η Αναλυτική Μηχανή

Οι πρώτοι ηλεκτρονικοί υπολογιστές εμφανίστηκαν στις αρχές του 1940 διαθέτοντας περιορισμένη ταχύτητα και μνήμη. Η έλλειψη ταχύτητας και μνήμης, των τότε υπολογιστικών μοντέλων,



Εικόνα 25. ENIAC - Ο πρώτος Ηλεκτρονικός

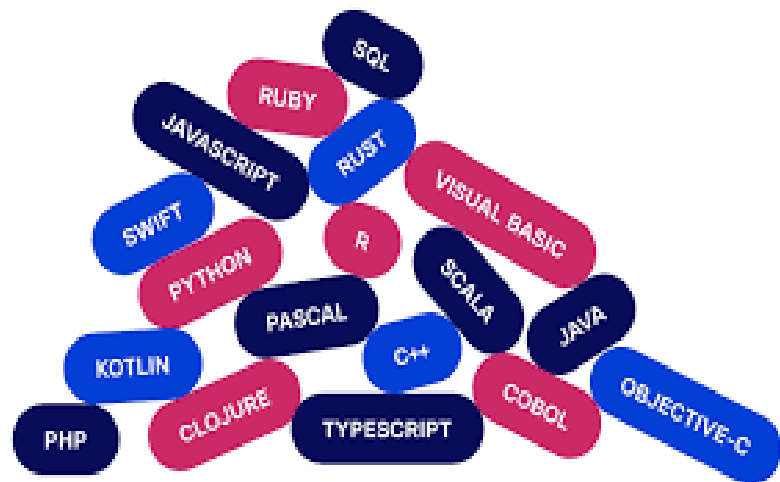
επέβαλαν στους προγραμματιστές να γράφουν σε γλώσσα Assembly. Πολύ σύοτομα όμως ανακάλυψαν ότι η χρήση της ήταν προβληματική λόγω της μεγάλης προσπάθειας που επέβαλε η σύνταξη προγραμμάτων καθώς και η διόρθωση των σφαλμάτων τους.

Το 1948, ο μηχανικός Konrad Zuse επινόησε μία καινούρια γλώσσα προγραμματισμού, την οποία ονόμασε Plankalkül. Η γλώσσα

αυτή όμως δεν υλοποιήθηκε τότε αλλά μετά από πολλά χρόνια, λόγω πολέμου. Άλλες σημαντικές γλώσσες που αναπτύχθηκαν εκείνη την πρώτη περίοδο των υπολογιστών ήταν η C-10 και η ENIAC coding system. Όλες οι γλώσσες αυτές ήταν σχεδιασμένες για τον υπολογιστή ENIAC.¹⁷ Από την δεκαετία του 1950 δημιουργούνται 2 μοντέρνες γλώσσες προγραμματισμού, η FORTRAN (FORmula TRANslation) και η COBOL (COMmon Business Oriented Language), των οποίων απόγονοι υπάρχουν ακόμα και σήμερα. Η FORTAN είναι η πρώτη γλώσσα προγραμματισμού υψηλού επιπέδου ευρείας χρήσης που είχε λειτουργική

υλοποίηση και δημιουργήθηκε το 1957 από την IBM. Η FORTAN είναι μια γλώσσα για επιστημονικές εφαρμογές και μελέτες. Αν και παραμένει δημοφιλής στην επιστημονική κοινότητα και χρησιμοποιείται μέχρι σήμερα σε υπολογιστές υψηλής απόδοσης όταν πρωτοεμφανίστηκε αντιμετώπισε αρκετά προβλήματα. Χρησιμοποιείται κυρίως για πρόβλεψη καιρού, ανάλυση πεπερασμένων στοιχείων, γεωφυσική, υπολογιστική φυσική, κρυσταλλογραφία και υπολογιστική χημεία. Στο σχεδιασμό την FORTAN στηρίχθηκαν αρκετές μεταγενέστερες γλώσσες προγραμματισμού όπως η BASIC.

Η COBOL η οποία κατέχει κυρίαρχο ρόλο στις επιχειρήσεις αναπτύχθηκε το 1959. Η δημοτικότητά της έχει μειωθεί αρκετά τα τελευταία χρόνια και πλέον χρησιμοποιείται σε παλαιού τύπου συστήματα για την διατήρηση ήδη υπαρχόντων εφαρμογών. Το 2002 έγινε αντικειμενοστρεφής γλώσσα βρίσκοντας εφαρμογές σε επιχειρηματικά, οικονομικά και διοικητικά συστήματα για εταιρείες και κυβερνήσεις.



Εικόνα 26. Γλώσσες Προγραμματισμού

Την δεκαετία του 1980 δημιουργείται η πρώτη αντικειμενοστρεφής γλώσσα C++ βασισμένη σε πολύ μεγάλο βαθμό πάνω στην ήδη υπάρχουσα, C. Επιπλέον, ένα νέο προγραμματιστικό στυλ, το να γράφει κανείς προγράμματα σε κομμάτια (modules), καταλήγει στην δημιουργία γλωσσών όπως η Modula.

Η δεκαετία του 1990 αποτελεί σταθμό στην εξέλιξη των γλωσσών προγραμματισμού καθώς ξεκινάει η εποχή του Internet, κάτι που δημιουργεί νέες ανάγκες και τάσεις στην τεχνολογία των γλωσσών. Οι γλώσσες πλέον αρχίζουν να ωριμάζουν και να αποκτούν προδιαγραφές. Γίνεται μεγάλη προσπάθεια για την διευκόλυνση των προγραμματιστών πάνω στην ανάπτυξη ταχύτητας και την ευκολία του κώδικα, κάτι που οδηγεί στην δημιουργία των πρώτων IDE (ολοκληρωμένα προγράμματα ανάπτυξης). Γενικά αναπτύσσονται πολύ οι αντικειμενοστρεφείς γλώσσες και πλέον μπαίνει δυναμικά στο παιχνίδι και η εταιρεία Microsoft με την C# και την Visual BASIC. Σημαντικές γλώσσες που αναπτύχθηκαν τη δεκαετία αυτή είναι η Haskell, η Python, η Java, η Ruby, η Javascript και η PHP.

Καθημερινά, η εξέλιξη των γλωσσών προγραμματισμού προχωράει. Νέες δημιουργούνται, παλιές ωριμάζουν ή εξελίσσονται. Από τον 1940, πλέον έχουν προστεθεί πάρα πολλά χαρακτηριστικά στις γλώσσες, ανάλογα και με το τι επέβαλαν οι εξελίξεις στα υπολογιστικά συστήματα όλα αυτά τα χρόνια. Η ύπαρξη πολύπλοκων, εξελιγμένων, ασφαλών και διαφορετικών συστημάτων οδήγησε στην εξέλιξή τους ακόμη και σήμερα.¹⁷

¹⁷ <https://spacezilotes.wordpress.com/2012/11/17/%CE%AF-%CF%8E/>

3.2 Γλώσσα προγραμματισμού Python

Η Python είναι μια γλώσσα προγραμματισμού με απλό συντακτικό, εξαιρετική αναγνωσιμότητα, φορητότητα (portability) και μοντέρνα χαρακτηριστικά που την καθιστούν κατάλληλη ως πρώτη γλώσσα προγραμματισμού.

Η Python είναι μία γλώσσα «υψηλού επιπέδου» (άλλα παραδείγματα τέτοιων γλωσσών είναι η FORTRAN, C, Java κλπ). Ο κώδικας μία τέτοιας γλώσσας πρέπει να μετατραπεί σε «γλώσσα μηχανής» ώστε να εκτελεστεί από τον Η/Υ. Η επεξεργασία αυτή γίνεται από διερμηνευτές (interpreters) και μεταγλωττιστές (compilers). Στην περίπτωση της python η επεξεργασία γίνεται από διερμηνευτή.



Η γλώσσα αυτή γράφτηκε από τον Ολλανδό προγραμματιστή Guido van Rossum στα τέλη της δεκαετίας 1980-90. Η έκδοση 2.0 δημοσιεύτηκε στις 16 Οκτωβρίου 2000 και η έκδοση 3.0, η οποία δεν είναι, εν γένει, σύμφωνη (compatible) με τις προηγούμενες εκδόσεις, στις 3 Δεκεμβρίου 2008.

3.2.1 Γιατί επιλέχθηκε η συγκεκριμένη γλώσσα

Ο μεγάλος αριθμός γλωσσών προγραμματισμού που χρησιμοποιούνται σήμερα αποτέλεσε ένα σημαντικό δίλημμα για την επιλογή της γλώσσας πάνω στην οποία θα βασιζόταν ο κώδικας. Τα διαφορά πλεονεκτήματα της Python ήταν ένας κατασταλτικός παράγοντας στην τελική απόφαση. Παρακάτω αναγράφονται κάποια από αυτά τα προτερήματα που κάνουν την εν λόγω γλώσσα να ξεχωρίζει μεταξύ άλλων.

1. **Ευκολία Μάθησης:** Η Python είναι γνωστή για την απλότητα και την ευανάγνωστη σύνταξή της, κάτι που την καθιστά εύκολη στην κατανόηση ακόμα και για αρχάριους. Αυτό επιτρέπει στους προγραμματιστές να επικεντρωθούν περισσότερο στην επίλυση προβλημάτων παρά στις λεπτομέρειες της σύνταξης.
2. **Πολυμορφία:** Η Python είναι μια πολυμορφική γλώσσα προγραμματισμού που μπορεί να χρησιμοποιηθεί για διάφορες εφαρμογές όπως η ανάπτυξη ιστοσελίδων, η ανάλυση δεδομένων, η μηχανική μάθηση, η τεχνητή νοημοσύνη, ο επιστημονικός υπολογισμός, η αυτοματοποίηση και άλλα.
3. **Μεγάλη Κοινότητα και Υποστήριξη από την Βιομηχανία:** Η Python έχει μια μεγάλη και ενεργή κοινότητα προγραμματιστών και χρηστών. Αυτό σημαίνει ότι υπάρχουν πλούσιοι πόροι διαθέσιμοι online, συμπεριλαμβανομένης της τεκμηρίωσης, των εκπαιδευτικών φύλλων, των φόρουμ και των βιβλιοθηκών. Η μεγάλη αυτή κοινότητα είναι ο λόγος που η γλώσσα λαμβάνει τακτικά ενημερώσεις και βελτιώσεις, εξασφαλίζοντας ότι παραμένει σχετική και ανταγωνιστική στον διαρκώς εξελισσόμενο κόσμο της τεχνολογίας. Η Python είναι ευρέως υιοθετημένη από την τεχνολογική βιομηχανία όσο και από την ακαδημαϊκή κοινότητα. Πολλοί τεχνολογικοί κολοσσοί όπως η Google, το Facebook, το Instagram, το Spotify και το Netflix χρησιμοποιούν την Python για διάφορους σκοπούς.

4. **Βιβλιοθήκες** : Η Python προσφέρει πολλές βιβλιοθήκες και πλαίσια προσαρμοσμένα για συγκεκριμένες εργασίες. Για παράδειγμα, το Django για την ανάπτυξη ιστοσελίδων, το Pandas για τη διαχείριση δεδομένων. Η χρήση αυτών των βιβλιοθηκών μπορεί να επιταχύνει σημαντικά τον χρόνο ανάπτυξης.
5. **Συμβατότητα Λειτουργικών Συστημάτων**: Η Python είναι διαθέσιμη σε διάφορες πλατφόρμες, συμπεριλαμβανομένων των Windows, macOS και Linux. Αυτό εξασφαλίζει ότι ο κώδικάς σας μπορεί να τρέξει άνετα σε διαφορετικά λειτουργικά συστήματα χωρίς σημαντικές τροποποιήσεις.

Συνολικά, η απλότητα, η πολυμορφία και η ισχυρή υποστήριξη από την κοινότητα καθιστούν την Python μια εξαιρετική επιλογή για μια ευρεία γκάμα προγραμματιστικών εργασιών, από έργα αρχαρίων έως εφαρμογές επιχειρησιακού επιπέδου.¹⁸

3.2.2 PyCharm

Για τις προγραμματιστικές ανάγκες της διπλωματικής επιλέχθηκε η πλατφόρμα του PyCharm στην οποία γράφηκε ο κώδικας. Η εφαρμογή προσφέρει ένα οπτικά εντυπωσιακό περιβάλλον το οποίο ο κάθε χρήστης μπορεί να εξατομικεύσει μέσω πρόσθετων. Το PyCharm αναπτύσσεται από την τσέχικη εταιρεία JetBrains και είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE), ίσως από τα καλύτερα, που χρησιμοποιείται για προγραμματισμό στην γλώσσα προγραμματισμού Python. Κυκλοφόρησε για πρώτη φορά τον Φεβρουάριο του 2010 και εκτός από την ανάλυση κώδικα, παρέχει γραφικό εντοπισμό σφαλμάτων, ενσωματωμένο ελεγκτή μονάδας, ενοποίηση με συστήματα ελέγχου εκδόσεων και υποστηρίζει την ανάπτυξη ιστού.



Το PyCharm είναι συμβατό με Linux, macOS και Windows και μπορεί να χρησιμοποιηθεί ως πρόγραμμα πολλαπλών πλατφορμών. Παρέχει μια σειρά από πρόσθετα εργαλεία για να επιταχύνει την ανάπτυξη της Python και ταυτόχρονα να ελαχιστοποιήσει την προσπάθεια για αυτό.

Η δυνατότητα περιήγησης κώδικα διευκολύνει πολύ τους προγραμματιστές την πλοήγηση σε μια κατηγορία, δυνατότητα ή αρχείο. Βοηθά επίσης να ελαχιστοποιηθεί σημαντικά η προσπάθεια και ο χρόνος που απαιτείται για την επεξεργασία και την τροποποίηση του κώδικα Python. Οι εξειδικευμένες προβολές έργων και οι προβολές δομής αρχείων είναι εύκολα προσβάσιμες¹⁹.

¹⁸ <https://datanalysis.net/research-design/spss-r-python-stata-meionehtimata-pleonehtimata/>

¹⁹ <https://www.educba.com/what-is-pycharm/>

3.3 Βάση δεδομένων SQL

Η χρήση ενός συγχρόνου μοντέλου αποθήκευσης πληροφοριών θα πρόσθετε στην εφαρμογή αξιοπιστία και ταχύτητα, κάτι που δεν θα μπορούσε επιτευχθεί με μία απαρхайωμένη τεχνική όπως είναι αυτή της δημιουργίας αρχείων. Για αυτό το λόγο η εργασία αξιοποιεί τα εργαλεία που μπορεί να προσφέρει μια βάση δεδομένων SQL.



Η SQL (Structured Query Language) είναι μια γλώσσα υπολογιστή για την εργασία με σύνολα γεγονότων και τις σχέσεις μεταξύ τους. Τα προγράμματα σχεσιακών βάσεων δεδομένων, όπως η Microsoft Office Access, χρησιμοποιούν SQL για να εργαστούν με δεδομένα. Σε αντίθεση με πολλές γλώσσες υπολογιστή, η SQL δεν είναι δύσκολο να διαβαστεί και να κατανοηθεί, ακόμη και από κάποιον αρχάριο. Όπως πολλές γλώσσες υπολογιστή, η SQL είναι ένα διεθνές πρότυπο που αναγνωρίζεται από οργανισμούς προτύπων όπως το ISO και το ANSI.

Η SQL δεν χρησιμοποιείται μόνο για το χειρισμό δεδομένων, αλλά επίσης για τη δημιουργία και την τροποποίηση της σχεδίασης αντικειμένων βάσης δεδομένων, όπως οι πίνακες. Το τμήμα της SQL που χρησιμοποιείται για τη δημιουργία και την τροποποίηση αντικειμένων βάσης δεδομένων ονομάζεται γλώσσα ορισμού δεδομένων (DDL).²⁰

3.3.1 MySQL

Επιλέχθηκε το σύστημα διαχείρισης βάσεων δεδομένων MySQL για τους παρακάτω λόγους:

1. **Ευκολία εγκατάστασης:** Η MySQL σχεδιάστηκε για να είναι φιλική προς το χρήστη. Είναι πολύ απλό να δημιουργήσουμε μια βάση δεδομένων MySQL καθώς περιέχει πολλά εργαλεία, τα οποία βελτιστοποιούν περαιτέρω τη διαδικασία εγκατάστασης. Η MySQL είναι διαθέσιμη για όλα τα μεγάλα λειτουργικά συστήματα.
2. **Ταχύτητα:** Η MySQL αποτελεί μια εξαιρετικά γρήγορη λύση βάσης δεδομένων, ενώ έχει σχετικά μικρότερο αποτύπωμα και είναι εξαιρετικά επεκτάσιμη μακροπρόθεσμα.
3. **Προνόμια χρήστη και ασφάλεια:** Η MySQL επιτρέπει τον ορισμό ενός επιπέδου ασφαλείας κωδικού πρόσβασης, την εκχώρηση κωδικών διαχειριστή και την πρόσθεση και αφαίρεση δικαιωμάτων λογαριασμού χρήστη.

²⁰ <https://support.microsoft.com/el-gr>

Για την ενσωμάτωση της γλώσσας προγραμματισμού Python και του συστήματος διαχείρισης βάσης δεδομένων MySQL χρησιμοποιείται το πρόγραμμα οδήγησης Python MySQL Connector. Αυτή η βιβλιοθήκη Python MySQL επιτρέπει τη μετατροπή μεταξύ τύπων δεδομένων Python και MySQL. Το MySQL Connector API υλοποιείται χρησιμοποιώντας καθαρή Python και δεν απαιτεί βιβλιοθήκη τρίτων²¹.

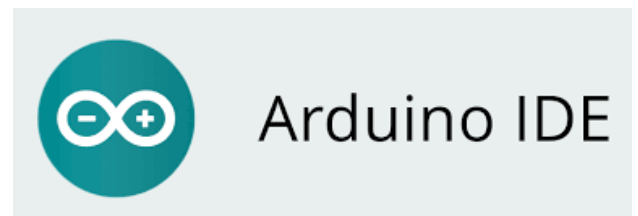


Καθώς οι περισσότερες εφαρμογές λογισμικού πρέπει να αλληλεπιδρούν με δεδομένα σε κάποια μορφή, οι γλώσσες προγραμματισμού όπως η Python παρέχουν εργαλεία για την αποθήκευση και την πρόσβαση σε αυτές τις πηγές δεδομένων²².

Αποτελεί ανοιχτό κώδικα από την έναρξή της το 1995, η MySQL έγινε γρήγορα ηγέτης στην αγορά μεταξύ των βάσεων δεδομένων SQL. Η MySQL είναι επίσης μέρος του οικοσυστήματος της Oracle. Επί του παρόντος, η MySQL χρησιμοποιείται από όλες τις μεγάλες εταιρείες τεχνολογίας, συμπεριλαμβανομένων των Google, LinkedIn, Uber, Netflix, Twitter και άλλων.

3.4 Arduino IDE

Για την αρχική επικοινωνία υπολογιστή-συστήματος χρησιμοποιήθηκε το Arduino IDE, ένα δωρεάν software που παρέχεται από την ίδια την Arduino. Ο κώδικας που χρησιμοποιήθηκε για τον προγραμματισμό του μικροεπεξεργαστή δίνεται από την εταιρία (Sparkfun) που κατασκευάζει την πλακέτα ανάγνωσης Rfid tag και παρατίθεται στο [Παράρτημα Β](#). Διατίθεται δωρεάν στο διαδίκτυο για χρήση από το ευρύ κοινό, ή προφανώς τους αγοραστές της πλακέτας τους. Με τις κατάλληλες παραλλαγές προσαρμόζεται στην εκάστοτε εφαρμογή, κάνοντας χρήση της εφαρμογής Arduino IDE. Για την σωστή συνεργασία της πλακέτας Arduino με την πλακέτα Sparkfun Rfid Reader M6E-NANO είναι απαραίτητη η εγκατάσταση της βιβλιοθήκης “SparkFun_UHF_RFID_Reader.h” στην εφαρμογή του IDE (<https://www.arduino.cc/reference/en/libraries/sparkfun-simultaneous-rfid-tag-reader-library/>).



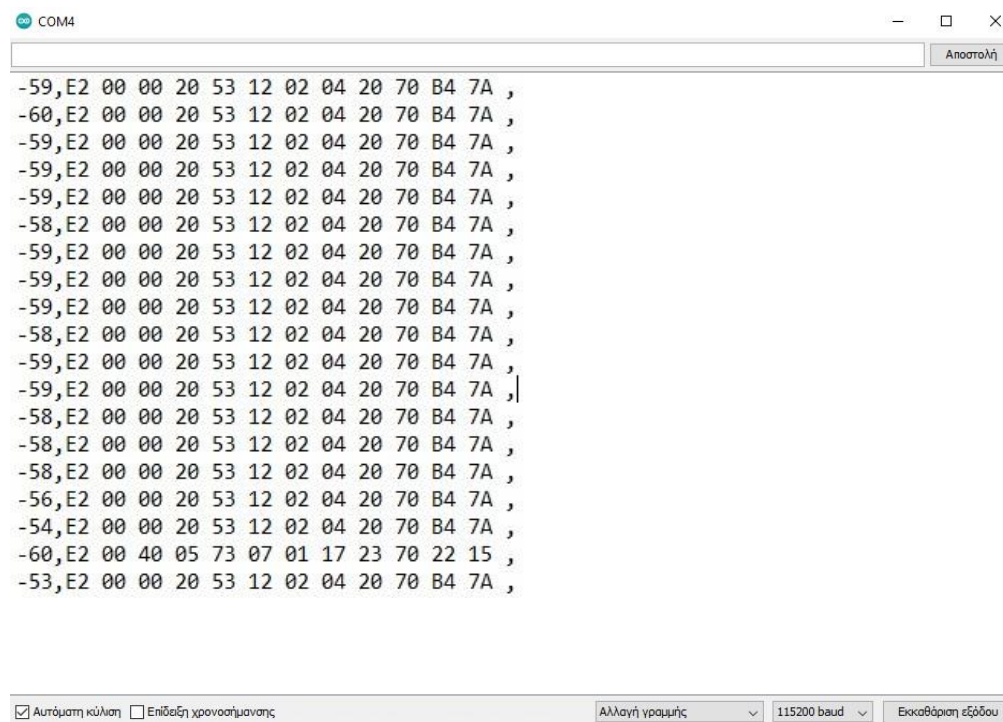
Μέσω του IDE γίνεται η πρώτη επαφή με τον εξοπλισμό και το πρόγραμμα, ενώ ταυτόχρονα πραγματοποιείται ο έλεγχος σωστής λειτουργίας του κώδικα Arduino και των εξαρτημάτων. Όταν ολοκληρωθεί το ανέβασμα του κώδικα στην πλακέτα, ανοίγουμε το

²¹ <https://www.geeksforgeeks.org/python-mysql/>

²² <https://realpython.com/python-mysql/>

Serial plotter. Από τις επιλογές στο κάτω μέρος του παραθύρου επιλέγεται τιμή Baud rate 115200.

Κατά το πέρασμα των ετικετών πάνω από το reader εμφανίζονται κάποια από τα βασικά τους στοιχεία, όπως η RSSI τιμή και το αναγνωριστικό τους (Tag ID). Ανάλογα με τις απαιτήσεις η εξαγωγή δεδομένων μπορεί να αλλάξει και με τις κατάλληλες ρυθμίσεις μέσα από τον κώδικα μπορούν να εμφανίζονται πληροφορίες, όπως η χρονική στιγμή που πέρασε κάποια ετικέτα πάνω από την κεραία. Επίσης τροποποιήσεις μπορούν να γίνουν στο Baud rate και στην μέγιστη ισχύ τροφοδοσίας της θύρας USB. Οι αλλαγές αυτές μπορούν να εντοπιστούν μέσα στον κώδικα σε μορφή σχολίων. Ωστόσο, οποιαδήποτε τροποποίηση στον κώδικα του Arduino επηρεάζει άμεσα τη λειτουργία της εφαρμογής RFID Asset Management System καθώς τα προκαθορισμένα στοιχεία RSSI και Tag ID έχουν οριστεί ως δεδομένα εισόδου.



Εικόνα 27. Απεικόνιση του serial plotter με δεδομένα.

3.5 Εγχειρίδιο χρήσης εφαρμογής (User Manual)

Σε αυτό το υποκεφάλαιο θα επισημανθούν τα σημεία του κώδικα που οφείλει ο εκάστοτε χρήστης να τροποποιήσει ώστε να επιτευχθεί η σύνδεση του εξοπλισμού με οποιοδήποτε Η/Υ. Επίσης γίνεται αναφορά στην εγκατάσταση και χρήση των προγραμμάτων του κεφαλαίου 3, τα οποία είναι απαραίτητα για την ορθή λειτουργία της εφαρμογής.

Ο κώδικας της εφαρμογής, παρατίθεται στο [Παράρτημα Α](#), είναι γραμμένος σε γλώσσα Python και συγκεκριμένα στην έκδοση 3.9.13 η οποία είναι διαθέσιμη στην ιστοσελίδα <https://www.python.org/downloads/release/python-3913/>. Το περιβάλλον που επιλέχθηκε είναι το PyCharm, ένα δωρεάν λογισμικό που διατίθεται στο διαδίκτυο κατεβάζοντας την community edition 2023 (<https://www.jetbrains.com/pycharm/>). Οι βιβλιοθήκες που χρησιμοποιήθηκαν αναγράφονται για ευκολία στην αρχή του κώδικα με την ετικέτα “import” και είναι απαραίτητη η εγκατάστασή τους.

Όσον αφορά την δημιουργία της βάσης δεδομένων τα εργαλεία που χρησιμοποιήθηκαν για τον σκοπό αυτό είναι το MySQL Server Installer (<https://dev.mysql.com/downloads/installer/>), υπεύθυνο για την εγκατάσταση μιας βάσης στον υπολογιστή και έπειτα η εφαρμογή WorkBench 8.0 CE (Visual Database Designer) (<https://dev.mysql.com/downloads/workbench/>), όπου πραγματοποιείται η επεξεργασία της.

3.5.1 Σύνδεση Python με MySQL database

Ο κώδικας χρησιμοποιεί μια βάση δεδομένων SQL για την αποθήκευσή των πληροφοριών και δεμένων των RFID tags που καταγράφει και χειρίζεται το σύστημα. Είναι απαραίτητη η δημιουργία μιας τοπικής βάσης δεδομένων MySQL στον υπολογιστή που τρέχει την εφαρμογή μας. Η σύνδεση γίνεται μέσα στον κώδικα αλλάζοντάς τις αντίστοιχες παραμέτρους ώστε να ταιριάζουν με αυτές του εκάστοτε συστήματος λειτουργίας.

Στο παρακάτω κομμάτι κώδικα φαίνονται τα στοιχεία τα οποία πρέπει να ρυθμίσει ο κάθε χρήστης στον προσωπικό του υπολογιστή για να επιτύχει την επικοινωνία της βάσης με τον κώδικα.

```
import mysql.connector

# mySQL Database connection parameters
host = "localhost"
user = "root"
password = "test123"
database = "rfid_database"
```

```
# Attempt to establish a connection to the database
connection = mysql.connector.connect(
    host=host,
    user=user,
    password=password,
    database=database
)

# Create a cursor to interact with the database
cursor = connection.cursor()
```

3.5.2 Σύνδεση Arduino με Python

Για να επιτευχθεί η σειριακή επικοινωνία μεταξύ του Arduino και του κώδικα στην Python είναι απαραίτητο να αντικατασταθεί η τιμή της μεταβλητής `serial_port` στο απόσπασμα που φαίνεται παρακάτω. Ο εντοπισμός της θύρας στην οποία είναι συνδεδεμένη η συσκευή γίνεται μέσω της διαχείρισης συσκευών των Windows (Universal Serial Bus Controllers).

Επίσης είναι σημαντικό να συμβαδίζει η τιμή της μεταβλητής Baud Rate με αυτή που δόθηκε στον κώδικα του Arduino (Arduino IDE), όπως αναφέρθηκε στο υποκεφάλαιο 3.4.

```
import serial

#Define the serial port and baud rate for communication with Arduino
serial_port = 'COM3'

baud_rate = 115200

#Connect to Arduino through serial port
arduino = serial.Serial(serial_port, baud_rate)
```

3.5.3 Δημιουργία πινάκων SQL

Μέσα στην βάση δεδομένων πρέπει να δημιουργηθούν οι παρακάτω 4 πίνακες για την αποθήκευση δεδομένων που ανιχνεύει το Reader. Οι πίνακες αντίστοιχα είναι

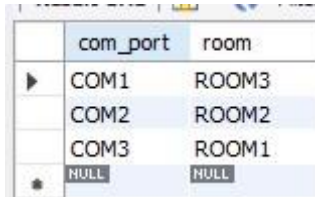
A) **all tags**: Ο πίνακας που αποθηκεύει όλες τις ετικέτες που εισάγονται στο σύστημα

B) **inside tags**: Οι ετικέτες που βρίσκονται εντός κάποιου δωματίου.

Γ) **outside tags**: Οι ετικέτες που δεν βρίσκονται σε κάποιο δωμάτιο.

Δ) **com_port_mapping**: Σε αυτόν τον πίνακα περιέχονται οι αντιστοιχίσεις θυρών `com_port` (συνδεδεμένων Arduino), με τα τρία διαθέσιμα δωμάτια που προσφέρει η

εφαρμογή. Ο πίνακας πρέπει να συμπληρωθεί από τον χρήστη όπως φαίνεται στην εικόνα 27B, τροποποιώντας ανάλογα την στήλη των θυρών σύμφωνα με τον εκάστοτε υπολογιστή. Τα περιεχόμενα αυτού του πίνακα φορτώνονται στο πρόγραμμα στην έναρξη της εφαρμογής.



com_port	room
COM1	ROOM3
COM2	ROOM2
COM3	ROOM1
NULL	NULL

Εικόνα 28. Παράδειγμα συμπληρωμένου πίνακα com_port_mapping.

Οι εντολές για την δημιουργία πινάκων μέσα στην βάση δεδομένων αναγράφονται παρακάτω:

CREATE database → Δημιουργία ενός Schema μέσα στην βάση δεδομένων.

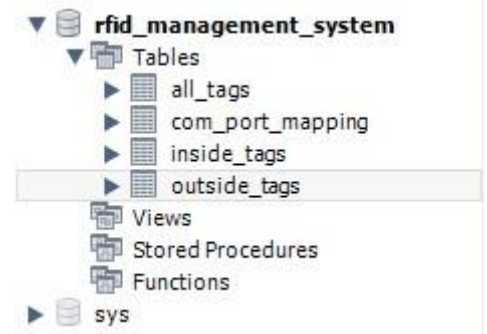
USE database → Επιλογή και επεξεργασία του συγκεκριμένου Schema.

```
CREATE TABLE all_tags (  
    tag_id int NOT NULL AUTO_INCREMENT,  
    tag varchar(255) NOT NULL,  
    PRIMARY KEY (tag_id)  
);
```

```
CREATE TABLE com_port_mapping (  
    com_port varchar(255) NOT NULL,  
    room varchar(255) NOT NULL,  
    PRIMARY KEY (com_port)  
);
```

```
CREATE TABLE inside_tags (  
    tag varchar(255) NOT NULL,  
    room varchar(255) DEFAULT NULL,  
    PRIMARY KEY (tag)  
);
```

```
CREATE TABLE outside_tags (  
    tag varchar(255) NOT NULL,  
    PRIMARY KEY (tag)  
);
```



Εικόνα 29. Οι πίνακες της βάσης

3.5.4 Δημιουργία κωδικού πρόσβασης

Για την ασφάλεια των δεδομένων της εφαρμογής απαιτείται η εισαγωγή κωδικού πρόσβασης. Αυθαίρετα έχει οριστεί η λέξη “test” και μπορεί να αλλάξει τροποποιώντας την τιμή της μεταβλητής όπως φαίνεται παρακάτω.

```
# Check the password of the intro screen
```

```
def check_password():
```

```
→ correct_password = 'test' ←
```

```
    entered_password = password_entry.get()
```



Εικόνα 30. Παράθυρο εισαγωγής κωδικού.

Κεφάλαιο 4 : Παρουσίαση εργασίας - Περιγραφή εφαρμογής

4.1 Σύντομη περιγραφή της εφαρμογής

Η βασική λειτουργία της εφαρμογής είναι η συνεχής παρακολούθηση των tags τα οποία μπορεί να έχουν τοποθετηθεί σε διαφορά προϊόντα, περιουσιακά στοιχεία, ή ακόμα και σε ζώα, ώστε να υπάρχει μία αδιάκοπη μετάδοση της πληροφορίας και της τοποθεσίας της κάθε ετικέτας.

Στη συνέχεια θα παρουσιαστεί σε πραγματικό χρόνο πως επιτυγχάνεται η ανάλυση των δεδομένων. Αρχικά τοποθετείται η κεραία σε κάποιο κεντρικό σημείο διέλευσης των tags, όπως για παράδειγμα η είσοδος ενός δωματίου. Η κεραία συνδέεται με την πλακέτα του Arduino/Sparkfun και τέλος τοποθετείται το καλώδιο δεδομένων στον υπολογιστή. Στόχος είναι κάθε ετικέτα που θα εισέρχεται ή θα εξέρχεται από το δωμάτιο να ελέγχεται ανά πάσα στιγμή.

Όπως θα φανεί και παρακάτω η εφαρμογή είναι χωρισμένη σε δύο βασικά παράθυρα-πίνακες, ένα με τα προϊόντα που βρίσκονται εντός και ένα με αυτά που βρίσκονται εκτός του δωματίου. Γενικά έχει οριστεί αυθαίρετα πως όταν εντοπίζεται μια καινούρια ετικέτα από το σύστημα, θα καταχωρείται αυτόματα στον πίνακα 'έξω'. Να σημειωθεί ότι παρέχεται η δυνατότητα της χειροκίνητης μεταφοράς μίας ετικέτας, για να μπορέσουν να καλυφθούν όλα τα πιθανά σενάρια που αφορούν την αρχική θέση μιας ετικέτας. Αν μια ετικέτα είναι τοποθετημένη στο εξωτερικό του δωματίου, με το πέρασμά της μπροστά από την κεραία, η κατάσταση της θα αλλάξει αυτόματα και θα μεταφερθεί στο εσωτερικό του, και το αντίθετο. Με αυτό τον τρόπο υπάρχει άμεση εικόνα για την μετακίνηση των ετικετών και την τοποθεσία τους. Πηγαίνοντας ένα βήμα παρακάτω, ο έλεγχος των tags θα μπορούσε να επεκταθεί σε μεγαλύτερη εμβέλεια και ίσως με τα κατάλληλα εργαλεία, σε παγκόσμιο επίπεδο.

Όλα τα δεδομένα αυτά αποθηκεύονται στη βάση δεδομένων MySQL, όπου ο κώδικας συνεργάζεται με τις εντολές «ανάγνωση», «ανάκτηση» και «τροποποίηση» των πληροφοριών μέσα από τους πίνακες της βάσης. Γίνεται συνεχής έλεγχος κάθε φορά που ένα νέο tag διαβάζεται, μεταφέρεται ή επεξεργάζεται ώστε να αποτραπεί κάποιο λάθος είτε λόγω κακών αναγνώσεων από τον reader, είτε λόγω κακού χειρισμού από τον χρήστη. Στόχος είναι να διατηρείται η ακεραιότητα της αποθηκευμένης πληροφορίας σε όλη τη διάρκεια της εκτέλεσης του προγράμματος.

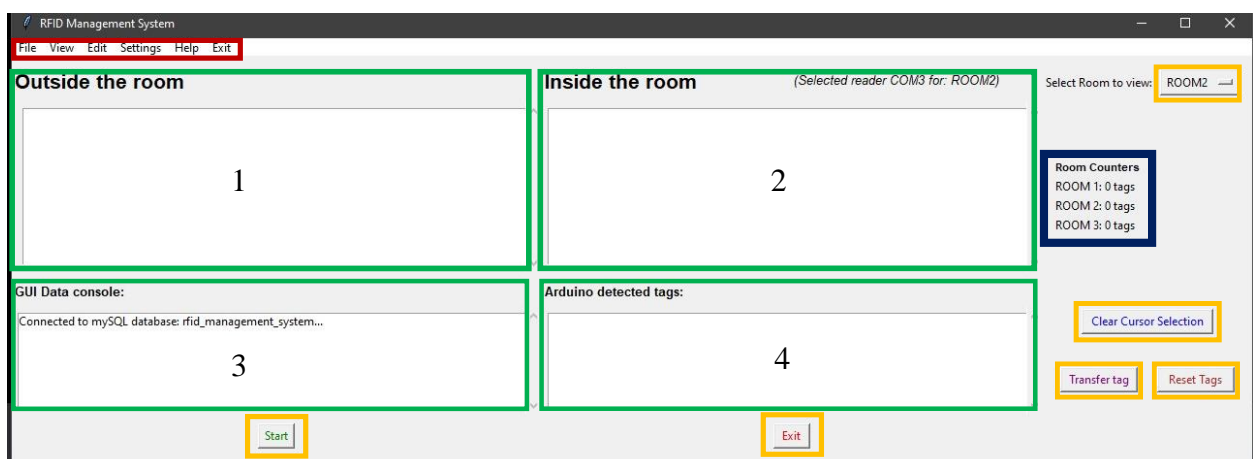
Μια σημαντική παρατήρηση για τις δυνατότητες του κώδικα είναι ότι το πρόγραμμα μπορεί να διαχειριστεί μέχρι τρία ξεχωριστά δωμάτια, όπου το καθένα ελέγχεται από διαφορετικό reader-Arduino. Στην παρούσα φάση της εργασίας, δεν υποστηρίζεται η ταυτόχρονη χρήση πολλαπλών Arduino, οπότε πρέπει να επιλεγθεί χειροκίνητα η κάθε αλλαγή δωματίου.

4.2 Γραφικό περιβάλλον και λειτουργίες

Σε αυτό το σημείο της εργασίας θα γίνει η αποτύπωση του γραφικού περιβάλλοντος της εφαρμογής και του τελικού αποτελέσματος που δημιουργήθηκε, μέσω ορισμένων φωτογραφιών. Επιλέχθηκε το πακέτο (βιβλιοθήκη) Tkinter, καθώς είναι εύκολο στη χρήση, κάτι που το καθιστά μια εξαιρετική λύση για αρχάριους.

➤ Γραφικό περιβάλλον

- Το παρακάτω παράθυρο είναι η πρώτη εικόνα που αντικρίζει ο χρήστης κατά την έναρξη της εφαρμογής, το αρχικό GUI (Graphical User Interface).
- Το κελί 1 είναι το παράθυρο στο οποίο εμφανίζονται οι ετικέτες οι οποίες έχουν τοποθετηθεί εκτός δωματίου. Στο κελί 2 βρίσκονται οι ετικέτες που είναι εντός του δωματίου, ενώ στο κελί 3 απεικονίζονται διάφορες πληροφορίες για την λειτουργία του προγράμματος. Τέλος στο κελί 4 δίνονται αυτούσιες οι πληροφορίες που διαβάζει η κεραία μέσω της πλακέτας Arduino.
- Με κόκκινο πλαίσιο έχει σημειωθεί η μπάρα εργαλείων η οποία αποτελείται από έξι βασικά πλήκτρα File, View, Edit, Settings, Help, Exit. Κάθε ένα από αυτά μπορεί έχει μια συγκεκριμένη λειτουργία με βάση τις επιθυμίες του χρήστη, καθώς βοηθάει στην ευκολότερη χρήση του προγράμματος.
- Στο μπλε πλαίσιο παρέχονται πληροφορίες σχετικά με το πόσα tags περιέχει το κάθε δωμάτιο. Η επιλογή δωματίου γίνεται από το menu Select Room, στο πάνω δεξιά μέρος του παραθύρου.
- Με πορτοκαλί πλαίσιο έχουν σημειωθεί βασικά πλήκτρα για την λειτουργία και τον χειρισμό όλου το προγράμματος.



Εικόνα 31. Γραφικό περιβάλλον εφαρμογής (αρχικό παράθυρο)

➤ **Λειτουργίες (πλήκτρα) της εφαρμογής**

- **Start:** Το πλήκτρο αυτό έχει πολύ βασικό ρόλο στο πρόγραμμα καθώς ενεργοποιεί την λήψη των δεδομένων από τον αναγνώστη-arduino. Ένα κύριο πρόβλημα που αντιμετωπίστηκε είναι ότι η λειτουργία του γραφικού περιβάλλοντος αποτελεί μια συνεχή «λούπα», με αποτέλεσμα να αγνοούνται οι πληροφορίες που παρέχει η πλακέτα. Την λύση στο πρόβλημα δίνει το πλήκτρο Start, μέσω της μεθόδου Multi-threading. Το πάτημα του Start δημιουργεί ένα ξεχωριστό thread στο πρόγραμμα που του επιτρέπει να εκτελεί ταυτόχρονα και τις δύο υπηρεσίες.

- **Exit:** Πλήκτρο το οποίο τερματίζει το πρόγραμμα και κλείνει το παράθυρο της εφαρμογής. Παράλληλα τερματίζει την σύνδεση με το Arduino και την βάση δεδομένων κάνοντας τους κατάλληλους ελέγχους για την ομαλή διακοπή όλων των λειτουργιών.

- **Transfer tag:** Χειροκίνητη μεταβίβαση ετικέτας. Υπάρχει η επιλογή χειροκίνητης μεταφοράς ενός tag από το εσωτερικό στο εξωτερικό του δωματίου, ή αντίστροφα, για λόγους διόρθωσης λάθους (κακή ανάγνωση), ή για οποιαδήποτε ανάγκη ορισμού αρχικοποίησης των θέσεων των ετικετών.

- **Reset tags:** Ενεργοποιεί την εκκαθάριση δεδομένων του προγράμματος. Διαγραφεί όλα τα δεδομένα του κώδικα, τόσο του γραφικού περιβάλλοντος αλλά και των πινάκων της βάσης δεδομένων.

- **Clear Cursor Selection:** Κατά την εκτέλεση του προγράμματος παρέχεται η δυνατότητα επιλογής αντικειμένων με τον κέρσορα. Το πλήκτρο «clear cursor» διαγράφει την επιλογή.

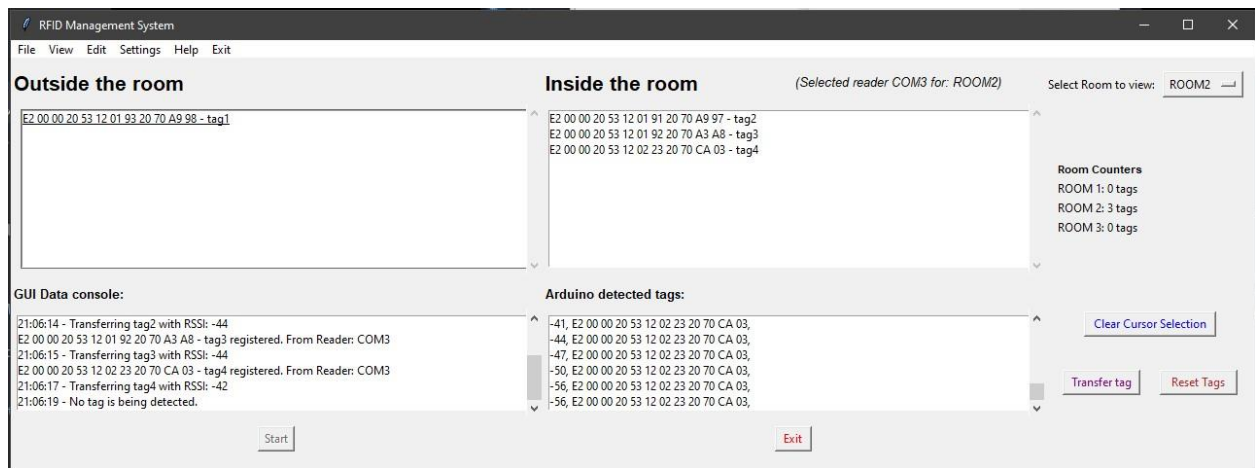
- **Select room to view:** Το τελευταίο πλήκτρο στο πανελ της εργασίας, επιτρέπει την εναλλαγή δωματίων. Το επιλεγμένο δωμάτιο αντικατασθεί το παράθυρο 'Inside Room' στο οποίο εμφανίζονται τα δεδομένα.

➤ **Data Console:** Είναι το παράθυρο παρακολούθησης όλων των αλλαγών στο πρόγραμμα, για παράδειγμα εμφανίζονται δεδομένα όπως η αλλαγή δωματίου, η προσθήκη νέου tag και γενικότερα σημαντικές πληροφορίες.

➤ **Arduino detected tags:** Στο παράθυρο αυτό απεικονίζεται ζωντανά η ανάγνωση της κάθε ετικέτας από το Arduino με την αντίστοιχη τιμή RSSI.

Κάθε tag που ανιχνεύεται παίρνει μια αύξουσα αριθμητική τιμή (1,2,3,4) με βάση την σειρά ανάγνωσης (tag ID). Αυτή η τιμή διατηρείται σε όλο το εύρος του προγράμματος και λειτουργεί σωστά ανεξάρτητα από την ανάγνωση νέων ετικετών.

Η εφαρμογή μεταφέρει αυτόματα ένα tag από το εξωτερικό στο εσωτερικό ενός δωματίου, εάν αυτό περάσει μπροστά από τον αναγνώστη ανιχνεύοντας την μέγιστη τιμή rssi. Εάν το tag εισέλθει στο εύρος ανάγνωσης του reader, αλλά δεν προσεγγίσει την μέγιστη τιμή, τότε θα καταχωρηθεί στο εξωτερικό κελί και δε θα μεταφερθεί. Η μεταφορά πραγματοποιείται εφόσον, και αν, το tag περάσει ολοκληρωτικά από πάνω του.

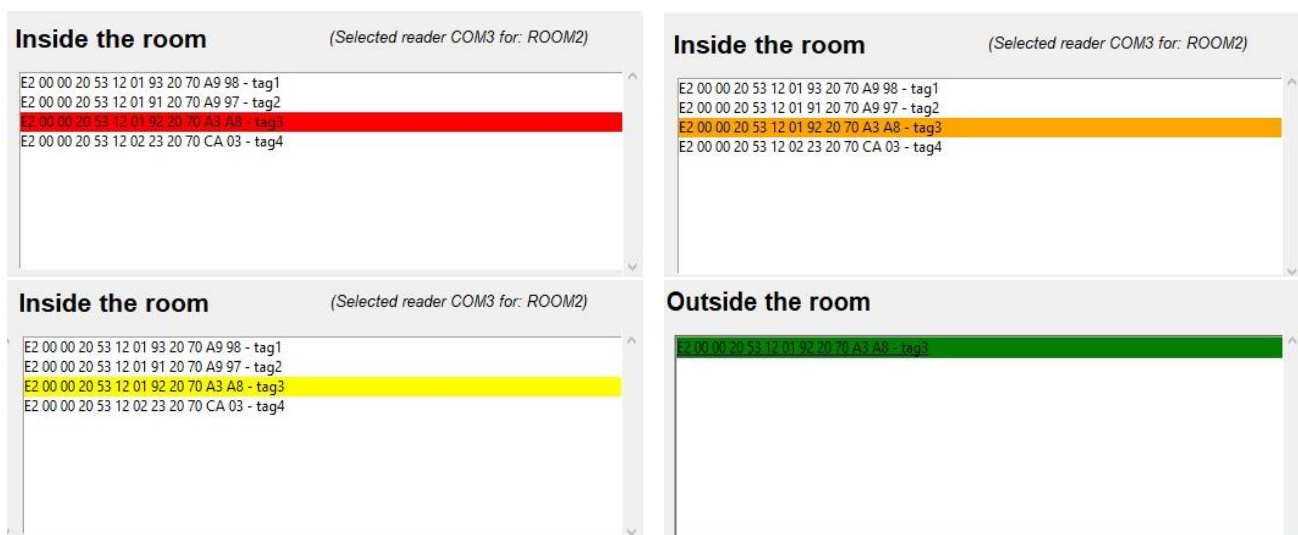


Χρωματική απεικόνιση των Tags

Με βάση την rssi τιμή, τα tags απεικονίζονται με διαφορετικά χρώματα (font color) τα οποία αντιστοιχούν στην απόστασή τους από τον reader. Τα τέσσερα αυτά χρώματα είναι κόκκινο, πορτοκαλί, κίτρινο και πράσινο, τα οποία έχουν οριστεί ώστε να αντιπροσωπεύουν ένα συγκεκριμένο εύρος τιμών.

Ο αρχικός εντοπισμός του tag γίνεται σε τιμές rssi περίπου -65 έως -55 και αντιστοιχεί στο Gui στο κόκκινο χρώμα. Όσο πλησιάζει η ετικέτα στην κεραία, η τιμή rssi μειώνεται μέχρι το εύρος -55 έως -50 όπου το χρώμα γίνεται πορτοκαλί. Με κίτρινο χρώμα συμβολίζονται οι τιμές -50 έως -45 όπου το tag βρίσκεται σχεδόν πάνω από το reader. Τέλος το πράσινο χρώμα συμβολίζει τιμές rssi μεγαλύτερες του -45 και σηματοδοτεί το πέρασμα του tag ακριβώς πάνω από την κεραία. Μόλις εντοπιστεί η μέγιστη τιμή -45 γίνεται αυτόματη μεταφορά του tag στο άλλο list box.

Για την αυτόματη μεταφορά έχει γίνει προεργασία ώστε να αποφευχθούν οι άσκοπες πολλαπλές μεταφορές μεταξύ κελιών, προσθέτοντας στον κώδικα μια δικλείδα ασφαλείας όπου εάν το tag ανιχνευτεί στην peak τιμή του, δεν μπορεί να ξανά μεταφερθεί για τα επόμενα 3 δευτερόλεπτα. Αυτός ο χρόνος αντιστοιχεί περίπου στο διάστημα που χρειάζεται μια ετικέτα για να διανύσει όλη την εμβέλεια της κεραίας.



Κεφάλαιο 5: Έρευνα αγοράς, περιθώρια βελτίωσης

5.1 Εισαγωγή στις εταιρίες RFID

Ένα σύστημα RFID βρίσκει εφαρμογή σε πολλούς τομείς της σύγχρονης ζωής, όπως έχει ήδη αναφερθεί. Κάθε εφαρμογή έχει διαφορετικά κριτήρια τα οποία καθορίζουν την αγορά του απαραίτητου εξοπλισμού. Για την αγορά των υλικών τα δεδομένα που πρέπει να λαμβάνονται υπόψιν είναι η απόσταση, ο χρόνος απόκρισης του συστήματος, η ασφάλεια, η αξιοπιστία καθώς και η οικονομία. Για τον λόγο αυτό υπάρχουν διάφορες εταιρίες οι οποίες παράγουν προϊόντα που ανταποκρίνονται στις απαιτήσεις του καθενός. Μερικές από αυτές είναι οι παρακάτω:

Zebra Technologies Corporation

Η Zebra είναι μια Αμερικανική εταιρία με έτος ίδρυσης το 1969. Κατά την ίδρυση της ειδικευόταν στην κατασκευή ηλεκτρομηχανολογικών προϊόντων υψηλής ταχύτητας. Με το πέρασμα των χρόνων εξαγόραζε διάφορες εταιρίες τεχνολογίας και το 2004 μπήκε στο χώρο των RFID εφαρμογών. Τα τελευταία χρόνια διακρίνεται για της αξιόπιστες και οικονομικές λύσεις που προσφέρει ενώ ειδικεύεται σε λύσεις υλικού και λογισμικού εταιρικού επιπέδου. Μεγάλο μέρος της εταιρίας ασχολείται με τους σαρωτές γραμμωτού κώδικα και την RFID τεχνολογία.



Impinj, Inc.

Κορυφαίος κατασκευαστής στον χώρο των UHF RFID GEN 2 τεχνολογιών θεωρείται η Αμερικανική εταιρία Impinj. Ιδρύθηκε το 2000 και από τότε προσφέρει λύσεις για σύνδεση καθημερινών αντικειμένων με το διαδίκτυο (IoT). Η εταιρία δραστηριοποιείται κυρίως στην παραγωγή RFID Tag UHF GEN2, RFID readers speedway, τσιπάκια ανάγνωσης Indy RFID καθώς και στην ανάπτυξη εφαρμογών.



ThingMagic

Η εταιρία ThingMagic αποτελεί πλέον μέρος της JADAK. Εξειδικεύεται στην κατασκευή RFID readers, προσφέροντας λύσεις τόσο σε επιχειρήσεις όσο και σε εφαρμογές μικρότερων απαιτήσεων. Παράδειγμα αποτελεί το τσιπάκι της πλακέτας Sparkfun RFID Reader.



Avery Dennison

Από τις μεγαλύτερες στο χώρο της RFID τεχνολογίας με εμπειρία πάνω από 70 χρόνια είναι η εταιρία Avery Dennison. Με τεράστια ποικιλία υλικών και λύσεων σημείωσε το 2015 πωλήσεις 7 δις δολαρίων κατατάσσοντας την στο νούμερο 435 στη λίστα Fortune 500. Tags UHF, HF NFC, ετικέτες ενδυμάτων και ιατρικά προϊόντα είναι κάποιες από τις ελάχιστες παροχές που προσφέρει η εταιρία. Μεγάλο κομμάτι αποτελούν επίσης τα logistics, οι βιομηχανικές εφαρμογές και οι εφοδιαστικές αλυσίδες.



Alien Technology

Άλλη μια αξιοσημείωτη εταιρία είναι η Alien Technology. Με έδρα την Καλιφόρνια και έτος ίδρυσης το 1994 προσφέρει ποιοτικές λύσεις σε θέματα τεχνολογίας. Στους βασικούς στόχους της συμπεριλαμβάνεται η παραγωγή ετικετών υψηλής αντοχής για διάφορες χρήσεις εξωτερικές και εσωτερικές, η κατασκευή κεραιών και αναγνώστων.



Neosid Pemetzrieder GmbH & Co. KG

Τέλος μια ευρωπαϊκή και συγκεκριμένα μια Γερμανική εταιρία που ειδικεύεται στην κατασκευή ηλεκτρομαγνητικών εξαρτημάτων κυρίως στον τομέα της βιομηχανίας. Ιδρύθηκε το 1989 με έδρα το Βερολίνο και έκτοτε εξελίσσεται και αναπτύσσεται συνεχώς στον τομέα της τεχνολογίας.



5.2 Έρευνα αγοράς

Με την εξέλιξη της τεχνολογίας και την ανάπτυξη νέων εταιρειών που διαρκώς δημιουργούνται και αναπτύσσονται διεθνώς στον τομέα των νέων τεχνολογιών, η ποικιλία στον χώρο των RFID συστημάτων είναι τεράστια. Η γκάμα των προϊόντων περιλαμβάνει από οικονομικές λύσεις για το ευρύ κοινό, μέχρι εξειδικευμένες λύσεις για βιομηχανίες και ερευνητικά project. Το ερώτημα που γεννάται είναι ποσά χρήματα αξίζει να διαθέσει ο καταναλωτής για την επίτευξη του στόχου του και τι ακριβώς θέλει να πετύχει.

Στην συνέχεια θα παρουσιαστούν προϊόντα του εμπορίου τα οποία έχουν παρόμοιο τρόπο λειτουργίας με τα υλικά που χρησιμοποιήθηκαν στην διπλωματική εργασία.

5.2.1 Κεραίες RFID

1. <https://www.sparkfun.com/products/14131>



- Ρεύμα τροφοδοσίας:
Min: 8 V
Max: 12 V
- Συχνότητα:
Min: 860 MHz
Max: 960 MHz
- Κέρδος: 6dbi
- Διαστάσεις: 233x200x60 mm
- Βάρος: 1.5Kg
- Πόλωση: Γραμμική Κατακόρυφη

Τιμή : 40,00€

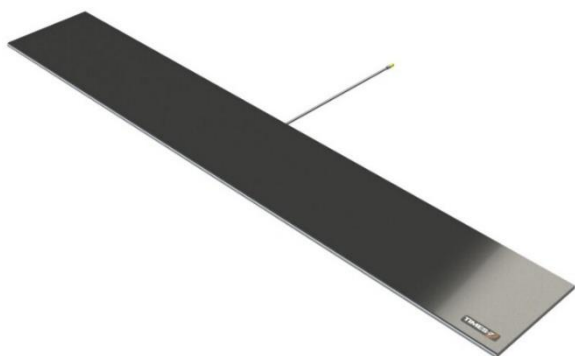
2. https://www.amazon.de/SR681-Outdoor-Antenna-Wiegand-Integrated/dp/B08KDSCDJT/ref=sr_1_1_sspa?crd=ITT88HGQDEAP&keywords=rfid%2Bantenne&qid=1679343481&srefix=rfid%2Ban%2Caps%2C135&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&smid=ACVM1KOS4WVL0&th=1



- Ρεύμα τροφοδοσίας :
Min: 8 V
Max: 12 V
- Συχνότητα:
Min: 860 MHz
Max: 960 MHz
- Κέρδος: 9dbi
- Διαστάσεις : 39.1x27.5x7.4cm
- Βάρος: 1.45Kg
- Πόλωση: Κυκλική

Τιμή : 239.67€

3. <https://www.cisper.nl/en/times-7-a5531c-ground-antenna-etsi-1200-x-195-x-10mm>



- Ρεύμα τροφοδοσίας :
Min: -
Max: -
- Συχνότητα:
Min: 864 MHz
Max: 928 MHz
- Κέρδος: 10dbi
- Διαστάσεις : 1200x195 x 10 mm
- Βάρος: 3.8 Kg
- Πόλωση: Οριζόντια Γραμμική

Τιμή : 498.63 €

5.2.2 Αναγνώστες RFID

1. <https://www.cisper.nl/en/keonn-advanreader-10-1-port-rfid-uhf-high-performance-usb-reader-with-enclosure>



- Ικανότητα ανάγνωσης: 150 tags/second
- Κέρδος: 27dBm
- Διαστάσεις: 68 x 68 x 10.7 mm
- Συχνότητα: 865,6 MHz - 867,6 MHz

Τιμή : 450.23 €

2. <https://www.cisper.nl/en/kathrein-rru-4500-reader-unit-4port-poe-krai-linux-ip67-etsi>



- Κέρδος 33 dBm
- Διαστάσεις: 300 x 300 x 71 mm
- Συχνότητα: 865–868 MHz

Τιμή : 1,770.00 €

3. <https://www.cisper.nl/en/impinj-r720-rain-rfid-reader-etsi>

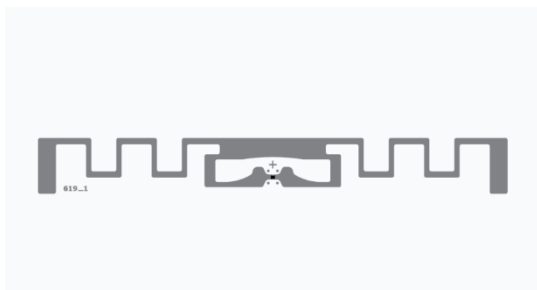


- Κέρδος: 10 - 33 dBm
- Διαστάσεις: 21.5 x 18.7 x 3.0 cm
- Συχνότητα: 902 – 928 MHz

Τιμή : 2,169.28 €

5.2.3 Ετικέτες RFID

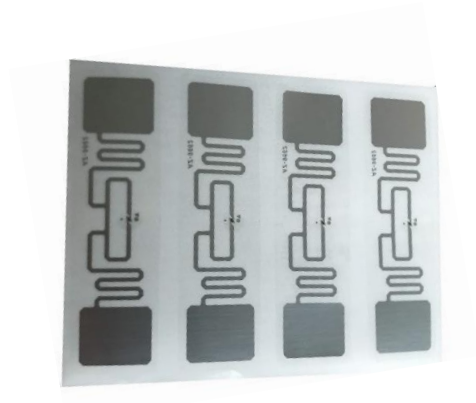
1. <https://www.cisper.nl/en/avery-dennison-squarewave-m730-label>



- EPC και User Μνήμη: 128-bit / 0-bit
- Διαστάσεις: – 97 x 15 mm
- Συχνότητα: UHF 860 - 960 MHz
- Θερμοκρασία λειτουργίας: -40 °C / 85 °C
- Chip: Impinj M730

Τιμή 0,61€/Τεμάχιο

2. <https://www.amazon.com/AZ9662-ISO18000-6C-Range-73-5x21-2mm-Adhesive/dp/B01LYBKMYM?th=1>



- EPC και User Μνήμη: 96bits / 512bits
- Διαστάσεις 73.5x21.2mm
- Συχνότητα: UHF 860-960MHZ
- Θερμοκρασία λειτουργίας: -40 °C / 80 °C
- Chip: ALIEN Higgs-3

Τιμή 0,18€/Τεμάχιο

3. <https://www.amazon.sa/-/en/UHF-860-960MHZ-73-5x22-5mm-Application-Management/dp/B09B784T4J>



- Μνήμη : 512Bits
- Διαστάσεις: 73*23mm
- Συχνότητα: 860-960Mhz
- Θερμοκρασία λειτουργίας: -35 °C / 75 °C
- Chip -

Τιμή 0,39€/Τεμάχιο

4. <https://www.cisper.nl/en/perfect-id-rock-112>



- EPC Μνήμη: 128 bits /96 bits
- Διαστάσεις :112 x 20 x 14 mm
- Συχνότητα: 902-928MHz
- Θερμοκρασία λειτουργίας: -40 °C / 85 °C
- Chip: IMPINJ MONZA R6P

Τιμή 2,5 €/Τεμάχιο

5. <https://www.cisper.nl/en/confidex-ironside-classic-m4qt-with-background-adhesive>



- EPC και User Μνήμη: 496 bit / 128 bit
- Διαστάσεις :51,5 x 47,5 x 10 mm
- Συχνότητα: 865 - 928 MHz
- Θερμοκρασία λειτουργίας: -40°C / +85°C
- Chip: Impinj M780

Τιμή 4,93 €/Τεμάχιο

6. <https://www.cisper.nl/en/xerafy-roswell-alien-higgs-3-fcc>



- Μνήμη EPC: 96 bits
- Διαστάσεις : 148 x 28 x 13.50 mm
- Συχνότητα: 865-868 MHz
- Θερμοκρασία λειτουργίας: -40°C to +85°C
- Chip: Alien Higgs 3

Τιμή 13,45 €/Τεμάχιο

5.3 Προτάσεις βελτίωσης του υπάρχοντος εξοπλισμού

Όπως συμπεραίνεται από τα παραπάνω, οι επιλογές ποικίλουν τόσο σε τιμή όσο και σε ποιότητα καθώς και σε τρόπο χρήσης. Κάθε κατασκευαστής προσφέρει μια γκάμα επιλογών ανάλογα με τις ανάγκες του πελάτη. Για παράδειγμα υπάρχουν ετικέτες με 18 λεπτά του ευρώ ενώ ταυτόχρονα κάποιες άλλες κοστίζουν αρκετά παραπάνω. Η έρευνα που έγινε είναι ενδεικτική και δεν συμπεριλαμβάνει όλες τις διαβαθμίσεις τιμών και δυνατοτήτων. Όσον αφορά τα υλικά που χρησιμοποιήθηκαν στην διπλωματική εργασία, συμπεραίνεται ότι αυτά καλύπτουν το μεγαλύτερο εύρος των απαιτήσεων.

Μια πιθανή βελτίωση θα ήταν στις ετικέτες, καθώς κάποιες ακριβότερες πιθανόν να έδιναν καλύτερα αποτελέσματα, χωρίς αυτό να σημαίνει ότι το ακριβότερο είναι πάντοτε και καλύτερο. Ένα σύστημα RFID αποτελείται από τρία βασικά μέρη, οπότε εάν αλλάξει κάποιο από αυτά χωρίς να βελτιωθούν τα άλλα δύο τα αποτελέσματα μπορεί να παραμείνουν αμετάβλητα. Συμπερασματικά πριν την έναρξη κάποιου project θα πρέπει να γίνεται σωστή έρευνα αγοράς ανάλογα με τις ανάγκες της εκάστοτε περίπτωσης και τους διαθέσιμους πόρους.

Βιβλιογραφία

1. The Kimball Group Reader: Relentlessly Practical Tools for Data Warehousing and Business Intelligence, 2nd Edition John Wiley & Sons, Indianapolis USA, 2016
2. Sweeney P. J., “RFID for Dummies”, 1st Edition, Wiley Publishing Inc., Indiana, USA, 2005
3. Bill Glover, Himanshu Bhatt, RFID Essentials O'Reilly Media Inc., California, USA, 2006
4. Vacca R. John, Computer and Information Security Handbook, Morgan Kaufmann Publishers, Burlington USA, 2009
5. Keskilammi, Sydanheimo & Kivikoski, Passive RFID Systems and the Effects of Antenna Parameters on Operational Distance in Radio Frequency Technology for Automated Manufacturing and Logistics Control, 2003
6. Παναγιώτης Παπάζογλου – Σπύρος Πολυχρόνης Λιώνης, (2018), «Ανάπτυξη Εφαρμογών με το Arduino», 2η έκδοση, Εκδόσεις Τζιόλα, Θεσσαλονίκη.
7. Kamran AHSAN - Hanifa SHAH - Paul KINGSTON, (2010), «RFID Applications: An Introductory and Exploratory Study», Stafford, United Kingdom.
8. S.Bagirathi - Sharmila Sankar and Sandhya, (2016), «Tag detection in RFID system based on RSSI technique for LF and HF Passive Tags», India.
9. Kamran AHSAN - Hanifa SHAH - Paul KINGSTON, (2010), «RFID Applications: An Introductory and Exploratory Study», Stafford, United Kingdom.

Διαδικτυακές Πηγές

- 1) <https://www.pemptousia.gr/2021/06/rfid-radio-frequency-identification/>
- 2) <https://www.rfidjournal.com/faq/whats-the-difference-between-read-only-and-read-write-rfid-tags>
- 3) <https://gaorfid.com/el/rfid-tags-market-report/>
- 4) <https://www.graphicarts.gr/enimerosi/arthra-themata/tecnologia-rfid>
- 5) <https://www.linkedin.com/pulse/rfid-middleware-introduction-bella-gao-1c>
- 6) <https://www.techtarget.com/iotagenda/definition/RFID-radio-frequency-identification>
- 7) <https://www.everythingrf.com/community/microwave-frequency-shf-rfid-tags-systems>
- 8) https://www.slideshare.net/ashtopustech/rfid-technology-next-generation-application-solutions-49590687?from_action=save
- 9) <https://www.trendmicro.com/vinfo/pl/security/news/security-technology/next-gen-payment-processing-tech-rfid-credit-cards>
- 10) <https://fastercapital.com/content/Rfid-in-Smart-Cities--Enabling-Seamless-Urban-Living.html>
- 11) <https://spacezilotes.wordpress.com/2012/11/17/%CE%AF-%CF%8E/>
- 12) <https://datanalysis.net/research-design/spss-r-python-stata-meionehtimata-pleonehtimata/>
- 13) <https://www.educba.com/what-is-pycharm/>
- 14) <https://support.microsoft.com/el-gr>
- 15) <https://www.geeksforgeeks.org/python-mysql/>
- 16) <https://realpython.com/python-mysql/>
- 17) <https://www.sparkfun.com/products/14131>
- 18) https://www.amazon.de/SR681-Outdoor-Antenna-Wiegand-Integrated/dp/B08KDSCDJT/ref=sr_1_1_sspa?crid=ITT88HGQDEAP&keywords=rfid%2Bantenne&qid=1679343481&sprefix=rfid%2Ban%2Caps%2C135&sr=8-1-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&smid=ACVM1KOS4WVL0&th=1

- 19) <https://www.cisper.nl/en/times-7-a5531c-ground-antenna-etsi-1200-x-195-x-10mm>
- 20) <https://www.cisper.nl/en/keonn-advanreader-10-1-port-rfid-uhf-high-performance-usb-reader-with-enclosure>
- 21) <https://www.cisper.nl/en/kathrein-rru-4500-reader-unit-4port-poe-krai-linux-ip67-etsi>
- 22) <https://www.cisper.nl/en/impinj-r720-rain-rfid-reader-etsi>
- 23) <https://www.cisper.nl/en/avery-dennison-squarewave-m730-label>
- 24) <https://www.amazon.com/AZ9662-ISO18000-6C-Range-73-5x21-2mm-Adhesive/dp/B01LYBKMYM?th=1>
- 25) <https://www.amazon.sa/-/en/UHF-860-960MHZ-73-5x22-5mm-Application-Management/dp/B09B784T4J>
- 26) <https://www.cisper.nl/en/perfect-id-rock-112>
- 27) <https://www.cisper.nl/en/confidex-ironside-classic-m4qt-with-background-adhesive>
- 28) <https://www.cisper.nl/en/xerafy-roswell-alien-higgs-3-fcc>
- 29) <https://el.wikipedia.org/wiki/Arduino>
- 30) https://store.arduino.cc/products/arduino-uno-rev3?_gl=1*1qqvs6w*_ga*MTQ4OTE4ODk4MC4xNjc2OTExNTI2*_ga_NEXN8H46L5*MTY5Njg0ODgwNy42LjEuMTY5Njg0ODg4MC4wLjAuMA
- 31) https://en.wikipedia.org/wiki/Arduino_Uno
- 32) <https://www.sparkfun.com/products/14066>
- 33) https://cdn.sparkfun.com/assets/4/e/5/5/0/SEN-14066_datasheet.pdf?_gl=1*93ouzk*_ga*MjExMzU1NzIxOC4xNjc5MzQzNzE0*_ga_T369JS7J9N*MTY5Njg3ODIxNi4zLjEuMTY5Njg4MDM1Ni4zNy4wLjAu
- 34) <https://www.techtarget.com/iotagenda/definition/RFID-radio-frequency-identification>
- 35) <https://www.linkedin.com/pulse/evolution-programming-languages-past-present-future-mohindroo->
- 36) <https://us.metoree.com/categories/rfid/>
- 37) <http://www.go-online.gr/ebusiness/specials/article.html?articleid=1591>

- 38) <http://www.go-online.gr/ebusiness/specials/article.html?articleid=1598>
- 39) <https://www.aimglobal.org/rfidfaq.html>
- 40) <https://www.slideshare.net/dimitriosnikolaidis144/rfid-79712634>
- 41) <https://gaorfid.com/el/rfid-tags-market-report/>
- 42) <https://www.marketsandmarkets.com/Market-Reports/rfid-market-446.html>
- 43) <https://www.zebra.com/us/en/about-zebra.html>
- 44) <https://neosid.de/>
- 45) https://en.wikipedia.org/wiki/Alien_Technology
- 46) <https://www.averydennison.com/en/home/company/overview.html>
- 47) <https://www.jadaktech.com/products/thingmagic-rfid/>
- 48) https://www.chainway.net/About?gad_source=1&gclid=Cj0KCQjwir2xBhC_ARIsAMTXk85ayV9sxXqiNGZ17yL7VaLGMHEpi12hZPMzv1OaI76BQW6_Sq_ZteEaAiDaEALw_wcB

Παράρτημα Α

Στο παράρτημα θα δοθεί αυτούσιος ο κώδικας που γράφτηκε για την υλοποίηση της εφαρμογής.

Έναρξη κώδικα:

```
import serial

import threading

import tkinter as tk

from tkinter import messagebox, Menu

import os

import sys

import time

import mysql.connector

# Check the password of the intro screen

def check_password():

    correct_password = 'test'

    entered_password = password_entry.get()

    if entered_password == correct_password:

        # Destroy the intro window if the correct password is entered

        intro_window.destroy()

        # Set the flag to indicate the correct password is entered

        password_correct.set(True)

    else:

        messagebox.showerror("Incorrect Password", "Please enter the correct password.")
```

```

# Create the intro window

intro_window = tk.Tk()

intro_window.title("RFID Management System - Login Screen")

intro_window.geometry("400x150")

intro_label = tk.Label(intro_window, text="Please enter the correct password to continue:")

intro_label.pack(pady=10)

password_entry = tk.Entry(intro_window, show="*", width=20)

password_entry.pack(pady=5)

submit_button = tk.Button(intro_window, text="Submit", command=check_password)

submit_button.pack(pady=10)


# Create a BooleanVar to store whether the correct password is entered

password_correct = tk.BooleanVar()


# Start the intro window main event loop

intro_window.mainloop()


# mySQL Database connection parameters

host = "localhost"

user = "kotsos"

password = "kotsospro99"

database = "rfid_management_system"

# Attempt to establish a connection to the database

connection = mysql.connector.connect(

    host=host,

```

```

    user=user,

    password=password,

    database=database
)

# Create a cursor to interact with the database

cursor = connection.cursor()


# Define the serial port and baud rate for communication with Arduino

serial_port = 'COM3'

baud_rate = 115200


# Fetch com_port and room values from the com_port_mapping database table

com_port_mapping = {}

cursor.execute("SELECT com_port, room FROM com_port_mapping")

for com_port, room in cursor.fetchall():

    com_port_mapping[com_port] = room

com_port = serial_port

room = com_port_mapping.get(com_port)


# Connect to Arduino through serial port

arduino = serial.Serial(serial_port, baud_rate)


# Event to signal when to stop reading data from Arduino

stop_event = threading.Event()


# Minimum interval (in seconds) between consecutive transfers of the same tag between
listboxes

```

```
min_transfer_interval = 3
```

```
# Dictionary to track timestamps of tag transfers
```

```
last_transfer_times = {}
```

```
# Global variables to store the last detected tag and the timer object
```

```
last_detected_tag = None
```

```
last_detection_timer = None
```

```
# Global variable to store the timer identifier
```

```
last_detection_timer_id = ""
```

```
is_timer_active = False
```

```
# Create a variable to store the last detected data
```

```
last_detected_data = ""
```

```
# Define different tag color configurations (for different RSSI levels)
```

```
listbox_tag_configurations = {
```

```
    "green": {"foreground": "black", "background": "green"},
```

```
    "yellow": {"foreground": "black", "background": "yellow"},
```

```
    "orange": {"foreground": "black", "background": "orange"},
```

```
    "red": {"foreground": "black", "background": "red"},
```

```
    "black": {"foreground": "black", "background": "white"},
```

```
}
```

```
data_thread = None # Declare data_thread as a global variable at the module level
```



```

# Start the data reading thread (button: Start)

def start_data_reading():

    # Start receiving data from the Arduino.

    start_button.config(state=tk.DISABLED)

    stop_event.clear()

    global data_thread

    data_thread = threading.Thread(target=data_reading)

    data_thread.start()


# Handle program exit and clean up resources. (button: Exit)

def on_exit():

    confirm = messagebox.askokcancel("Exit Confirmation", "Are you sure you want to exit?")

    if confirm:

        root.destroy()

        stop_event.set()

        arduino.close()

        # Wait for the data thread to finish (if running)

        if 'data_thread' in globals() and data_thread is not None and data_thread.is_alive():

            data_thread.join()

        # Close the database connection

        connection.close()

        sys.exit()


# Read data from the Arduino and update the GUI accordingly.

```

```

def data_reading():

    global    last_detected_tag,    last_detection_timer_id,    is_timer_active,    room,
    last_detected_data

    while not stop_event.is_set():

        data = arduino.readline().decode().strip()

        data = data.split(",")

        data = [d.strip() for d in data]

        # Check if the data list is empty

        if not data:

            continue # Continue to read the next data

        last_detected_data = ', '.join(data) # Save the last detected data as a string

        if len(data) >= 2:

            rssi = int(data[0])

            tag_value = data[1]

            # Update the gui with the received tags

            update_gui(tag_value, rfids_listbox3)

            # Call the update_listbox4 function to update the data reading console with the
            detected tag data

            update_listbox4()

            # Call the check_rssi function to handle RSSI-based automatic tag transfers and font
            color configuration

```

```

check_rssi(rssi, tag_value)

# Reset the timer whenever a tag is detected

if is_timer_active:

    root.after_cancel(last_detection_timer_id)

start_no_tag_timer() # Start the new timer for no tag detection


# Update the last detected tag

last_detected_tag = tag_value


# Update the listbox4 data console with detected arduino data.

def update_listbox4():

    global last_detected_data

    if last_detected_data:

        rfids_listbox4.insert(tk.END, last_detected_data)

        rfids_listbox4.see(tk.END) # Scroll to the bottom to show new data


# Update the GUI with RFID data and add it to the appropriate listbox and database table.

def update_gui(data, listbox):

    global room

    # Check if the tag is in the 'all_tags' table

    select_query = "SELECT * FROM all_tags WHERE tag = %s"

    cursor.execute(select_query, (data,))

    row = cursor.fetchone()

    if row:

```

```

# If the tag exists in the table, get its ID from the table

tag_id = row[0]


# Check if the data is already in the listbox to avoid duplicates
if not any(data in item for item in listbox.get(0, tk.END)):

    # Check if the tag exists in the 'inside_tags' database table
    inside_query = "SELECT * FROM inside_tags WHERE tag = %s"
    cursor.execute(inside_query, (data, ))
    inside_row = cursor.fetchone()

    if inside_row is not None:

        # Only insert if the tag is in the "correct" room
        if inside_row[1] == selected_room:

            listbox.insert(tk.END, f"{data} - tag{tag_id} detected. From Reader: {com_port}")
        else:

            # Tag is not inside the correct room, so check if it's in the 'outside_tags' table
            outside_query = "SELECT * FROM outside_tags WHERE tag = %s"
            cursor.execute(outside_query, (data,))
            outside_row = cursor.fetchone()

            if outside_row is None:

                # Since the tag is not inside and not already in the 'outside_tags' table, insert it to
                outside

                insert_outside_query = "INSERT INTO outside_tags (tag) VALUES (%s)"
                cursor.execute(insert_outside_query, (data,))
                connection.commit()

```

```

        listbox.insert(tk.END, f"{data} - tag{tag_id} detected. From Reader: {com_port}")

elif outside_row is not None:

    listbox.insert(tk.END, f"{data} - tag{tag_id} detected. From Reader: {com_port}")

else:

    # If the tag is new, add it to the 'all_tags' table

    insert_query = "INSERT INTO all_tags (tag) VALUES (%s)"

    values = (data,)

    cursor.execute(insert_query, values)

    connection.commit()

    # Fetch the newly inserted row to get its ID

    cursor.execute(select_query, (data,))

    row = cursor.fetchone()

    tag_id = row[0]

    # Check if the tag exists in the 'inside_tags' database table

    inside_query = "SELECT * FROM inside_tags WHERE tag = %s"

    cursor.execute(inside_query, (data,))

    inside_row = cursor.fetchone()

    if inside_row is not None:

        # Only insert if the tag is in the "correct" room

        if inside_row[1] == selected_room:

            listbox.insert(tk.END, f"{data} - tag{tag_id} detected. From Reader: {com_port}")

    else:

```

```

# Tag is not inside the "correct" room, so check if it's in the 'outside_tags' table

outside_query = "SELECT * FROM outside_tags WHERE tag = %s"

cursor.execute(outside_query, (data,))

outside_row = cursor.fetchone()


if outside_row is None:

    # Since the tag is not inside and not already in the 'outside_tags' table, insert it to
    outside

    insert_outside_query = "INSERT INTO outside_tags (tag) VALUES (%s)"

    cursor.execute(insert_outside_query, (data,))

    connection.commit()


    listbox.insert(tk.END, f"{data} - tag{tag_id} registered. From Reader: {com_port}")

elif outside_row is not None:

    listbox.insert(tk.END, f"{data} - tag{tag_id} detected. From Reader: {com_port}")


# Ensure the listbox scrolls to the last item

listbox.yview(tk.END)

# Update the listboxes to reflect the changes

update_listboxes()


# Update the GUI listboxes to always display the current RFID data.

def update_listboxes():

    global selected_room, room


# Clear the listboxes

```

```

rfids_listbox1.delete(0, tk.END)

rfids_listbox2.delete(0, tk.END)


# Fetch data from the 'all_tags' table
select_query = "SELECT * FROM all_tags"

cursor.execute(select_query)

all_tags_data = cursor.fetchall()


# Initialize counters for each room
room1_count = 0

room2_count = 0

room3_count = 0


# Update the listboxes with the fetched data
for row in all_tags_data:

    tag_id = row[0]

    tag_value = row[1]


# Check if the tag exists in the 'inside_tags' database table
inside_query = "SELECT * FROM inside_tags WHERE tag = %s"

cursor.execute(inside_query, (tag_value,))

inside_row = cursor.fetchone()

if inside_row:

    inside_tag, inside_room = inside_row[0], inside_row[1]


    if inside_room == "ROOM1":

```



```

        room1_count += 1

elif inside_room == "ROOM2":

    room2_count += 1

elif inside_room == "ROOM3":

    room3_count += 1


if inside_room == selected_room:

    # Display the tag in rfids_listbox2 if room matches selected_room

    rfids_listbox2.insert(tk.END, f"{tag_value} - tag{tag_id}")


else:

    # Check if the tag is in 'outside_tags' table

    outside_query = "SELECT * FROM outside_tags WHERE tag = %s"

    cursor.execute(outside_query, (tag_value,))

    outside_row = cursor.fetchone()


if outside_row:

    # Display the tag in the outside tags listbox

    rfids_listbox1.insert(tk.END, f"{tag_value} - tag{tag_id}")


    # Increment the counters based on the tag's room

    outside_room_query = "SELECT room FROM inside_tags WHERE tag = %s"

    cursor.execute(outside_room_query, (tag_value,))

    outside_room = cursor.fetchone()


if outside_room == "ROOM1":

```

```

        room1_count += 1

    elif outside_room == "ROOM2":

        room2_count += 1

    elif outside_room == "ROOM3":

        room3_count += 1

# Update the room counters text label

room1_tags_label.config(text=f"ROOM 1: {room1_count} tags")

room2_tags_label.config(text=f"ROOM 2: {room2_count} tags")

room3_tags_label.config(text=f"ROOM 3: {room3_count} tags")


# Check the RSSI value of a detected tag and transfer it to the other listbox if criteria is met,
with its updated font.

def check_rssi(rssi, tag_value):

    global last_transfer_times, room

    font_color = None

    if tag_value is None:

        return

    # Check if the tag_value exists in the 'outside_tags' database table

    outside_query = "SELECT * FROM outside_tags WHERE tag = %s"

    cursor.execute(outside_query, (tag_value,))

    outside_row = cursor.fetchone()

    # Get the current font color of the tag in the listbox

    tag_listbox = rfids_listbox1 if outside_row is not None else rfids_listbox2

```

```

# Fetch the tag_id from the 'all_tags' table in the database

cursor.execute("SELECT tag_id FROM all_tags WHERE tag = %s", (tag_value,))

result = cursor.fetchone()

if result:

    tag_id = result[0]

else:

    tag_id = None

# Find the index of the tag in the listbox based on the tag_value and tag_id

if tag_id is not None:

    try:

        tag_index = tag_listbox.get(0, tk.END).index(f"{tag_value} - tag{tag_id}")

    except ValueError:

        tag_index = None

    else:

        tag_index = None

current_font_color = font_color

# Only update the font color in the listbox if it's different from the current color

if font_color != current_font_color and room == selected_room and tag_index is not None:

    tag_listbox.itemconfig(tag_index, background=font_color)

if rssi >= -45:

    font_color = "green"

    current_time = time.monotonic()

```

```

transfer_time = last_transfer_times.get(tag_value, 0)

# Check if enough time has passed since the last transfer of this tag
if current_time - transfer_time >= min_transfer_interval:

    # Check if the tag exists in the 'inside_tags' database table with a specific room
    inside_query = "SELECT * FROM inside_tags WHERE tag = %s AND room = %s"
    cursor.execute(inside_query, (tag_value, room))

    inside_row = cursor.fetchone()

    if inside_row is not None:

        inside_row_room = inside_row[1]

    else:

        inside_row_room = None

# Check if the tag_value exists in the 'outside_tags' database table
outside_query = "SELECT * FROM outside_tags WHERE tag = %s"
cursor.execute(outside_query, (tag_value,))

outside_row = cursor.fetchone()

# Perform the transfer based on the presence of the tag in the database tables
if inside_row:

    # Transferring from inside to outside

    delete_query = "DELETE FROM inside_tags WHERE tag = %s"
    cursor.execute(delete_query, (tag_value,))

    connection.commit()

    insert_query = "INSERT INTO outside_tags (tag) VALUES (%s)"
    cursor.execute(insert_query, (tag_value,))

```

```

        connection.commit()

    elif outside_row:

        # Transferring from outside to inside

        delete_query = "DELETE FROM outside_tags WHERE tag = %s"

        cursor.execute(delete_query, (tag_value,))

        connection.commit()

        insert_query = "INSERT INTO inside_tags (tag, room) VALUES (%s, %s)"

        cursor.execute(insert_query, (tag_value, room))

        connection.commit()

    if inside_row_room != room and outside_row is None:

        log_transfer_info(tag_value, 10) # 10 assigned for room mismatch

    else:

        log_transfer_info(tag_value, rssi)


    # Update the last transferred tag timestamp

    last_transfer_times[tag_value] = current_time

    update_listboxes()

elif rssi >= -50:

    font_color = "yellow"

elif rssi >= -55:

    font_color = "orange"

elif rssi >= -60:

    font_color = "red"

elif rssi < -60:

    font_color = "black"

```

```

# Check if the tag_value exists in the 'outside_tags' database table

outside_query = "SELECT * FROM outside_tags WHERE tag = %s"

cursor.execute(outside_query, (tag_value,))

outside_row = cursor.fetchone()


# Check if the tag_value exists in the 'inside_tags' database table

inside_query = "SELECT * FROM inside_tags WHERE tag = %s"

cursor.execute(inside_query, (tag_value,))

inside_row = cursor.fetchone()

tag_listbox = rfids_listbox1 if outside_row is not None else rfids_listbox2


# Fetch the tag_id from the 'all_tags' table in the database

cursor.execute("SELECT tag_id FROM all_tags WHERE tag = %s", (tag_value,))

result = cursor.fetchone()

if result:

    tag_id = result[0]

else:

    tag_id = None


# Find the index of the tag in the listbox based on the tag_value and tag_id

if tag_id is not None:

    try:

        tag_index = tag_listbox.get(0, tk.END).index(f"{tag_value} - tag{tag_id}")

    except ValueError:

        tag_index = None

```

else:

 tag_index = None

 # Only update the font color in the listbox if the tag is in the correct room and is visible in the listbox

 if inside_row is not None or room == selected_room:

 font_color = listbox_tag_configurations.get(font_color)

 if font_color and tag_index is not None:

 tag_listbox.itemconfig(tag_index, **font_color)

 elif outside_row is not None:

 font_color = listbox_tag_configurations.get(font_color)

 if font_color and tag_index is not None:

 tag_listbox.itemconfig(tag_index, **font_color)

 return

def log_transfer_info(tag_value, rssi):

 current_time = time.strftime("%H:%M:%S")

 # Fetch the tag_id from the 'all_tags' table in the database

 cursor.execute("SELECT tag_id FROM all_tags WHERE tag = %s", (tag_value,))

 result = cursor.fetchone()

 if result:

 tag_id = result[0]

 else:

 tag_id = None

 if rssi < 0:

 # Rssi value provided meaning it is an automatic transfer

```

message = f"{current_time} - Transferring tag{tag_id} with RSSI: {rssi}"

rfids_listbox3.insert(tk.END, message)

rfids_listbox3.yview(tk.END) # Scroll to the bottom of the listbox to show the latest
message

elif rssi == 0:

    # Rssi = 0 meaning it was called from the manual transfer tag function

    message = f"{current_time} - tag{tag_id} has been manually transferred."

    rfids_listbox3.insert(tk.END, message)

    rfids_listbox3.yview(tk.END) # Scroll to the bottom of the listbox to show the latest
    message

elif rssi == 10:

    # Called for room mismatch

    message = f"{current_time} - tag{tag_id} is inside another room, can't perform transfer."

    rfids_listbox3.insert(tk.END, message)

    rfids_listbox3.yview(tk.END) # Scroll to the bottom of the listbox to show the latest
    message


# Timer for the no_tag_detected function

def start_no_tag_timer():

    global last_detection_timer_id, is_timer_active

    last_detection_timer_id = root.after(2000, no_tag_detected) # 2sec = 2000ms

    is_timer_active = True

# Function to be called when no tag is detected for a set time

def no_tag_detected():

    global last_detection_timer_id, last_detected_tag, is_timer_active

    current_time = time.strftime("%H:%M:%S")

    # Reset the timer identifier

```



```

last_detection_timer_id = ""

if last_detected_tag is None:

    # The timer expired, but no tag has been detected during the interval

    pass

else:

    # Change the font color of the last detected tag to black

    # Check if the tag_value exists in the 'outside_tags' database table

    outside_query = "SELECT * FROM outside_tags WHERE tag = %s"

    cursor.execute(outside_query, (last_detected_tag,))

    outside_row = cursor.fetchone()

    tag_listbox = rfids_listbox1 if outside_row is not None else rfids_listbox2

    try:

        # Fetch the tag_id from the 'all_tags' table in the database

        cursor.execute("SELECT tag_id FROM all_tags WHERE tag = %s", (last_detected_tag,))

        result = cursor.fetchone()

        if result:

            tag_id = result[0]

        else:

            tag_id = None

        if tag_id is not None:

            tag_index = tag_listbox.get(0, tk.END).index(f"{last_detected_tag} - tag{tag_id}")

            tag_listbox.itemconfig(tag_index, foreground="black", background="white")

    except ValueError:

        # Handle the case when the tag is not found in the listbox

```

```

        pass

rfids_listbox3.insert(tk.END, f"{current_time} - No tag is being detected.")

rfids_listbox3.yview(tk.END) # Scroll to the bottom of the listbox to show the latest
message

# Reset the timer status flag

is_timer_active = False


# Handle double-click event on listbox items and show RFID info in a message box.
def on_double_click(event):

    selected_listbox = None

    index = None

    # Determine the source listbox based on the selected item

    selection = rfids_listbox1.curselection()

    if selection:

        index = selection[0]

        selected_listbox = rfids_listbox1

    else:

        selection = rfids_listbox2.curselection()

        if selection:

            index = selection[0]

            selected_listbox = rfids_listbox2

    if selected_listbox is not None:

        selected_item = selected_listbox.get(index)

        rfid_value = selected_item.split(" - ")[0]

        rfid_tag_id = selected_item.split(" - ")[1]

        inside_query = "SELECT * FROM inside_tags WHERE tag = %s"

```

```

cursor.execute(inside_query, (rfid_value,))

inside_row = cursor.fetchone()


# Determine the location based on the selected listbox

if selected_listbox == rfids_listbox1:

    location = "Outside any room"

else:

    location = inside_row[1]


# Show a message box with the RFID value, tag id and location

messagebox.showinfo("Tag Info", f"The selected RFID value is: {rfid_value}\nLocation: {location}\nWith tag id: {rfid_tag_id}")


# Manual Transfer of RFID tags between 'Inside the room' and 'Outside the room' listboxes.

def transfer_tag():

    global last_transfer_times

    selected_listbox = None

    destination_listbox = None

    index = None

    # Determine the source and destination listboxes based on the selected item

    selection = rfids_listbox1.curselection()

    if selection:

        index = selection[0]

        selected_listbox = rfids_listbox1

        destination_listbox = rfids_listbox2

```

```

else:

    selection = rfids_listbox2.curselection()

    if selection:

        index = selection[0]

        selected_listbox = rfids_listbox2

        destination_listbox = rfids_listbox1

if selected_listbox is None:

    # If no tag is selected, display a message in the GUI console

    messagebox.showinfo("Warning", "Please select a tag first.")

    return

# Get the selected tag from the source listbox

selected_tag = selected_listbox.get(index)

if selected_tag not in last_transfer_times:

    last_transfer_times[selected_tag] = 0 # Assign 0 to indicate a manual transfer

# Check if the selected tag has the expected format (tag_value - tagX)

tag_parts = selected_tag.split(" - ")

if len(tag_parts) == 2:

    tag_value, tag_id = tag_parts[0], tag_parts[1]

# Update the destination listbox and remove the selected tag from the source listbox

destination_listbox.insert(tk.END, selected_tag)

selected_listbox.delete(index)

```

```

# Check if the tag_value exists in the 'inside_tags' database table

inside_query = "SELECT * FROM inside_tags WHERE tag = %s"

cursor.execute(inside_query, (tag_value,))

inside_row = cursor.fetchone()


# Check if the tag_value exists in the 'outside_tags' database table

outside_query = "SELECT * FROM outside_tags WHERE tag = %s"

cursor.execute(outside_query, (tag_value,))

outside_row = cursor.fetchone()


# Perform the transfer based on the presence of the tag in the database tables

if inside_row:

    # Delete the tag from inside_tags table

    delete_query = "DELETE FROM inside_tags WHERE tag = %s"

    cursor.execute(delete_query, (tag_value,))

    connection.commit()


    # Insert the tag into outside_tags table

    insert_query = "INSERT INTO outside_tags (tag) VALUES (%s)"

    cursor.execute(insert_query, (tag_value,))

    connection.commit()


# Log the transfer in the "Data console" listbox

log_transfer_info(tag_value, 0)

```

```

elif outside_row:

    # Delete the tag from outside_tags table

    delete_query = "DELETE FROM outside_tags WHERE tag = %s"

    cursor.execute(delete_query, (tag_value,))

    connection.commit()


    # Insert the tag into inside_tags table with the correct room

    insert_query = "INSERT INTO inside_tags (tag, room) VALUES (%s, %s)"

    cursor.execute(insert_query, (tag_value, room))

    connection.commit()


    # Log the transfer in the "Data console" listbox

    log_transfer_info(tag_value, 0)

# Update the listboxes to reflect the changes

update_listboxes()


# Function to clear the cursor selection in the listboxes

def clear_selection():

    rfids_listbox1.selection_clear(0, tk.END)

    rfids_listbox2.selection_clear(0, tk.END)

# Function to remove all saved tags from listboxes and database tables.

def reset_tags():

    # Check if there are any tags in the listboxes

    cursor.execute("SELECT COUNT(*) FROM all_tags")

    result = cursor.fetchone()

    if result[0] == 0:

```

```

    messagebox.showinfo("No RFID Tags Detected", "There is no need to reset the tags.")

    return

# Fetch the number of tags before the removal from the 'all_tags' table
cursor.execute("SELECT COUNT(*) FROM all_tags")

result = cursor.fetchone()

tags_before_removal = result[0]

# Display a confirmation dialog box to ask for user confirmation

response = messagebox.askokcancel("Confirmation Warning", "Are you sure you want to
reset all tags? This action cannot be undone.")

if response:

    # Clear all database tables

    truncate_query = "TRUNCATE TABLE all_tags"

    cursor.execute(truncate_query)

    connection.commit()

    truncate_query = "TRUNCATE TABLE inside_tags"

    cursor.execute(truncate_query)

    connection.commit()

    truncate_query = "TRUNCATE TABLE outside_tags"

    cursor.execute(truncate_query)

    connection.commit()

# Clear the listboxes

rfids_listbox1.delete(0, tk.END)

```

```

rfids_listbox2.delete(0, tk.END)

rfids_listbox3.delete(0, tk.END)

rfids_listbox4.delete(0, tk.END)


# Update the listboxes to reflect the changes (show empty listboxes)

update_listboxes()


# Fetch the number of tags after the removal from the 'all_tags' table

cursor.execute("SELECT COUNT(*) FROM all_tags")

result = cursor.fetchone()

tags_after_removal = tags_before_removal - result[0]


# Log the removal of the tags in the "Data console" listbox

current_time = time.strftime("%H:%M:%S")

rfids_listbox3.insert(tk.END, f"{current_time} - Database tables and app GUI have been
reset.")

rfids_listbox3.insert(tk.END, f"{current_time} - Number of tags removed:
{tags_after_removal}")

rfids_listbox3.yview(tk.END) # Scroll to the bottom of the listbox to show the latest
message

else:

    # If the user cancels the operation, do nothing

    pass

def on_room_selection(event):

    global selected_room

    selected_room = selected_room_var.get()


if selected_room:

```



```

    message = f"{selected_room} has been selected from the dropdown menu."

    rfids_listbox3.insert(tk.END, message)

    rfids_listbox3.yview(tk.END) # Scroll to the bottom of the listbox to show the latest
message

    update_listboxes()

# Function to handle the About menu bar option

def show_about_info():

    messagebox.showinfo("About", "RFID Management System\nVersion 1.0\n© 2023
IHU\nDeveloped by Tsagkarakis Konstantinos and Iordanidis Xristos")

# Function to set up the menu bar

def setup_menu():

    global com_port, room

    menu_bar = Menu(root)

    # File menu

    file_menu = Menu(menu_bar, tearoff=0)

    menu_bar.add_cascade(label="File", menu=file_menu)

    # View menu

    view_menu = Menu(menu_bar, tearoff=0)

    menu_bar.add_cascade(label="View", menu=view_menu)

    # Edit menu

    edit_menu = Menu(menu_bar, tearoff=0)

    menu_bar.add_cascade(label="Edit", menu=edit_menu)

```

```

# Settings menu

settings_menu = Menu(menu_bar, tearoff=0)

menu_bar.add_cascade(label="Settings", menu=settings_menu)


# Help menu

help_menu = Menu(menu_bar, tearoff=0)

help_menu.add_command(label="About", command=show_about_info)

menu_bar.add_cascade(label="Help", menu=help_menu)


# Exit menu

exit_menu = tk.Menu(menu_bar, tearoff=0)

menu_bar.add_cascade(label="Exit", menu=exit_menu)

exit_menu.add_command(label="Exit", command=on_exit)


# Configure the root window to use the menu_bar

root.config(menu=menu_bar)


# Check if the correct password was entered before proceeding to create the main GUI
if password_correct.get():

    # Setup for the main GUI window

    root = tk.Tk()

    setup_menu()


# Labels and listboxes setup for the GUI


# Outside Listbox

```

```
header_label1 = tk.Label(root, text="Outside the room", font=("Helvetica", 16, "bold"))
```

```
header_label1.grid(row=0, column=0, pady=(10, 5), sticky="w")
```

```
rfids_listbox1 = tk.Listbox(root, width=15, height=10)
```

```
rfids_listbox1.grid(row=1, column=0, padx=(10, 5), pady=5, sticky="nsew")
```

```
scrollbar1 = tk.Scrollbar(root, command=rfids_listbox1.yview)
```

```
scrollbar1.grid(row=1, column=0, sticky='nse')
```

```
# Inside Listbox
```

```
header_label2 = tk.Label(root, text="Inside the room", font=("Helvetica", 16, "bold"))
```

```
header_label2.grid(row=0, column=1, pady=(10, 5), sticky="w")
```

```
rfids_listbox2 = tk.Listbox(root, width=55, height=10)
```

```
rfids_listbox2.grid(row=1, column=1, padx=(5, 10), pady=5, sticky="nsew")
```

```
scrollbar2 = tk.Scrollbar(root, command=rfids_listbox2.yview)
```

```
scrollbar2.grid(row=1, column=1, sticky='nse')
```

```
# Create a label for viewing_room, port header
```

```
viewing_room_label = tk.Label(root, text=f"(Selected reader {com_port} for: {room})",  
font=("Helvetica", 10, "italic"))
```

```
viewing_room_label.grid(row=0, column=1, columnspan=2, padx=120)
```

```
# Console listbox
```

```
header_label3 = tk.Label(root, text="GUI Data console:", font=("Helvetica", 10, "bold"))
```

```
header_label3.grid(row=2, column=0, pady=(10, 5), sticky="w")
```

```
rfids_listbox3 = tk.Listbox(root, width=60, height=6)
```

```

rfids_listbox3.grid(row=3, column=0, padx=(5, 10), pady=5, sticky="nsew")

scrollbar3 = tk.Scrollbar(root, command=rfids_listbox3.yview)
scrollbar3.grid(row=3, column=0, sticky='nse')

# Data reading listbox

header_label4 = tk.Label(root, text="Arduino detected tags:", font=("Helvetica", 10,
"bold"))

header_label4.grid(row=2, column=1, pady=(10, 5), sticky="w")

rfids_listbox4 = tk.Listbox(root, width=55, height=6)

rfids_listbox4.grid(row=3, column=1, padx=(5, 10), pady=5, sticky="nsew")

scrollbar4 = tk.Scrollbar(root, command=rfids_listbox4.yview)

scrollbar4.grid(row=3, column=1, sticky='nse')

rfids_listbox1.configure(yscrollcommand=scrollbar1.set)
rfids_listbox2.configure(yscrollcommand=scrollbar2.set)
rfids_listbox3.configure(yscrollcommand=scrollbar3.set)
rfids_listbox4.configure(yscrollcommand=scrollbar4.set)

root.columnconfigure(0, weight=1)
root.columnconfigure(1, weight=1)

# GUI Buttons config

exit_button = tk.Button(root, text="Exit", command=on_exit, height=1, width=4, fg="red")

exit_button.grid(row=4, column=1, pady=(10, 10), padx=100)

start_button = tk.Button(root, text="Start", command=start_data_reading, height=1,
width=4, fg="green")

start_button.grid(row=4, column=0, pady=(10, 10), padx=10)

transfer_button = tk.Button(root, text="Transfer tag", command=transfer_tag, height=1,
width=10, fg="purple")

```

```

transfer_button.grid(row=3, column=2, pady=(50, 10), padx=(0, 100))

clear_selection_button = tk.Button(root, text="Clear Cursor Selection",
command=clear_selection, height=1, width=18, fg="blue")

clear_selection_button.grid(row=3, column=2, pady=(0, 80), padx=(0, 0))

reset_tags_button = tk.Button(root, text="Reset Tags", command=reset_tags, height=1,
width=10, fg="brown")

reset_tags_button.grid(row=3, column=2, pady=(50, 10), padx=(100, 0))


# Initialize the selected_room variable with the default value

selected_room = room


# Create a variable to store the selected room

selected_room_var = tk.StringVar(value=selected_room)


# Dropdown menu to select the room

room_label = tk.Label(root, text="Select Room to view:")

room_label.grid(row=0, column=2, pady=(10, 5), padx=(0, 100))


room_dropdown = tk.OptionMenu(root, selected_room_var, "ROOM1", "ROOM2",
"ROOM3", command=on_room_selection)

room_dropdown.grid(row=0, column=2, pady=(10, 5), padx=(110, 0))


# Create label widgets to display the number of tags in each room

room_counters_label = tk.Label(root, text="Room Counters", font=("Helvetica", 8, "bold"))

room_counters_label.grid(row=1, column=2, padx=10, pady=(0, 40), sticky="w")

room1_tags_label = tk.Label(root, text="ROOM 1: 0 tags")

room1_tags_label.grid(row=1, column=2, padx=10, sticky="w")

room2_tags_label = tk.Label(root, text="ROOM 2: 0 tags")

```

```

room2_tags_label.grid(row=1, column=2, padx=10, pady=(40, 0), sticky="w")

room3_tags_label = tk.Label(root, text="ROOM 3: 0 tags")

room3_tags_label.grid(row=1, column=2, padx=10, pady=(80, 0), sticky="w")


# GUI window config

root.title("RFID Management System")

root.geometry("1280x420")

root.protocol("WM_DELETE_WINDOW", on_exit)

rfids_listbox1.bind("<Double-Button-1>", on_double_click)

rfids_listbox2.bind("<Double-Button-1>", on_double_click)


# Call the update_listboxes function at the start of the program to populate the listboxes
on each launch.

update_listboxes()

# Connected to sql database

rfids_listbox3.insert(tk.END, f"Connected to mySQL database: {database}...")

# Start the main GUI event loop

root.mainloop()

```

Τέλος κώδικα

Παράρτημα Β

Στο παράρτημα Β δίνεται ο κώδικας του Arduino που παρέχεται από την εταιρία SparkFun.

Έναρξη κώδικα

```
/*
```

```
Reading multiple RFID tags, simultaneously!
```

```
By: Nathan Seidle @ SparkFun Electronics
```

```
Date: October 3rd, 2016
```

```
https://github.com/sparkfun/Simultaneous\_RFID\_Tag\_Reader
```

```
Constantly reads and outputs any tags heard
```

```
If using the Simultaneous RFID Tag Reader (SRTR) shield, make sure the serial slide  
switch is in the 'SW-UART' position
```

```
*/
```

```
#include <SoftwareSerial.h> //Used for transmitting to the device
```

```
SoftwareSerial softSerial(2, 3); //RX, TX
```

```
#include "SparkFun_UHF_RFID_Reader.h" //Library for controlling the M6E Nano module
```

```
RFID nano; //Create instance
```

```
void setup()
```

```
{
```

```
Serial.begin(115200);
```

```
while (!Serial); //Wait for the serial port to come online
```

```
if (setupNano(38400) == false) //Configure nano to run at 38400bps
```

```
{
```

```
  //Serial.println(F("Module failed to respond. Please check wiring."));
```

```
  //while (1); //Freeze!
```

```
}
```

```
nano.setRegion(REGION_EUROPE); //Set to EUROPE
```

```
nano.setReadPower(2300); //5.00 dBm. Higher values may causes USB port to brown out
```

```
//Max Read TX Power is 27.00 dBm and may cause temperature-limit throttling
```

```
//Serial.println(F("Press a key to begin scanning for tags."));
```

```

//while (!Serial.available()); //Wait for user to send a character

//Serial.read(); //Throw away the user's character

nano.startReading(); //Begin scanning for tags
}

void loop()
{
  if (nano.check() == true) //Check to see if any new data has come in from module
  {
    byte responseType = nano.parseResponse(); //Break response into tag ID, RSSI,
    frequency, and timestamp

    if (responseType == RESPONSE_IS_KEEPALIVE)
    {
      //Serial.println(F("Scanning"));
    }

    else if (responseType == RESPONSE_IS_TAGFOUND)
    {
      //If we have a full record we can pull out the fun bits

      int rssi = nano.getTagRSSI(); //Get the RSSI for this tag read

      //long freq = nano.getTagFreq(); //Get the frequency this tag was detected at

      //long timeStamp = nano.getTagTimestamp(); //Get the time this was read, (ms) since
      last keep-alive message

      byte tagEPCBytes = nano.getTagEPCBytes(); //Get the number of bytes of EPC from
      response

      //Serial.print(F(" rssi["));
      Serial.print(rssi);
      Serial.print(F(", "));

      //Serial.print(F(" freq["));
      //Serial.print(freq);
      //Serial.print(F("]"));

      //Serial.print(F(" time["));
      //Serial.print(timeStamp);
      //Serial.print(F("]"));
    }
  }
}

```



```

//Print EPC bytes, this is a subsection of bytes from the response/msg array
//Serial.print(F(" epc["));
for (byte x = 0 ; x < tagEPCBytes ; x++)
{
  if (nano.msg[31 + x] < 0x10) Serial.print(F("0")); //Pretty print
  Serial.print(nano.msg[31 + x], HEX);
  Serial.print(F(" "));
}
Serial.print(F(", "));
Serial.println();
}
else if (responseType == ERROR_CORRUPT_RESPONSE)
{
  //Serial.println("Bad CRC");
}
else
{
  //Unknown response
  //Serial.print("Unknown error");
}
}
}

//Gracefully handles a reader that is already configured and already reading continuously
//Because Stream does not have a .begin() we have to do this outside the library
boolean setupNano(long baudRate)
{
  nano.begin(softSerial); //Tell the library to communicate over software serial port
  //Test to see if we are already connected to a module
  //This would be the case if the Arduino has been reprogrammed and the module has
  stayed powered
  softSerial.begin(baudRate); //For this test, assume module is already at our desired baud
  rate

```

```

while (softSerial.isListening() == false); //Wait for port to open

//About 200ms from power on the module will send its firmware version at 115200. We
need to ignore this.

while (softSerial.available()) softSerial.read();

nano.getVersion();

if (nano.msg[0] == ERROR_WRONG_OPCODE_RESPONSE)
{
    //This happens if the baud rate is correct but the module is doing a continuous read
    nano.stopReading();

    //Serial.println(F("Module continuously reading. Asking it to stop..."));

    delay(1500);
}
else
{
    //The module did not respond so assume it's just been powered on and communicating
    at 115200bps

    softSerial.begin(115200); //Start software serial at 115200

    nano.setBaud(baudRate); //Tell the module to go to the chosen baud rate. Ignore the
    response msg

    softSerial.begin(baudRate); //Start the software serial port, this time at user's chosen
    baud rate

    delay(250);
}

//Test the connection

nano.getVersion();

if (nano.msg[0] != ALL_GOOD) return (false); //Something is not right

//The M6E has these settings no matter what

nano.setTagProtocol(); //Set protocol to GEN2

nano.setAntennaPort(); //Set TX/RX antenna ports to 1

return (true); //We are ready to rock
}

```

Τέλος κώδικα