Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Εξειδικευμένη συσκευή για την συστηματική παρακολούθηση και βελτίωση της φαρμακευτικής συμμόρφωσης

Διπλωματικη Εργασια

της

ΓΑΛΑΤΑ ΑΓΛΑΙΑΣ ΕΛΛΗΣ

Επιβλέπων: Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2018

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Εξειδικευμένη συσκευή για την συστηματική παρακολούθηση και βελτίωση της φαρμακευτικής συμμόρφωσης

Διπλωματικη Εργασια

της

ΓΑΛΑΤΑ ΑΓΛΑΙΑΣ ΕΛΛΗΣ

Επιβλέπων: Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την .....

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Παναγιώτης Τσανάκας	Δημήτριος Κουτσούρης	Ηλίας Μαγκλογιάννης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Π.Π.

Αθήνα, Ιούνιος 2018

(Υπογραφή)

.....

Γαλατα Αγλαΐα - Ελλη
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Ευχαριστίες

Πρώτο από όλους θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής εργασίας, Καθηγητή Παναγιώτη Τσανάκα για την πολύτιμη καθοδήγηση του και την εμπιστοσύνη και εκτίμηση που μου έδειξε. Στη συνέχεια θα ήθελα να ευχαριστήσω τον κύριους Δημήτρη Νικητόπουλο και Σόλων Ζάννο, οι οποίοι με τα πλούσια πνευματικά προσόντα και το ήθος τους συνέβαλαν ουσιαστικά στην ολοκλήρωση αυτής της εργασίας. Τις ευχαριστίες μου εκφράζω και στους καθηγητές κ.Κουτσούρη και κ.Μαγκλογιάννη που δέχτηκαν να είναι μέλη της τριμελής επιτροπής αξιολόγησης. Ιδιαίτερες ευχαριστίες θέλω να απευθύνω στους συμφοιτητές μου για την καθοριστική τους βοήθεια, οι οποίοι στάθηκαν σημαντικοί αρωγοί στην προσπάθειά μου και με υποστήριξαν σε κάθε φάση της πορείας μου. Τέλος, θέλω να ευχαριστήσω τη μητέρα μου καθώς και τον αδερφό μου Κωστή, που με υπομονή και κουράγιο πρόσφεραν την απαραίτητη ηθική συμπαράσταση για την ολοκλήρωση της διπλωματικής μου εργασίας.

Περίληψη

Αντικείμενο της διπλωματικής εργασίας είναι η ανάπτυξη και υλοποίηση μιας εξειδικευμένης συσκευής η οποία θα στοχεύει στη συστηματική παρακολούθηση και βελτίωση της φαρμακευτικής συμμόρφωσης των ασθενών.

Σύμφωνα με τον Παγκόσμιο Οργανισμό Υγείας μόνο το 50% των χρόνιων ασθενών ακολουθούν πιστά τη θεραπεία τους στον Δυτικό Κόσμο. Η μειωμένη τήρηση της αγωγής αυξάνει τις πιθανότητες για εξέλιξη της νόσου, αποτυχία της θεραπείας, νοσηλεία και εμφάνιση ανεπιθύμητων φαρμακευτικών ενεργειών (adverse drug events-ADEs), το σύνολο των οποίων είναι δυνητικά απειλητικό για τη ζωή του ασθενή. Επιπλέον, οι ίδιες οι ανεπιθύμητες ενέργειες και αλληλεπιδράσεις των χορηγουμένων φαρμακευτικών σκευασμάτων, δύνανται να δράσουν ως ανασταλτικός παράγοντας στη συμμόρφωση του ασθενούς στο προτεινόμενο θεραπευτικό σχήμα.

Η εργασία επικεντρώνεται κυρίως στην ανάπτυξη της συσκευής η οποία θα ειδοποιεί τον ασθενή για τη λήψη της φαρμακευτικής του αγωγής και θα παρακολουθεί τη συμμόρφωση του ως προς αυτή σε πραγματικό χρόνο. Αυτό θα επιτυγχάνεται με την εκμετάλλευση των δυνατοτήτων της συσκευής, οι οποίες θα επεκτείνονται από τις απλές λειτουργίες αφύπνισης στην ασύρματη αποστολή δεδομένων σε απομακρυσμένο server.

Η υλοποίηση της συσκευής πραγματοποιήθηκε με τον μικροελεγκτή ATmega2560, οποίος είναι ενσωματωμένος στην πλακέτα του Arduino Mega. Ο σχεδιασμός και η ανάπτυξη της συσκευής βασίστηκε στο χαμηλό κόστος και την εφαρμογή της σε μαζική παραγωγή blisters. Σχετικές έρευνες που πραγματοποιήθηκαν στο παρελθόν έρχονται αντιμέτωπες με προβλήματα κόστους και η προώθησή τους στην αγορά είναι αδύνατη. Η προτεινόμενη λύση απαρτίζεται από έξυπνη συσκευασία blister, η οποία θα διαθέτει παθητικό ηλεκτρονικό κύκλωμα τυπωμένο πάνω στο blister με αγώγιμα ή μεγάλης αντίστασης μελάνια ενός στρώματος και μικρή συσκευή ενεργοποίησης του blister και επικοινωνίας με το κεντρικό πληροφοριακό σύστημα τεχνολογίας υπολογιστικού νέφους. Ακόμα, η προτεινόμενη συσκευή ενεργοποίησης θα είναι επαναχρησιμοποιήσιμη και θα μπορεί να προσαρτάται σε όλα τα blisters.

Λέξεις Κλειδιά

Φαρμακευτική Συμμόρφωση, Arduino Mega, Blister, Serial Peripheral Interface, Universal Synchronous/Asynchronous Receiver/Transmitter, Analog to Digital Converter, Hardware/Timer Interrupts, Sleep Mode

Abstract

The subject of the diploma thesis is the development and implementation of a specialized device aiming at the systematic monitoring and improvement of the patient medical adherence

According to the World Health Organization, only 50% of chronic patients are closely following their treatment in the Western World. Reduced adherence increases the chances of disease progression, treatment failure, hospitalization, and adverse drug events (ADEs), all of which are potentially life-threatening, while their undesirable effects and interactions of the pharmaceutical formulations in the context of a vicious cycle can act as an inhibitory factor in patient compliance in the proposed regimen.

The work focuses mainly on the development of the device, which will alert the patient to receive medication and monitor his compliance in real time. This will be achieved by exploiting the capabilities of the package, which will be expanded from simple alarm functions to wireless data sending to a remote server.

The implementation of the device was performed with the micro controller ATmega2560, which is built into the Arduino Mega board. The design and development of the device was based on its low cost and its application to mass production blisters. Surveys conducted in the past are facing cost problems and their promotion on the market is impossible. The proposed solution consists of a smart blister pack containing a passive electronic circuit printed on the blister with conductive or high resistance single layer inks and a small blister activation device, which will communicate with the central cloud computing technology. In addition, the suggested activation device will be reusable and can be attached to all blisters.

Keywords

Medical Adherence, Arduino Mega, Blister, Serial Peripheral Interface, Universal Synchronous/Asynchronous Receiver/Transmitter, Analog to Digital Converter, Hardware/Timer Interrupts, Sleep Mode

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Η εξειδικευμένη συσκευή συστηματικής παρακολούθησης και βελτίωσης της φαρμακευτικής συμμόρφωσης στοχεύει στην εξάλειψη ενός μείζονος προβλήματος στον ιατρικό τομέα. Το μέγεθος του ζητήματος της μη φαρμακευτικής συμμόρφωσης αγγίζει περίπου το 50% του παγκόσμιου πληθυσμού, γεγονός που ενισχύει την αναγκαιότητα επίλυσής του. Η εξέλιξη της τεχνολογίας μας δίνει τη δυνατότητα για ασύρματη παρακολούθηση της ιατρικής μας θεραπείας καθώς επίσης και το τεχνολογικό υπόβαθρο για να συντηρηθεί αυτή η καινοτομία σε παγκόσμιο επίπεδο και να εξομαλύνει τις επιπτώσεις της μη φαρμακευτικής συμμόρφωσης.

1.1 Αντικείμενο της διπλωματικής

Η συσκευή ηλεκτρονικής παρακολούθησης των φαρμάκων θα έχει ως στόχο τον περιορισμό και κατ' επέκταση την εξάλειψη του προβλήματος της μη φαρμακευτικής συμμόρφωσης. Η πρότυπη συσκευή θα προγραμματίζεται από τον φαρμακοποιό σε μία ιστοσελίδα, ώστε να συμπεριλάβει τις προβλεπόμενες ώρες λήψης χαπιών από τους ασθενής. Οι αντίστοιχες ώρες θα αποθηκεύονται σε μία βάση δεδομένων για τη μετέπειτα επίβλεψή τους από το θεράποντα ιατρό. Τέλος, η συσκευή θα ανανεώνει τη βάση δεδομένων με τα χρονικά στίγματα κατά τα οποία ο ασθενής έλαβε κάποιο χάπι. Έτσι, ο ιατρός θα μπορεί να ενημερωθεί σε οποιαδήποτε χρονική στιγμή για τη συνέπεια του ασθενή ως προς τη συνταγογραφημένη του θεραπεία.

Η συσκευή προγραμματίστηκε με τη χρήση Arduino Mega πλακέτας σε γλώσσα c/c++ μέσω του ολοκληρωμένου περιβάλλοντος ανάπτυξης(IDE) του Arduino. Η βάση δεδομένων που χρησιμοποιήθηκε για την αποθήκευση των χρονικών σημάτων λήψης των φαρμάκων από τους ασθενείς είναι η NoSQL MongoDB Database. Ο προγραμματισμό της ιστοσελίδας και η σύνδεσή της με τη βάση δεδομένων έγινε στο server περιβάλλον ανοιχτού κώδικα Node.js.

1.1.1 Συνεισφορά

Οι αλλαγές που θα επιφέρει η λειτουργία των ηλεκτρονικών συσκευασιών φαρμάκων θα είναι άμεσες και θα αφορούν τόσο τον τομέα υγείας όσο και τον οικονομικό. Αναλυτικότερα, η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

- Γρηγορότερη ανάρρωση ασθενών
- Μείωση παρενεργειών από λανθασμένες συνταγογραφήσεις
- Ελαχιστοποίηση των άσκοπων νοσοκομειακών νοσηλειών
- Περιορισμός στη σπατάλη ιατρικών πόρων
- Αποφυγή αγοράς περιττών φαρμάκων

1.2 Οργάνωση του τόμου

Η δομή αυτής της διπλωματικής εργασίας είναι:

Κεφάλαιο 2 Στο κεφάλαιο αυτό αναπτύχθηκε αναλυτικά το εύρος του προβλήματος της μη φαρμακευτικής συμμόρφωσης, στατιστικά στοιχεία που το αποδεικνύουν, παράγοντες που το προκαλούν καθώς και μελέτες που έχουν παρουσιαστεί στο παρελθόν και προσεγγίζουν το ζήτημα.

Κεφάλαιο 3 Πρόκειται για μια περιληπτική παρουσίαση των διαδικασιών που απαιτούνται για το σχεδιασμό, τη δημιουργία και την προώθηση ενός νέου προϊόντος στην αγορά.

Κεφάλαιο 4 Στο επόμενο κεφάλαιο παρουσιάζεται η προτεινόμενη πλατφόρμα με εις βάθος ανάλυση των περιφερειακών συσκευών που την απαρτίζουν.

Κεφάλαιο 5 Αποτελεί μια επιστημονική παρουσίαση των δυνατοτήτων του ATmega2560 του Arduino που λήφθηκαν υπόψη για τη δημιουργία της συσκευής, με έμφαση στη μνήμη του και την κατανάλωση ενέργειας.

Κεφάλαιο 6 Πρόκειται για τη θεωρητική επεξήγηση των σεναρίων που υλοποιεί η πρότυπη συσκευή.

Κεφάλαιο 7 Στο κεφάλαιο αυτό παρουσιάζεται η λεπτομερής ανάπτυξη της συσκευής, με στόχο την εις βάθος κατανόηση της από τον αναγνώστη.

Κεφάλαιο 8 Στο προτελευταίο κεφάλαιο εμφανίζονται τα αποτελέσματα των δοκιμών που πραγματοποιήθηκαν και κατ΄ επέκταση ο προβλεπόμενος τρόπος απόκρισης της συσκευής σε οποιαδήποτε δεδομένα.

Κεφάλαιο 9 Τέλος, αναλύονται συνοπτικά οι μελλοντικές εργασίες που απαιτούνται για την ολοκλήρωση της συσκευής καθώς και ιδέες για επεκτάσεις της λειτουργικότητάς της.

Κεφάλαιο 2

Το Πρόβλημα της Μη-Φαρμακευτικής Συμμόρφωσης

2.1 Εισαγωγή

Η συμπεριφορά που καθορίζει τη συνέπεια του ασθενούς στη λήψη των φαρμάκων του είναι μια περίπλοκη και καθαρά ατομική διαδικασία, η οποία απαιτεί πολλαπλές και διαφορετικές μεθόδους για να βελτιωθεί. Ένας τεράστιος όγκος ερευνών έχει οδηγήσει στην ανάπτυξη θεραπειών με αποδεδειγμένη αποτελεσματικότητα και ανθεκτικότητα στον κίνδυνο. Ωστόσο, μεταξύ της θεραπείας και των αποτελεσμάτων διακρίνεται ένα ενδιάμεσο στάδιο, αυτό της φαρμακευτικής συμμόρφωσης.

Θεραπεία → *Συμμόρφωση* → *Αποτελέσματα*

Ως συμμόρφωση των ασθενών στη θεραπευτική αγωγή (patient adherence) προσδιορίζεται ο βαθμός στον οποίο η συμπεριφορά των ασθενών συμβαδίζει με τις αντίστοιχες κάθε φορά ιατρικές συστάσεις.

Σύμφωνα με τον Παγκόσμιο Οργανισμό Υγείας, η φαρμακευτική συμμόρφωση συνδέεται άρρηκτα με τη λήψη του σωστού φαρμακευτικού σκευάσματος, την κατάλληλη στιγμή, στην απαιτούμενη δόση και για το χρονικό διάστημα που συστήνει ο εκάστοτε θεράπων ιατρός.

2.2 Στατιστικά στοιχεία

Η θεραπεία των χρόνιων ασθενειών συνήθως απαιτεί τη μακροπρόθεσμη ιατρική θεραπεία. Αν και πολλές φαρμακευτικές αγωγές θεωρούνται αποτελεσματικές στην καταπολέμηση των αντίστοιχων παθήσεων, τα αποτελέσματα δεν είναι πάντα τα αναμενόμενα. Τα οφέλη της αγωγής δεν συνειδητοποιούνται εξ ολοκλήρου, καθώς σχεδόν το 50% των ασθενών δε τη λαμβάνουν ακριβώς όπως είναι συνταγογραφημένη.

Σε σχετική έκθεση που πραγματοποιήθηκε το 2003 από το Διεθνή Οργανισμό Υγείας (World Health Organization - WHO) σημειώθηκε μεταξύ άλλων ότι η υγεία των ασθενών θα βελτιωθεί σε σπουδαίο βαθμό αν γίνει πιο αποτελεσματική η παρέμβαση μας στη συνέπεια της λήψης των φαρμάκων από τους ασθενείς. Συγκεκριμένα, η πρόοδος που θα προκύψει θα είναι σπουδαιότερη από την αναβάθμιση οποιασδήποτε ιατρικής θεραπείας. Η αδυναμία προσκόλλησης στην φαρμακευτική αγωγή οδηγεί σε αυξημένα ποσοστά θνησιμότητας και εκτιμάται ότι για τα δεδομένα της Αμερικής το αντίστοιχο κόστος είναι περίπου 100 δισεκατομμύρια δολάρια το χρόνο.

Όπως προαναφέρθηκε, ποικίλες ακριβείς αναλύσεις έχουν αποδείξει ότι το ποσοστό της φαρμακευτικής συμμόρφωσης για ασθενείς με χρόνιες παθήσεις στις αναπτυγμένες χώρες υπολογίζεται περίπου στο 50%. Παράλληλα, στις αναπτυσσόμενες χώρες το ποσοστό αυτό θεωρείται αρκετά μεγαλύτερο, θεωρώντας σα δεδομένο την έλλειψη των απαιτούμενων πόρων υγείας, καθώς και τη δυσκολία παροχής ιατρικής περίθαλψης.

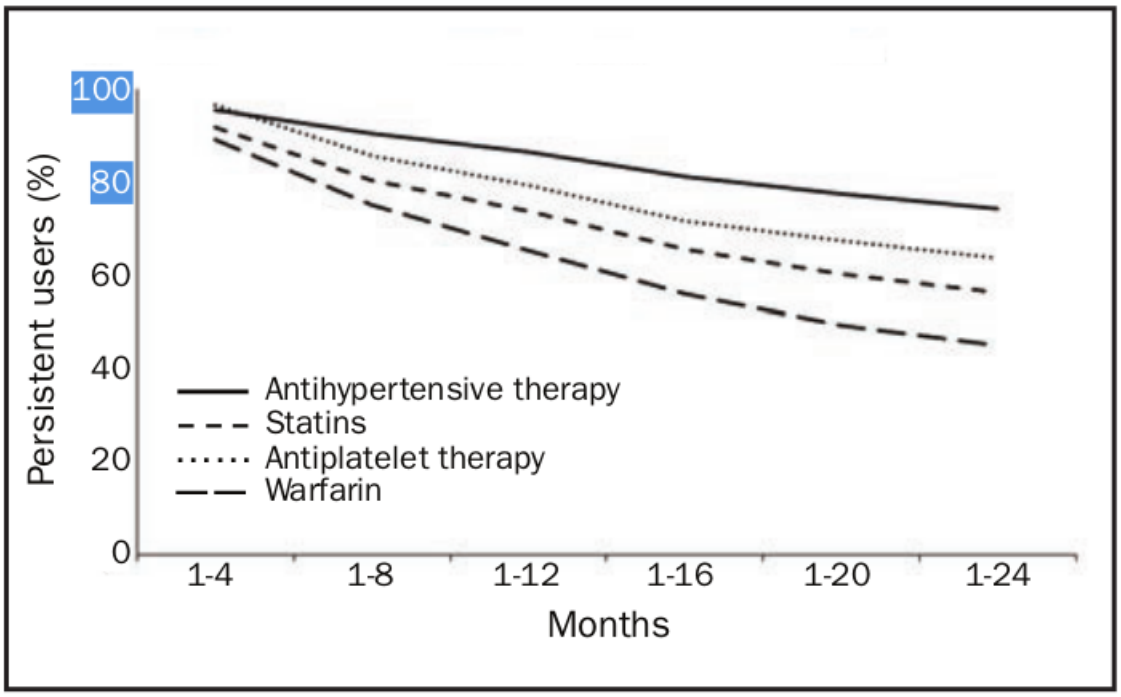
Για παράδειγμα, στην Κίνα, στην Γκάμπια και στις Σεϋχέλλες μόνο το 43%, 27% και 26% αντίστοιχα των ασθενών που πάσχουν από υπέρταση είναι συνεπείς στη λήψη των φαρμακευτικών τους συνταγών. Από την άλλη πλευρά, στις αναπτυγμένες χώρες, όπως οι ΗΠΑ, το ποσοστό αυτό υπολογίζεται στο 51%. Σε ανάλογη έρευνα, έχει προκύψει ότι οι ασθενείς με κατάθλιψη που είναι συνεπείς στις αντικαταθλιπτικές τους θεραπείες κυμαίνονται σε ποσοστό από 40% έως 70%. Ακόμα, στην Αυστραλία μόνο το 43% των ασθενών με άσθμα λαμβάνει την θεραπεία του όπως έχει συνταγογραφηθεί, ενώ μόνο το 28% λαμβάνει τα προκαθορισμένα προληπτικά φάρμακα. Τα αποτελέσματα για τον ιό HIV κυμαίνονται επίσης σε ένα ευρύ διάστημα μεταξύ 37-83%, ποσοστό που εξαρτάται από το φάρμακο του ασθενή και τα δημογραφικά χαρακτηριστικά της περιοχής του.

Παρακάτω παρατίθεται ένα απόσπασμα από την ομιλία του *Kofi Annan*, Γενικού Γραμματέα των Ηνωμένων Εθνών, με αφορμή τη δημοσίευση της έρευνας της επιτροπής για τη Μακροοικονομία και την Υγεία, στο Λονδίνο, στις 20 Δεκεμβρίου 2001.

"When we are sick, working is hard and learning is harder still. Illness blunts our creativity, cuts out opportunities. Unless the consequences of illness are prevented, or at least minimized, illness undermines people, and leads them into suffering, despair and poverty."

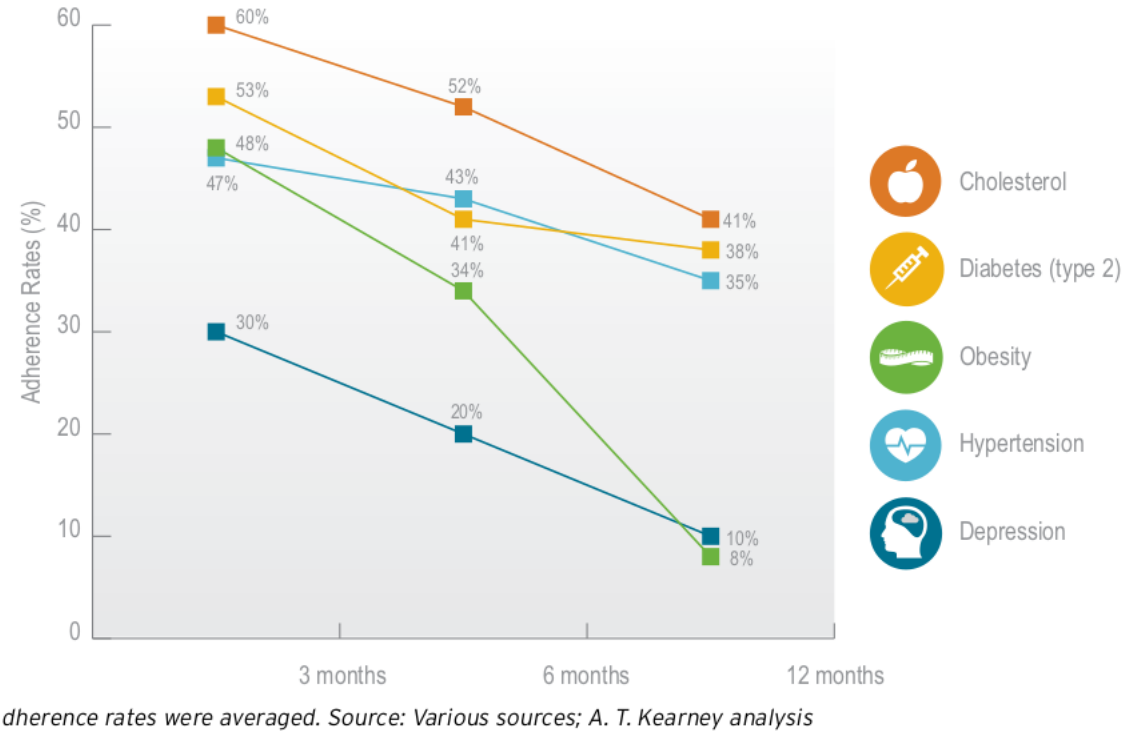
Η φαρμακευτική συμμόρφωση είναι ο κύριος παράγοντας που σχετίζεται με την αποτελεσματικότητα όλων των φαρμακευτικών θεραπειών, ιδιαίτερα στις αγωγές που συνταγογράφονται για χρόνιες παθήσεις. Σύμφωνα, με έρευνα του Διεθνούς Οργανισμού Υγείας, στις Ηνωμένες Πολιτείες Αμερικής, οι μισές περίπου νοσηλείες σε νοσοκομείο οφείλονται σε λανθασμένη λήψη φαρμακευτικής αγωγής. Σχετικά δεδομένα έχουν εξαχθεί για καρδιαγγειακές παθήσεις, καθώς για κάποιους ριψοκίνδυνους συντελεστές, η φαρμακευτική συμμόρφωση μπορεί να προσεγγιστεί με τη βοήθεια υποκατάστατων δεικτών. Για παράδειγμα, η σωστή αντι-υπερτασική θεραπεία μπορεί να προσεγγιστεί με τον έλεγχο της πίεσης του αίματος και η σωστή θεραπεία μείωσης λιπιδίων μπορεί να προσεγγιστεί με τη μέτρηση των επιπέδων λιπιδίων. Ιδιαίτερα για τους ασθενείς με καρδιαγγειακές παθήσεις είναι κρίσιμη η συνέπεια του

ασθενή, καθώς επιπτώσεις όπως μερική αναπηρία, νοσηρότητα, ακόμα και θνησιμότητα είναι συνήθεις, ωστόσο τα αποτελέσματα δεν είναι τα επιθυμητά. Όπως φαίνεται και στο σχήμα 2.1, οι ασθενείς γίνονται όλο και πιο μη συνεπείς με την πάροδο του χρόνου.



Σχήμα 2.1: Φαρμακευτική Συμμόρφωση Ασθενών με Καρδιαγγειακές παθήσεις

Οι μελέτες που έχουν διεξαχθεί αποδεικνύουν ότι η μη φαρμακευτική συμμόρφωση των ασθενών με την πάροδο του χρόνου είναι ένα γενικό φαινόμενο. Επιπλέον, ένα μεγάλο ποσοστό μεγέθους 50% – 90% σταματάει εντελώς τη συνταγογραφημένη θεραπεία του, μετά το τέλος του πρώτου χρόνο. Η εικόνα 2.2 αναπαριστά αυτά τα δεδομένα για 5 διαφορετικές χρόνιες ασθένειες και συγκεκριμένα για τη χοληστερίνη, το διαβήτη 2ου τύπου, την παχυσαρκία, την υπέρταση και την κατάθλιψη.



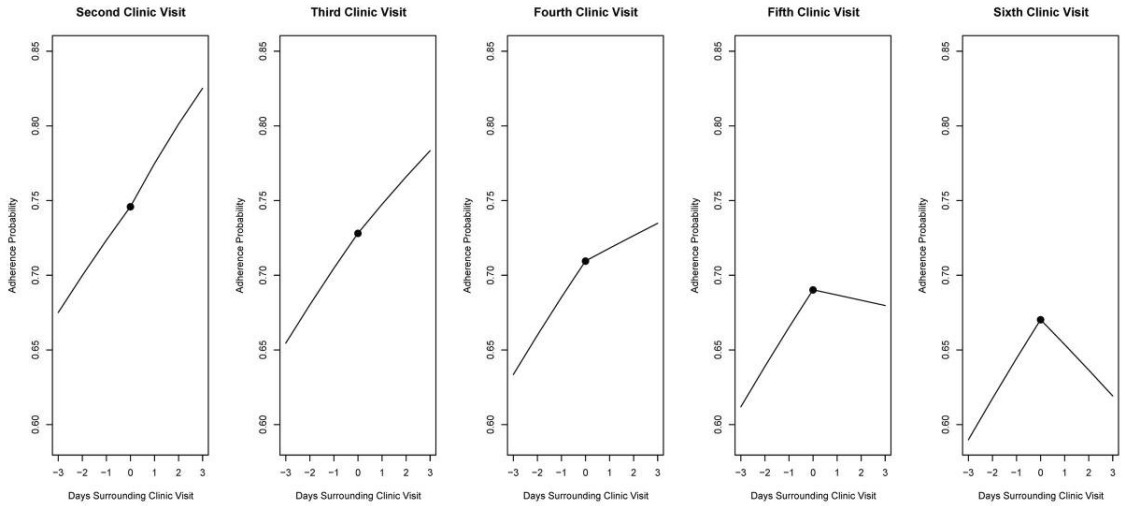
Σχήμα 2.2: Φαρμακευτική Συμμόρφωση Ασθενών με Χρόνιες παθήσεις

2.3 White-coat Adherence

Αναλυτικότερα, με τον όρο *White-coat Adherence* ορίζουμε την κατάσταση στην οποία ο ασθενής δείχνει να γίνεται πιο συνεπής στη λήψη της φαρμακευτικής του αγωγής γύρω από την περίοδο της επίσκεψης του στην κλινική – νοσοκομείο. Το γεγονός αυτό υποδηλώνει εσφαλμένα οτι ο ασθενής λαμβάνει τη συνταγογραφημένη του αγωγή με τον τρόπο που έχει προγραμματιστεί. Αυτή η συμπεριφορά ωστόσο μπορεί να επηρεάσει αρνητικά τη λήψη κάποιων αποφάσεων, τον προγραμματισμό της θεραπείας και ως εκ τούτου τα αποτελέσματα αυτής. Για παράδειγμα, η *White-coat Adherence* μπορεί τεχνητά να επηρεάσει κάποιες ουσίες του οργανισμού οδηγώντας τον ιατρό στην πεποίθηση ότι ο ασθενής λαμβάνει σωστά την αγωγή του. Έτσι, στην περίπτωση που τα συμπτώματα του ασθενή συνεχιστούν, ο ιατρός μπορεί να προχωρήσει σε περιττές αλλαγές της θεραπευτικής αγωγής, πιστεύοντας ότι η αγωγή δεν είναι κατάλληλη για τον ασθενή, με αποτέλεσμα να επηρεαστεί δυσμενώς τόσο η καταπολέμηση της νόσου όσο και οι παρενέργειες της φαρμακευτικής αγωγής. Η μελέτη της συσχέτισης του φαινόμενου αυτού με διάφορες ασθένειες θα πρέπει να ερευνηθεί περισσότερο. Σχετικά με τους ασθενείς που πάσχουν από επιληψία, η απότομη διακοπή της λήψης φαρμάκων μετά την επίσκεψη στο αντίστοιχο κέντρο υγείας, διακινδυνεύει σε μεγαλύτερο βαθμό της πιθανότητες κρίσης των ασθενών.

Συνεπώς, η καταγραφή και η ενημερότητα αυτού του φαινομένου είναι ουσιώδης στη βελτίωση της ιατρικής φροντίδας. Μια έρευνα που πραγματοποιήθηκε το 2013 και δημοσιεύτηκε

στο διαδίκτυο από την .J Pediatr αποδεικνύει την ύπαρξη του φαινομένου, συνοψίζοντας τα στοιχεία της εικόνας 2.3.



Σχήμα 2.3: White-Coat Adherence

Ο συμπαγής κύκλος αντικατοπτρίζει την πιθανότητα του ασθενή να είναι συνεπής στη φαρμακευτική του αγωγή τη μέρα της επίσκεψης του στην κλινική. Όπως φαίνεται και στην φωτογραφία 2.3, στις πρώτες επισκέψεις του στην κλινική/ νοσοκομείο ο ασθενής είναι συνεπής, ενώ με την πάροδο του χρόνου, τείνει να παίρνει τα χάπια του μόνο πριν και μετά την επίσκεψη του.

2.4 Μέθοδοι μέτρησης του βαθμού τήρησης της φαρμακευτικής αγωγής

Οι μέθοδοι για τη μέτρηση της προσκόλλησης στη φαρμακευτική αγωγή καθίστανται δύσκολοι, καθώς το αν θα τηρεί ή όχι ο ασθενής την αγωγή είναι μια καθαρά ατομική διαδικασία. Οι ασθενείς θεωρούνται ότι ανταποκρίνονται στη φαρμακευτική τους αγωγή αν το ποσοστό που ορίζεται από τον αριθμό των χαπιών που δε λήφθηκαν προς το συνολικό αριθμό των συνταγογραφημένων χαπιών, σε ένα συγκεκριμένο χρόνο ξεπερνάει το 80%.

$$\frac{Αριθμός\ χαπιών\ που\ λήφθηκαν\ σε\ χρόνο\ \mathrm{X}}{Συνολικός\ αριθμός\ συνταγογραφημένων\ χαπιών\ σε\ χρόνο\ \mathrm{X}} * 100 \geqslant 80\%$$

Αυτή η μέθοδος, ωστόσο, δεν θεωρείται αντιπροσωπευτική για την αναπαράσταση ενός μακροπρόθεσμου μοτίβου, επειδή οι ασθενείς παρουσιάζουν συνήθως το λεγόμενο *White-Coat Adherence*, δηλαδή γίνονται πιο συνεπείς 5 μέρες πριν και 5 μέρες μετά τη συνάντησή τους με τον θεράποντα ιατρό.

Η ακριβής αξιολόγηση της τήρησης της φαρμακευτικής αγωγής καθίσταται απαραίτητη για έναν αποτελεσματικό και αποδοτικό σχεδιασμό της θεραπείας, καθώς και για τη διασφάλιση

ότι όλες οι μεταβολές στην υγεία του ασθενή θα οφείλονται στη συνιστάμενη δοσολογία. Επιπλέον, ποικίλες αποφάσεις των ιατρών θα πρέπει να βασίζονται σε έγκυρα και αξιόπιστα δεδομένα ανάλογα με τη συνέπεια του ασθενή στη λήψη της φαρμακευτικής του αγωγής. Αναλυτικά, τέτοιες αποφάσεις συνιστούν οι αλλαγές στη σύσταση των συνταγών, η διαφοροποίηση των φαρμάκων και το είδος της επικοινωνίας που στοχεύει στη συνέπεια του ασθενή. Αναμφισβήτητα, δεν υπάρχει κάποιος "χρυσός κανόνας", ο οποίος καταγράφει τη συμπεριφορά του ασθενή σχετικά με τη λήψη των συνταγογραφημένων του φαρμάκων. Για το λόγο αυτό, έχουν εφευρεθεί και αναλυθεί διάφορες στρατηγικές που στοχεύουν σε μια ακριβή μέτρηση αυτής της συμπεριφοράς.

Οι προσεγγίσεις μέτρησης της συμμόρφωσης στη φαρμακοθεραπεία διακρίνονται σε τρεις κατηγορίες:

- Υποκειμενικές Μέθοδοι
- Αντικειμενικές Μέθοδοι
- Βιοχημικές Μέθοδοι

2.4.1 Υποκειμενικές Μέθοδοι

Η πρώτη κατηγορία περιλαμβάνει ερωταπαντήσεις γιατρών και ασθενών για τις υποκειμενικές τους απόψεις στη συμμόρφωση τους στη λήψη της αγωγής. Η διαδικασία αυτή συχνά διευρύνεται σε μέλη της οικογένειας του ασθενή ή στους προσωπικούς του φροντιστές. Ωστόσο, προέκυψε ότι οι ασθενείς υπερτιμούσαν τη συνέπεια τους, με αποτέλεσμα η πλειοψηφία των ασθενών να ισχυρίζεται εσφαλμένα ότι ακολουθεί πιστά τις ιατρικές συμβουλές.

Άλλες υποκειμενικές μέθοδοι περιλαμβάνουν τη συμπλήρωση ερωτηματολογίων. Έτσι προσπαθούν να κατηγοριοποιήσουν τους ασθενείς βάσει των χαρακτηριστικών της προσωπικότητας τους. Ωστόσο, έχει αποδειχθεί ότι τέτοιες στρατηγικές είναι αναξιόπιστες. Μόνο σε περιπτώσεις συγκεκριμένων ιατρικών συστάσεων οι υποκειμενικές μέθοδοι έχουν καταλήξει σε αξιόπιστα δεδομένα. Για παράδειγμα, σε περίπτωση διαχείρισης της παχυσαρκίας, ερωτηματολόγια που αφορούν τη συχνότητα λήψης φαγητού, έχουν αποφέρει έγκυρα αποτελέσματα.

2.4.2 Αντικειμενικές Μέθοδοι

Αν και οι αντικειμενικές μέθοδοι εμφανίζονταν αρχικά πιο βελτιωμένες από τις υποκειμενικές, προέκυψε ότι η κάθε μια έχει τα ελαττώματα της. Οι υπολειπόμενες δόσεις της φαρμακευτικής αγωγής μπορούν να μετρούνται στην κλινική. Ωστόσο, οι ανακρίβειες στις μετρήσεις αποτελούν ένα συχνό φαινόμενο και συνήθως καταλήγουν σε μία υπερτίμηση της φαρμακευτικής συμπεριφοράς του ασθενούς, ενώ παράλληλα πολλές σημαντικές πληροφορίες δεν καταγράφονται, όπως το χρονοδιάγραμμα της λήψης των φαρμάκων και μοτίβα που να αποτυπώνουν τις μη λαμβάνουσας δόσεις. Πρόσφατες καινοτομίες αποτελούν οι συσκευές ηλεκτρονικής παρακολούθησης της φαρμακευτικής συμμόρφωσης των ασθενών, οι οποίες καταγράφουν την ώρα και τη μέρα που μια συσκευή φαρμάκων ανοίχτηκε και κατ' επέκταση

περιγράφουν καλύτερα τη συχνότητα με την οποία οι ασθενείς λαμβάνουν την αγωγή τους. Δυστυχώς, το κόστος αυτών των συσκευών αποκλείει την ευρεία χρήση τους. Παρ' όλα αυτά, οι ληφθείς πληροφορίες θα μπορούσαν να συσσωρευτούν στις βάσεις δεδομένων των φαρμακείων και να καταλήξουν σε ακριβή και αξιόπιστα αποτελέσματα.

Ο ασθενής, ωστόσο, δε θα μπορούσε να χαρακτηριστεί ως συνεπής ή όχι. Στην πραγματικότητα τέτοια αξιολόγηση δε θα μπορούσε να υπάρξει καθώς το φαινόμενο αυτό είναι συνεχές, εννοώντας ότι η συνάρτηση μεταξύ της δόσης φαρμάκου και της επιτυχημένης ή όχι λήψης του είναι συνεχής. Αν και μια τέτοια συνάρτηση θα ήταν δύσκολο να κατασκευαστεί δεδομένου του πραγματικού χρόνου λήψης της αγωγής, η σπουδαιότητά της καθίσταται αναγκαία για τον καθορισμό της κατάλληλης θεραπείας.

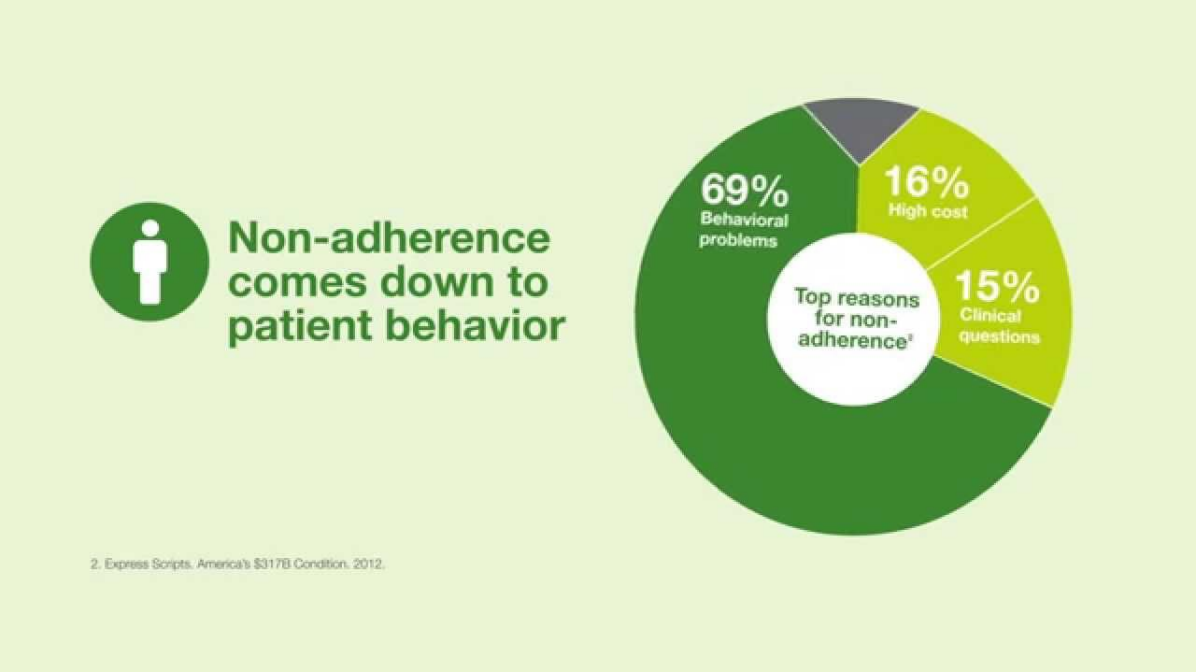
2.4.3 Βιοχημικές Μέθοδοι

Η μέτρηση αυτή αποτελεί μια τρίτη προσέγγιση για την αξιολόγηση της συμπεριφοράς του ασθενή στη τήρηση της αγωγής του. Κάποιοι μη τοξικοί βιολογικοί δείκτες μπορούν να προστεθούν στα φάρμακα και στη συνέχεια, η παρουσία τους στο αίμα ή στα ούρα μπορεί να αποδείξει ότι ο ασθενής έλαβε πρόσφατα μια δόση του εξεταζόμενου φαρμάκου. Αυτή η στρατηγική αξιολόγησης, όπως και οι προηγούμενες διαθέτει κάποια σχετικά μειονεκτήματα, καθώς τα ευρήματα μπορεί να είναι παραπλανητικά και να επηρεάζονται από διάφορους μεμονωμένους παράγοντες, συμπεριλαμβανομένης της διατροφής του ασθενή, της απορρόφησης του και του ρυθμού απέκκρισης.

2.5 Παράγοντες που οδηγούν στη μη φαρμακευτική συμμόρφωση

Οι παράγοντες που επηρεάζουν τους ασθενείς και συντελούν στη μη λήψη των φαρμάκων τους είναι ποικίλοι. Πολλές έρευνες αναλύουν ότι υπάρχει μια αμφίδρομη εξάρτηση μεταξύ των χρόνιων ασθενειών και των οικονομικών προβλημάτων που δημιουργούνται. Πολλοί άνθρωποι με οικονομικές δυσκολίες, ασχέτως την περιοχή που ζούνε και την εκάστοτε κουλτούρα βιώνουν έναν ατέρμον κύκλο. Το να είναι κανείς υγιής απαιτεί χρήματα για φαγητό, υγειονομική και ιατρική περίθαλψη, αλλά για να κερδίσει κανείς λεφτά πρέπει να είναι υγιής. Η μη τήρηση της φαρμακοθεραπείας συχνά δημιουργεί προβλήματα στους φτωχούς πληθυσμούς και έχει ως αποτέλεσμα τη σπατάλη και την υποαπορρόφηση των ήδη περιορισμένων ιατρικών πόρων. Επιπλέον, πολλοί ιατροί ισχυρίζονται ότι οι περίπλοκες φαρμακευτικές συνταγές καθιστούν αδύνατο σε αυτούς να εξάγουν έγκυρα συμπεράσματα για τη λήψη ή όχι των φαρμάκων από τους ασθενείς. Στην εικόνα 2.4 παρουσιάζονται οι κύριοι παράγοντες μη φαρμακευτικής συμμόρφωσης.

Μια λεπτομερής κατηγοριοποίηση πραγματοποιήθηκε από το Διεθνή Οργανισμό Υγείας. Αντιλαμβανόμενοι το εύρος του προβλήματος, προσπάθησαν να εστιάσουν στους παράγοντες που οδηγούν στη μη φαρμακευτική συμμόρφωση και κατέληξαν στις παρακάτω πέντε διακριτές κατηγορίες:



Σχήμα 2.4: Παράγοντες μη Φαρμακευτικής Συμμόρφωσης

- Κοινωνικοοικονομικοί παράγοντες
- Παράγοντες Υγείας
- Πάθηση του ασθενή
- Θεραπεία της ασθένειας
- Παράγοντες σχετικοί με τον ασθενή

Μια πιο γενική κατηγοριοποίηση, ωστόσο, είναι η εξής:

- παράγοντες σχετικοί με τον ασθενή
- παράγοντες σχετικοί με το γιατρό
- παράγοντες σχετικοί με το σύστημα υγείας

2.5.1 Παράγοντες Σχετικοί με τον Ασθενή

Οι πιο συχνοί λόγοι περιλαμβάνουν τη μη εμπλοκή του ασθενή στη διαδικασία αποφάσεων της θεραπείας του και την υποβέλτιστη υγειονομική παιδεία. Στις ΗΠΑ σημειώνεται ότι περίπου 90 εκατομμύρια ενήλικες δεν διαθέτουν κατάλληλη υγειονομική πεδία, γεγονός που διακινδυνεύει την υγεία τους. Ακόμα, η άποψη του ασθενή σχετικά με την υγεία του, οι προηγούμενες εμπειρίες του με αντίστοιχες φαρμακοθεραπείες και η έλλειψη κινήτρου αποτελούν σημαντικούς παράγοντες για τη συμμόρφωση του ασθενή στην αγωγή του. Πολλές

φορές, επίσης, οι ασθενείς έρχονται αντιμέτωποι με οικονομικές δυσκολίες και δυσκολεύονται να ανταπεξέλθουν στα κόστη των φαρμάκων τους. Επιπρόσθετα, παράγοντες αργοπορίας στη λήψη φαρμάκων καθίστανται η έλλειψη κατάλληλων μεταφορικών μέσων και η δυσκολία πρόσβασης στα κατάλληλα φαρμακεία της εκάστοτε περιοχής. Σύμφωνα με στοιχεία ασθενών με καρδιαγγειακές παθήσεις, ένας ακόμη παράγοντας που οδηγεί στη μη τήρηση της φαρμακευτικής αγωγής από τους ασθενείς είναι η έλλειψη κάποιου οικογενειακού προσώπου ή της απαραίτητης ψυχολογικής υποστήριξης. Συγκεκριμένα, ασθενείς με κατάθλιψη παρουσιάζουν τα πιο χαμηλά ποσοστά φαρμακευτικής συμμόρφωσης.

2.5.2 Παράγοντες Σχετικοί με το Γιατρό

Όχι μόνο οι γιατροί συχνά αποτυγχάνουν να αναγνωρίσουν την έλλειψη συνέπειας των ασθενών στη λήψη των φαρμάκων τους, αλλά συχνά συμβάλλουν σε αυτό με τη συνταγογράφηση σύνθετων φαρμακευτικών αγωγών, αποτυγχάνοντας να εξηγήσουν αποτελεσματικά τα οφέλη και τις ανεπιθύμητες ενέργειες ενός φαρμάκου και μη λαμβάνοντας υπόψη το οικονομικό βάρος για τον ασθενή. Η ελλιπής επικοινωνία μεταξύ του ιατρού πρωτοβάθμιας φροντίδας και του ασθενούς με χρόνιες ασθένειες, όπως η CVD υπονομεύει περαιτέρω τη σε βάθος κατανόηση της νόσου από τον ασθενή και τη σημασία της προσκόλλησης στην αγωγή. Μια άλλη πηγή αναποτελεσματικής επικοινωνίας παρατηρείται στην αδυναμία του ιατρού να αποσπάσει το ιστορικό του ασθενή για τις εναλλακτικές και συμπληρωματικές θεραπείες που έχει ακολουθήσει στο παρελθόν. Ακόμα, η επικοινωνία μεταξύ των ιατρών είναι συχνά ανεπαρκής και μπορεί να συμβάλλει στη μη φαρμακευτική συμμόρφωση του ασθενή. Η άμεση επικοινωνία μεταξύ των νοσηλευτών και των γιατρών της πρωτοβάθμιας φροντίδας εμφανίζεται σε λιγότερο από το 20% των νοσηλειών. Ακόμα και η σύνοψη της ιατρικής γνωμάτευσης είναι διαθέσιμη μόνο στο 34% των πρώτων επισκέψεων στα νοσοκομεία. Τέλος, η ανεπαρκής επικοινωνία μεταξύ ιατρών και νοσοκομειακών συμβούλων συντελεί σε εσφαλμένη φαρμακευτική αγωγή και κατ' επέκταση αναπόφευκτη νοσοκομειακή επανεισδοχή.

2.5.3 Παράγοντες Σχετικοί με το Σύστημα Υγείας

Τα περισσότερα συστήματα υγειονομικής περίθαλψης υπολειτουργούν, γεγονός που δημιουργεί εμπόδια στο συντονισμό των ασθενών και την πρόσβασή τους στην ιατρικά κέντρα. Ακόμα το απαγορευτικό κόστος αρκετών φαρμάκων, όπως και των ασφαλιστικών φορέων υγείας έχει ως αποτέλεσμα την αδυναμία των ασθενών να αγοράσουν τα φάρμακα τους. Τέλος, οι κλινικοί ιατροί δεν έχουν τον κατάλληλο χρόνο για να αναλύσουν στους ασθενείς τα συμπτώματα της μη λήψης των φαρμάκων τους ή την παράλειψη κάποιων δόσεων, γεγονός που προκαλείται από το μειωμένο προσωπικό και το μεγάλο φόρτο εργασίας.

Συνοψίζοντας, λοιπόν, οι παράγοντες που συμβάλλουν στην ασυνέπεια της λήψης φαρμάκων είναι ποικίλοι και περιλαμβάνουν εκείνους που σχετίζονται με τους ασθενείς (π.χ. υποβέλτιστη υγειονομική παιδεία και έλλειψη συμμετοχής στη διαδικασία λήψης αποφάσεων), εκείνες που σχετίζονται με τους γιατρούς (π.χ. συνταγογράφηση σύνθετων φαρμακευτικών αγωγών, τα εμπόδια επικοινωνίας, η αναποτελεσματική διαβίβαση των πληροφοριών σχετικά

με τις ανάστροφες συνέπιες και η περίθαλψη από πολλούς διαφορετικούς γιατρούς) και εκείνες που σχετίζονται με συστήματα υγειονομικής περίθαλψης (π.χ. περιορισμός χρόνου επίσκεψης γραφείου, περιορισμένη πρόσβαση σε κέντρα υγείας και έλλειψη τεχνολογίας ιατρικής πληροφόρησης).

2.6 Βιβλιογραφικές Μελέτες

Έχουν γίνει σημαντικές προσπάθειες από την παγκόσμια ερευνητική κοινότητα προς την κατεύθυνση εύρεσης αντικειμενικών μεθόδων μέτρησης της φαρμακευτικής συμμόρφωσης και αύξησης του βαθμού της. Οι σημαντικότερες από αυτές είναι:

- Μέθοδοι Αξιολόγησης της φαρμακευτικής συμμόρφωσης των ασθενών μέσω αυτοματοποιημένων βάσεων δεδομένων
- Χρήση συσκευών ενσωματωμένης ηλεκτρονικής συσκευασίας φαρμάκων
- Έξυπνη συσκευασία φαρμάκων με RFID τεχνολογία
- Ηλεκτρονική συσκευασία φαρμάκων 5 στρωμάτων

2.6.1 Μέθοδοι Αξιολόγησης της φαρμακευτικής συμμόρφωσης των ασθενών μέσω αυτοματοποιημένων βάσεων δεδομένων

Αυτή η μέθοδος αξιολόγησης δημοσιεύθηκε το 2006 με τίτλο: *"Methods for evaluation of medication adherence and persistence using automated databases"* από τους: *Susan E. Andrade ScD, Kristijan H. Kahler Msc, Feride Frech MPH* και *K. Arnold Chan MD, ScD.*

Σκοπός της έρευνας ήταν η πραγματοποίηση μιας συστηματικής μελέτης όλων των μεθόδων που χρησιμοποιούνταν για την αξιολόγηση της τήρησης της φαρμακευτικής αγωγής από τους ασθενείς με τη χρήση αυτοματοποιημένων βάσεων δεδομένων. Τέτοιες μελέτες είχαν διεξαχθεί στο παρελθόν στα πλαίσια φαρμακευτικών και οικονομικών μελετών. Αυτοματοποιημένα, λοιπόν, πραγματοποιήθηκε μια προσπάθεια αναγνώρισης όλων των σχετικών άρθρων από τον Ιανουάριο του 1980 έως τις 31 Μαρτίου το 2004. Όλες οι σχετικές μελέτες που αφορούσαν τη μη συνέπεια των ασθενών στη συνταγογραφημένη τους αγωγή, τη μη ισορροπημένη λήψη φαρμάκων ή την απότομη διακοπή της θεραπείας συγκεντρώθηκαν και αποθηκεύτηκαν σε βάσεις δεδομένων. Διάφοροι αλγόριθμοι εφαρμόστηκαν στα δεδομένα που λήφθηκαν, ωστόσο τα αποτελέσματα δεν ήταν επιθυμητά. Συγκεκριμένα, πολλές έρευνες θεωρούσαν εσφαλμένα ότι από τη στιγμή της αγοράς του φαρμάκου, αυτό πρόκειται να καταναλωθεί.

2.6.2 Χρήση συσκευών ενσωματωμένης ηλεκτρονικής συσκευασίας φαρμάκων

Η δημιουργία τέτοιων συσκευών στοχεύει κυρίως στην αγορά τους από κλινικά/ ερευνητικά κέντρα για την διεξαγωγή δοκιμών και τη συλλογή ερευνητικών δεδομένων.

Η βασική αρχή λειτουργίας είναι ότι παρακολουθείται το πότε αφαιρέθηκε ένα χάπι από την ηλεκτρονική συσκευασία (που έχει τοποθετηθεί εξαρχής) και αποστέλλεται σε ένα πληροφοριακό σύστημα. Ορισμένες λύσεις προσφέρουν και λειτουργίες ειδοποίησης του ασθενούς την ώρα που πρέπει να λάβει μια φαρμακευτική δόση. Αυτές οι λύσεις προσφέρουν αυξημένα επίπεδα ελέγχου της συμμόρφωσης του ασθενή αλλά και άμεση ειδοποίηση σε περίπτωση μη λήψης κάποιας δόσης. Η ιδιαιτερότητα του είναι το ενσωματωμένο ηλεκτρονικό σύστημα στη συσκευή, το οποίο επιτρέπει την αποθήκευση μεγάλου όγκου δεδομένων. Στη συνέχεια, τα δεδομένα αυτά μπορούν να εισαχθούν είτε σε κάποιον ηλεκτρονικό υπολογιστή είτε σε ένα πιο περίπλοκο λογισμικό, το οποίο θα πραγματοποιήσει κάποια στατιστική έρευνα. Οι πληροφορίες καταγράφονται και μεταφέρονται από την ηλεκτρονική συσκευασία στο ενσωματωμένο σύστημα με τη βοήθεια αγώγιμων μελανιών.

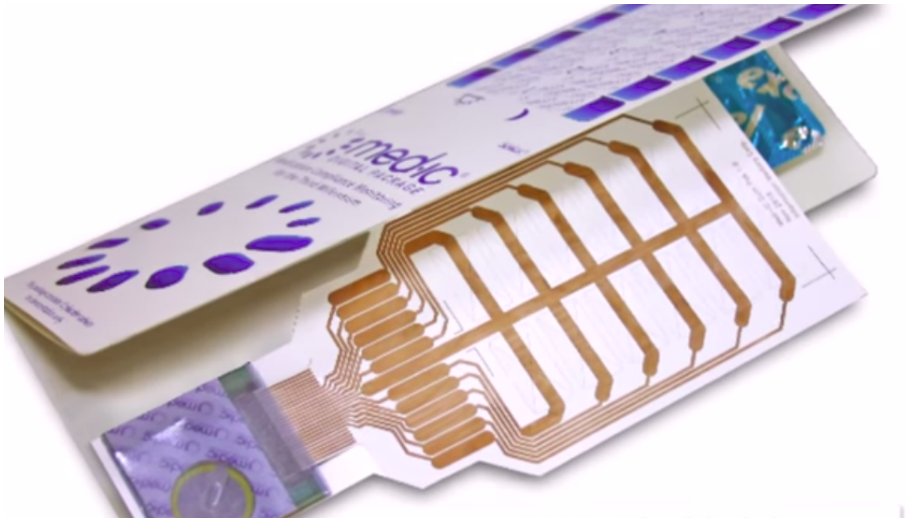
Σημαντικό μειονέκτημα αυτής της τεχνολογίας είναι ο όγκος της συσκευής (δοχείο φαρμάκων) και το αυξημένο κόστος της. Για το λόγο αυτό, η συσκευή αυτή δεν ενδείκνυται για μαζική παραγωγή.

2.6.3 Έξυπνη συσκευασία φαρμάκων με RFID τεχνολογία

Πρόκειται για μια μέθοδο έξυπνης συσκευασίας, όπου η βασική αρχή λειτουργίας της περιλαμβάνει αναγνώριση ραδιοσυχνοτήτων (RFID) για την παρακολούθηση των φαρμάκων. Συγκεκριμένα, χρησιμοποιούν ένα μοναδικό αναγνωριστικό σε κάθε συσκευασία, σαν ηλεκτρονικό barcode.

Μία γενικότερη εφαρμογή αυτής της συσκευασίας σχετίζεται με τη γνησιότητα των φαρμάκων. Καθώς το φάρμακο εισέρχεται και φεύγει από κάθε στάδιο της φαρμακευτικής αλυσίδας εφοδιασμού, περνά μέσα από μια ειδικά σχεδιασμένη πύλη που μπορεί να διαβάσει τις πληροφορίες στο τσιπ RFID για να επιβεβαιώσει ότι το φάρμακο είναι γνήσιο. Με αυτό τον τρόπο, τα chip RFID μπορούν να αποτρέψουν την είσοδο αναλώσιμων φαρμάκων στην αγορά και κατ' επέκταση την αποφυγή βλαβών και την προστασία της φήμης των κατασκευαστών.

Με τη χρήση τσιπ πυριτίου, αυτού του είδους οι εφαρμογές είναι εφικτές και σε επίπεδα κιβωτίων φαρμάκων. Ωστόσο, τα ευέλικτα ηλεκτρονικά συστήματα, με τις δυνατότητες τους για χαμηλότερο κόστος και ευκολότερη ενσωμάτωση, καθιστούν δυνατή την παρακολούθηση της φαρμακευτικής αγωγής σε μεμονωμένα πακέτα ή φιάλες. Επιπλέον, τα ευέλικτα ηλεκτρονικά συστήματα έχουν τη δυνατότητα να ξεπερνούν τις απλές εφαρμογές παρακολούθησης και αντικατάστασης φαρμάκων. Για παράδειγμα, ενσωματώνοντας μια ποικιλία αισθητήρων παράλληλα με τη λειτουργικότητα RFID, τα ευέλικτα ηλεκτρονικά συστήματα αποκτούν τη δυνατότητα έγκαιρης καταγραφής των περιβαλλοντικών συνθηκών στις οποίες έχει εκτεθεί το φάρμακο καθ 'όλη τη διάρκεια ζωής του. Αυτή είναι μια εξαιρετικά χρήσιμη ικανότητα, καθώς παράγοντες όπως η θερμότητα και η υγρασία μπορούν να επηρεάσουν σημαντικά την αποτελεσματικότητα πολλών φαρμάκων. Για παράδειγμα, ορισμένα εμβόλια και φάρμακα για τη ρευματοειδή αρθρίτιδα πρέπει να φυλάσσονται σε ψυγείο. Υπάρχουν περιπτώσεις στις οποίες η έκθεση σε θερμοκρασίες δωματίου για μια ημέρα ή δύο μπορεί να καταστήσει ένα φάρμακο άχρηστο ή ακόμη και δυνητικά επιβλαβές. Στη συγκεκριμένη συσκευή λοιπόν, πριν



Σχήμα 2.5: Έξυπνη συσκευασία φαρμάκων με RFID τεχνολογία

χρησιμοποιήσει ο ασθενής το φάρμακο, ο υπεύθυνος σαρώνει τη συσκευή με μια τυπική συσκευή ανάγνωσης RFID και μπορεί να παρακολουθήσει το πλήρες προφίλ του ιστορικού του φαρμάκου. Αυτό μπορεί να περιλαμβάνει έναν κωδικό αναγνώρισης προϊόντος για να επιβεβαιώσει ότι το φάρμακο είναι γνήσιο, συν μια πλήρη καταγραφή των συνθηκών αποθήκευσης. Ο χρήστης μπορεί αμέσως να δει αν η θερμοκρασία περιβάλλοντος έχει υπερβεί την επιτρεπτή μέγιστη τιμή για αυτό το φάρμακο και, αν ναι, για πόσο καιρό. Κάθε φάρμακο που έχει αποθηκευτεί εσφαλμένα μπορεί να απορριφθεί, βοηθώντας στην αποτελεσματική θεραπεία και την αποφυγή βλάβης στον ασθενή.

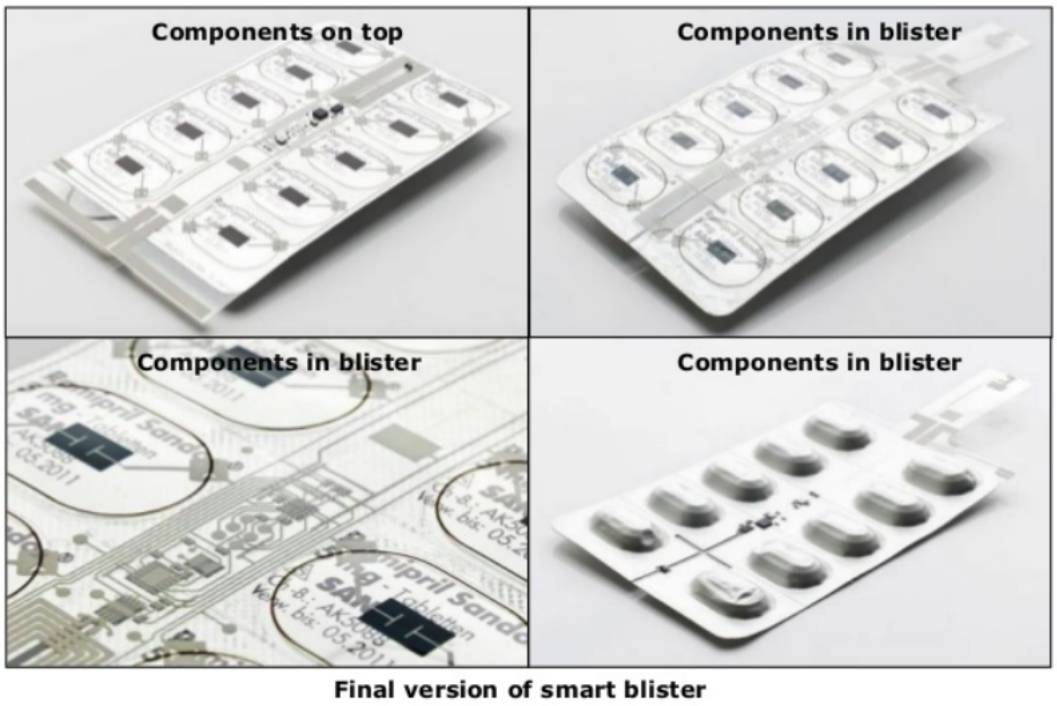
Σχετικά με τη δυνατότητα ανίχνευσης της λήψης χαπιών, η συσκευή αυτή αξιοποιώντας τεχνολογίες RFID και ηλεκτρονικών κυκλωμάτων ενσωματωμένων σε blister μπορεί να αναγνωρίσει τον χρόνο αφαίρεσης του κάθε χαπιού. Αναλυτικότερα, χρησιμοποιεί μία συσκευή που αποτελείται από μια Κεντρική Μονάδα Επεξεργασίας (CPU) και έχει προσαρτημένο ένα πλέγμα αισθητήρων, ενσωματωμένο στη συσκευή φυσαλίδων φαρμάκων (blister). Η CPU καταγράφει την ώρα που κάθε δισκίο ή μια κάψουλα εκδιώκεται από την κυψέλη της και αποθηκεύει τα δεδομένα για μεταγενέστερη απεικόνιση ή / και ανάλυση. Κατά τη στιγμή της επαναπλήρωσης της συνταγής ή κατά την επίσκεψη παρακολούθησης της κλινικής μελέτης, τα δεδομένα του ασθενούς σχετικά με τη συχνότητα και τη λήψη των φαρμάκων μεταφορτώνονται σε έναν υπολογιστή μέσω ενός αναγνώστη RFID. Αυτό πραγματοποιείται μέσω μιας συσκευής ανάγνωσης (εικόνα 2.5) για να αποστείλει τα δεδομένα σε πληροφοριακό σύστημα διαχείρισης.

Αυτή η μέθοδος προσφέρει πολλαπλά πλεονεκτήματα στον έλεγχο των φαρμάκων και ταυτόχρονα λύνει το πρόβλημα του αυξημένου όγκου της συσκευής, όμως παραμένει υψηλού κόστους για ατομική χρήση και έχει το μειονέκτημα της ετεροχρονισμένης μετάδοσης της πληροφορίας.

2.6.4 Ηλεκτρονική συσκευασία φαρμάκων 5 στρωμάτων

Στην ίδια κατηγορία με την παραπάνω βρίσκεται και η πιο προηγμένη μέθοδος ηλεκτρονικού κυκλώματος πέντε στρωμάτων. Ενσωματώνει στο blister τυπωμένα ηλεκτρονικά κυκλωμάτα πέντε στρωμάτων και ενεργά ηλεκτρονικά στοιχεία. Τα στρώματα αυτά είναι:

- Κύκλωμα, το οποίο θα έχει καλή αγωγιμότητα
- Κεραία με *αντίσταση* $\leqslant 40\ \Omega$
- Γέφυρες που θα εξυπηρετούν τις ηλεκτρονικές επαφές μεταξύ των στοιχείων
- Τυπωμένες αντιστάσεις με απόκλιση $\geqslant 5\%$ που θα ανιχνεύουν ποιο χάπι εξήχθη.



Σχήμα 2.6: Ηλεκτρονική συσκευασία φαρμάκων 5 στρωμάτων

Η εν λόγω συσκευασία παρουσιάζεται στο σχήμα 2.6. Ενσωματωμένο στο blister υπάρχει ένα chip RDIF για την επικοινωνία και την αποθήκευση δεδομένων. Η κεραία χρησιμοποιείται προφανώς για τη μεταφορά των δεδομένων. Χρησιμοποιείται επικοινωνία κοντινού πεδίου (near field communication, NFC), η οποία είναι μια μικρής εμβέλειας ασύρματη τεχνολογία, που λειτουργεί σε συχνότητα 13,56MHz και μεταφέρει δεδομένα με ρυθμό έως και 424kbps. Η λειτουργία της βασίζεται στην επαφή ή την προσέγγιση, σε απόσταση περίπου τεσσάρων με πέντε εκατοστών, της συσκευής σε κάποια άλλη που περιλαμβάνει τον κατάλληλο αισθητήρα.

Αυτή η μέθοδος λύνει το πρόβλημα του αυξημένου όγκου της συσκευής και το μειονέκτημα της ετεροχρονισμένης μετάδοσης της πληροφορίας, όμως είναι υψηλού κόστους και δεν μπορεί να εφαρμοστεί σε μαζική παραγωγή παρά μόνο σε περιορισμένη.

Κεφάλαιο 3

Διαδικασία Προώθησης Προϊόντος στην Αγορά

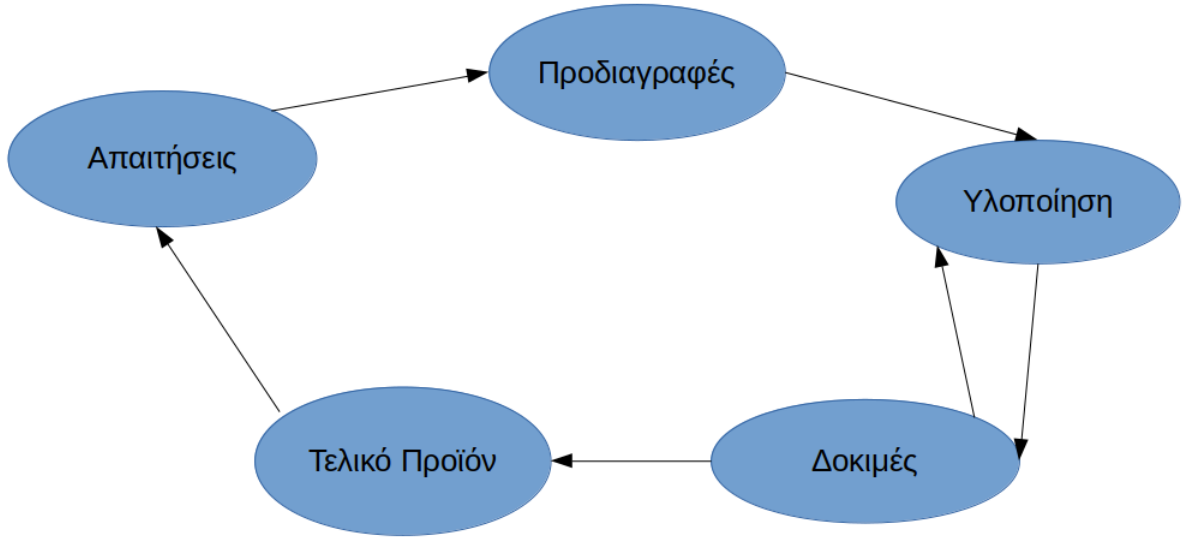
3.1 Κύκλος Δημιουργίας του Προϊόντος

Η παρούσα έρευνα αφορά τη δημιουργία μιας ηλεκτρονικής συσκευασίας φαρμάκων που θα στοχεύει στον περιορισμό ή ακόμα και στη επίλυση του προβλήματος της μη φαρμακευτικής συνέπειας των ασθενών.

Το να εισάγει κανείς ένα νέο προϊόν στην αγορά είναι μια περίπλοκη διαδικασία. Ο ακριβής αρχικός σχεδιασμός του προϊόντος καθίσταται αναγκαίος, καθώς οποιοδήποτε λάθος στην πορεία μπορεί να παραλύσει την τρέχουσα πρόοδο. Το γεγονός αυτό έχει συμβεί αρκετές φορές και προϊόντα που είναι έτοιμα να βγουν στην αγορά, βρίσκονται και πάλι στο μηδέν. Η τεχνική που ακολουθήθηκε σε αυτή την έρευνα είναι η εξής:

- Απαιτήσεις καταναλωτών
- Προδιαγραφές συσκευής
- Υλοποίηση συσκευής
- Δοκιμές
- Τελικό Προϊόν

Όπως φαίνεται και στο σχήμα 3.1, το πρώτο βήμα αποτελείται από μια ευρεία έρευνα των απαιτήσεων που θα έχουν οι καταναλωτές για το προϊόν. Στη συνέχεια, λοιπόν, θα ξεκινήσει ο σχεδιασμός του προϊόντος που θα καλύπτει αυτές τις απαιτήσεις και κατ' επέκταση η υλοποίηση του. Κατά τη διάρκεια του σταδίου των δοκιμών θα παρατηρηθούν προβλήματα, οπότε η διαδικασία θα επιστρέψει στο στάδιο της υλοποίησης. Εφόσον, όλες οι δοκιμές στεφθούν με επιτυχία, το προϊόν θα καταλήξει στην αγορά. Δεν πρόκειται ωστόσο, για μια διαδικασία που θα πραγματοποιηθεί μόνο μια φορά. Οποιαδήποτε ενημέρωση, εκσυγχρονισμός ή αλλαγή που θα υλοποιηθεί, θα πρέπει να περάσει πρώτα από τα προαναφερθέντα στάδια.



Σχήμα 3.1: Κύκλος Δημιουργία Προϊόντος

3.1.1 Κάλυψη Αναγκών Ομάδων Καταναλωτών

Η συσκευή αυτή απευθύνεται κυρίως στις παρακάτω κατηγορίες καταναλωτών.

- Ασθενείς
- Γιατροί
- Οικογενειακά Πρόσωπα/ Νοσηλευτές κατ' οίκον
- Φαρμακοποιοί
- Φαρμακευτικές Εταιρίες/ Ερευνητικοί Οργανισμοί

Η συσκευή εξυπηρετεί κάθε μία από αυτές τις κατηγορίες με διαφορετικό τρόπο, αφού κάθε μια χρειάζεται διαφορετικές υπηρεσίες και διευκολύνσεις.

3.1.1.1 Ασθενείς

Όπως έχει αναλυθεί σε προηγούμενο κεφάλαιο, οι ασθενείς δεν αντιλαμβάνονται τη σπουδαιότητα της τήρησης της φαρμακευτικής τους αγωγής και συχνά απλά ξεχνάνε να τη λάβουν. Ακόμα, σε περιπτώσεις περίπλοκων συνταγών, οι ασθενείς μπερδεύονται και δεν ξέρουν ποιο χάπι θα πρέπει να λάβουν. Παράλληλα, η ψυχολογική κατάσταση του ασθενή διαδραματίζει κύριο παράγοντα στη μη τήρηση του προβλεπόμενου χρονοδιαγράμματος για τη λήψη των φαρμάκων. Τέλος, πολλές συσκευασίες φαρμάκων συχνά τελειώνουν χωρίς να το αντιληφθούμε και δυστυχώς δεν υπάρχουν πάντα τα αντίστοιχα αποθέματα στα κοντινά φαρμακεία.

Συνεπώς, προκειμένου η συσκευή να αντιμετωπίζει αυτά τα προβλήματα, θα πρέπει να διαθέτει τις εξής δυνατότητες:

1. Υπενθύμιση λήψης χαπιού
2. Επαναλαμβανόμενες επόμενες υπενθυμίσεις, σε περίπτωση που ο ασθενής δεν είναι διαθέσιμος/ δεν ακούσει την πρώτη
3. Μετά από χρονικό διάστημα μισής ώρας, προγραμματισμός επόμενης δόσης
4. Επιβράβευση ασθενή για την προσπάθεια του
5. Ειδοποίηση για το τελείωμα των χαπιών
6. Χαμηλό κόστος
7. Πρακτική συσκευή με γνώριμες λειτουργίες στον ασθενή
8. Ανθεκτική Μπαταρία

3.1.1.2 Γιατροί

Το πιο σύνηθες πρόβλημα των γιατρών είναι η αμφιβολία για την αποτελεσματικότητα μιας θεραπευτικής αγωγής. Ένας ασθενής μπορεί να εμφανίζει συμπτώματα είτε σε περίπτωση αναποτελεσματικής αγωγής είτε επειδή δε λαμβάνει τα χάπια του όπως είναι προγραμματισμένα. Αυτό το ζήτημα οδηγεί πολλές φορές σε επιβάρυνση του ασθενή με ισχυρότερες θεραπείες και κατ' επέκταση περισσότερες παρενέργειες. Για την εξάλειψη του παραπάνω προβλήματος, η συσκευή θα παρέχει στον ιατρό τις παρακάτω πληροφορίες:

1. Ενημέρωση σε πραγματικό χρόνο για τη συνέπεια του ασθενή στη λήψη της φαρμακευτικής του αγωγής
2. Στατιστικά συγκεντρωτικά αποτελέσματα για την πορεία του.

3.1.1.3 Οικογενειακά Πρόσωπα/ Νοσηλευτές κατ' οίκον

Τα οικογενειακά πρόσωπα των ασθενών επιφορτίζονται κι αυτά με άγχη και επιπλέον υποχρεώσεις. Αντίστοιχα, οι νοσηλευτές θεωρούν πλέον δικιά τους υποχρέωση τη συνέπεια των ασθενών στην φαρμακευτική αγωγή τους.

Προκειμένου να μειωθούν τα άγχη των οικογενειακών και φιλικών προσώπων και να προληφθεί η παρατεταμένη παράλειψη κάποιου φαρμάκου, η ηλεκτρονική συσκευή θα συμβάλει ως εξής:

1. Έμμεση ενημέρωση τους από τον θεράποντα ιατρό, είτε μέσω e-mail είτε τηλεφωνικά

3.1.1.4 Φαρμακοποιοί

Οι αγορές των καταναλωτών στα φαρμακεία δεν ακολουθούν πάντα μια προβλεπόμενη πορεία. Υπάρχουν διαστήματα που προϊόντα βρίσκονται σε έλλειψη, ενώ άλλα που προϊόντα λήγουν στα ράφια των φαρμακείων. Το γεγονός αυτό προκαλεί τόσο οικονομικά προβλήματα

όσο και προβλήματα υγείας, αφού οι ασθενείς δεν μπορούν να τηρήσουν τη φαρμακευτική τους αγωγή. Παράλληλα, διάφορες υπάρχουσες ηλεκτρονικές συσκευασίες φαρμάκων απαιτούν τόσο περίπλοκο χειρισμό από το φαρμακοποιό, που ο ίδιος καταλήγει να μην τις συστήνει στους ασθενείς. Με στόχο, λοιπόν, την ικανοποίηση τόσο των φαρμακοποιών όσο και των ασθενών, η ηλεκτρονική συσκευασία θα έχει τη δυνατότητα:

1. Εύκολης αρχικοποίησης από το φαρμακοποιό
2. Ειδοποίηση του για τον τελειωμό των χαπιών των συσκευασιών από τους συνδεδεμένους πελάτες του, με στόχο την ενημέρωση των αποθεμάτων του

3.1.1.5 Φαρμακευτικές Εταιρίες/ Ερευνητικοί Οργανισμοί

Τόσο οι φαρμακευτικές εταιρίες όσο και οι ερευνητικοί οργανισμοί ψάχνουν να ανακαλύψουν τα αίτια που οδηγούν στη μη λήψη των συνταγογραφημένων φαρμάκων από τους ασθενείς. Προσπαθούν να σχηματίσουν μοτίβα και να ομαδοποιήσουν τους ασθενείς έτσι ώστε να εξηγηθεί επιστημονικά η συμπεριφορά τους. Τα πορίσματα που προκύπτουν από τα έως τώρα ευρήματα δείχνουν ότι η μη φαρμακευτική συνέπεια είμαι μια καθαρά ατομική συμπεριφορά. Ωστόσο συλλέγοντας περισσότερα δεδομένα, θα μπορούσαμε να εξάγουμε πιο συγκεκριμένα αποτελέσματα και να συνεισφέρουμε σημαντικά στον τομέα της υγείας.

Με τη χρήση της συσκευής λοιπόν θα είναι δυνατή:

1. Η ανώνυμη αποστολή δεδομένων σε ερευνητικά κέντρα/ φαρμακευτικές εταιρίες σχετικά με τη συνέπεια των ασθενών στη λήψη της φαρμακευτικής τους αγωγής.

Με τον τρόπο αυτό, θα μπορεί να αξιολογηθεί και η αποτελεσματικότητα της συσκευής.

3.1.2 Προδιαγραφές Συσκευής

Οι προδιαγραφές της συσκευής θα πρέπει να συμβαδίζουν με τις απαιτήσεις των διαφόρων ομάδων καταναλωτών. Αρχικά, η αφύπνιση των ασθενών προϋποθέτει την ύπαρξη ηχείου και οθόνης. Επίσης σημαντική καθίσταται η δυνατότητα μέτρησης της ώρας, ώστε να συγχρονίζεται η συσκευή με τις ώρες που πρέπει να ληφθούν τα χάπια από τον ασθενή. Ακόμα, απαραίτητη είναι και η ύπαρξη μίας κάρτας μνήμης ώστε να αποθηκεύονται τα ηχητικά αρχεία που θα χρησιμοποιηθούν ως μέσα αφύπνισης. Ασφαλώς, ο συνδυασμός όλων των παραπάνω θα επιτευχθεί μέσω ενός μικροελεγκτή. Παράλληλα, οι φαρμακοποιοί χρειάζονται ένα χρηστικό front-end περιβάλλον αλληλεπίδρασης με τη συσκευή, προκειμένου να την αρχικοποιήσουν. Τέλος, η βάση δεδομένων που θα αποθηκεύονται τα δεδομένα θα πρέπει να συμπεριλαμβάνει όλα τα σχετικά πρωτόκολλα υπεράσπισης και προστασίας των προσωπικών δεδομένων των καταναλωτών.

Συνεπώς, οι προδιαγραφές που θα πρέπει να εμπεριέχει η ηλεκτρονική συσκευή συνοψίζονται ως εξής:

- Μικροελεγκτής

- Οθόνη
- Υποδοχή για κάρτα μνήμης
- Ηχείο
- Ρολόι
- Wireless επικοινωνία
- Ανθεκτική μπαταρία

Κεφάλαιο 4

Προτεινόμενη Πλατφόρμα

4.1 Εισαγωγή

Η λύση που προτείνεται στην παρούσα μελέτη βασίζεται κυρίως στη χρήση των δυνατοτήτων του Arduino. Τα εξαρτήματα που χρησιμοποιήθηκαν και συναρμολογήθηκαν μεταξύ τους είναι τα παρακάτω:

- Arduino Mega
- Οθόνη LCD
- Memory Card Module
- Speaker
- Real Time Clock
- 24 Αντιστάσεις 68KΩ
- 6 Αντιστάσεις 20KΩ
- 1 Αντίσταση 47KΩ
- 1 Αντίσταση 20Ω
- Electronic Blister Medical Package

Το κάθε ένα από αυτά θα εξεταστεί αναλυτικότερα στη συνέχεια.

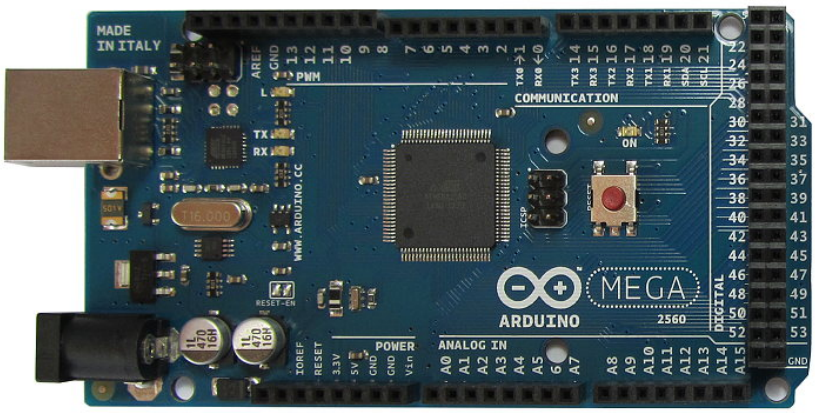
4.2 Λόγοι επιλογής Arduino

Πριν αναπτύξουμε τους λόγους για την επιλογή των επιμέρους συσκευών, θα πρέπει να αναλυθούν οι αιτίες επιλογής του Arduino, καθώς αποτελεί το βασικό στοιχείο της συσκευής. Βασικός παράγοντας της χρήσης του Arduino αποτελεί το χαμηλό του κόστος. Αντίστοιχα, το κόστος και όλων των υπολοίπων συσκευών συμπεριλαμβάνεται στο εύρος από 1€-10€.

Μία εκτεταμένη έρευνα πραγματοποιήθηκε για την επιλογή του Arduino Mega έναντι του Raspberry Pi.

4.2.1 Arduino Mega

Στη συγκεκριμένη έρευνα το Arduino αποτελεί την κύρια συσκευή προγραμματισμού της συσκευασίας λήψης χαπιών (Σχήμα 4.1). Το Arduino θα μετρήσει την τάση που θα έχει κάθε στιγμή το blister χαπιών, θα ενημερωθεί για την συνέπεια του ασθενή σχετικά με τη λήψη της φαρμακευτικής του αγωγής και στη συνέχεια θα αποστείλει τα δεδομένα αυτά στην εξωτερική βάση δεδομένων. Ακόμα είναι η κύρια συσκευή (master) που συνδέει όλες τις υπόλοιπες περιφερειακές συσκευές.



Σχήμα 4.1: Arduino Mega Πλακέτα

Πρόκειται για μία πλακέτα επέκτασης του Arduino UNO. Το Arduino είναι ένας μικροελεγκτής μονής πλακέτας, δηλαδή μια απλή μητρική πλακέτα *ανοικτού κώδικα* με ενσωματωμένο μικροελεγκτή και εισόδους/εξόδους, η οποία μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για τη γλώσσα προγραμματισμού C++ και ένα σύνολο από βιβλιοθήκες, υλοποιημένες επίσης στην C++). Το Arduino χρησιμοποιείται για την ανάπτυξη διαδραστικών αντικειμένων αλλά συνδέεται και με υπολογιστή με χρήση ειδικών προγραμμάτων. Συγκεκριμένα το Arduino Mega βασίζεται στο ATmega2560 της Atmel. Το ATmega2560 παρέχει σειριακή επικοινωνία (5V), η οποία είναι διαθέσιμη στις ψηφιακές εισόδους RX και TX του Arduino. Έχει τη δυνατότητα να δεχθεί μονάδες εισόδου/εξόδου. Περιλαμβάνει αναλυτικά:

- 54 ψηφιακές μονάδες εισόδου/εξόδου (εκ των οποίων οι 14 μπορούν να χρησιμοποιηθούν ως PWM έξοδοι, παλμούς ρυθμιζόμενου πλάτους)
- 16 αναλογικές μονάδες εισόδου/εξόδου
- 4 USART (Universal Synchronous Asynchronous Receiver and Transmitter) Σειριακές Θύρες Επικοινωνίας
- έναν 16 MHz ταλαντωτή κρυστάλλου

- μία USB υποδοχή
- μια υποδοχή τροφοδοσίας
- ICSP header
- κουμπί επαναφοράς Reset

Η συσκευή προγραμματίζεται σε Arduino IDE (ολοκληρωμένο περιβάλλον ανάπτυξης). Για το προγραμματισμό της συσκευής χρησιμοποιήθηκαν οι εξής βιβλιοθήκες:

- *LiquidCrystal.h* για τον προγραμματισμό της οθόνης κρυστάλλων
- *SD.h* για την κάρτα μνήμης
- *TMRpcm.h* για τις λειτουργίες του ηχείου
- *string.h*
- *ArduinoJson.h* για την ανάγνωση Json αρχείων
- *EEPROM.h* για τη χρήση της EEPROM μνήμης που διαθέτει το Arduino
- *RTClib.h* για τον προγραμματισμό του εξωτερικού ρολογιού πραγματικού χρόνου
- *Wire.h*
- *avr/sleep.h* Για τον ορισμό του Sleep Mode σε Power Down Mode
- textitavr/wdt.h Για τον προγραμματισμό του WatchDog Interrupt

4.2.2 Raspberry Pi

Το Raspberry Pi είναι ένας μικρός υπολογιστής σε μέγεθος μιας πιστωτικής κάρτας. Δημιουργήθηκε στο Ηνωμένο Βασίλειο από το Raspberry Pi Foundation για την ευκολότερη εκμάθηση της επιστήμης των υπολογιστών στα σχολεία.

Το τελευταίο μοντέλο που έχει λανσαριστεί στην αγορά είναι το *Raspberry Pi3* (Σχήμα 4.2). Περιλαμβάνει 512 MB RAM μνήμης και 700MHz μικροεπεξεργαστή. Τα χαρακτηριστικά του είναι τα παρακάτω:

- SoC: Broadcom BCM2837
- CPU: 4× ARM Cortex-A53, 1.2GHz
- GPU: Broadcom VideoCore IV
- RAM: 1GB LPDDR2 (900 MHz)
- Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless

Σχήμα 4.2: Πλακέτα Raspberry Pi3

- Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
- Storage: microSD
- GPIO: 40-pin header
- Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

4.2.2.1 Διαφορές μεταξύ Arduino και Raspberry Pi

Η κάθε μία συσκευή έχει τα πλεονεκτήματα και τα μειονεκτήματα της. Η βέλτιστη επιλογή εξαρτάται μόνο από την εφαρμογή που θέλουμε να αναπτύξουμε.

Το Raspberry Pi ενδείκνυται για εφαρμογές με περιορισμένο Hardware κομμάτι και πιο περίπλοκο Software. Ακόμα εξυπηρετεί περιπτώσεις, στις οποίες χρειάζεται να χρησιμοποιηθούν ποικίλες γλώσσες προγραμματισμού και όχι μόνο C και C++ . Αντίθετα, σε εφαρμογές με αλληλεπίδραση διαφορετικών συσκευών, όπου απαιτείται η ανάγνωση δεδομένων ή ο έλεγχος αυτών, το Raspberry Pi δεν καθίσταται ιδανική επιλογή. Συνοπτικά, μπορούμε να καταλήξουμε στα εξής:

Πλεονεκτήματα Raspberry Pi

(α΄) Η συσκευή μπορεί να τρέχει εξολοκλήρου με λειτουργικό σύστημα Linux. Αντίθετα, στο Arduino δεν υπάρχει λειτουργικό σύστημα. Πρόκειται για προγραμματισμό

χαμηλού επιπέδου. Ο ίδιος ο προγραμματιστής γράφει το firmware. Αντίστοιχα, θα πρέπει να προγραμματίζει την κάθε μονάδα εισόδου/εξόδου στην κατάλληλη τιμή (0 – 5V). Ακόμα, οι Hardware Interrupts είναι πολύ περιορισμένες και θα αναλυθούν περαιτέρω σε επόμενο κεφάλαιο.

(β′) Ευκολία σύνδεσης στο διαδίκτυο. Από την άλλη πλευρά, θα πρέπει να υπάρχει κατάλληλο Hardware/Software για τη σύνδεση του Arduino στο διαδίκτυο.

(γ′) Μπορεί να προγραμματιστεί σε ποικίλες γλώσσες προγραμματισμού, όπως σε Python, HTML5, Javascript, Jquery, JAVA, Perl, Erlang, C και C++. Αντίθετα, το Arduino περιορίζεται μόνο στις δύο τελευταίες ή σε Assembly.

Μειονεκτήματα Raspberry Pi

(α′) Η Hardware επικοινωνία δε γίνεται σε πραγματικό χρόνο. Σε περίπτωση που η CPU (Central Processing Unit – Κεντρική Μονάδα Επεξεργασίας) είναι απασχολημένη, η διασύνδεση με το Hardware μπορεί να καθυστερήσει. Από την άλλη πλευρά, το Arduino μπορεί να χρησιμοποιηθεί σε εφαρμογές πραγματικού χρόνου.

(β′) Δε διαθέτει ενσωματωμένο *Άναλογικό σε Ψηφιακό* μετατροπέα σε αντίθεση με το Arduino.

(γ′) Ο σχεδιασμός του Hardware συστήματός του δεν είναι *Open - Source* αντίθετα με το Arduino.

Λαμβάνοντας υπόψη όλα τα παραπάνω, επιλέχθηκε η χρήση του Arduino στην παρούσα διπλωματική εργασία. Κύριος λόγος αποτέλεσε η ευκολία στη λήψη σημάτων πραγματικού χρόνου από το Blister των χαπιών.

4.3 LCD Οθόνη

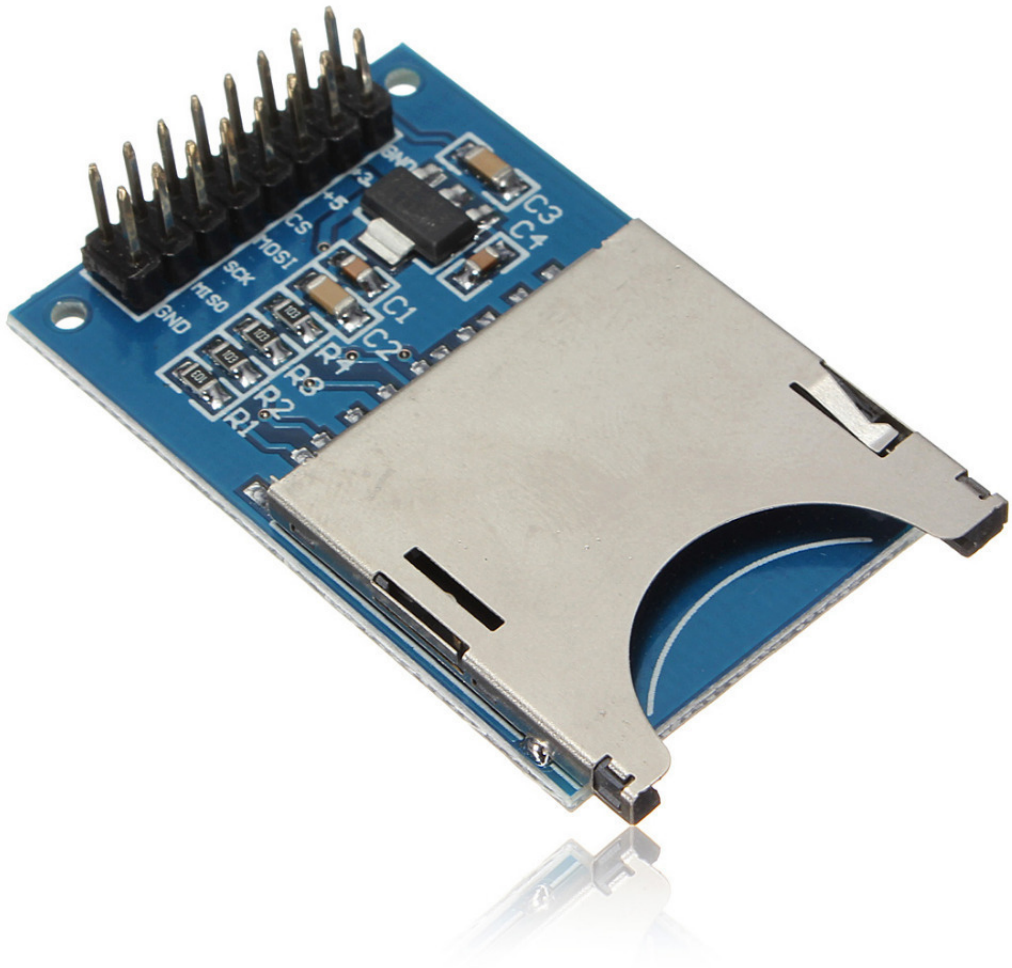
Στην παρούσα έρευνα, η LCD οθόνη χρησιμοποιείται για την απεικόνιση μηνυμάτων στον ασθενή, σχετικά με την ενημέρωση του για τη λήψη κάποιου χαπιού, την επιβράβευση του μετά από την επίδειξη της απαιτούμενης συνέπεια, την προτροπή του για ανανέωση της αγωγής λόγω τελειωμού της υπάρχουσας ή την ενημέρωση του για κάποιο εσωτερικό πρόβλημα της συσκευής (Σχήμα 4.3).

Σχήμα 4.3: LCD Οθόνη

Η οθόνη υγρών κρυστάλλων αποτελεί ένα Module ηλεκτρονικής απεικόνισης. Διαθέτει δύο σειρές συμβολοσειρών, όπου η κάθε μία χωράει μέχρι 16 χαρακτήρες. Κάθε χαρακτήρας εμφανίζεται σε ένα πλαίσιο 5×7 pixel. Αποτελείται από δύο καταχωρητές, αυτόν για τις εντολές (Command) και αυτόν για τα δεδομένα (Data). Ο πρώτος αποθηκεύει τις εντολές που δίνονται στην LCD οθόνη. Κάθε εντολή που δίνεται έχει ένα προκαθορισμένο έργο. Για παράδειγμα μπορεί να αρχικοποιεί την οθόνη, να την καθαρίζει, να θέτει τον κέρσορα σε μια συγκεκριμένη θέση, να αλλάζει το χρώμα στο φόντο, κλπ. Από την άλλη πλευρά, ο Data καταχωρητής αποθηκεύει τα δεδομένα που θα εμφανιστούν στην LCD οθόνη. Τα δεδομένα μετατρέπονται σε ASCI μορφή και αποθηκεύονται αντίστοιχα. Μπορεί να συνδεθεί τόσο με αναλογικές όσο και με ψηφιακές εισόδους. Έχει πολλά πλεονεκτήματα έναντι των 7-Segment οθονών, καθώς μπορεί να εμφανίσει οποιοδήποτε χαρακτήρα και δεν περιορίζεται στα αριθμητικά ψηφία 0-9. Ακόμα, είναι οικονομική και προγραμματίζεται εύκολα.

4.4 Memory Card Module (SD Card)

Το module υποδοχής κάρτας μνήμης αποτελεί έναν επιπλέον χώρο αποθήκευσης για το Arduino (Σχήμα 4.4). Οι χρήστες μπορούν είτε να γράψουν είτε να διαβάσουν την SD Card μέσω της υπάρχουσας SD βιβλιοθήκης. Περιλαμβάνει 8 εξωτερικές θύρες όπως φαίνεται και στο παρακάτω σχήμα. Καταλαμβάνει μόνο την SPI θύρα του Arduino, η λειτουργία της οποίας θα αναλυθεί παρακάτω.

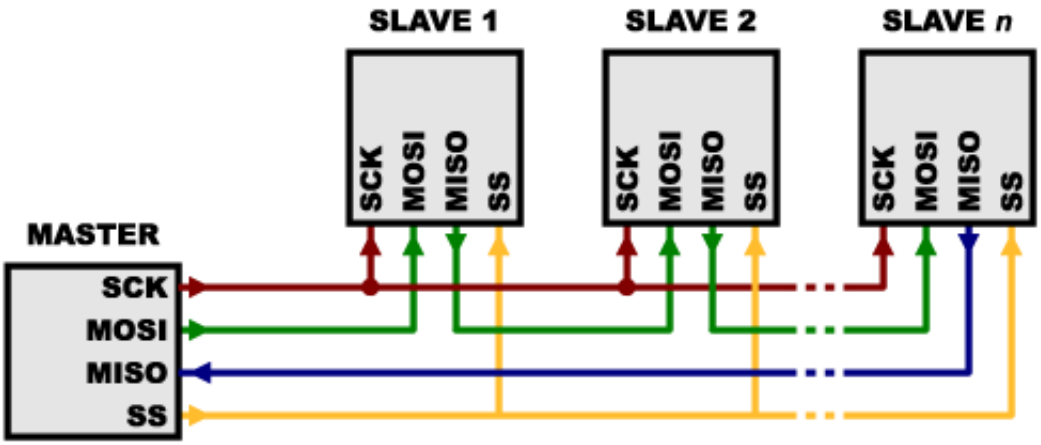


Σχήμα 4.4: Module Υποδοχής Κάρτας Μνήμης

4.4.1 Serial Peripheral Interface (SPI)

Πρόκειται για μια διεπαφή σύγχρονης και σειριακής επικοινωνίας, η οποία χρησιμοποιείται από μικροελεγκτές για τη μεταφορά δεδομένων μεταξύ περιφερειακών συσκευών (Σχήμα 4.5). Η επικοινωνία μπορεί να επιτευχθεί άμεσα σε κοντινές αποστάσεις. Επίσης, χρησιμοποιείται και για την επικοινωνία μεταξύ μικροελεγκτών. Στην SPI σύνδεση υπάρχει πάντα μία κύρια συσκευή (συνήθως ο μικροελεγκτής), η οποία ελέγχει τις περιφερειακές συσκευές. Τυπικά, υπάρχουν τρεις κύριες γραμμές επικοινωνίας σε όλες τις συσκευές:

- MISO (Master In Slave Out)- Επικοινωνία από την δευτερεύουσα συσκευή (slave) στην κύρια (master).
- MOSI (Master Out Slave In)- Επικοινωνία από την κύρια συσκευή (master) στη δευτερεύουσα (slave)



Σχήμα 4.5: Serial Peripheral Interface (SPI

- SCK (Serial Clock) – Συγχρονισμός της μετάδοσης δεδομένων με παλμούς ρολογιού, οι οποία παράγονται από την κύρια συσκευή

και μία η οποία εξαρτάται από την κάθε συσκευή:

- SS (Slave Select) – Η μονάδα εισόδου (pin) των συσκευών, που ενεργοποιεί ή απενεργοποιεί την επικοινωνία της κύριας συσκευής με αυτές. Όταν το pin κάποιας συσκευής είναι 0V, τότε αυτή επικοινωνεί με την κύρια, ενώ όταν είναι 5V την αγνοεί. Αυτό επιτρέπει την ύπαρξη πολλών *slave* συσκευών, οι οποίες θα μοιράζονται τις ίδιες MISO, MOSI, CLK γραμμές.

Το SD card Module βασίζεται στην SPI επικοινωνία. Στον πίνακα 4.1 παρουσιάζονται οι συμβατές θύρες διάφορων Arduino πλακετών.

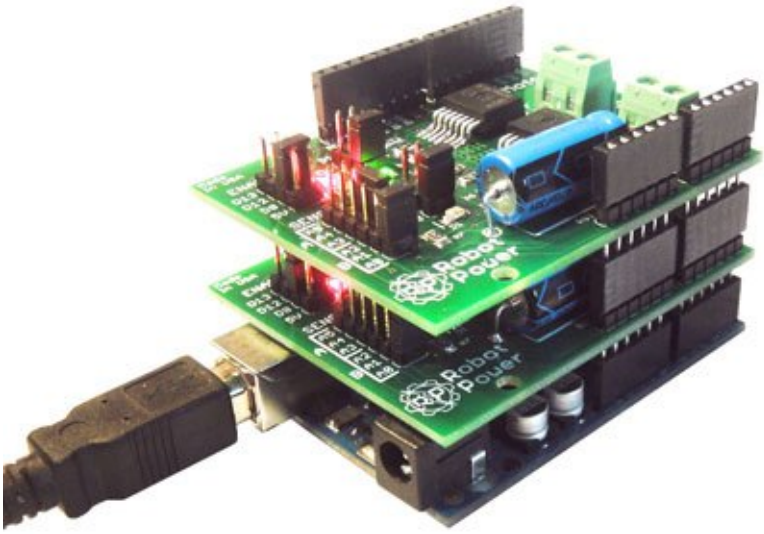
Arduino/Genuino Boards	MOSI	MISO	SCK	SS
Uno, Duemilanove	11 ή ICSP-4	12 ή ICSP-1	13 ή ICSP-3	10
Mega1280, Mega2560	51 ή ICSP-4	50 ή ICSP-1	52 ή ICSP-3	53
Leonardo	ICSP-4	ICSP-1	ICSP-3	-
MKP1000	8	10	9	-

Πίνακας 4.1: SPI συσχετισμός θυρών πλακετών Arduino

4.4.2 Διαφορά Shield και Module

Με τον όρο *Shield* (Σχήμα 4.6) θεωρούμε ένα *Module*, το οποίο είναι σχεδιασμένο με τρόπο που να συνδέεται ακριβώς με την κεντρική πλακέτα του Arduino. Συνδέεται στις ελεύθερες εισόδους του χωρίς να απαιτεί κάποια ιδιαίτερη συνδεσμολογία με επιπλέον καλώδια.

Το Module είναι μία ανεξάρτητη ηλεκτρονική συσκευή και γι αυτό θεωρείται συχνά ευκολότερο στην προσαρμογή και κατ' επέκταση στη χρήση της.



Σχήμα 4.6: Arduino Shield

4.5 Real Time Clock (RTC)

Το Tiny RTC clock (Σχήμα 4.7) επικοινωνεί με το Arduino χρησιμοποιώντας I^2C τεχνολογία. Ακόμα περιλαμβάνει 32K EEPROM χώρο αποθήκευσης και ενσωματώνει αισθητήρα θερμοκρασίας (DS18B20). Όλες οι λειτουργίες του συμπεριλαμβάνονται σε μία πλακέτα μεγέθους 25mm x 28mm x 8.4mm. Ακόμα, φορτίζεται αυτόνομα αφού περιλαμβάνει τη δική της επαναφορτιζόμενη μπαταρία λιθίου (LIR2303). Συγκεκριμένα, όταν ο αισθητήρας θερμοκρασίας είναι απενεργοποιημένος, η μπαταρία μπορεί να λειτουργεί συνεχόμενα για ένα χρόνο.

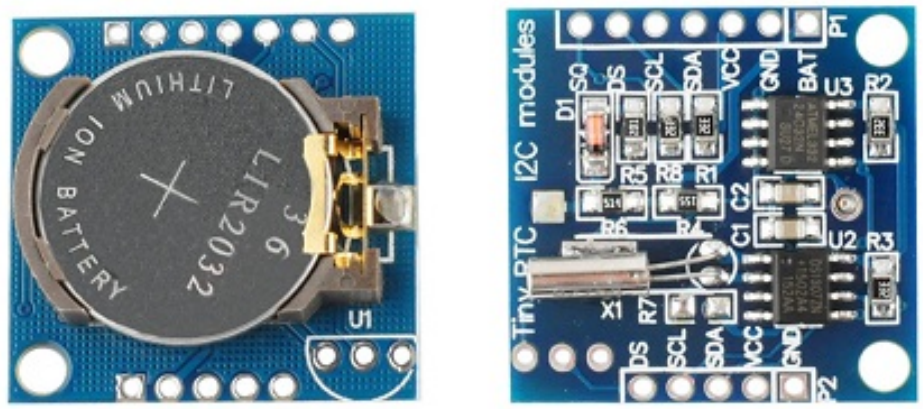
Το module αυτό χρησιμοποιείται κυρίως σε εφαρμογές καταγραφής δεδομένων ή χρονισμού όπως για παράδειγμα η ενεργοποίηση κάποιας συσκευής τρεις φορές τη μέρα. Από τη στιγμή που το module έχει αυτόνομη λειτουργία, μπορεί να διατηρήσει δεδομένα ακόμα και σε περίπτωση απενεργοποίησης του Arduino, επιτρέποντας έτσι τη χαμηλή κατανάλωση συστημάτων, τα οποία μπορούν να βρίσκονται σε αδράνεια αρκετές ώρες.

Τα χαρακτηριστικά του DS18B20 αισθητήρα είναι τα παρακάτω:

- 5.0-5.5V input voltage
- Αδιάβροχο
- Το έυρος των θερμοκρασιών που μπορεί να μετρήσει είναι από -55°C έως +125°C
- Ακρίβεια ±0.5°C από -10°C έως +85°C
- ένα καλώδιο διεπαφής

Τα πιο σημαντικά pins υπάρχουν δύο φορές. Παρακάτω παρουσιάζεται η λειτουργικότητα των pins του RTC module:

Παρακάτω αναλύεται η I^2C επικοινωνία του RTC με το Arduino.



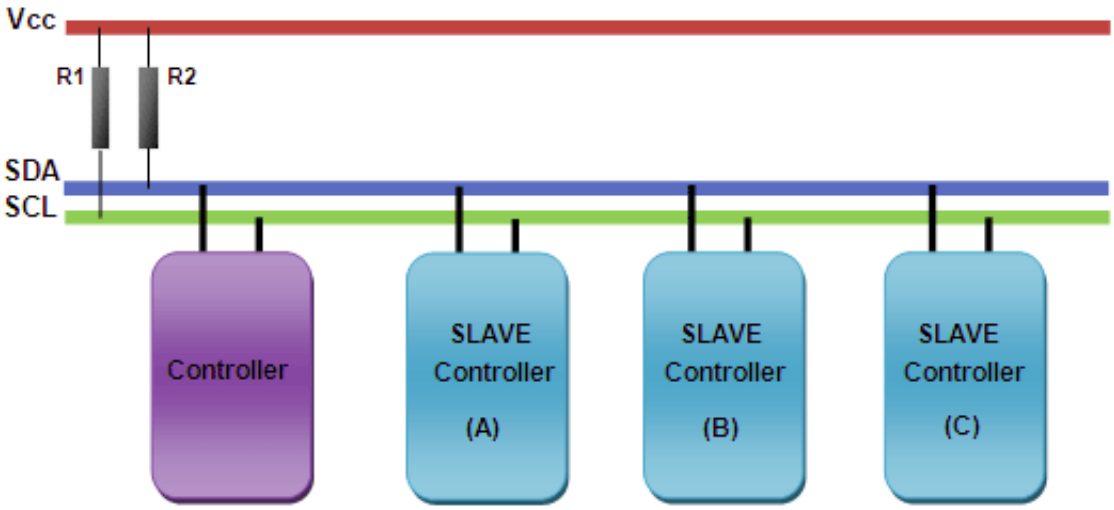
Σχήμα 4.7: Real Time Clock

4.5.1 Two wire Serial Interface (TWI/I^2C)

Πρόκειται για ένα πρωτόκολλο που επιτρέπει την σύγχρονη μετάδοση σειριακών δεδομένων. Χρησιμοποιεί δύο μόνο διαύλους για την επικοινωνία μεταξύ δύο ή περισσοτέρων ολοκληρωμένων (Σχήμα 4.8). Αποτελεί ένα πρωτόκολλο πολλαπλών σημείων αφού μπορεί να συνδέσει μέχρι και 27 περιφερειακές συσκευές. Υπάρχουν δύο αμφίδρομης επικοινωνίας γραμμές, η SDA (Serial Data) και η SCL (Serial Clock) που υλοποιούν την μετάδοση δεδομένων μεταξύ των συσκευών. Η επικοινωνία βασίζεται στη λογική Master – Slave. Μία από τις συσκευές ελέγχει όλη τη διαδικασία και οι άλλες λειτουργούν βάσει αυτής. Η SCL αποτελεί το ρολόι και ελέγχεται από τη Master συσκευή, ενώ η SDA γραμμή αποτελεί τον διάδρομο μεταφοράς δεδομένων. Ο ρυθμός μεταφοράς δεδομένων εξαρτάται από τη συχνότητα των Slave συσκευών. Στην ουσία αποτελεί τη συχνότητα του ρολογιού στην SCL γραμμή.

PIN	Περγραφή	Σχόλια
BAT	Battery Voltage	Χρησιμοποιείται είτε για τη μέτρηση της τάσης της μπαταρίας είτε δεν είναι συνδεδεμένο
GND	Ground	Γείωση
Vcc	5V τροφοδοσία	Ενεργοποιεί το Arduino και φορτίζει τη μπαταρία
SDA	I2C data	I2C data for the RTC
SCL	I2C clock	I2C clock for the RTC
DS	DS18B20	Αισθητήρας Θερμοκρασίας
SQ	Square wave output	Συνήθως δεν χρησιμοποιείται

Πίνακας 4.2: Λειτουργικότητα των Θυρών του RTC module:



Σχήμα 4.8: Πρωτόκολλο I^2C για τη Σειριακή Μετάδοση Δεδομένων

4.5.1.1 Διαφορές μεταξύ SPI και I^2C

Οι βασικές διαφορές μεταξύ αυτών των δύο σειριακών πρωτοκόλλων επικοινωνίας είναι οι παρακάτω:

- I^2C απαιτεί μόνο δύο γραμμές επικοινωνίας, ενώ το SPI απαιτεί τρεις ή τέσσερις
- Η SPI υποστηρίζει υψηλότερες ταχύτητες μεταφοράς δεδομένων
- Η I^2C αντλεί μεγαλύτερη ενέργεια
- Η I^2C μπορεί να επικοινωνήσει με πολλαπλές συσκευές μέσω του ίδιου διαδρόμου, χωρίς να χρειάζεται επιπλέον γραμμές σήματος
- Σε αντίθεση με την SPI, η I^2C διασφαλίζει ότι τα δεδομένα λήφθηκαν σωστά από τις Slave συσκευές.
- Η I^2C είναι πιο οικονομική στην υλοποίηση

- Η SPI υποστηρίζει μία μοναδική Master συσκευή, ενώ η I^2C δέχεται πολλαπλές
- Η I^2C είναι λιγότερο ευαίσθητη σε θόρυβο
- Η SPI μπορεί να μεταφέρει δεδομένα σε κοντινές μόνο αποστάσεις, ενώ το εύρος της I^2C είναι μεγαλύτερο

Γενικά, η SPI προτιμάται για επικοινωνίες με μεγαλύτερη ταχύτητα και χαμηλή κατανάλωση ενέργειας, ενώ η I^2C διακρίνεται για την επικοινωνία με μεγάλο αριθμό περιφερειακών συσκευών και δυναμική αλλαγή της Master συσκευής. Και οι δύο τεχνολογίες ωστόσο είναι αποτελεσματικές με σταθερά πρωτόκολλα επικοινωνίας για ενσωματωμένες εφαρμογές.

Στον πίνακα 4.3 παρουσιάζεται η I^2C συνδεσμολογία θυρών των διαφόρων Arduino πλακετών.

Arduino Board	SDA	SCL
UNO, Ethernet	A4	A5
Mega	20	21
Leonardo	2	3
Due	20	21

Πίνακας 4.3: Συνδεσμολογία θυρών Arduino σε I^2C επικοινωνία

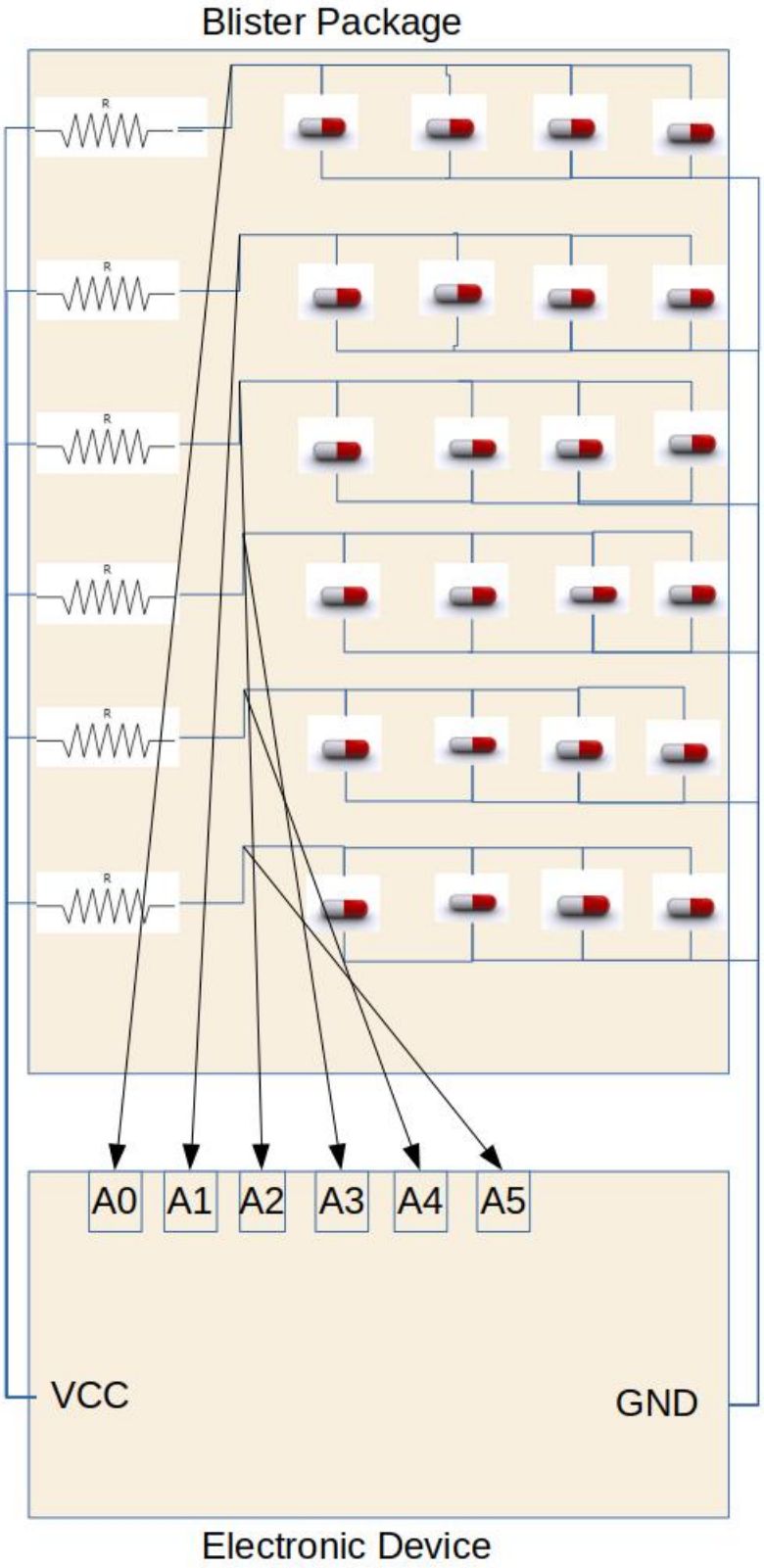
4.6 Speaker

Αποτελεί μία μικρή συσκευή ήχου, με δυνατότητας αυξομείωσης της έντασης. Πρόκειται για τη συσκευή που αναπαράγει το αρχείο ήχου, που χρησιμοποιείται σαν αφύπνιση για τον ασθενή. Η συνδεσμολογία της συσκευής περιλαμβάνει τρεις μόνο θύρες, την τροφοδοσία, τη γείωση και το σήμα το οποίο θα παράξει τη μουσική.

Σχήμα 4.9: Ηχείο Μουσικής

4.7 Blister Package

Στην παρούσα έρευνα, τα χάπια θα τοποθετηθούν σε μία συσκευασία blister. Υπάρχουν οχτώ αγώγιμες συνδέσεις μεταξύ του blister και της ηλεκτρονικής συσκευής. Μία για την πηγή τάσης, μία για τη γείωση και έξι για τη μέτρηση της τάσης. Κάθε μία από τις έξι παράλληλες αγώγιμες συνδέσεις αποτελείται από τέσσερις παράλληλες γραμμές, όπου η κάθε μια διατηρεί και από ενα χάπι, δηλαδή μία αντίσταση (Σχήμα 4.10). Μέσα σε χρονικό διάστημα δέκα δευτερολέπτων, η συσκευή μετράει την τάση κάθε μίας από τις έξι γραμμές ξεχωριστά. Έχοντας αποθηκευμένη την προηγούμενη τιμή της κάθε γραμμής χαπιών, υπολογίζει τη διαφορά τάσης στο blister και αντιλαμβάνεται αν έχει ληφθεί η απαιτούμενη φαρμακευτική αγωγή από τον ασθενή. Στο παρακάτω σχήμα, παρουσιάζεται ο σχεδιασμός του blister και η σύνδεση του με την ηλεκτρονική συσκευή. Η συσκευή φαίνεται απλά σαν ένα *black box*, που διαθέτει μόνο τις απαιτούμενες εξωτερικές συνδέσεις.



Σχήμα 4.10: Blister φαρμάκων

Κεφάλαιο 5

Θεωρητικό Υπόβαθρο

5.1 Εισαγωγή

Στην παρούσα έρευνα, η ηλεκτρονική συσκευή προγραμματίστηκε με τρόπο που να εξυπηρετεί καλύτερα τη λειτουργία της ως συσκευή ειδοποίησης λήψης χαπιών. Κάποιοι από τους παράγοντες που λήφθηκαν υπ' όψη είναι οι δυνατότητες της μνήμης της και οι μέθοδοι εξοικονόμησης ενέργειας.

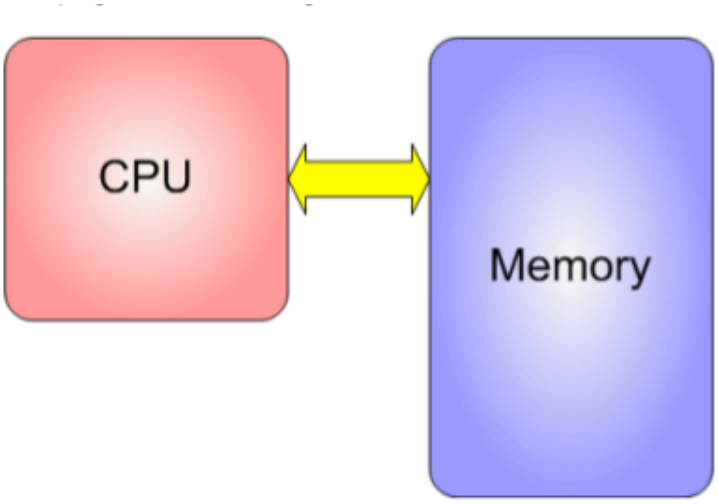
5.2 Μέθοδοι Βέλτιστης Διαχείρισης της Μνήμης της Συσκευής

5.2.1 Αρχιτεκτονικές Διαχείρισης Μνήμης

Σήμερα οι περισσότεροι ηλεκτρονικοί υπολογιστές νέας γενιάς είναι υβριδικά σχεδιασμένοι, συνδυάζοντας δύο διαφορετικές αρχιτεκτονικές για τη διαχείριση της μνήμης τους, την *Von Neumann* και τη *Harvard*.

5.2.1.1 Von Neumann Architecture

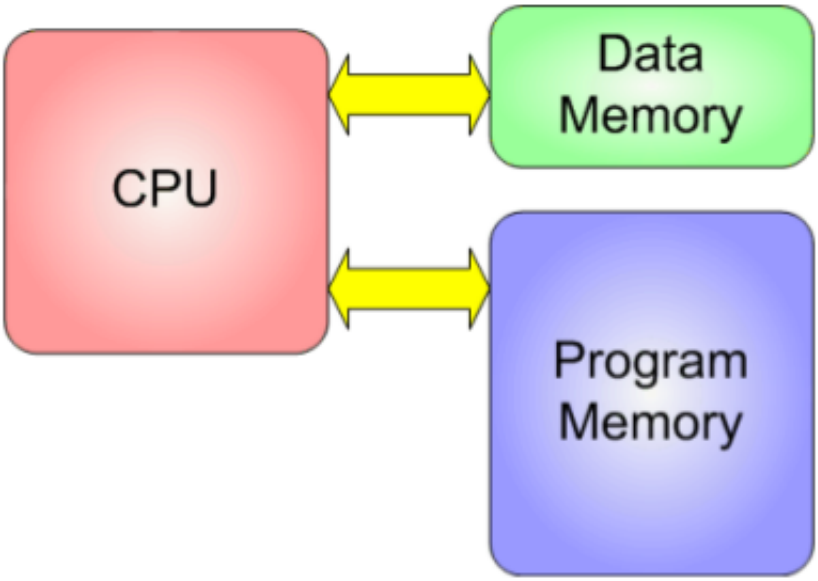
Σχεδιάστηκε από τον διάσημο μαθηματικό και φυσικό *John Von Neumann* το 1945. Μέχρι να προταθεί η έννοια της *Von Neumann* αρχιτεκτονικής για το σχεδιασμό των ηλεκτρονικών υπολογιστών, οι υπολογιστικές μηχανές σχεδιαζόντουσαν για έναν και μόνο προκαθορισμένο σκοπό που θα απέφευγε την πολυπλοκότητα, λόγω της χειροκίνητης επανασύνδεσης των κυκλωμάτων. Πρόκειται για θεωρητικό σχεδιασμό, βασισμένο στην έννοια των υπολογιστών αποθηκευμένου προγράμματος *(stored-program)*, όπου τα δεδομένα του προγράμματος και οι εντολές αποθηκεύονται στην ίδια μνήμη. Η ιδέα πίσω από τις αρχιτεκτονικές Von Neumann είναι η δυνατότητα αποθήκευσης οδηγιών στη μνήμη μαζί με τα δεδομένα στα οποία λειτουργούν αυτές οι οδηγίες.



Σχήμα 5.1: VonNeumann Αρχιτεκτονική

5.2.1.2 Harvard Architecture

Πρόκειται για μια αρχιτεκτονική ηλεκτρονικών υπολογιστών με ξεχωριστό τρόπο αποθήκευσης των δεδομένων και των εντολών. Σε αντίθεση με την αρχιτεκτονική *Von Neumann*, η οποία χρησιμοποιεί έναν μόνο δίαυλο επικοινωνίας για να έχει πρόσβαση σε εντολές από τη μνήμη και να μεταφέρει δεδομένα σε διάφορα σημεία του υπολογιστή, η Harvard έχει διαφορετικό χώρο μνήμης για τις εντολές και τα δεδομένα. Οι δύο έννοιες είναι παρόμοιες και διαφέρουν μόνο στον τρόπο με τον οποίο έχουν πρόσβαση σε μνήμες. Η ιδέα πίσω από την Harvard αρχιτεκτονική είναι ο χωρισμός της μνήμης σε δύο μέρη – μία για τα δεδομένα και μία για τις εντολές.



Σχήμα 5.2: Harvard Αρχιτεκτονική

5.2.1.3 Βασικές Διαφορές των δύο αρχιτεκτονικών

Von Neumann Architecture	Harvard Architecture
Ίδια φυσική μνήμη για δεδομένα και εντολές	Διαφορετική φυσική μνήμη για δεδομένα και εντολές
Ο επεξεργαστής χρειάζεται δύο κύκλους ρολογιού για την εκτέλεση εντολών	Ο επεξεργαστής χρειάζεται έναν κύκλο ρολογιού για την εκτέλεση εντολών
Ο σχεδιασμός και η ανάπτυξη της μονάδας ελέγχου είναι πιο απλός, άρα πιο γρήγορος και πιο απλός	Η μονάδα ελέγχου αποτελείται από δύο διαύλους επικοινωνίας, γεγονός που αυξάνει την πολυπλοκότητα της και κατά συνέπεια το κόστος ανάπτυξης
Η μεταφορά δεδομένων και εξαγωγή εντολών δε μπορούν να εκτελεστούν ταυτόχρονα	Η μεταφορά δεδομένων και εξαγωγή εντολών μπορούν να εκτελεστούν την ίδια στιγμή
Ο σχεδιασμός του οφείλεται στην ιδέα του stored-program υπολογιστή	Είναι μοντέρνα αρχιτεκτονική που βασίζεται σε αναμετάδοση του Harvard Mark I υπολογιστικού μοντέλου
Χρησιμοποιείται κυρίως σε προσωπικούς υπολογιστές	Χρησιμοποιείται σε μικροεπεξεργαστές και στην ψηφιακή επεξεργασία σημάτων (Digital Signal Processing (DSP))

Πίνακας 5.1: Βασικές Διαφορές των αρχιτεκτονικών Von Neumann και Harvard

Οι περισσότεροι μικροεπεξεργαστές, όπως και αυτός που έχει το Arduino, είναι σχεδιασμένοι σύμφωνα με το *Harvard* μοντέλο για τη διαχείριση της μνήμης τους.

5.2.2 Κατηγορίες Μνήμης Arduino

Υπάρχουν τρεις ομάδες μνήμης στον μικροελεγκτή που χρησιμοποιείται στις πλακέτες του Arduino. Αυτές είναι οι παρακάτω:

1. Flash Memory (χώρος προγράμματος): αποθηκεύεται το Arduino sketch (το περιβάλλον στο οποίο είναι γραμμένο το πρόγραμμα)
2. SRAM (Static Random Access Memory): το sketch δημιουργεί και χειρίζεται μεταβλητές όταν τρέχει
3. EEPROM : χώρος μνήμης, που ο προγραμματιστής μπορεί να αποθηκεύσει μακροπρόθεσμα πληροφορίες.

5.2.2.1 Flash Memory

Χρησιμοποιείται για την αποθήκευση της εικόνας του προγράμματος και την αρχικοποίηση των εντολών. Από τη Flash μνήμη μπορείτε να εκτελέσετε τον κώδικα του προγράμματος

αλλά δεν μπορείτε να αλλάξατε τα δεδομένα στη Flash μνήμη από τον εκτελούμενο κώδικα. Προκειμένου να μεταβάλετε τα δεδομένα, θα πρέπει πρώτα ο εκτελούμενος κώδικας να αντιγραφεί στη SRAM. Η Flash Memory έχει την ίδια τεχνολογία που χρησιμοποιούν και οι SD cards. Είναι μη πτητική, δηλαδή δε χάνει τα δεδομένα της με τη διακοπή της τροφοδοσίας ρεύματος. Επίσης, έχει μια πεπερασμένη διάρκεια ζωής περίπου 100.000 κύκλων εγγραφής.

5.2.2.2 SRAM

Η μνήμη αυτή μπορεί να διαβαστεί και να γραφτεί από το εκτελούμενο πρόγραμμα (Σχήμα 5.3). Χρησιμοποιείται για διάφορους σκοπούς όταν τρέχει ένα πρόγραμμα, όπως:

5.2.2.2.1 Στατικά Δεδομένα

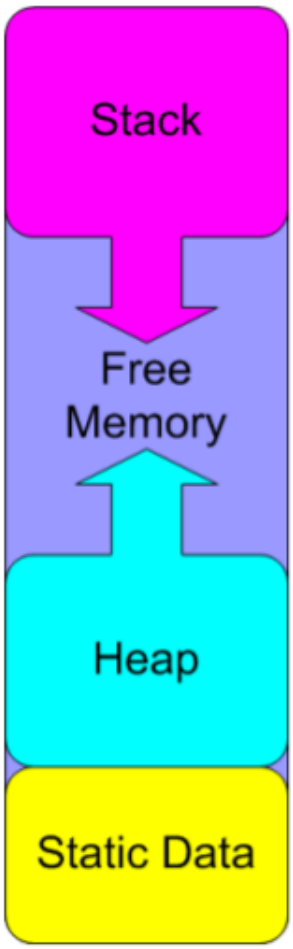
Είναι ένα μπλοκ δεσμευμένου χώρου στη SRAM για όλες τις global και τις στατικές μεταβλητές του προγράμματος. Επίσης, οι μεταβλητές με αρχικοποιημένες τιμές, αντιγράφονται κατά τη διάρκεια της εκτέλεσης του προγράμματος από τη Flash memory στη SRAM.

5.2.2.2.2 Σωρός (Heap)

Η μνήμη στον σωρό χρησιμοποιείται για δυναμικά κατανεμημένα δεδομένα. Όσο κατανέμονται δεδομένα, η μνήμη αυτή αυξάνεται από την κορυφή της περιοχής των στατικών δεδομένων.

5.2.2.2.3 Στοίβα (Stack)

Πρόκειται για τη μνήμη στην οποία αποθηκεύονται οι τοπικές μεταβλητές (local variables) και διατηρούνται οι πληροφορίες σχετικά με τις διακοπές και τις κλήσεις συναρτήσεων. Η στοίβα αυξάνεται από την κορυφή της SRAM προς τον σωρό. Κάθε διακοπή, κλήση συνάρτησης ή δέσμευση χώρου για την αποθήκευση τοπικών μεταβλητών αυξάνει το χώρο της στοίβας. Επιστρέφοντας από μία διακοπή ή συνάρτηση, οι κλήσεις θα ανακτήσουν όλο τον χώρο στη στοίβα που είχε δεσμευτεί αρχικά για την κλήση τους.



Σχήμα 5.3: Περιγραφή SRAM μνήμης

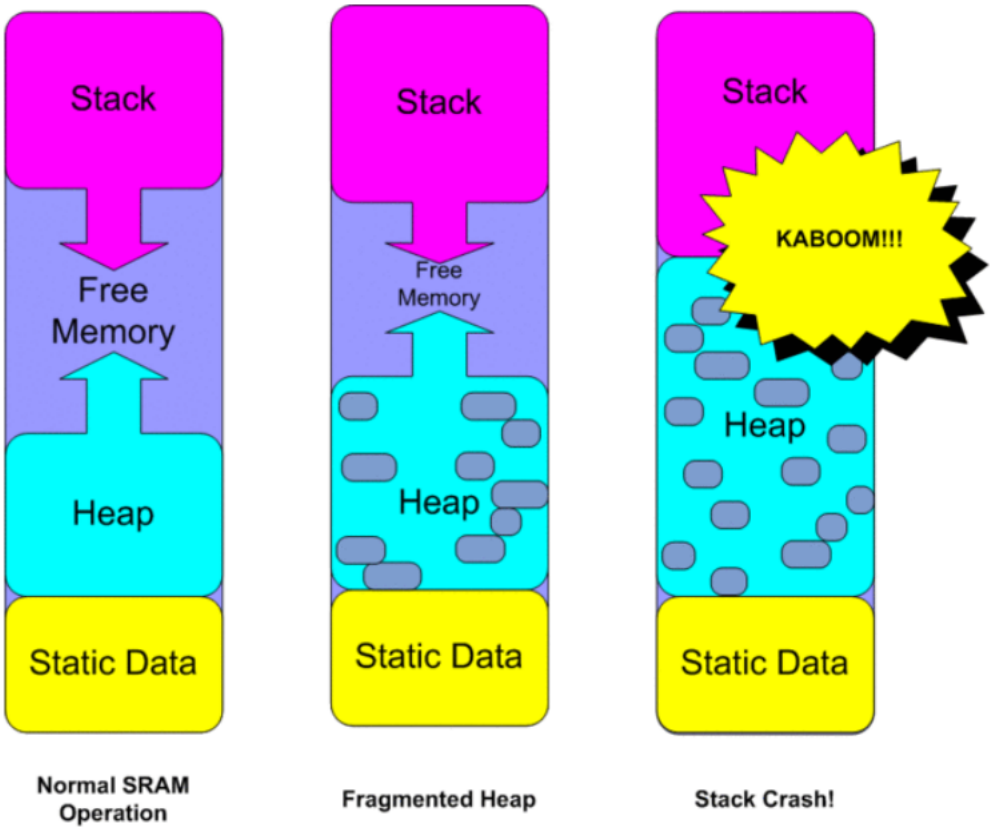
Τα περισσότερα προβλήματα μνήμης εμφανίζονται όταν συγκρούονται η στοίβα και ο σωρός. Όταν συμβεί αυτό, μία ή και οι δύο από αυτές τις περιοχές μνήμης θα καταστραφούν με απρόβλεπτα αποτελέσματα. Σε ορισμένες περιπτώσεις θα προκαλέσει άμεσο τερματισμό λειτουργίας του υπολογιστή. Σε άλλες, τα αποτελέσματα μπορεί να μην παρατηρηθούν μέχρι πολύ αργότερα. Η SRAM μνήμη είναι πιο δυναμική και κατά συνέπεια είναι πιο δύσκολο να μετρηθεί. Ένας τρόπος για να μετρήσουμε την ελεύθερη SRAM στο πρόγραμμα μας είναι καλώντας την παρακάτω freeMemory() συνάρτηση. Με την προσθήκη της συνάρτησης 5.4 στο πρόγραμμα, ο προγραμματιστής μπορεί να την καλέσει από διάφορα σημεία και να ενημερωθεί για τον ελεύθερο χώρο στη SRAM.

```
#ifdef __arm__
// should use uinstd.h to define sbrk but Due causes a conflict
extern "C" char* sbrk(int incr);
#else  // __ARM__
extern char *__brkval;
#endif  // __arm__

int freeMemory() {
  char top;
#ifdef __arm__
  return &top - reinterpret_cast<char*>(sbrk(0));
#elif defined(CORE_TEENSY) || (ARDUINO > 103 && ARDUINO != 151)
  return &top - __brkval;
#else  // __arm__
  return __brkval ? &top - __brkval : &top - __malloc_heap_start;
#endif  // __arm__
}
```

Σχήμα 5.4: Συνάρτηση freeMemory()

Στην ουσία, η παραπάνω συνάρτηση υπολογίζει τον χώρο μεταξύ του σωρού και της στοίβας. Δεν υπολογίζει τον μη δεσμευμένο χώρο που μπορεί να βρίσκεται μέσα στο σωρό. Η στοίβα δεν μπορεί να εκμεταλλευτεί αυτόν τον χώρο και πολλές φορές είναι τόσο κατακερματισμένος που δεν μπορεί να χρησιμοποιηθεί ούτε στον σωρό για επιπλέον δέσμευση χώρου. Ο χώρος μεταξύ του σωρού και της στοίβας είναι αυτός που θα πρέπει να ελέγχεται προκειμένου να αποφευχθούν τα προβλήματα στη στοίβα. Η εικόνα 5.5, αναπαριστά τον τρόπο με τον οποίο δημιουργείται συχνά η έλλειψη μνήμης στη SRAM.



Σχήμα 5.5: Συντριβή της SRAM μνήμης

5.2.2.3 EEPROM

Πρόκειται για μία ακόμα μη-πτητική μνήμη (δε χάνει τα δεδομένα της με τη διακοπή της τροφοδοσίας ρεύματος) που μπορεί να διαβάζεται και να γράφεται από το εκτελούμενο πρόγραμμα. Μπορεί να διαβάζεται μόνο ως byte-by-byte και γι αυτό είναι λίγο άβολη στη χρήση. Επίσης, είναι πιο αργή μνήμη από τη SRAM και έχει περιορισμένο χρόνο ζωής σε 100.000 κύκλους γραφής. Αντίθετα, μπορεί να διαβάζεται απεριόριστες φορές.

5.2.2.4 Σύγκριση Μνήμης διαφορετικών μικροελεγκτών

Το ATmega2560 τσιπ που βρίσκεται στο Arduino Mega περιλαμβάνει τους παρακάτω χώρους μνήμης.

Arduino board	Processor	Flash	SRAM	EEPROM
UNO, Uno Ethernet, Menta, Boardino	Atmega328	32K	2K	1K
Mega, MegaADK	Atmega2560	256K	8K	4K
Leonardo, Micro, Flora, 32U4 Breakout, Teensy, Esplora	Atmega 32U4	32K	2.5K	1K

Πίνακας 5.2: Σύγκριση Μνήμης διαφορετικών μικροελεγκτών

5.2.3 Βελτιστοποίηση μνήμης Arduino

Κατά τη μεταγλώττιση του προγράμματος, το IDE (open-source Arduino Software) μας ενημερώνει τόσο για το μέγεθος του προγράμματος όσο και για τον ελεύθερο χώρο που περισσεύει. Σε περίπτωση που το πρόγραμμα φτάσει ή υπερβεί τον διαθέσιμο χώρο, τότε ορισμένες από τις παρακάτω ενέργειες θα βοηθήσουν για τη μείωση του δεσμευμένου χώρου σε κάθε μία από τις διαφορετικές μνήμες του Arduino. Στο πείραμα μας, προσπαθήσαμε να ενσωματώσουμε όλες τις παρακάτω τεχνικές, προκειμένου να αρκεί η μνήμη του Arduino Mega. Για τον παραπάνω λόγο, εκμεταλλευτήκαμε και τις τρεις διαφορετικές μονάδες μνήμης που διαθέτει. Ακόμα, προτιμήσαμε την χρήση του Memory Card Module για να διατηρήσουμε τη μνήμη σε ένα ασφαλές και επιθυμητό όριο.

5.2.3.1 Βελτιστοποίηση Μνήμης Προγράμματος

Για τη βελτίωση της Flash memory χρησιμοποιήσαμε τις παρακάτω τεχνικές:

1. Αφαίρεση Dead – Code
2. Ενοποίηση Επαναλαμβανόμενου Κώδικα
3. Εξάλειψη του BootLoader

Αφαίρεση Dead – Code

Συνήθως, ο συνολικός κώδικας ενός προγράμματος είναι μια συσσώρευση συναρτήσεων από διαφορετικές πηγές. Για το λόγο αυτό υπάρχουν πιθανότητες η τελική μορφή του κώδικα να μην χρησιμοποιεί όλες τις πληροφορίες που αρχικά είχαμε συλλέξει. Έτσι, κάποιες πληροφορίες μπορούν να εξαλειφθούν έτσι ώστε να εξοικονομήσουμε χώρο στη μνήμη προγράμματος. Ο περιττός κώδικας βρίσκεται συχνά στα παρακάτω σημεία:

- Σε βιβλιοθήκες που συμπεριλαμβάνονται
- Σε συναρτήσεις που έχουμε ορίσει και τελικά δεν καλούνται
- Σε ορισμένες μεταβλητές που δεν χρησιμοποιούνται

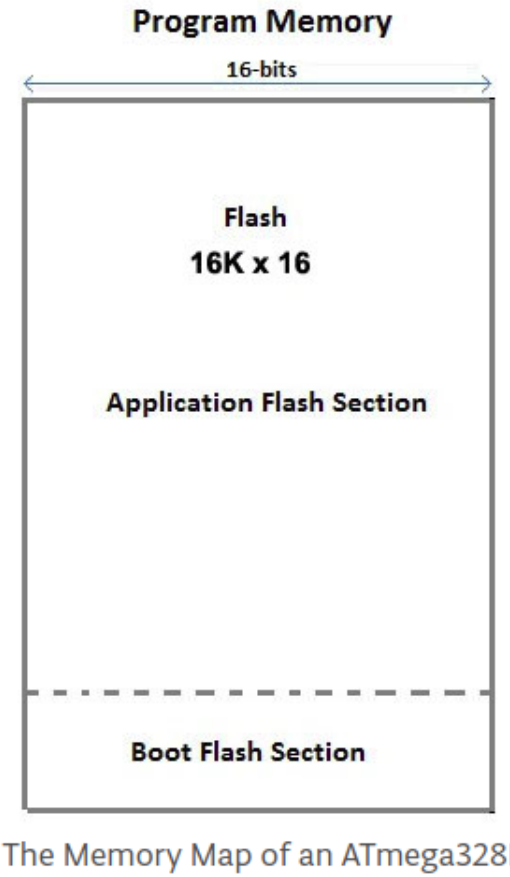
- Σε κομμάτια κώδικα που είναι λογικά αδύνατο να προσπελαστούν, λόγω συνθηκών που δεν είναι ποτέ αληθείς.

Ενοποίηση Επαναλαμβανόμενου Κώδικα

Σε πολλά σημεία του προγράμματος εφαρμόζουμε παρόμοιες λογικές που είναι δυνατό να υλοποιηθούν με ίδιες ακολουθίες εντολών κώδικα. Η δημιουργία μίας γενικής φύσεως συνάρτησης που θα υλοποιεί και τις δύο λειτουργίες και θα καλείται από δύο ή περισσότερα σημεία θα έχει ως αποτέλεσμα την μείωση του δεσμευμένου χώρου στη μνήμη προγράμματος.

Εξάλειψη του BootLoader

Σαν τελευταία επιλογή προτείνεται η διαγραφή του BootLoader σε περίπτωσης που ο χώρος είναι υπερβολικά περιορισμένος (Σχήμα 5.6). Ο Bootloader είναι το κομμάτι κώδικα που κάνει όλες τις Arduino συσκευές να προγραμματίζονται εύκολα μέσω του Arduino Software (IDE). Με την εκμετάλλευση αυτού του κώδικα είναι δυνατή η σύνδεση της συσκευής στον υπολογιστή με USB και με το πάτημα της εντολής *Upload* η έναρξη της διαδικασίας μεταφοράς του sketch στη Flash μνήμη του μικροελεγκτή. Ο κώδικας αυτός εκτελείται κάθε φορά που γίνεται reset (επαναφορά) στον μικροελεγκτή και ψάχνει να φορτώσει κάποιο sketch από τη σειριακή/USB θύρα, χρησιμοποιώντας συγκεκριμένο πρωτόκολλο και ταχύτητα. Στον ATmega2560, αποτελείται από 80KB και βρίσκεται στο τέλος του χώρου μνήμης. Δεν μπορεί να προγραμματιστεί, όπως ο κανονικός κώδικας στο sketch και έχει σχεδιαστεί για τον παραπάνω συγκεκριμένο σκοπό.



Σχήμα 5.6: Περιγραφή BootLoader

Προκειμένου να προγραμματίσουμε τον BootLoader και να παρέχουμε στο μικροελεγκτή τη συμβατότητα που χρειάζεται με το Arduino Software (IDE), θα πρέπει να χρησιμοποιηθεί ο In-circuit Serial Programmer (ISP). Αυτή είναι η συσκευή που συνδέεται σε ένα συγκεκριμένο σύνολο από pins του μικροελεγκτή ώστε να προγραμματίσει όλη τη Flash memory, συμπεριλαμβανομένων των 4KB του BootLoader. Η διαδικασία προγραμματισμού του ISP περιλαμβάνει επίσης τη γραφή ορισμένων διακοπών ασφαλείας, δηλαδή κάποιες διαδικασίες που ορίζουν τη συμπεριφορά του μικροελεγκτή σε συγκεκριμένες καταστάσεις.

5.2.3.2 Βελτιστοποίηση SRAM

Η SRAM είναι το πιο σημαντικό είδος μνήμης στο Arduino. Η ανεπάρκεια της αποτελεί το πιο συχνό πρόβλημα μνήμης του Arduino και είναι δύσκολο να διαγνωσθεί. Κάποιες από τις μεθόδους που χρησιμοποιούνται για τη βέλτιστη διαχείριση της SRAM είναι οι παρακάτω.

- Αφαίρεση μη χρησιμοποιούμενων μεταβλητών
- Χρήση της συνάρτησης F()

- Χρήση της Reserve() συνάρτησης
- Περιορισμός του μεγέθους του Buffer
- Μείωση υπέρ-μεγεθών μεταβλητών
- Χρήση Τοπικών Μεταβλητών

Αφαίρεση μη χρησιμοποιούμενων μεταβλητών

Σε πολλές περιπτώσεις υπάρχουν αναθέσεις μεταβλητών στον κώδικα, οι οποίες δεν χρησιμοποιούνται ουσιαστικά. Έτσι λαμβάνουν περιττό χώρο στη μνήμη.

Χρήση της συνάρτησης F()

Οι συμβολοσειρές (Strings) είναι ένας από τους κύριους λόγους της δραματικής μείωσης της SRAM μνήμης. Αρχικά λαμβάνουν χώρο στη Flash Memory και στη συνέχεια με την εκτέλεση του προγράμματος αντιγράφονται στη SRAM ως στατικές μεταβλητές. Το γεγονός αυτό αποτελεί σημαντική σπατάλη της μνήμης από τη στιγμή που δεν πρόκειται να γράψουμε πάνω στις συμβολοσειρές.
Για παράδειγμα μία εντολή ανάθεσης μεταβλητής όπως:

char message[]="I am studying in National Technical University of Athens."; (5.1)

χρειάζεται 57bytes χώρο στη SRAM (κάθε χαρακτήρας χρειάζεται ένα byte και επιπλέον άλλο ένα απαιτείται από τον *0΄* χαρακτήρα, ο οποίος δηλώνει το τέλος της συμβολοσειράς). Αντίστοιχες αναθέσεις γίνονται συνέχεια όταν στέλνουμε κάποιο κείμενο για να απεικονιστεί σε μία LCD οθόνη. Στην πρότυπη συσκευή παρακολούθησης και βελτίωσης της φαρμακευτικής συμμόρφωσης των ασθενών υπάρχουν ποικίλα μηνύματα ανάλογου μεγέθους που απεικονίζονται στην LCD οθόνη. Υπάρχει μήνυμα ενημέρωσης για την ενεργοποίηση της συσκευής, μήνυμα προειδοποίησης ή ενημέρωσης του ασθενή για τον τελειωμό των χαπιών, μήνυμα δήλωσης εσωτερικής βλάβης της συσκευής κ.α. Ένας τρόπος να μειώσουμε τη δέσμευση μνήμης αυτών των συμβολοσειρών είναι η χρήση της F() macro, η οποία εισήχθη στη Version 1.0 του Arduino IDE. Η F() χαρακτηρίζεται σαν μία μακροεντολή που βρίσκεται στην *Wstring.h* βιβλιοθήκη. Η λειτουργία της βασίζεται στην αποθήκευση των Strings στη Flash Memory και όχι στη SRAM. Συγκεκριμένα, όταν καλείται η F() με μία συμβολοσειρά ως όρισμα, στην ουσία ενημερώνει το μεταγλωττιστή να αφήσει τα δεδομένα αυτά στη μνήμη προγράμματος. Στη συνέχεια, όταν έρθει η ώρα να προσπελάσουμε την εντολή, ένα byte αντιγράφεται μόνο στη RAM. Ωστόσο, υπάρχουν κάποιοι περιορισμοί στη λειτουργία της:

- Η συγκεκριμένη εντολή δεν μπορεί να χρησιμοποιηθεί για δεδομένα που πρόκειται να μεταβληθούν.
- Λειτουργεί μόνο σε Strings
- Δεν ενδείκνυται για μεγάλα μπλοκ κειμένων, όπως HTML σελίδες

- Δεν δέχεται κάποια βελτιστοποίηση, με την έννοια ότι η χρήση της ίδιας συμβολοσειράς σε πολλά σημεία θα καταναλώνει αρκετή μνήμη προγράμματος

Εν κατακλείδι αποτελεί έναν πολύ αποτελεσματικό τρόπο μείωσης του δεσμευμένου χώρου στη SRAM. Κάθε φορά, λοιπόν, που εφαρμόζεται η εντολή *print()* ή η απεικόνιση κάποιου μηνύματος σε LCD οθόνη είναι βέλτιστο να καλούμε την F() macro προηγουμένως.

Χρήση της Reserve() συνάρτησης

Η συνάρτηση Reserve() επιτρέπει τη δέσμευση συγκεκριμένου χώρου στη μνήμη για τον χειρισμό συμβολοσειρών. Η ιδέα της συνάρτησης είναι να αποτραπεί ο κατακερματισμός του σωρού από το string, για να προ-καταχωρηθεί ένα string που αναπτύσσεται. Ορίζεται ως εξής: string.reserve(size) :όπου size: insigned int που δηλώνει τον αριθμό των bytes στη μνήμη που θα δεσμευτούν για τον χειρισμό του συγκεκριμένου string. Έχοντας ήδη δεσμευμένη τη μνήμη, αποτρέπουμε την κλήση της realloc() στις περιπτώσεις που το string θα μεγαλώσει. Συνήθως, πολλά διαφορετικά Strings χρησιμοποιούνται προσωρινά για την τροποποίηση των Strings, με αποτέλεσμα να δεσμεύεται νέος χώρος στο σωρό και να αφήνεται ένα κενό στα σημεία που βρισκόταν αρχικά το string (memory fragmentation). Το μόνο που χρειάζεται για να αποφευχθεί ο κατακερματισμός της μνήμης του σωρού είναι η χρήση της συνάρτησης reserve σε όλα τα String που γνωρίζουμε ότι θα αυξηθεί το μέγεθός τους.

Περιορισμός του μεγέθους του Buffer

1. Buffer και Array Allocation: Σε περίπτωση δέσμευσης χώρου στη μνήμη για κάποιο buffer ή πίνακα , θα πρέπει να βεβαιωνόμαστε ότι ο χώρος αυτός δεν υπερβαίνει τον επιθυμητό.

2. Buffer σε Libraries: Πολλές βιβλιοθήκες επίσης δεσμεύουν προσωρινή μνήμη, η οποία είναι δυνατό να περιοριστεί.

3. System Buffers: Ακόμα ένας buffer που είναι κρυμμένος στο σύστημα είναι ο εσωτερικός buffer σειριακής λήψης δεδομένων 64-bytes. Αν το πρόγραμμα δε λαμβάνει αρκετά σειριακά δεδομένα υψηλής ταχύτητας, τότε είναι εφικτό να μειωθεί το μέγεθος του buffer ακόμα και στο μισό. Το μέγεθος του σειριακού buffer ορίζεται στο αρχείο *HardwareSerial.cpp* και ο ορισμός του μεγέθους ορίζεται στο σημείο: #define SERIAL_BUFFER_SIZE 64 Το σημείο αυτό θα μπορούσε να τροποποιηθεί σε 32 bytes ή και λιγότερο.

Μείωση υπέρ-μεγεθών μεταβλητών

Κάθε τύπος δεδομένων που χρησιμοποιείται απαιτεί διαφορετικό μέγεθος μνήμης. Για παράδειγμα, είναι σπατάλη χώρου να ορίζουμε μία μεταβλητή ως float, ενώ θα μπορούσε να οριστεί ως int. Αντίστοιχα, αν μπορούμε να ορίσουμε μία μεταβλητή ως byte, δεν θα πρέπει

να την ορίζουμε ως int. Στον πίνακα 5.7 παρουσιάζεται ένας πίνακας που ορίζει το μέγεθος κάθε τύπου δεδομένων.

Data Types	Size in Bytes	Can contain:
boolean	1	true (1) or false (0)
char	1	ASCII character or signed value between -128 and 127
unsigned char, byte, uint8_t	1	ASCII character or unsigned value between 0 and 255
int, short	2	signed value between -32,768 and 32,767
unsigned int, word, uint16_t	2	unsigned value between 0 and 65,535
long	4	signed value between -2,147,483,648 and 2,147,483,647
unsigned long, uint32_t	4	unsigned value between 0 and 4,294,967,295
float, double	4	floating point value between -3.4028235E+38 and 3.4028235E+38 (Note that double is the same as a float on this platform.)

Σχήμα 5.7: Μέγεθος των τύπων δεδομένων

Χρήση Τοπικών Μεταβλητών

Οι global και οι static μεταβλητές είναι τα πρώτα δεδομένα που φορτώνονται στη SRAM. Ωθούν προς τα πάνω την αρχή του σωρού προς τη στοίβα και καταλαμβάνουν μόνιμα τον χώρο που θα αποθηκευτούν. Για το λόγο αυτό προτιμάται πάντα η χρήση τοπικών μεταβλητών. Η κλήση κάθε συνάρτησης δημιουργεί ένα πλαίσιο στη στοίβα και την ωθεί να μεγαλώσει προς τον σωρό. Κάθε τέτοιο πλαίσιο περιλαμβάνει:

1. Όλες τις παραμέτρους οι οποίες περνάνε στη συνάρτηση
2. Όλες τις τοπικές μεταβλητές που ορίζονται μέσα στη συνάρτηση

Το πλεονέκτημα των τοπικών μεταβλητών έγκειται στο γεγονός της πλήρης αποδέσμευσης του χώρου που καταλαμβάνουν, εφόσον επιστρέψουμε από τη συνάρτηση στην οποία είναι ορισμένες.

5.3 Μέθοδοι Εξοικονόμησης Ενέργειας της Συσκευής

Υπάρχουν ποικίλα ζητήματα που σχετίζονται με την κατανάλωση ενέργειας στο Arduino, σε ένα AVR σύστημα. Μέσα στον Atmega2560 μικροελεγκτή υπάρχουν διάφορα κυκλώματα που δουλεύουν ταυτόχρονα προκειμένου να μην υπερφορτώνεται ο επεξεργαστής. Κάθε ένα από τα κυκλώματα αυτά καταναλώνει ένα ποσό ενέργειας. Για παράδειγμα, η συνάρτηση analogWrite() του Arduino δεν χρησιμοποιεί τον επεξεργαστή για να μετρήσει τους κύκλους του ρολογιού ώστε να παράγει ένα PWM (pulse-width modulation) σήμα. Αντίθετα χρησιμοποιεί έναν από τους ενσωματωμένους Timers (χρονοδιακόπτες) για τη μέτρηση των παλμών του ρολογιού και στη συνέχεια στέλνει μία αίτηση διακοπής στον επεξεργαστή. Έτσι, ο επεξεργαστής σταματάει τις υπόλοιπες εργασίες και χειρίζεται τη διακοπή, αλλάζοντας την κατάσταση του συγκεκριμένου pin. Με την αποφόρτιση κάποιων εργασιών, ο μικροεπεξεργαστής έχει τη δυνατότητα να χειρίζεται πολλαπλές εργασίες ταυτόχρονα.

5.3.1 Χαρακτηριστικά κυκλώματα ATmega2560

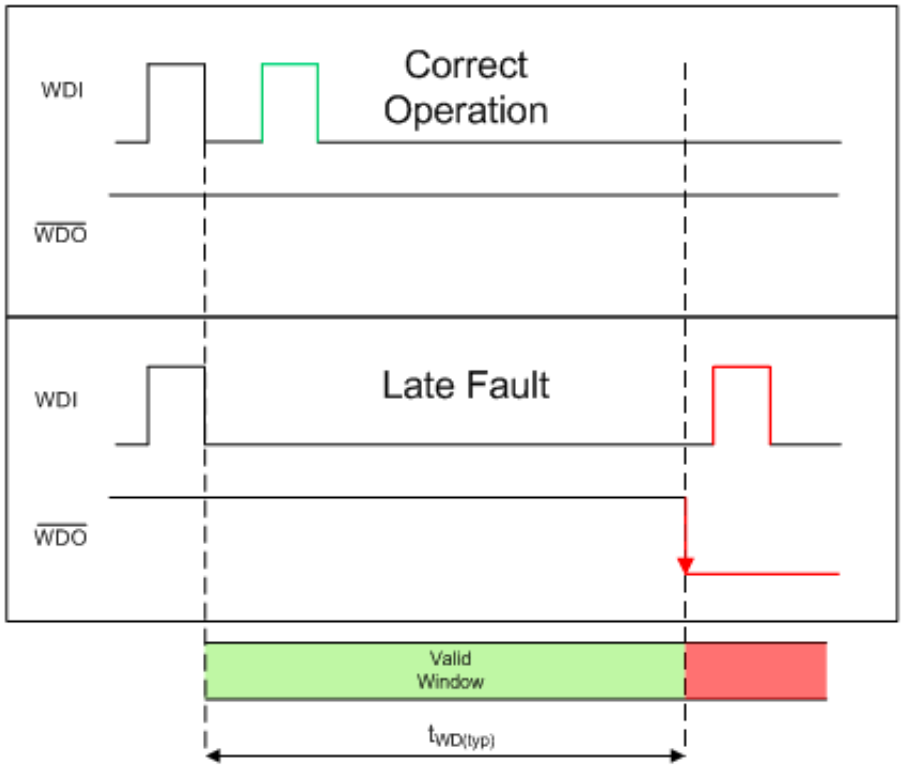
Κάποια από τα κυκλώματα που είναι ενσωματωμένα στον ATmega2560 είναι τα παρακάτω:

- Watchdog Timer
- Brown-out Detector(BOD)
- Analog To Digital Conversion (ADC)
- Two wire Serial Interface (TWI/I2C)
- Universal Synchronous Asynchronous Receiver and Transmitter (USART)
- SPI

5.3.1.1 Watchdog Timer

Πρόκειται για έναν ανεξάρτητο μετρητή, που λειτουργεί με δικό του ρολόι και συχνότητα 128kHz. Ο Watchdog Timer συμβάλει σημαντικά στην παρακολούθηση της λειτουργίας της συσκευής, καθώς ελέγχει το σύστημα για βλάβες και κατ' επέκταση τις αντιμετωπίζει ανάλογα. Ο Watchdog Timer μπορεί να προκαλέσει μία διακοπή όταν έχει λήξει ένα ορισμένο χρονικό διάστημα. Ακόμα έχει τη δυνατότητα να ξυπνήσει τη συσκευή από κάποιες *Sleep Mode* καταστάσεις ή μπορεί απλά να λειτουργεί σαν γενικής φύσεως χρονιστής. Ο μικροεπεξεργαστής στέλνει περιοδικούς παλμούς στην είσοδο του timer, προκειμένου να υποδείξει ότι το σύστημα λειτουργεί σωστά. Αν ο παλμός αυτός δεν ληφθεί μέσα σε ένα χρονικό πλαίσιο (time out), τότε ο Watchdog timer στέλνει σήμα επαναρχικοποίησης (reset) στο Arduino.

Συγκεκριμένα, επαναφέρει το Arduino στην αρχική του λειτουργία κάθε φορά που ο μικροεπεξεργαστής παγώνει. Το χρονικό πλαίσιο που αναμένει ο timer τον παλμό μπορεί να οριστεί από 16ms έως 8s. Στο διάγραμμα 5.8 απεικονίζεται η λειτουργία του Watchdog timer. Στην πρώτη εικόνα ο timer λαμβάνει τον περιοδικό παλμό στην είσοδό του WDI μέσα στο προβλεπόμενο χρονικό πλαίσιο t_{WD} και παράγει ένα θετικό παλμό στην έξοδο $\overline{WDO}$. Αντίθετα, στην δεύτερη εικόνα υπάρχει καθυστέρηση στην έλευση του παλμού, με αποτέλεσμα η έξοδος του timer να αλλάζει στο τέλος του t_{WD} διαστήματος.

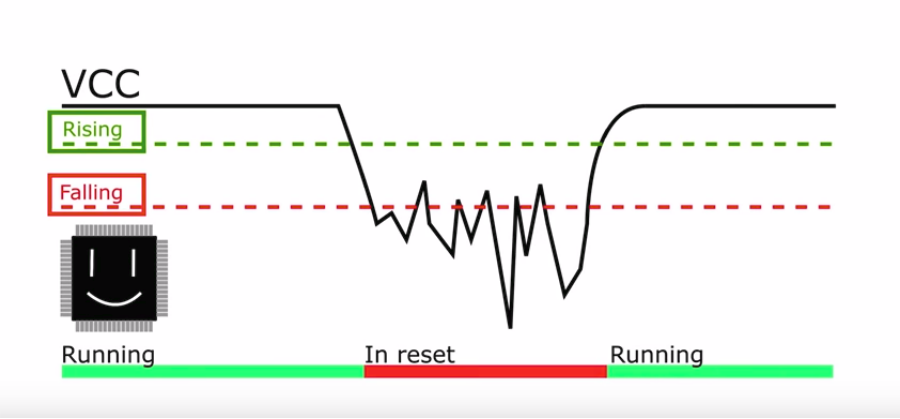


Σχήμα 5.8: Λειτουργία του WatchDog Timer

5.3.1.2 Brown-out Detector(BOD)

Πρόκειται για κύκλωμα ελέγχου της τροφοδοσίας της συσκευής. Οι AVR μικροελεγκτές όπως και ο ATmega2560 διαθέτουν brown-out (χαμηλής τάσης) detectors, με προγραμματισμένα επίπεδα κατωφλίου για την τάση τροφοδοσίας. Όταν η τάση στο V_{cc} pin του Arduino πέσει κάτω από ένα συγκεκριμένο όριο, η λειτουργία του Arduino μπορεί να είναι απρόσμενη. Ο BOD παρακολουθεί, λοιπόν, τις τιμές της τάσεις στην τροφοδοσία της συσκευής και την κάνει *reset* κάθε φορά που η τάση είναι χαμηλή. Αναλυτικότερα, ο BOD περιλαμβάνει δύο thresholds, που ονομάζονται *Rising* και *Falling*. Όταν η τάση στο V_{cc} pin πέσει κάτω από το Falling threshold, τότε το Arduino πέφτει σε λειτουργία *reset*. Αντίστοιχα, όταν η τάση

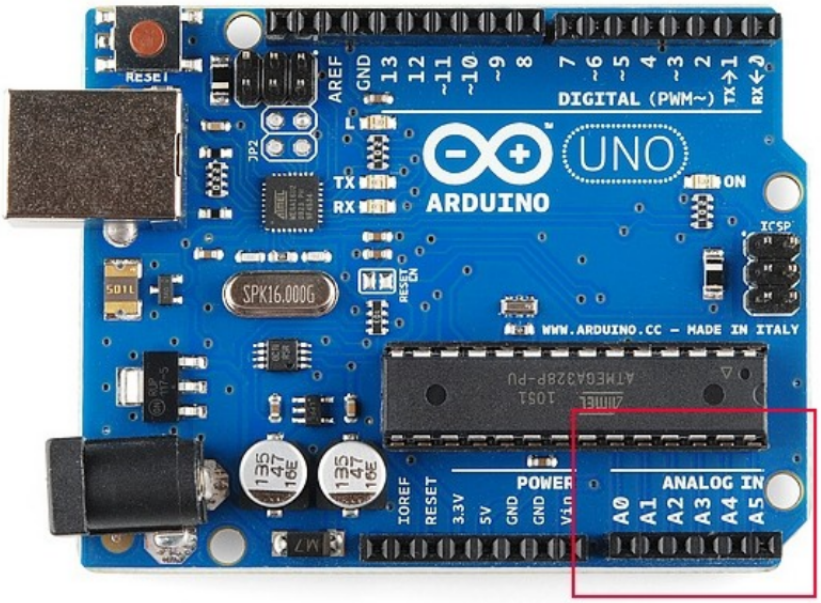
στο V_{cc} pin ανέβει πάνω από το Rising threshold τότε το Arduino συνεχίζει κανονικά τη λειτουργία του. Η λειτουργία αυτή παρουσιάζεται και στο διάγραμμα 5.9.



Σχήμα 5.9: Λειτουργία του Brown-out Detector(BOD)

5.3.1.3 Analog To Digital Conversion (ADC)

Οι μικροεπεξεργαστές είναι ικανοί να ανιχνεύουν δυαδικά σήματα, πχ. είναι το κουμπί πατημένο ή όχι; Αυτά είναι ψηφιακά σήματα. Όταν ο μικροεπεξεργαστής είναι τροφοδοτημένος με 5Volts , αντιλαμβάνεται σαν 0Volts το δυαδικό 0 και 5Volts το δυαδικό 1. Τι γίνεται όμως όταν ένα σήμα έχει τιμή 2.72Volts; Συχνά, χρειάζεται να μετρήσουμε σήματα, η τιμή των οποίων ποικίλει. Τα σήματα αυτά ονομάζονται αναλογικά σήματα. Σχεδόν όλοι οι μικροεπεξεργαστές ενσωματώνουν έναν αντιστροφέα αναλογικών – ψηφιακών σημάτων προκειμένου να είναι εφικτή η μέτρηση αυτών των σημάτων. Όπως αναφέρθηκε παραπάνω, δεν έχουν όλα τα pin του Arduino τη δυνατότητα για αναλογική σε ψηφιακή μετατροπή, αλλά μόνο τα pins A0 – A5 (Σχήμα 5.10). Το *"A"* που έχουν σαν πρόθεμα συμβολίζει τη δυνατότητα για μέτρηση αναλογικών τάσεων.



Σχήμα 5.10: Αναλογικά pins Arduino

Οι ADC (αναλογικοί-ψηφιακοί μετατροπείς) ποικίλουν σε κάθε μικροεπεξεργαστή. Η ADC συσκευή του Arduino περιλαμβάνει έναν 10-bit ADC, δηλαδή έχει την ικανότητα να διακρίνει 1,024 (2^{10}) διαφορετικές αναλογικές τιμές. Κάποιοι μικροεπεξεργαστές έχουν μόνο 8-bit ADC, δηλαδή μπορούν να διακρίνουν έως 256(2^8) διαφορετικές αναλογικές τιμές, ενώ άλλοι έχουν 16-bit ADC, δηλαδή διακρίνουν 65,536(2^{16}) αναλογικές τάσεις.

Από τον ADC προκύπτει μία τιμή από 0 – 5Volts. Θεωρεί ότι τα 5Volts αντιστοιχίζονται στο 1023 και οποιαδήποτε τιμή μικρότερη του πέντε προκύπτει αναλογικά από την παρακάτω εξίσωση.

$$\frac{Ανάλυση\ του\ \text{ADC}}{Τροφοδοσία\ του\ Συστήματος} = \frac{\text{ADC}\ τιμή\ τάσης}{Μετρούμενη\ Αναλογική\ Τάση} \tag{5.2}$$

Συνεπώς, για το 10-bit ADC του Arduino, η εξίσωση απλοποιείται ως εξής:

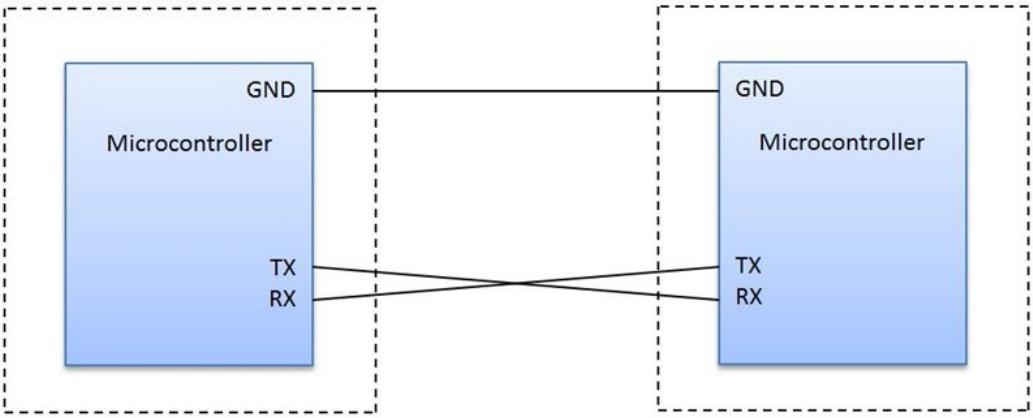
$$\frac{1023}{5} = \frac{\text{ADC}\ τιμή\ τάσης}{Μετρούμενη\ Αναλογική\ Τάση} \tag{5.3}$$

Σε περίπτωση που το σύστημα χρησιμοποιεί 3.Volt αντί για Volt, η εξίσωση προσαρμόζεται αλλάζοντας απλά τα 5Volt με 3.Volt.

5.3.1.4 USART (Universal Synchronous Asynchronous Receiver and Transmitter)

Οι περισσότεροι μικροεπεξεργαστές περιλαμβάνουν έναν ή περισσότερους UART στην πλακέτα τους. Ο ATmega2560 που χρησιμοποιεί το Arduino Mega περιλαμβάνει τέσσερις

USARTs, δηλαδή θύρες που επιτρέπουν τόσο τη σύγχρονη όσο και την ασύγχρονη επικοινία. Αντίθετα, το Arduino UNO περιλαμβάνει μόνο μία USART θύρα. Η λειτουργία τους βασίζεται στη σειριακή επικοινωνία δεδομένων. Έχουν μία γραμμή για τη μετάδοση δεδομένων (TxD: Transmit Data) και μία ξεχωριστή για τη λήψη αυτών (RxD: Receive Data) (Σχήμα 5.11). Πρόκειται για ένα-προς-ένα επικοινωνία, οπότε δεν χρειάζονται γραμμές επιλογής. Υποστηρίζει προκαθορισμένο ρυθμό στη μετάδοση των δεδομένων, ο οποίος θα πρέπει να αρχικοποιηθεί πριν την επικοινωνία, προκειμένου να μην αλλοιωθούν τα δεδομένα στην επικοινωνία. Τέλος, οι USARTs χρησιμοποιούν καταχωρητές ολίσθησης οι οποίοι συνεισφέρουν στη μετατροπή των δεδομένων σε σειριακή ή παράλληλη μορφή. Η χρήση τους προτιμάται όταν τα διαθέσιμα pins για μεταφορά δεδομένων είναι περιορισμένα.



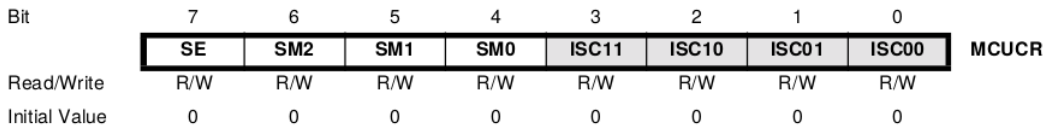
Σχήμα 5.11: Usart επικοινωνία συσκευών

5.3.2 Sleep Modes

Ο ATmega2560 μικροελεγκτής έχει τη δυνατότητα ποικίλων *Sleep modes*, που μπορούν να κληθούν προκειμένου να μειωθεί η κατανάλωση ενέργειας, απενεργοποιώντας τη λειτουργία κάποιων κυκλωμάτων στο Arduino. Τα *Sleep Modes* χρησιμοποιούνται κυρίως για να παραταθεί η λειτουργία της μπαταρίας. Το Arduino Mega μπορεί να προγραμματιστεί στα έξι παρακάτω sleep modes:

1. Idle
2. ADC Noise reduction
3. Power-Down
4. Power-Save
5. Standby
6. Extended Standby

Για να εισαχθεί το Arduino με κατάσταση αδράνειας, θα πρέπει να οριστεί στον MUCR) καταχωρητή το Sleep Mode που θα χρησιμοποιηθεί. Η δομή του καταχωρητή παρουσιάζεται στην εικόνα 5.12.



Σχήμα 5.12: Microcontroller Unit Control Register

Ακόμα, το (Sleep Enable) bit θα πρέπει να βρίσκεται στο λογικό ένα. Συνιστάται η μεταβολή του από μηδέν σε ένα να γίνεται ακριβώς πριν την εκτέλεση της *SLEEP* εντολής και η αναίρεση της ακριβώς μετά την εκτέλεση της εντολής. Η επιλογή του κατάλληλου *SLEEP Mode* ορίζεται από τα bit 4, 5 και 6. Στον πίνακα 5.3 παρουσιάζεται η αρχικοποίηση του καταχωρητή σε κάθε μία από τις καταστάσεις.

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noice Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby
1	1	1	Extended Standby

Πίνακας 5.3: Τρόπος επιλογής Sleep Mode

5.3.2.1 Idle Mode

Όταν τα SM0 – SM2 bits είναι γραμμένα ως 000, τότε η εντολή SLEEP οδηγεί τον μικροελεγκτή να βρεθεί σε *Idle Mode*, σταματώντας τη λειτουργία της CPU (Central Processing Unit) αλλά επιτρέποντας τη λειτουργία των SPI, USART (Universal Synchronous/Asynchronous Receiver/Transmitter), Analog Comparator, ADC, Two-wire Serial Interface (I^2C Interface), Timer/Counters, Watchdog και του συστήματος διακοπών. Αυτού του είδους το sleep mode σταματάει το ρολόι της CPU και της FLASH, ενώ επιτρέπει στα υπόλοιπα ρολόγια να λειτουργούν κανονικά. Η Idle Mode επιτρέπει στην MCU να ξυπνάει τόσο από εξωτερικές διακοπές όσο και από εσωτερικές όπως η υπερχείλιση στον χρονιστή/μετρητή και οι USART διακοπές. Ακόμα, στην κατάσταση αυτή οι ADC μετατροπές ξεκινάνε αυτόματα από τη συσκευή.

5.3.2.2 ADC Noise Reduction Mode

Όταν τα SM0 – SM2 bits είναι γραμμένα ως 001, τότε η εντολή SLEEP οδηγεί τον μικροελεγκτή να βρεθεί σε *ADC Noise Reduction Mode*, σταματώντας τη λειτουργία της CPU (Central Processing Unit) αλλά επιτρέποντας τη λειτουργία των ADC, των εξωτερικών

διακοπών, του Two-wire Serial Interface (I^2C Interface), των Timer/Counters και του Watchdog Timer. Η βασικά της λειτουργία είναι η παύση των $clk_{I/O}$(Βασικό Ρολόι του μικροεπεξαργαστή), clk_{CPU} και clk_{FLASH} και η συνέχιση της λειτουργίας των υπολοίπων ρολογιών. Η κατάσταση αυτή βελτιώνει τους θορύβους στην ADC, προκαλώντας μετρήσεις υψηλότερης ανάλυσης. Όταν η ADC mode είναι ενεργοποιημένη, τότε η ADC μετατροπή ξεκινάει αυτόματα. Η MCU μπορεί να φύγει από την αδράνεια όταν μία από τις επόμενες διακοπές προκληθούν.

- ADC Conversion Complete interrupt
- External Reset
- Watchdog Reset
- Brown-out Reset
- Two-wire Serial Interface Address Match Interrupt
- Timer/Counter2 interrupt
- SPM/EEPROM ready interrupt
- External interrupt on INT7

5.3.2.3 Power-Down Mode

Όταν τα SM0 – SM2 bits είναι καταχωρημένα ως 010, τότε η εντολή SLEEP οδηγεί τον μικροελεγκτή να βρεθεί σε *Power-Down Mode*, σταματώντας τη λειτουργία του Εξωτερικού ταλαντωτή, αλλά επιτρέποντας τη λειτουργία των Two-wire Serial Interface (I^2C Interface) και του Watchdog timer. Η κατάσταση αυτή φέρνει σε παύση όλα τα ρολόγια της συσκευής, ενώ επιτρέπει μόνο ασύγχρονες λειτουργίες. Οι διακοπές, λοιπόν, που ξυπνάνε την MCU από την Power-Down Mode είναι οι παρακάτω:

- External Reset
- Watchdog Reset
- Brown-out Reset
- Two-wire Serial Interface address match interrupt
- External interrupt on INT7
- External interrupt on INT3

Σε περίπτωση που μία από τις παραπάνω διακοπές εφαρμοστούν, θα πρέπει να διατηρηθεί ο παλμός τους για κάποιο χρονικό διάστημα, ώστε η συσκευή να ξυπνήσει από το SLEEP Mode. Όταν συμβαίνει αυτό, υπάρχει μία καθυστέρηση μέχρι να γίνει αποτελεσματική η διακοπή και να επανέλθει το Arduino σε κανονική λειτουργία. Αυτή η διακοπή επιτρέπει την επανεκκίνηση του ρολογιού και την σταθεροποίηση του.

5.3.2.4 Power-Save Mode

Όταν τα SM0 – SM2 bits είναι ίσα με 011, τότε η εντολή SLEEP οδηγεί τον μικροελεγκτή να βρεθεί σε *Power-Save Mode*. Η κατάσταση αυτή είναι παρόμοια με τη Power-Down Mode με μόνο μία διαφορά. Σε περίπτωση που ο Timer/Counter είναι ρυθμισμένος ασύγχρονα, τότε θα μετράει κανονικά κατά τη διάρκεια που η συσκευή θα βρίσκεται σε Sleep Mode. Αντίθετα, αν ο Timer/Counter δεν είναι ασύγχρονα ρυθμισμένος, τότε ενδείκνυται η χρήση της Power-Down Mode, αφού το περιεχόμενο των καταχωρητών στον Ασύγχρονο Timer θα θεωρούνται μη-καθορισμένα. Αυτή η κατάσταση οδηγεί στην παύση όλων των ρολογιών με εξαίρεση του clk_{ASY}, επιτρέποντας μόνο ασύγχρονες λειτουργίες.

5.3.2.5 Standby Mode

Όταν τα SM0 – SM2 bits είναι γραμμένα ως 110 και υπάρχει εξωτερικό κρυσταλλικό ρολόι, τότε η εντολή SLEEP οδηγεί τον μικροελεγκτή να βρεθεί σε *Standby Mode*. Η κατάσταση αυτή είναι πανομοιότυπη με την *Power-Down Mode* με εξαίρεση τον ταλαντωτή που συνεχίζει να λειτουργεί. Η συσκευή ξυπνάει από αυτήν την κατάσταση μετά από έξι κύκλους ρολογιού.

5.3.2.6 Extended Standby Mode

Όταν τα SM0 – SM2 bits είναι γραμμένα ως 110 και υπάρχει εξωτερικό κρυσταλλικό ρολόι, τότε η εντολή SLEEP οδηγεί τον μικροελεγκτή να βρεθεί σε *Extended Standby Mode*. Η κατάσταση αυτή είναι πανομοιότυπη με την Power-save Mode με εξαίρεση τον ταλαντωτή που συνεχίζει να λειτουργεί. Η συσκευή ξυπνάει από αυτήν την κατάσταση μετά από έξι κύκλους ρολογιού. Ο πίνακας 5.4 εμφανίζει συνολικά όλα τα Sleep-Modes καθώς και τις πηγές που προκαλούν τις διακοπές τους.

	Active Clock Domains					Oscillators		Wake-up Sources						
Sleep Mode	clk $_{CPU}$	clk $_{I/O}$	clk $_{FLASH}$	clk $_{ADC}$	clk $_{ASY}$	Main Clock	Timer Oscillator	INT 7	I^2C	Timer 2	eeprom	ADC	WDT	Other I/O
Idle			x	x	x	x	$x^{(2)}$	x	x	x	x	x	x	x
ADC Noise				x	x	$x^{(2)}$	$x^{(2)}$	x	x	x	x	x	x	
Power-down								x	x				x	
Power-save					x		$x^{(2)}$	x	x	x			x	
Standby$^{(1)}$						x		x	x				x	
Extended *Standby*$^{(1)}$						$x^{(2)}$	$x^{(2)}$	x	x	x			x	

Πίνακας 5.4: Τρόπος επιλογής Sleep Mode

Σημειώσεις:

1. Εξωτερικός Κρύσταλλος επιλεγμένος ως ρολόι
2. Ο Timer/Counter είναι ασύγχρονα ρυθμισμένος

Το Arduino μπορεί να ξυπνήσει από τη *Sleep mode* λειτουργία είτε μέσω εξωτερικών σημάτων είτε εσωτερικά, σε κάποιες προκαθορισμένες χρονικές στιγμές. Έτσι, η συσκευή μπορεί να είναι σε λειτουργία όταν καθίσταται αναγκαίο και να είναι σε αδράνεια τις υπόλοιπες στιγμές. Ο ορισμός του είδους του *Sleep mode* γίνεται ανάλογα με το είδος της διακοπής που θα εφαρμοστεί.

5.3.3 Μετρήσεις Κατανάλωσης Ενέργειας

Κάθε μία από τις παραπάνω καταστάσεις συνεχίζει να καταναλώνει ένα ποσό ενέργειας. Στον πίνακα 5.5 παρουσιάζεται η αντιστοιχία των Sleep Modes με το μειωμένο ρεύμα που καταναλώνουν.

Sleep Mode	Current(mA)
SLEEP_MODE_IDLE	15
SLEEP_MODE_ADC	6.5
SLEEP_MODE_PWR_SAVE	1.62
SLEEP_MODE_EXT_STANDBY	1.62
SLEEP_MODE_STANDBY	0.84
SLEEP_MODE_PWR_DOWN	0.36

Πίνακας 5.5: Μετρήσεις Κατανάλωσης Ενέργειας των Sleep Modes

5.3.4 Interrupts (Διακοπές)

Σε γενικές γραμμές, τα περισσότερα προγράμματα μπορούν να εκτελεστούν χωρίς να καταφεύγουν σε διακοπές. Ωστόσο,η χρήση τους έχει ποικίλα πλεονεκτήματα καθώς ξεκαθαρίζει τον κώδικα και προσθέτει μια επιπλέον διάσταση. Τα ISR (Interrupt Service Routine) μπορούν να προσφέρουν τα εξής πλεονεκτήματα:

- Παροχή γρήγορης απόκριση σε εξωτερικές εισόδους και διεπαφή χρήστη.
- Απελευθέρωση του κύριου βρόγχου για βασικές λειτουργίες
- Παροχή ακριβής χρονισμού σε συνδυασμό με τη λειτουργία των Timers/Counters

5.3.4.1 External Interrupts

Η υπολογιστική πλατφόρμα Arduino Mega χρησιμοποιεί μέχρι και έξι εξωτερικά interrupts, τα INT0-INT5, τα οποία εκ κατασκευής είναι ορισμένα στις θύρες 2,3,21,20,19 και 18 αντίστοιχα. Η λειτουργία διακοπής θα ενεργοποιηθεί ανεξάρτητα από τον ορισμό των pins ως

εισόδους ή εξόδους. Η εξωτερική διακοπή είναι ένα σήμα που δέχεται ο μικροελεγκτής οποιαδήποτε χρονικά στιγμή, για ένα γεγονός το οποίο χρειάζεται άμεση προσοχή. Χαρακτηρίζονται ως ασύγχρονες διακοπές και μπορούν να επαναφέρουν τη συσκευή από όλα τα Sleep Modes. Όταν ένα interrupt ενεργοποιηθεί, ο μικροελεγκτής σταματά οποιαδήποτε εργασία πράττει και εκτελεί την ρουτίνα της διακοπής (Interrupt Service Routine). Μετά την εκτέλεση της ρουτίνας, το πρόγραμμα συνεχίζει από την εντολή στην οποία προέκυψε η διακοπή. Η λειτουργία αυτή είναι πολύ χρήσιμη καθώς μας επιτρέπει να εκτελούμε μια συνάρτηση άμεσα και ταχύτατα. Για να χρησιμοποιήσουμε μια εξωτερική διακοπή αρκεί να χρησιμοποιήσουμε τη συνάρτηση *attachInterrupt* στην οποία πρέπει να δηλώσουμε τον αριθμό της διακοπής(INT0...INT5), τη συνάρτηση που θα υλοποιήσει και τον τρόπο λειτουργίας της, δηλαδή:

attachInterrupt (interrupt, functionName, mode) (5.4)

Προφανώς στη θέση του *interrupt* επιλέγουμε ανάμεσα από τα INT0 με INT5. Η ρουτίνα του interrupt καλείται κάθε φορά που αυτό ενεργοποιείται , δεν χρειάζεται κάποια παράμετρο και δεν επιστρέφει τίποτα. Στη θέση του function_name γράφουμε το όνομα της συνάρτησης της διακοπής, ενώ όσον αφορά τον τρόπο λειτουργίας (mode) του interrupt υπάρχουν τέσσερις επιλογές οι οποίες είναι:

LOW Η διακοπή ενεργοποιείται όταν στη θύρα που είναι συνδεδεμένη προκύψει χαμηλό σήμα 0Volt.

CHANGE Η διακοπή ενεργοποιείται όταν στη θύρα που είναι συνδεδεμένη υπάρξει οποιαδήποτε μεταβολή στο σήμα.

RISING Η διακοπή ενεργοποιείται όταν στη θύρα που είναι συνδεδεμένη η διακοπή υπάρξει σήμα που μετατρέπεται από LOW σε HIGH.

FALLING Η διακοπή ενεργοποιείται όταν στη θύρα που είναι συνδεδεμένη η διακοπή υπάρξει σήμα που μετατρέπεται από HIGH σε LOW.

Τέλος, μέσα στη συνάρτηση του interrupt η συνάρτηση delay() δεν λειτουργεί, όπως και οι τιμές της εντολής millis() δεν αυξάνονται. Οι μεταβλητές που χρησιμοποιούνται μέσα και έξω από τη συνάρτηση της διακοπής πρέπει να είναι μορφής μη-πτητικές.

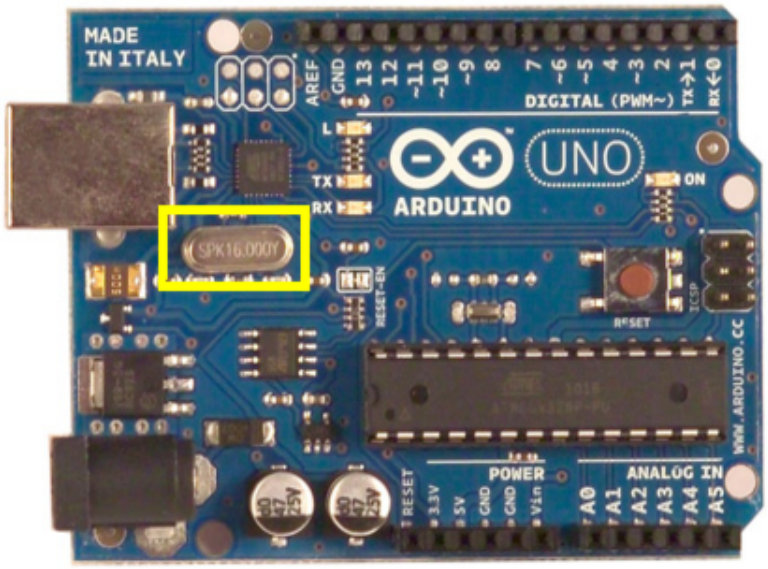
Οι θύρες που χρησιμοποιούνται για τις εξωτερικές διακοπές ποικίλουν ανάλογα με την Arduino πλακέτα που χρησιμοποιείται. Στον πίνακα 5.6 παρουσιάζονται τα interrupt pins σε κάθε συσκευή.

Πλακέτα	Ψηφιακές Θύρες που χρησιμοποιούνται από τις διακοπές
UNO, Nano, Mini, κα.	2,3
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21
Micro, Leonardo	0, 1, 2, 3, 7
Zero	Όλες οι ψηφιακές θύρες εκτός της 4
MKR1000 Rev.1	0, 1, 4, 5, 6, 7, 8, 9, A1, A2
Due	Όλες οι ψηφιακές θύρες

Πίνακας 5.6: Interrupt θύρες στις Arduino πλακέτες

5.3.4.2 Timer Interrupts

Στο Arduino υπάρχει ενσωματωμένος ένας κρύσταλλος 16MHz, που ξεκινάει αυτόματα να ταλαντώνεται κάθε φορά που το Arduino είναι ενεργοποιημένο (Σχήμα 5.13). Όλες οι συναρτήσεις που σχετίζονται με τον χρόνο χρησιμοποιούν τις μετρήσεις αυτού του κρυστάλλου. Για παράδειγμα, η συνάρτηση millis() μπορεί να μετρήσει μέχρι και πενήντα ημέρες και μετά θα κάνει *reset* και θα επαναλάβει τη μέτρηση από την αρχή. Ακόμα, η micros() μπορεί να μετρήσει έως και εβδομήντα λεπτά.



Σχήμα 5.13: 16MHz Κρύσταλλος Arduino

Ο Atmega2560 περιλαμβάνει έξι timers, που καλούνται timer0...timer5. Ο timer0 και ο timer2 αποτελούνται από 8bit, ενώ οι υπόλοιποι από 16bit. Όλοι οι timers περιλαμβάνουν έναν μετρητή, ο οποίος αυξάνεται με κάθε παλμό του ρολογιού. Όταν ο μετρητής φτάσει μια συγκεκριμένη τιμή τότε θα μηδενιστεί και θα ξεκινήσει πάλι από την αρχή. Σύμφωνα με τον αριθμό των bits, οι timers0 και timers2 δέχονται μέχρι 256 τιμές, ενώ οι υπόλοιποι έχουν

μεγαλύτερη ανάλυση αφού μετράνε έως και 65536 διαφορετικές τιμές. Συναρτήσεις όπως οι delay() και millis() λειτουργούν με τη βοήθεια του Timer0. Αντίστοιχα, η συνάρτηση tone() χρησιμοποιεί τον Timer2. Η λειτουργία τους επεκτείνεται στον έλεγχο της περιόδου των PWM παλμών, καθώς και τον χειρισμό των διακοπών. Ο έλεγχος των Timer Interrupts γίνεται μέσω της επιλογής της τιμής στην οποία θα κάνει *reset* ο timer (Compare Match Value) και του καθορισμού της ταχύτητας μέτρησης του μετρητή. Με τον τρόπο αυτό μπορεί να ελέγχεται η συχνότητα των Timer Interrupts.

Αρχικά, θα αναλύσουμε την ταχύτητα με την οποία αυξάνεται ο μετρητής. Το ρολόι του Arduino έχει συχνότητα 16MHz, άρα αυτή είναι η υψηλότερη ταχύτητα που μπορεί να λειτουργήσει ένας μετρητής. Στα 16MHz κάθε παλμός αναπαριστά χρόνο ίσο με $\frac{1}{16000000}$ του δευτερολέπτου, δηλαδή $\sim 63ns$. Υπάρχουν καταστάσεις στις οποίες αυτή η μέτρηση είναι υπερβολικά γρήγορη, ειδικά για τους Timer0 και Timer2 που είναι 8-bit καταχωρητές αφού θα πρέπει να κάνουν *reset* κάθε $\frac{256}{16000000}$ δευτερόλεπτα στην καλύτερη περίπτωση που η Compare Match Value έχει οριστεί στο μέγιστο, δηλαδή στο 255. Ακόμα και για τους Timers 1,3,4,5, που είναι 16-bit καταχωρητής, ο μέγιστος χρόνος μέτρησης θα είναι $\frac{65536}{16000000} \cong 4$ms. Για την επίλυση του προβλήματος χρησιμοποιείται ο λεγόμενος prescaler, ο οποίος προσαρμόζει την ταχύτητα του μετρητή ως εξής:

$$Ταχύτητα\ του\ μετρητή\ (\text{Hz}) = \frac{Ταχύτητα\ του\ ρολογιού\ του\ \text{Arduino(16MHz)}}{\text{prescaler}} \quad (5.5)$$

Οπότε, ο 1-prescaler θα αυξάνει την ταχύτητα του μετρητή στα 16MHz, ο 8-prescaler θα αυξάνει τον μετρητή με ταχύτητα 2MHz κτλ π. Οι διαθέσιμες ταχύτητες του μετρητή φαίνονται στον πίνακα 5.7.

Prescler	Clock Frequency (Hz)
1	16000000
8	2000000
64	250000
256	62500
1024	15625

Πίνακας 5.7: Ταχύτητες των Timers του Arduino

Λαμβάνοντας υπόψη όλα τα παραπάνω, μπορούμε πλέον να επιτύχουμε την επιθυμητή συχνότητα διακοπών με τη χρήση της παρακάτω συνάρτησης:

$$Συχνότητα\ διακοπών\ (\text{Hz}) = \frac{Ταχύτητα\ ρολογιού\ \text{Arduino (16MHz)}}{(\text{prescaler} * (\text{compare match register} + 1))} \quad (5.6)$$

Αντιστρέφοντας λοιπόν την συνάρτηση υπολογίζουμε την τιμή του Compare Match Register ως εξής:

$$\text{Compare Match Register} = [\frac{Ταχύτητα\ ρολογιού\ \text{Arduino (16MHz)}}{(\text{prescaler} * επιθυμητή\ συχνότητα\ διακοπών\ (\text{Hz}))}] - 1 \quad (5.7)$$

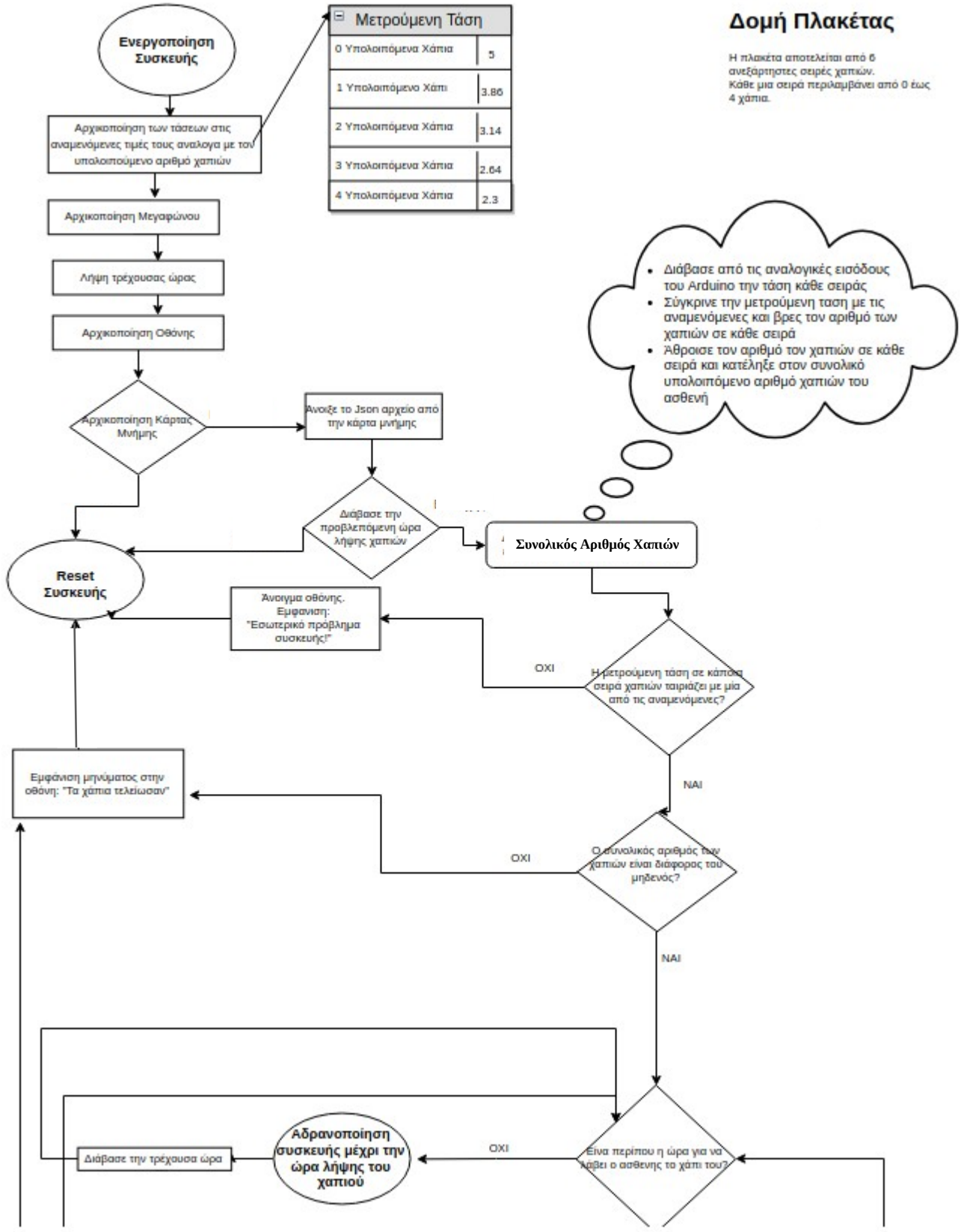
Με την παραπάνω μεθοδολογία, λοιπόν, μπορούμε να ορίσουμε τη συχνότητα των Χρονικών Διακοπών που επιθυμούμε.

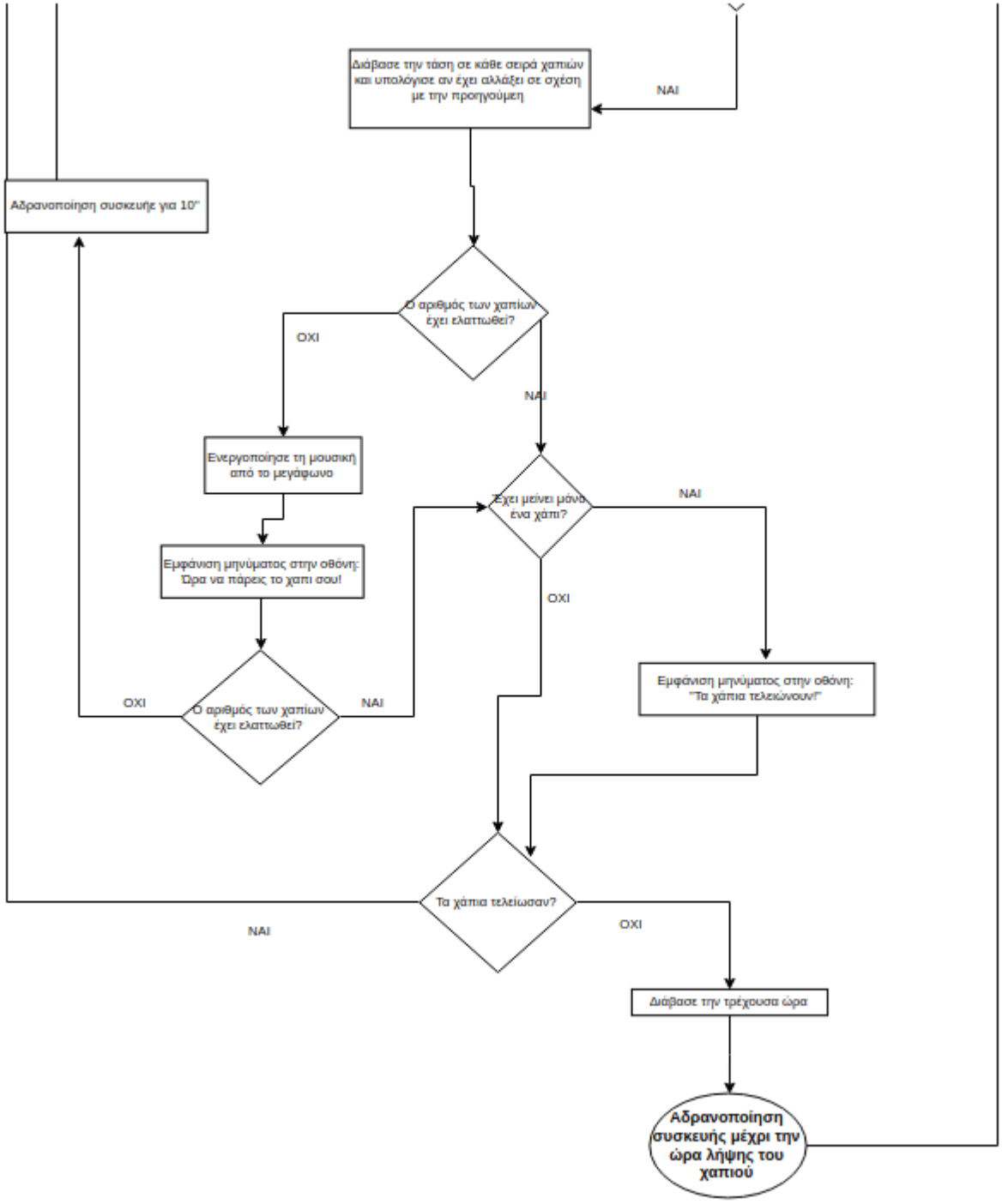
Κεφάλαιο 6

Σχεδιασμός της Πρότυπης Συσκευής

6.1 Εισαγωγή

Η λειτουργία της πλακέτας συνοψίζεται στο παρακάτω σχήμα 6.1.


Ενεργοποίηση Συσκευής
Αρχικοποίηση των τάσεων στις αναμενόμενες τιμές τους αναλογα με τον υπολοιπούμενο αριθμό χαπιών
Μετρούμενη Τάση
0 Υπολοιπόμενα Χάπια 5
1 Υπολοιπόμενο Χάπι 3.86
2 Υπολοιπόμενα Χάπια 3.14
3 Υπολοιπόμενα Χάπια 2.64
4 Υπολοιπόμενα Χάπια 2.3
Δομή Πλακέτας
Η πλακέτα αποτελείται από 6 ανεξάρτητες σειρές χαπιών.
Κάθε μια σειρά περιλαμβάνει από 0 έως 4 χάπια.
Αρχικοποίηση Μεγαφώνου
Λήψη τρέχουσας ώρας
Αρχικοποίηση Οθόνης
Αρχικοποίηση Κάρτας Μνήμης
Άνοιξε το Json αρχείο από την κάρτα μνήμης
Διάβασε την προβλεπόμενη ώρα λήψης χαπιών
Διαβασε από τις αναλογικές εισόδους του Arduino την τάση κάθε σειράς
Σύγκρινε την μετρούμενη ταση με τις αναμενόμενες και βρες τον αριθμό των χαπιών σε κάθε σειρά
Άθροισε τον αριθμό των χαπιών σε κάθε σειρά και κατέληξε στον συνολικό υπολοιπόμενο αριθμό χαπιών του ασθενή
Συνολικός Αριθμός Χαπιών
Reset Συσκευής
Άνοιγμα οθόνης. Εμφανιση: "Εσωτερικό πρόβλημα συσκευής!"
ΟΧΙ
Η μετρούμενη τάση σε κάποια σειρά χαπιών ταιριάζει με μία από τις αναμενόμενες?
ΝΑΙ
Εμφάνιση μηνύματος στην οθόνη: "Τα χάπια τελείωσαν"
ΟΧΙ
Ο συνολικός αριθμός των χαπιών είναι διάφορος του μηδενός?
ΝΑΙ
Διάβασε την τρέχουσα ώρα
Αδρανοποίηση συσκευής μέχρι την ώρα λήψης του χαπιού
ΟΧΙ
Είναι περίπου η ώρα για να λάβει ο ασθενης το χάπι του?



Σχήμα 6.1: Διάγραμμα Ροής

Όταν το Arduino συνδεθεί με τροφοδοσία θα προχωρήσει σε κάποιες βασικές ενέργειες. Αρχικά θα έρθουν σε λειτουργία όλες οι περιφερειακές συσκευές και θα αρχικοποιηθεί η συσκευή. Στη συνέχεια θα εισέλθει σε ένα βρόγχο μέχρι τη αποσύνδεσή της ή την παρουσίαση κάποιου εσωτερικού προβλήματος. Η ανάλυση, λοιπόν, του διαγράμματος ροής θα χωριστεί σε δύο μέρη:

1. Στάδιο Αρχικοποιήσεων
2. Στάδιο Αφύπνισης

6.2 Στάδιο Αρχικοποιήσεων

Αρχικά, με την ενεργοποίηση της συσκευής θα ενεργοποιηθεί η οθόνη της με το μήνυμα *Initialization*. Στη συνέχεια, θα αποθηκευτούν οι επιτρεπτές τιμές τάσης της κάθε σειράς φαρμάκων του blister. Όπως έχει αναλυθεί και στο κεφάλαιο 4, το blister αποτελείται από έξι πανομοιότυπες σειρές χαπιών, όπου κάθε μία περιλαμβάνει το πολύ τέσσερα χάπια. Έτσι, ανάλογα με τη τιμή τάσης που μετριέται μπορεί απευθείας να προσεγγίζεται ο αριθμός χαπιών της συγκεκριμένης σειράς. Με έξι, λοιπόν, διαδοχικές μετρήσεις του blister είναι δυνατό να εξαχθεί ο συνολικός υπολειπόμενος αριθμός χαπιών στη συσκευή. Στη συνέχεια, αρχικοποιείται το ηχείο, εννοώντας ότι αντιστοιχίζεται μία ψηφιακή θύρα του Arduino στο Module του Ηχείου.

Ακόμα, ενεργοποιείται η σειριακή επικοινωνία του Arduino με το Real Time Clock (RTC) για να ενημερωθεί η συσκευή με τη τρέχουσα ώρα, καθώς το RTC εξάρτημα λειτουργεί ανεξάρτητα από την υπόλοιπη συσκευή και μπορεί να μετράει ανελλιπώς το χρόνο.

Επιπλέον, θα πρέπει να εισαχθούν στη συσκευή οι συγκεκριμένες ώρες που ο ασθενής θα πρέπει να λάβει τα χάπια του. Οι ώρες αυτές έχουν εισαχθεί από το φαρμακοποιό κατά την αγορά της συσκευής από τον ασθενή. Συγκεκριμένα, ο φαρμακοποιός έχοντας πρόσβαση σε μία ιστοσελίδα θα μπορεί να εισάγει μία νέα συσκευή μαζί με τις αντίστοιχες ώρες λήψης των φαρμάκων. Τα δεδομένα αυτά θα αντιστοιχίζονται σε μία MongoDB βάσης δεδομένων μέσω μιας node.js εφαρμογής. Προκειμένου να προσομοιώσουμε την άμεση επικοινωνία του Arduino με τη MongoDb, θα αποθηκεύσουμε τα παραπάνω δεδομένα στην κάρτα μνήμης σε Json μορφή, καθώς η MongoDb είναι nonSQL βάση δεδομένων. Επομένως, ενεργοποιούμε την περιφερειακή συσκευή της κάρτας μνήμης (SD Card Module). Αν είναι επιτυχής η ενεργοποίηση της SD, διαβάζουμε το Json αρχείο και ανακτούμε τις απαιτούμενες πληροφορίες σχετικά με τις προβλεπόμενες ώρες λήψης των χαπιών από τον ασθενή. Διαφορετικά, γίνεται αυτόματη επαναρχικοποίηση της συσκευής (reset).

Τελευταίο βήμα στην αρχικοποίηση της συσκευής είναι η ενημέρωσή της για τον τρέχον αριθμό χαπιών που διαθέτει το blister. Για να το πετύχει αυτό, το Arduino διαβάζει από τις έξι αναλογικές εισόδους του την τάση στις ανεξάρτητες σειρές χαπιών του blister. Για κάθε μία από αυτές, συγκρίνει τη μετρούμενη τάση με τις αναμενόμενες και βάσει αυτών υπολογίζει τον αριθμό τον χαπιών σε κάθε μία από τις έξι σειρές. Σε περίπτωση που κάποια από τις μετρήσεις δεν συνάδει με τις αποδεκτές τιμές, η συσκευή εμφανίζει μήνυμα λάθους στην LCD

οθόνη ("Εσωτερικό Πρόβλημα Συσκευής!") και κάνει επανεκκίνηση. Διαφορετικά, αθροίζονται τα αποτελέσματα που έχουν εξαχθεί για κάθε μία από τις έξι σειρές και υπολογίζεται ο πραγματικός αριθμός υπολειπόμενων χαπιών του ασθενή σε πραγματικό χρόνο.

6.3 Στάδιο Αφύπνισης

Το στάδιο αφύπνισης αποτελεί στην ουσία ένα συνεχόμενο επαναλαμβανόμενο βρόγχο που ενεργοποιείται τις στιγμές λήψης χαπιών από τους ασθενείς. Αρχικά, η συσκευή επαναεκκινείται κάθε 24 ώρες, προκειμένου να αποφευχθεί η διατήρηση τυχόν λανθασμένων μετρήσεων. Ακόμα, επαληθεύεται ότι ο αριθμός των χαπιών που υπολογίσαμε προηγουμένως είναι διάφορος του μηδενός. Διαφορετικά, η συσκευή θα αδρανοποιηθεί. Όταν ο ασθενής ανανεώσει τη συσκευασία, τότε θα πατήσει το κουμπί επανεκκίνησης (reset) που βρίσκεται πάνω στη συσκευή και αυτή θα εφαρμόσει πάλι τις αρχικοποιήσεις του πρώτου σταδίου.

Στη συνέχεια, η συσκευή θα πρέπει να ειδοποιεί τον ασθενή την ώρα λήψης της φαρμακευτικής του αγωγής. Το Arduino επεξεργάζεται όλες τις πληροφορίες που έχει συλλέξει και ελέγχει αν η προβλεπόμενη ώρα έχει φτάσει. Σε περίπτωση που δεν έχει φτάσει η κατάλληλη ώρα αδρανοποιείται μέχρι τότε. Διαφορετικά, μετράει τη τάση σε κάθε σειρά χαπιών του blister και ελέγχει αν ο ασθενής έχει ήδη λάβει το χάπι του. Τη διαδικασία αυτή την επιτυγχάνει εξισώνοντας τις αναμενόμενες τιμές με τις μετρούμενες και εξάγοντας τη διαφορά των χαπιών. Αν λοιπόν ο ασθενής έχει προλάβει να λάβει τη φαρμακευτική του αγωγή τότε η διαφορά αυτή θα είναι θετική. Στην περίπτωση αυτή, η LCD οθόνη θα εμφανίσει μήνυμα επιβράβευσης στον ασθενή και θα αδρανοποιηθεί μέχρι την ώρα λήψης του επόμενου χαπιού. Αντίθετα, αν η διαφορά είναι ίση με μηδέν, τότε η συσκευή θα πρέπει να ειδοποιήσει τον ασθενή. Αυτό θα επιτευχθεί με τη χρήση οπτικοακουστικού υλικού, δηλαδή μέσω της LCD οθόνης και του ηχείου. Η διαδικασία αυτή θα επαναλαμβάνεται κάθε δέκα δευτερόλεπτα, αναμένοντας από τον ασθενή να ειδοποιηθεί και να λάβει τη φαρμακευτική αγωγή του.

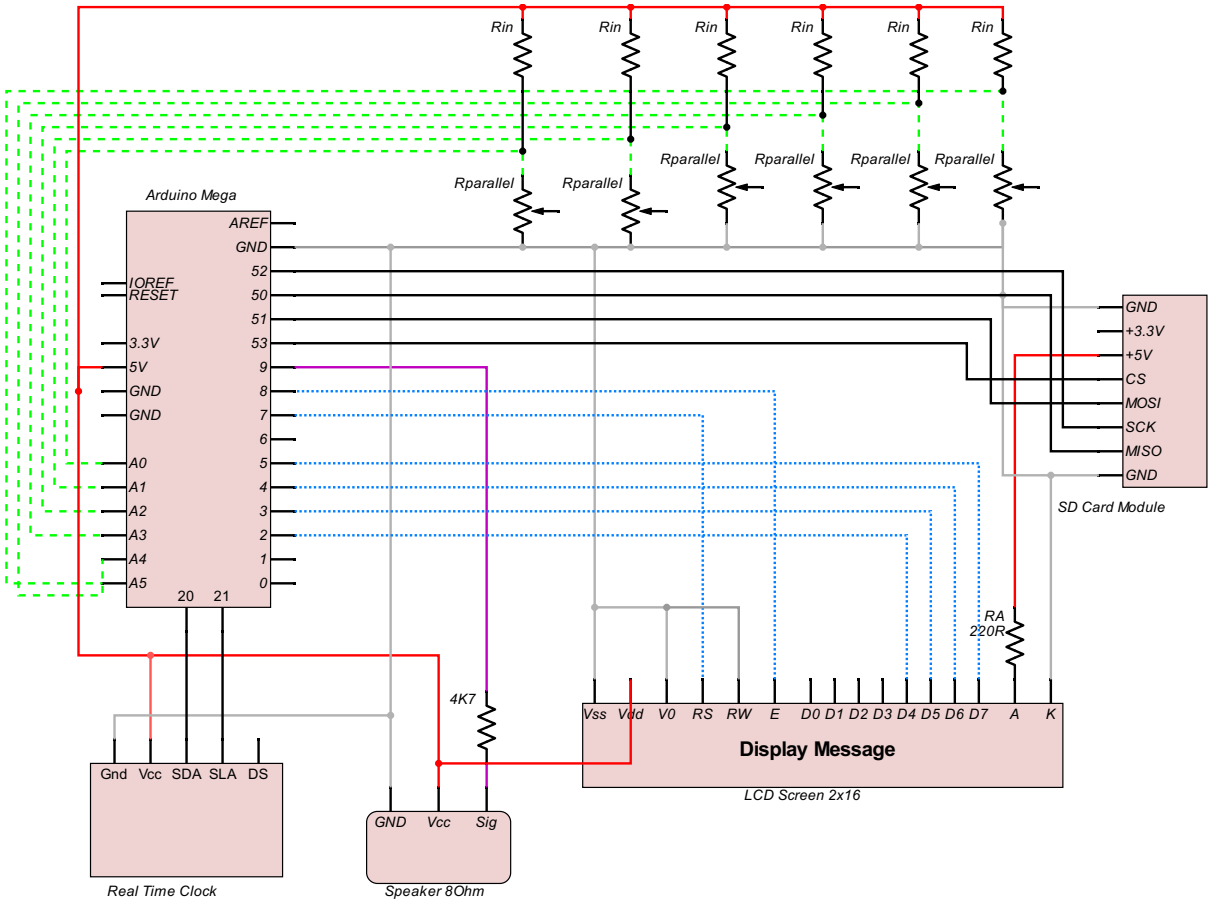
Όπως αναλύθηκε και παραπάνω, η συσκευή βρίσκεται σε αδράνεια (Sleep Mode) τις ώρες που δεν είναι προγραμματισμένη η λήψη χαπιών από τον ασθενή. Αυτό την καθιστά πιο οικονομική, χωρίς να επιβαρύνει την μπαταρία της περαιτέρω.

Κεφάλαιο 7

Τεχνική Υλοποίηση της Συσκευής

7.1 Εισαγωγή

Η υλοποίηση της συσκευής συμπεριλαμβάνει ποικίλες συνδέσεις μεταξύ της βασικής πλακέτας Arduino και των περιφερειακών κυκλωμάτων. Οι ακριβείς συνδέσεις των διαφόρων θυρών παρουσιάζονται στο σχήμα 7.1.



Σχήμα 7.1: Ηλεκτρονικό Κύκλωμα της Συσκευής

7.2 Εσωτερική Συνδεσμολογία Συσκευής

Όπως παρουσιάζεται στο διάγραμμα 7.1, η κάθε περιφερειακή συσκευή συνδέεται με συγκεκριμένες θύρες του Arduino. Η συνδεσμολογία εξαρτάται κυρίως από το είδος της σειριακής επικοινωνίας που έχει κάθε συσκευή με το Arduino. Παρακάτω αναλύεται η λογική της συνδεσμολογίας της κάθε περιφερειακής συσκευής.

- Το SD card module συνδέεται χρησιμοποιώντας το SPI πρωτόκολλο με το Arduino Mega. Για το λόγο αυτό, τα pins του Arduino που χρησιμοποιούνται είναι τα: 50(MISO), 51(MOSI), 52(SCK), 53(SS).
- Το Real Time Clock (CLK) συνδέεται σειριακά με το Arduino μέσω του I^2C πρωτοκόλλου. Όπως έχει αναλυθεί και στο κεφάλαιο 4, το Arduino Mega συνδέεται με στα pins 20 και 21 με το RTC.
- Το ηχείο της συσκευής συνδέεται αυθαίρετα με το pin9 του Arduino. Η συγκεκριμένη επιλογή θύρας μπορεί να μεταβληθεί ελεύθερα, με την αντίστοιχη προσαρμογή στον κώδικα.
- Η lcd screen μπορεί να λειτουργήσει αντίστοιχα και με διαφορετική συνδεσμολογία. Τα pins που επιλέξαμε στο πείραμα μας δεν είναι δεσμευτικά.
- Οι τάσεις που μετρούνται στο blister είναι αναλογικές. Συνεπώς, για τον έλεγχο τους θα μπορούσαν να χρησιμοποιηθούν οποιεσδήποτε από τις 16 αναλογικές θύρες του Arduino Mega.

7.3 Εύρεση Ιδανικής Αντίστασης Χαπιών στο Blister

Σημαντικό στοιχείο της συσκευής αποτελούν τόσο οι αντιστάσεις εισόδου του blister όσο και οι αντιστάσεις των ίδιων των χαπιών. Το κάθε χάπι που εισάγεται στη συσκευή αντιστοιχίζεται σε μια αντίσταση. Η συσκευή θα περιλαμβάνει 24 χάπια. Η βασική της λογική βασίζεται στη διαφορά τάσης που προκύπτει από την αφαίρεση ενός χαπιού. Η διαφορά αυτή αποτελεί στην ουσία τη μέγιστη διακριτική ικανότητα του Arduino. Για την επίτευξη της μέγιστης διακριτικής ικανότητας υλοποιήσαμε στο blister ένα κύκλωμα διαιρέτη τάσης, υπολογίζοντας τις κατάλληλες αντιστάσεις – χάπια.

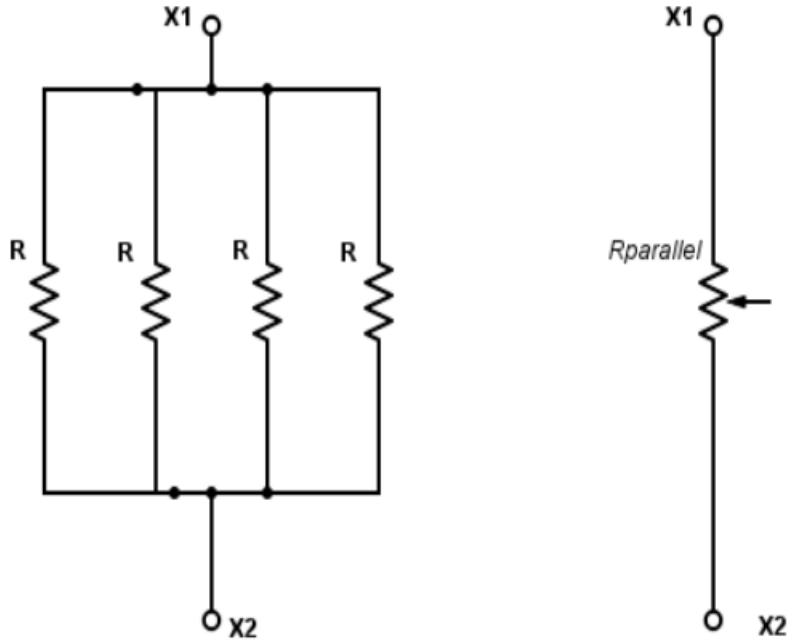
Το κύκλωμα του διαιρέτη τάσης συμπεριλαμβάνει μία πηγή τάσης(V_{in}) και δύο αντιστάσεις σε σειρά (R_1 και R_2). Η μετρούμενη τάση είναι αυτή που βρίσκεται στον κόμβο μεταξύ των δύο αντιστάσεων και συμβολίζεται ως V_{out}.

Σχήμα 7.2: Βασικό κύκλωμα Διαιρέτη Τάσης

Στην εξειδικευμένη συσκευή παρακολούθησης και βελτίωσης της φαρμακευτικής συμμόρφωσης ορίζονται οι παράμετροι της 7.2 ως εξής:

- V_{in}: τροφοδοσία του Arduino που ισούται με 5Volts.
- V_{out}: μέτρηση της τάσης σε κάθε σειρά φαρμάκων, από την οποία επάγεται και ο υπολειπόμενος αριθμός φαρμάκων.
- Η R_1 αποτελεί την αντίσταση εισόδου που εξυπηρετεί την αύξηση της διακριτικής ικανότητας του Arduino, εννοώντας τον υπολογισμό της μέγιστης διαφοράς τάσης από την εξαγωγή ενός χαπιού.
- Τέλος, η R_2 συμβολίζει τέσσερις παράλληλες αντιστάσεις-χάπια.

Συνεπώς, στο διάγραμμα 7.1 η R_{in} αντιστοιχίζεται στην R_1 και η $R_{parallel}$ στην R_2. Κάθε μία από της $R_{parallel}$ αποτελεί την παράλληλη σύνδεση τεσσάρων αντιστάσεων/χαπιών. Συνεπώς, $R_{parallel}$ είναι στην ουσία η 7.3.



Σχήμα 7.3: Κύκλωμα Αντιστάσεων - Χαπιών

Η μετρούμενη τάση (V_{out}) υπολογίζεται με την παρακάτω εξίσωση.

$$V_{out} = V_{in} * \frac{R_2}{R_1 + R_2} \tag{7.1}$$

Για την εύρεση της κατάλληλης τιμής στις αντιστάσεις R_1 και R_2 υπολογίσαμε τη πτώση τάσης (ΔV_{out}) στον κόμβο μεταξύ τους μετά από την εξαγωγή ενός χαπιού. Αρχικά η R_2 αποτελούταν από τέσσερα χάπια ίσης αντίστασης σε παράλληλη συνδεσμολογία. Έστω ότι το κάθε χάπι έχει αντίσταση R_x. Άρα η R_2 προκύπτει από τον τύπο:

$$R_2 = \frac{R_x}{4} \tag{7.2}$$

Μετά την εξαγωγή ενός χαπιού, η συνολική αντίσταση $R_2^{'}$ θα αυξηθεί, καθώς ο αριθμός των παράλληλων χαπιών έχει γίνει τρία. Η $R_2{}^{'}$ προκύπτει από τη σχέση:

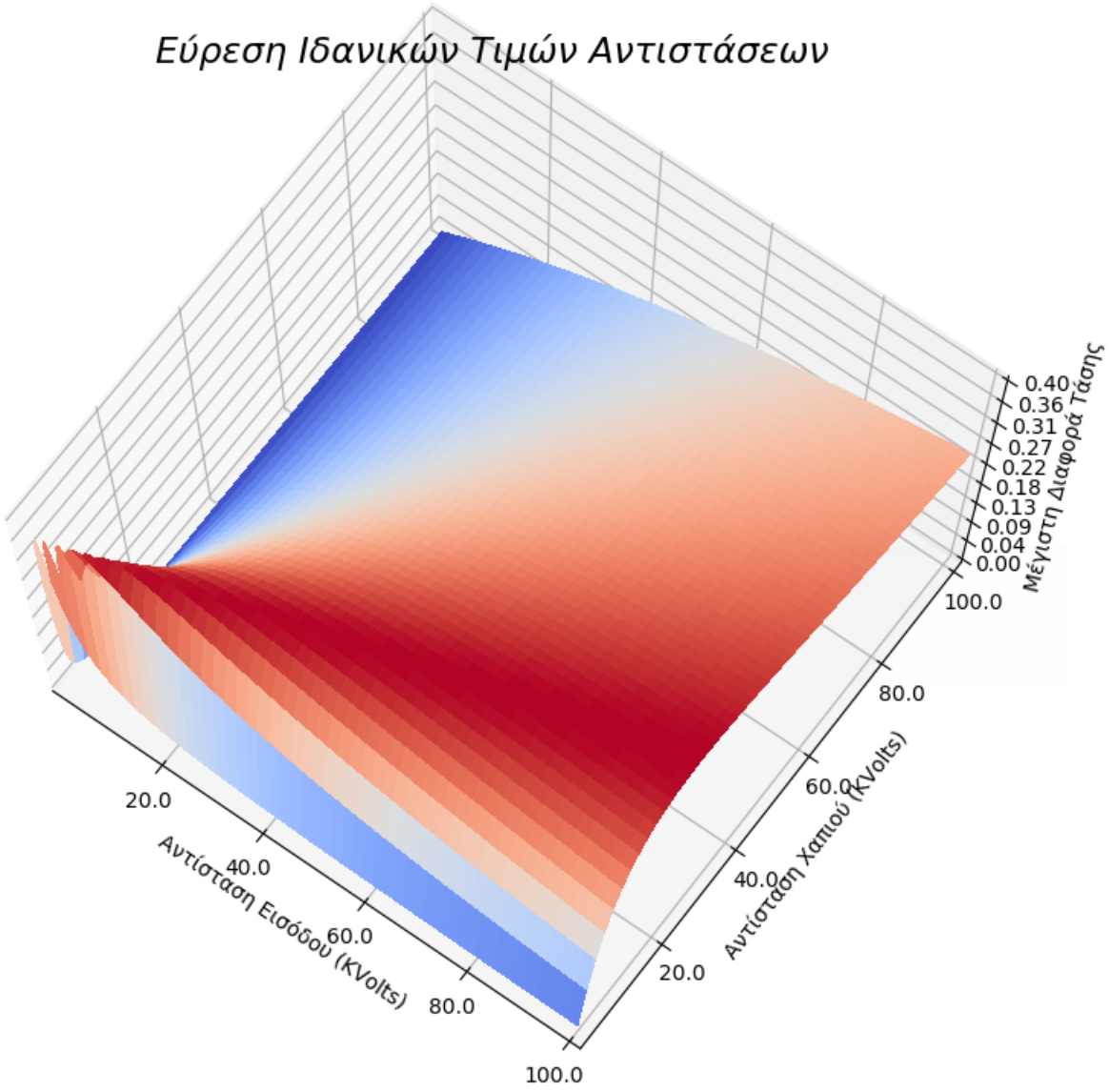
$$R_2^{'} = \frac{R_x}{3} \tag{7.3}$$

Συνεπώς, η πτώση τάσης V_{out} υπολογίζεται από τον παρακάτω τύπο.

$$V_{out} = V_i * (\frac{R_2^{'}}{R_1 + R_2^{'}} - \frac{R_2}{R_1 + R_2}) \tag{7.4}$$

Προκειμένου να καταλήξουμε στο $V_{out,max}$ δοκιμάσαμε συνδυασμούς διαφορετικών τιμών αντιστάσεων. Οι τιμές αυτές είναι βρίσκονται στο εύρος 220Ohm – 100KOhm. Η δοκιμή

έγινε με την εφαρμογή ενός αλγορίθμου σε γλώσσα Python. Ο κώδικας βρίσκεται αναλυτικά στο Παράρτημα 1. Στο σχήμα 7.4 παρουσιάζεται ένα τρισδιάστατο διάγραμμα, με άξονες τα R_1, R_2 και V_{out}.



Σχήμα 7.4: Εύρεση Ιδανικών Τιμών Αντιστάσεων

Όπως φαίνεται και στο διάγραμμα, η μέγιστη διαφορά τάσης, μετά την εξαγωγή ενός χαπιού προκύπτει στα πιο σκούρα σημεία του. Οι μέγιστες τιμές, λοιπόν είναι οι εξής:

- $R_1 = 69978\text{Ohm}$

- $R_x = 19517\text{Ohm}$
- $V_{out,max} = 0.41\text{Volt}$

Στην πρότυπη εξειδικευμένη συσκευή που δημιουργήσαμε, χρησιμοποιήσαμε τις παρακάτω τιμές αντιστάσεων προκειμένου να προσεγγίσουμε τις βέλτιστες.

- $R_1 = 68\text{KOhm}$
- $R_x = 20\text{KOhm}$
- $V_{out,max} = 0.41\text{Volt}$

Συμπεριλαμβάνοντας και τις φυσικές αποκλίσεις της συσκευής,η σχέση μεταξύ της μετρούμενες τάσης και του υπολειπόμενου αριθμού χαπιών εμφανίζεται στον πίνακα 7.1.

Αριθμός Χαπιών	Τάση Volts
0	5
1	3.86
2	3.14
3	2.64
4	2.3

Πίνακας 7.1: Σχέση υπολειπόμενων χαπιών - Επιτρεπτής Τάσης

Λόγω σφαλμάτων των μετρήσεων, αποδεκτές τιμές θεωρούνται αυτές που βρίσκονται σε εύρος ±0.1 από τις παραπάνω. Οι μετρούμενες τιμές που δεν συμπεριλαμβάνονται στο εύρος των παραπάνω υποδεικνύουν σφάλμα του συστήματος.

7.4 Προγραμματισμός Συσκευής

Όπως έχει αναφερθεί και στο προηγούμενο κεφάλαιο, το πρώτο στάδιο λειτουργίας της συσκευής αφορά της αρχικοποίηση της.

Αρχικά, η λειτουργία της συσκευής ξεκινάει από την ενεργοποίηση της LCD οθόνης. Όπως φαίνεται και στον αλγόριθμο 1 ο καθορισμός των θυρών του Arduino με την οθόνη καθορίζεται από εδώ. Αυτή η επιλογή των θυρών αντικατοπτρίζεται και στο παραπάνω συνολικό κύκλωμα 7.1.

Ακόμα, η εντολή *lcd.begin(16,2)* θα ενεργοποιήσει την οθόνη και συγκεκριμένα τις δύο σειρές, όπου η καθεμία διαθέτει 16bit. Με την εντολή *lcd.setCursor(1,0)* τοποθετούμε τον *cursor* στο δεύτερο bit της πρώτης στήλης (Η αρίθμηση ξεκινάει από το μηδέν). Έτσι στην επόμενη εντολή *lcd.print(F("Initialization"))* η λέξη *"Initialization"* θα φανεί στην οθόνη και το πρώτο της γράμμα θα ξεκινάει στο σημείο που τοποθετήσαμε τον *cursor* με την προηγούμενη εντολή.

Τέλος, χρησιμοποιείται πάντα η μακροεντολή *F()* σε εκτυπώσεις της LCD οθόνης, αφού οι συμβολοσειρές που εμφανίζονται δε θα χρησιμοποιηθούν σε άλλο μέρος του κώδικα. Έτσι, εμποδίζουμε τον επεξεργαστή να αποθηκεύσει αυτά τα bits στη SRAM και δεν σπαταλάμε επιπλέον χώρο στη μνήμη. Περισσότερες λεπτομέρειες σχετικά με τη λειτουργία της *F()* macro παρουσιάζονται στο κεφάλαιο 5. Χρησιμοποιώντας την F() macro στη συσκευή φαρμακευτικής συμμόρφωσης, προέκυψε ότι το 10% της μνήμης της SRAM ήταν δεσμευμένο με μη-χρησιμοποιούμενες συμβολοσειρές.

Algorithm 1 Initialization of LCD screen

```
Const int: rs = 7, en = 8, d4 = 2, d5 = 3, d6 = 4, d7 = 5;
LiquidCrystal: lcd(rs, en, d4, d5, d6, d7);
lcd.clear();
 lcd.begin(16, 2);
 lcd.setCursor(1, 0);
 lcd.print(F("Initialization"));
```

Ο αλγόριθμος 2 υλοποιεί την αρχικοποίηση της *RTC* συσκευής. Όπως έχει αναφερθεί, το *RTC Module* λειτουργεί ανεξάρτητα από το *Arduino Mega*. Προκειμένου να αντιμετωπίσουμε τυχόν προβλήματα με το *RTC* έχουν προστεθεί οι συνθήκες "! *rtc.begin*()" και "! *rtc.isrunning*()". Έτσι, μπορούμε να ενημερωθούμε απευθείας για προβλήματα όπως η αποφόρτιση της μπαταρίας λιθίου του *RTC*. Ακόμα, η δεύτερη συνθήκη ελέγχει αν το module έχει αρχικοποιηθεί ή είναι ανενεργό. Στη δεύτερη περίπτωση, θα συγχρονιστεί με την ώρα και μέρα που η συσκευή έκανε κάποια ανανέωση. Εφόσον, όλες οι προηγούμενες ενέργειες στεφθούν με επιτυχία, η ημερομηνία και η ώρα θα εμφανιστούν στην LCD οθόνη αντικαθιστώντας το προηγούμενο μήνυμα αρχικοποίησης *(printDateTime())*.

Algorithm 2 Initialization of Real Time Clock

```
RTC_DS1307: rtc; //global variable
if (! rtc.begin()) then
    printInternalError();
    resetFunc(); //call reset
if (! rtc.isrunning()) then
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__))); //following line sets the RTC to
   the date & time this sketch was compiled
//Print dateTime in LCD screen
 printDateTime();
```

Στον πίνακα 3 ορίζεται η συνάρτηση *resetFunc*. Η λειτουργία της συνιστά την επανεκκίνηση της συσκευής κάθε φορά που η συγκεκριμένη συνάρτηση καλείται.

Algorithm 3 Define Reset Function

```
void(* resetFunc) (void) = 0; //declare reset function @ address 0
```

Η κλήση της συνάρτησης *printInternalError()* βρίσκεται σε διάφορα σημεία του κώδικα, αφού μέσω αυτής ο χρήστης/ασθενής θα αντιληφθεί ότι η συσκευή αντιμετωπίσει εσωτερικό πρόβλημα και ότι ο ίδιος θα πρέπει είτε να την επανεκκινήσει είτε να ζητήσει επισκευή. Οποιαδήποτε, λοιπόν, δυσλειτουργία της συσκευής θα καταλήξει στην εμφάνιση του μηνύματος *"INTERNAL ERROR", "RESET DEVICE"* στην *LCD* οθόνη. Το μήνυμα θα παραμείνει για 4 δευτερόλεπτα στην οθόνη. Η λειτουργία αυτή έχει οριστεί μέσω της εντολής *delay*, η οποία μετράει σε ms. Συνεπώς, $4000ms$ αντιστοιχίζονται σε 4 δευτερόλεπτα αναμονής. Ο λόγος προσθήκης της συγκεκριμένης καθυστέρησης έγκειται στη δυνατότητα του χρήστη να προλάβει να δει το μήνυμα στην οθόνη.

Algorithm 4 Print Internal Error Function

```
void printInternalError()
{
  lcd.begin(16,2);
  lcd.setCursor(0,0);
  lcd.print(F("INTERNAL ERROR"));
  lcd.setCursor(0,1);
  lcd.print(F("RESET DEVICE"));
  lcd.display();
  delay(4000);
{
```

Η συνάρτηση *printDateTime* (5) καλείται από τη *set up function* και περιλαμβάνεται στο κομμάτι της αρχικοποίησης του *RTC Module*. Όπως αναφέρθηκε και στον αλγόριθμο 2, η συνάρτηση θα αντικαταστήσει το υπάρχον μήνυμα της *LCD* οθόνης με την τρέχουσα ημερομηνία και ώρα.

Algorithm 5 Print Date Time Function

```
void printDateTime()
{
  lcd.begin(16, 2);
  DateTime now = rtc.now();
  lcd.setCursor(0, 0);
  lcd.print(now.year(), DEC);
  lcd.setCursor(4, 0);
  lcd.print('/');
  lcd.setCursor(5, 0);
  lcd.print(now.month(), DEC);
  lcd.setCursor(7, 0);
  lcd.print('/');
  lcd.setCursor(8, 0);
  lcd.print(now.day(), DEC);
  lcd.setCursor(11, 0);
  lcd.print(now.hour(), DEC);
  lcd.setCursor(13, 0);
  lcd.print(':');
  lcd.setCursor(14, 0);
  lcd.print(now.minute(), DEC);
  lcd.display();
  delay(3000);
}
```

Στον αλγόριθμο 6 φαίνεται η αποθήκευση των προβλεπόμενων τιμών τάσεων στην μνήμη EEPROM (Κεφάλαιο 5.2.2.3). Η μεταβλητή *eeAddress* δείχνει το σημείο στη μνήμη που θα αποθηκευτεί η νέα μεταβλητή. Η διαδικασία αποθήκευσης γίνεται byte-by-byte και κατά συνέπεια ο δείκτης *eeAddress* αυξάνεται ανάλογα με το μέγεθος της μεταβλητής. Θεωρώντας, λοιπόν, ότι οι μεταβλητές είναι τύπου float θα δεσμεύσουν από 4bytes η καθεμία (πίνακας 5.7).

Algorithm 6 Initialization of EEPROM

```
int eeAddress = 0;
EEPROM.put(eeAddress, (float) 5);
eeAddress += sizeof(float);
EEPROM.put(eeAddress, (float) 3.86);
eeAddress += sizeof(float);
EEPROM.put(eeAddress, (float) 3.15);
eeAddress += sizeof(float);
EEPROM.put(eeAddress, (float) 2.64);
eeAddress += sizeof(float);
EEPROM.put(eeAddress, (float) 2.30);
eeAddress += sizeof(float);
EEPROM.put(eeAddress, (float) 0.02);
```

Η αρχικοποίηση του ηχείου βασίζεται εξολοκλήρου στη βιβλιοθήκη *tmrpcm*. Με την εντολή *tmrpcm.speakerPin = 9;* αντιστοιχίζεται το pin9 του Arduino στο Signal pin του ηχείου. Στη συνέχεια, ορίζεται η ένταση του ήχου και τέλος φορτώνεται και ακούγεται το αρχείο.

Algorithm 7 Initialize and Play a .wav music file

```
// Initialize
tmrpcm.speakerPin = 9;
//Play music
tmrpcm.setVolume(6);
tmrpcm.play("5.wav");
```

Κάθε φορά που ενεργοποιείται το Arduino ή επανεκκινεί θα πρέπει να ενημερώνεται τον συνολικό αριθμό χαπιών. Αυτό επιτυγχάνεται με το διάβασμα της αναλογικής τάσης κάθε μίας από τις έξι πανομοιότυπες σειρές χαπιών στο Blister. Συνεπώς, για να λάβουμε την πραγματική τιμή της χρησιμοποιήσαμε την εντολή *analogRead()* και στη συνέχεια τη μετατρέψαμε στην τελική της μορφή (Κεφάλαιο 5.3.1.3: Analog To Digital Conversion (ADC)). Σύμφωνα με τον πίνακα 7.1, μπορούμε να εξάγουμε τον υπολειπόμενο αριθμό χαπιών στη συγκεκριμένη σειρά. Τη διαδικασία αυτή την υλοποιεί η συνάρτηση *measurePinsInLine*. Η περίπτωση που το αποτέλεσμα της συνάρτησης ισούται με -1 υποδεικνύει πρόβλημα του συστήματος, καθώς μόνο οι τιμές του 7.1 πίνακα είναι αποδεκτές. Σε περίπτωση λοιπόν εσωτερικής βλάβης, θα καλεστεί η συνάρτηση *printInternalError()* για να εμφανιστεί στην *LCD* οθόνη αντίστοιχο μήνυμα. Στη συνέχεια, η συσκευή θα επανεκκινήσει μέσω της συνάρτησης *resetFunc()*. Σε περίπτωση που δεν υπάρχει κάποια βλάβη, η συσκευή θα ανανεώσει τον τρέχον αριθμό χαπιών σε κάθε σειρά και στη συνέχεια θα εξάγει τον υπολειπόμενο αριθμό χαπιών στο blister.

Algorithm 8 Initialize the total number of the remaining pills

```
int: currentPins[6],noPills; //global variables
for int i=0; i < 6; i++ do
    int initValue = analogRead(i);
    float voltage = initValue * (5.0 / 1023.0);
    int pinsInLine = measurePinsInLine(voltage);
  if (pinsInLine == -1) then
        printInternalError();
        resetFunc(); //call reset
    currentPins[i] = pinsInLine;
    noPills += pinsInLine;
end
```

Η συνάρτηση 9 καλείται σε δύο σημεία του κώδικα. Αρχικά, θα καλεστεί στην αρχικοποίηση των pins, εννοώντας στο σημείο που επανεκκινεί το Arduino και εξάγει το συνολικό αριθμό χαπιών στη συσκευή. Επιπλέον, καλείται από τη συνάρτηση *measureVoltage* η οποία βρίσκεται στο κεντρικό βρόγχο του προγράμματος και ελέγχει αν ο ασθενής έλαβε κάποιο χάπι και κατά συνέπεια αν ο συνολικός αριθμός χαπιών στο blister έχει μειωθεί. Ο τρόπος με τον οποίο λειτουργεί η συγκεκριμένη συνάρτηση βασίζεται στις αναμενόμενες τιμές τάσεις που έχουν αποθηκευτεί στην EEPROM από το σημείο της αρχικοποίησης. Έτσι, ελέγχοντας την μετρούμενη τάση σε κάποια σειρά χαπιών μπορούμε απευθείας να εξάγουμε τον αριθμό των χαπιών στη συγκεκριμένη σειρά. Συνεπώς, η συνάρτηση αυτή θα καλεστεί έξι φορές σε κάθε αίτημα για μέτρηση του αριθμού χαπιών στο blister. Όπως φαίνεται και στα σχόλια, οι αναμενόμενες τιμές τάσεις βρίσκονται σε συγκεκριμένη θέση στην EEPROM μνήμη οπότε θα πρέπει να την προσπελάσουμε με συγκεκριμένο τρόπο προκειμένου να τις ανακτήσουμε.

Algorithm 9 Measure Number of Pills in each line

```
int measurePinsInLine(float volt)
{
  int pills = 0;
  //[0 – 3bits] ⇒ 5Volts(0 pills)
  //[4 – 7] ⇒ 3.86Volts(1 pill)
  //[8, 11] ⇒ 3.14Volts(2 pills)
  //[12, 15] ⇒ 2.64Volts(3 pills)
  //[16, 19] ⇒ 2.31Volt(4 pills)
for (int eeAddress = 0; eeAddress < 23; eeAddress+ = sizeof(float)) do
    pills++;
    float f;
    EEPROM.get(eeAddress, f);
  if (pills == 5) then pills = 0; ;
  if (approximate(volt, f)) then return pills; ;
  return -1
end
}
```

Ο αλγόριθμος 10 καλείται για την εξίσωση των τάσεων των σειρών χαπιών του Blister με τις προβλεπόμενες που είναι αποθηκευμένες στην EEPROM μνήμη του Arduino. Λαμβάνοντας υπόψη τις αποκλίσεις των μετρήσεων του Arduino δικαιολογούμε ένα εύρος απόκλισης 0.05Volts, δηλαδή 1%. Συνεπώς, αν η μετρούμενη τάση είναι αποδεκτή, η συνάρτηση θα επιστρέψει true, διαφορετικά θα επιστρέψει false.

Algorithm 10 Find if two values are approximately equal

```
boolean approximate(float i, float j)
{
if (abs(i – j) < 0.05) then return true ;
else return false ;
}
```

Η συνάρτηση *ISR (Interrupt Service Routine)* αποτελεί τη ρουτίνα διακοπών, η οποία θα καλεστεί αυτόματα κάθε φορά που ο *Watchdog Timer* θα προκαλέσει κάποια διακοπή. Όταν καλεστεί, θα απενεργοποιήσει τον *Watchdog Timer* για λόγους ασφαλείας και θα επιστρέψει στο σημείο που αυτός καλέστηκε. Ωστόσο, για να λειτουργήσει αυτή η διαδικασία θα πρέπει να αρχικοποιηθεί ο *Watchdog Timer* με την επεξεργασία κάποιων καταχωρητών. Το *Sleep Mode* που επιλέχθηκε να χρησιμοποιηθεί είναι το *Power-Down* καθώς επιτυγχάνει τη μικρότερη κατανάλωση σε σχέση με τα υπόλοιπα (Πίνακας 5.5).

Algorithm 11 Define WatchDog Interrupt Routine

```
// watchdog interrupt
ISR(WDT_vect)
{
  wdt_disable(); // disable watchdog
}

void myWatchdogEnable(const byte interval)
{
  MCUSR = 0; // reset various flags
  WDTCSR | = 0b00011000; //set WDCE, WDE
  WDTCSR = 0b01000000 | interval; //set WDIE
  wdt_reset();
  set_sleep_mode (SLEEP_MODE_PWR_DOWN);
  sleep_mode(); //now goes to Sleep and waits for the interrupt
}
```

Ακόμα μία περιφερειακή συσκευή που χρησιμοποιήσαμε είναι το *SD Card Module*. Η σύνδεση του με το Arduino επιτυγχάνεται μέσω SPI επικοινωνίας. Για το λόγο αυτό στον πίνακα 12 χρησιμοποιείται η θύρα 53 για την ενεργοποίηση του. Σε περίπτωση, μη ενεργοποίησης της κάρτας μνήμης θα ακολουθηθεί η γνωστή διαδικασία βλάβης. Θα καλεστεί λοιπόν, η συνάρτηση *printInternalError()* για να εμφανιστεί στην *LCD* οθόνη αντίστοιχο μήνυμα και η συσκευή θα επανεκκινήσει μέσω της συνάρτησης *resetFunc()*. Διαφορετικά, θα ανοίξουμε το αρχείο *"test.txt"* το οποίο προσομοιώνει την επικοινωνία με τη MongoDB βάση δεδομένων. Το αρχείο θα αποθηκευτεί στη μεταβλητή *myFile*, η οποία είναι μία Global μεταβλητή και διαβαστεί μέσω της συνάρτησης *openFile()*.

Algorithm 12 Initialize SD card

```
File: myFile; //global variable
if (! SD.begin(53)) then
    printInternalError();
    resetFunc(); //call reset
myFile = SD.open("test.txt"); //open for read only
 openFile();
```

Στη συνάρτηση *openFile()* (Αλγόριθμος 13) ελέγχουμε αν η διαδικασία ανοίγματος του αρχείου από την SD Card ήταν πετυχημένη. Η μεταβλητή *myFile* είναι τύπου *File* και η συνθήκη *if (myFile)* θα είναι αληθής αν το συγκεκριμένο αρχείο ανοίχτηκε σωστά και ψευδής διαφορετικά. Στη δεύτερη περίπτωση θα ακολουθηθεί η διαδικασία εσωτερικής βλάβης, δηλαδή, θα καλεστεί η συνάρτηση *printInternalError()* για να εμφανιστεί στην *LCD* οθόνη αντίστοιχο

μήνυμα και η συσκευή θα επανεκκινήσει μέσω της συνάρτησης *resetFunc()*. Σε περίπτωση που το αρχείο ανοιχθεί σωστά η συνάρτηση *readTime()* θα διαβάσει το περιεχόμενο του.

Algorithm 13 Open file from the SD card

```
void openFile()
{
if (myFile) then
    readTime();
    myFile.close();
else
    printInternalError();
    resetFunc(); //call reset
end
}
```

Η συνάρτηση 15 διαβάζει τα δεδομένα από το αποθηκευμένο αρχείο στην κάρτα μνήμης και τα αποθηκεύει στις Global μεταβλητές *Time1* και *Time2*, οι οποίες συμβολίζουν τις δύο χρονικές στιγμές μέσα στη μέρα που ο ασθενής θα πρέπει να λάβει τα χάπια του. Στην πραγματικότητα, η διαδικασία αυτή θα γίνεται μέσω της επικοινωνίας της συσκευής με τη MongoDB βάση δεδομένων. Για την προσέγγιση λοιπόν της πραγματικής λειτουργίας της συσκευής, το αρχείο *"test.txt"* που ανοίχθηκε, έχει τη μορφή ενός *"Json"* αρχείου. Για την ανάλυση του χρησιμοποιήθηκε η βιβλιοθήκη < $ArduinoJson.h$ >. Η *read()* εντολή διαβάζει έναν χαρακτήρα κάθε φορά, ο οποίος προστίθεται στους προηγούμενους στη μεταβλητή *json2*. Για το λόγο αυτό η συνάρτηση βρίσκεται μέσα στο βρόγχο *While (myFile.available())*, η συνθήκη του οποίου θα είναι αληθής για όσο διάστημα περισσεύουν χαρακτήρες που δεν έχουν προσπελαστεί. Εφόσον, έχουμε λάβει όλα τα δεδομένα από το αρχείο, θα τα προσπελάσουμε με τη βοήθεια της *Json* βιβλιοθήκης και συγκεκριμένα μέσω της εντολής *parseObject()*. Έτσι, η μεταβλητή $JsonObjectroot$ θα περιλαμβάνει όλο το *Json* αρχείο, που θα έχει τη μορφή του αλγορίθμου 14.

Algorithm 14 Η μορφή του αρχείου text.txt

```
{
"device":"ABCD",
"timeForPill_1":"08:00",
"timeForPill_2":"19:30"
}
```

Στη συνέχεια, θα εξάγουμε τις τιμές/ ώρες λήψης των δύο χαπιών και θα τις μετατρέψουμε σε λεπτά. Ωστόσο, οι τιμές είναι αποθηκευμένες με συγκεκριμένο τρόπο και σε ASCII μορφή. Συνεπώς, για να επιτύχουμε τη μετατροπή, εξάγουμε αρχικά τα δύο πρώτα bits που συμβολίζουν την ώρα και τα πολλαπλασιάζουμε με 60 έτσι ώστε να τα μετατρέψουμε σε λεπτά. Στη

συνέχεια, παραλείπουμε το bit που αντιστοιχίζεται στην άνω-κάτω τελεία και λαμβάνουμε τα δύο επόμενα bits, τα οποία συμβολίζουν τα λεπτά και τα προσθέτουμε στα πρώτα. Η αφαίρεση του '0' είναι αναγκαία καθώς οι χαρακτήρες είναι αποθηκευμένοι σε ASCII μορφή (Σχήμα 7.5).

The ASCII code

American Standard Code for Information Interchange

www.theasciicode.com.ar

ASCII control characters

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

ASCII printable characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_			

theASCIIcode.com.ar

Extended ASCII characters

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	á	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	ë	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	240	F0h	
145	91h	æ	177	B1h	▒	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	▓	210	D2h	Ê	242	F2h	‗
147	93h	ô	179	B3h	│	211	D3h	Ë	243	F3h	¾
148	94h	ò	180	B4h	┤	212	D4h	È	244	F4h	¶
149	95h	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°
153	99h	Ö	185	B9h	╣	217	D9h	┘	249	F9h	¨
154	9Ah	Ü	186	BAh	║	218	DAh	┌	250	FAh	·
155	9Bh	ø	187	BBh	╗	219	DBh	▉	251	FBh	¹
156	9Ch	£	188	BCh	╝	220	DCh	▄	252	FCh	³
157	9Dh	Ø	189	BDh	¢	221	DDh	¦	253	FDh	²
158	9Eh	×	190	BEh	¥	222	DEh	▌	254	FEh	■
159	9Fh	ƒ	191	BFh	┐	223	DFh	▀	255	FFh	

Σχήμα 7.5: Πίνακας ASCII

Algorithm 15 Read Json File from SD Card

```
#define: JSON_BUFFER_SIZE 20
unsigned long: time1, time2; //global variable
void readTime()
 {
  StaticJsonBuffer< 200 > jsonBuffer;
  String json2;
while myFile.available() do
    char c = myFile.read();
    json2.concat(c);
end
  JsonObject& root = jsonBuffer.parseObject(const_cast< char* >(json2.c_str()));
  const char* timeForPill_1 = root["timeForPill_1"];
  const char* timeForPill_2 = root["timeForPill_2"];
if (! root.success()) then
     printInternalError();
     resetFunc(); //call reset
  //Count in minutes
  time1 = (10 * (timeForPill_1[0] - '0') + timeForPill_1[1] - '0') * 60 + 10 * (timeForPill_
  1[3] - '0') + timeForPill_1[4] - '0';
  time2 = (10 * (timeForPill_2[0] - '0') + timeForPill_2[1] - '0') * 60 + 10 * (timeForPill_
  2[3] - '0') + timeForPill_2[4] - '0';
}
```

Η συνάρτηση *loop()* (Αλγόριθμος 17) αποτελεί τον επαναλαμβανόμενο βρόγχο της συσκευής, που θα εκτελείται έως ότου ο ασθενής αποσυνδέσει τη συσκευή από την τροφοδοσία ή γίνει εσωτερική επανεκκίνηση της συσκευής. Αρχικά, η συσκευή θα επανεκκινεί κάθε 24 ώρες. Η λειτουργία αυτή θα γίνεται κυρίως για λόγους ασφάλειας και επαλήθευσης της ορθότητας των δεδομένων που επαναχρησιμοποιούνται. Στη συνέχεια, καλείται η συνάρτηση *printRestOf()* η οποία εμφανίζει στην LCD οθόνη τον υπολειπόμενο αριθμό χαπιών. Η ενέργεια αυτή διαρκεί επιπλέον τρία δευτερόλεπτα μέσω της εντολής *delay()*, προκειμένου να προλάβει ο ασθενής/χρήστης να δει την οθόνη. Εφόσον, εμφανιστεί το μήνυμα για αρκετό χρόνο η οθόνη θα σβήσει (*lcd.noDisplay()*). Επόμενη ενέργεια αποτελεί ο έλεγχος του υπολειπόμενου αριθμού χαπιών. Σε περίπτωση που το blister περιλαμβάνει μόνο ένα τελευταίο χάπι, η συσκευή θα προειδοποιήσει τον ασθενή προκειμένου να επιληφθεί την ανανέωση της συσκευασίας φαρμάκων. Ακόμα, σε περίπτωση που τα χάπια του blister έχουν τελειώσει η συσκευή θα ενημερώσει το χρήστη και κάνει επανεκκίνηση, αφού δεν μπορεί να εκτελέσει τη λειτουργία της χωρίς χάπια. Εφόσον οι παραπάνω ενέργειες ολοκληρωθούν και το blister δεν είναι άδειο, θα εμφανιστεί η υπολειπόμενη ώρα λήψης κάποιου χαπιού στην οθόνη, μέσω της *printTimeForThePill* συνάρτησης. Στο μέρος-2, λοιπόν, του βρόγχου η συσκευή ελέγχει αν

έχει φτάσει η κατάλληλη ώρα λήψης χαπιού. Ο έλεγχος αυτός γίνεται μέσω της συνάρτησης *(isTimeForPill())*. Στην περίπτωση που η προκειμένη συνάρτηση επιστρέψει *false* τότε θα υπολογίσει τον χρόνο μέχρι το επόμενο χάπι *getTimeForPill()*, θα εμφανίσει το χρόνο που υπολόγισε στην οθόνη και θα αδρανοποιήσει τη συσκευή για το παραπάνω χρονικό διάστημα. Ο Watchdog Timer ξυπνάει τη συσκευή από κάποιο Sleep Mode το πολύ κάθε οχτώ δευτερόλεπτα. Προκειμένου, λοιπόν να αδρανοποιήσουμε τη συσκευή για περισσότερη ώρα, ορίζουμε τον Watchdog Timer στο μέγιστο (8 δευτερόλεπτα) και τον ενεργοποιούμε όσες φορές χρειαστεί μέσω του For βρόγχου. Από την άλλη πλευρά, αν η συνθήκη *isTimeForPill()* επιστρέψει *true*, τότε ο ασθενής θα πρέπει να ειδοποιηθεί άμεσα. Αρχικά, η συσκευή μετράει τα υπολειπόμενα χάπια (*measureVoltage()*) και ελέγχει αν ο ασθενής έχει πάρει ήδη το χάπι του, έχοντας προνοήσει για τη λήψη της αγωγής του. Αν έχει συμβεί το παραπάνω ενδεχόμενο, τότε η συσκευή θα εμφανίσει στην οθόνη τον υπολειπόμενο αριθμό χαπιών (*printRestOf()*), θα υπολογίσει την ώρα μέχρι το επόμενο χάπι, δηλαδή την ώρα που θα είναι σε αδράνεια (*getTimeForNextPill()*), θα εμφανίσει αυτή την ώρα στην οθόνη (*printTheSleepTime()*) και στη συνέχεια θα ενεργοποιήσει τον Watchdog Timer. Ωστόσο, αν ο ασθενής δεν έχει λάβει πρωτύτερα το χάπι του, τότε το ηχείο θα αναπαράγει ένα μουσικό αρχείο και η οθόνη θα εμφανίσει το μήνυμα *"Time to take your pills"*. Η διαδικασία αυτή θα διαρκέσει δέκα δευτερόλεπτα, μαζί με τις καθυστερήσεις στο εσωτερικό τον συναρτήσεων. Στη συνέχεια, η συσκευή θα μετρήσει ξανά τον αριθμό τον χαπιών στο blister για να ενημερωθεί αν ο ασθενής ειδοποιήθηκε. Αν ο ασθενής έλαβε κάποιο χάπι στο παραπάνω διάστημα, τότε η συσκευή θα υπολογίσει το χρόνο για το επόμενο και θα ενεργοποιήσει το Watchdog Timer. Διαφορετικά, η συσκευή θα επαναλαμβάνει τη διαδικασία αφύπνισης κάθε δέκα δευτερόλεπτα.

Algorithm 16 Κύριος Βρόγχος Προγράμματος - Μέρος1

```
void loop()
{
//Reset the device in the end of the day
if (millis() == 86400000) then
 |  resetFunc();
printRestOf();
delay(3000);
lcd.noDisplay();
if (noPills == 1) then
 |  notifyPatient()
else if (noPills == 0) then
 |      printInternalError();
 |      resetFunc(); //call reset
int sum;
printTimeForThePill();
delay(5000);
}
```

Algorithm 17 Κύριος Βρόγχος Προγράμματος - Μέρος2

```
if (isTimeForPill()) then
    for (int i = 0; i < 6; i++) do
        pins = measureVotage(i);
        if (pins==-1) then
            printInternalError();
            resetFunc(); //call reset
    end
    sum += pins;
    printRestOf();
    if (sum ⩾ noPills) then
        playMusic();
        LightTheScreen();
        playMusic();
        for (int i = 0; i < 6; i++) do
            pins = measureVotage(i); if (pins==-1) then
                printInternalError();
                resetFunc(); //call reset
        end
        sum += pins;
    if (sum ⩾ noPills) then
        lcd.print(F("Notification in"));
        lcd.setCursor(3, 1);
        lcd.print(F("10 seconds"));
        lcd.display();
        myWatchdogEnable (0b100001); // 8 seconds
    else
        lcd.print(F("Congratulations"));
        lcd.display();
        noPills = sum;
        printRestOf();
        long nextPill = (getTimeForNextPill())-10;
        printTheSleepTime(nextPill);
        for (long i = 0; i < nextPill * 60/8; i++) do
            myWatchdogEnable (0b100001);
        end
    end
    else
        long nextPill = (getTimeForPill() -10);
        printTheSleepTime(nextPill);
        for (long i = 0; i < (nextPill * 60)/8); i++) do
            myWatchdogEnable (0b100001);
        end
    end
```

Όταν έρθει η στιγμή που ο ασθενής θα πρέπει να λάβει την αγωγή του, η συσκευή οφείλει να τον ενημερώσει με χρήση οπτικοακουστικού υλικού. Έτσι στη συνάρτηση 18 η οθόνη εμφανίζει το μήνυμα *Time to take your pills*, προκειμένου να ειδοποιήσει τον ασθενή ότι έφτασε η προβλεπόμενη στιγμή λήψης των χαπιών.

Algorithm 18 Remind the patient to take his pills

```
void LightTheScreen()
{
  lcd.begin(16, 2);
  lcd.setCursor(2, 0);
  lcd.print(F("Time to take "));
  lcd.setCursor(3, 1);
  lcd.print(F("your pills"));
  lcd.display();
}
```

Η λειτουργία της συνάρτηση 19 έγκειται στην ειδοποίηση του χρήστη για το τελείωμα των χαπιών του. Το μήνυμα *You are out of pills* εμφανίζεται στην LCD οθόνη.

Algorithm 19 Notify the patient that he is out of pills

```
void notifyPatient()
{
  lcd.begin(16, 2);
  lcd.setCursor(2, 0);
  lcd.print(F("You are out"));
  lcd.setCursor(3, 1);
  lcd.print(F("of pills"));
  lcd.display();
}
```

Στη συνάρτηση 20 αποτυπώνεται στην οθόνη ο υπολειπόμενος αριθμός χαπιών. Η μεταβλητή *noPills*, που αντιστοιχεί στο συνολικό αριθμό χαπιών του blister είναι global και συνεπώς δε χρειάζεται να την περάσουμε σαν όρισμα στην συνάρτηση. Η λειτουργία αυτή είναι σημαντική, καθώς ο χρήστης ενημερώνεται άμεσα για το απόθεμα των φαρμάκων του και μπορεί να προβεί στην αγορά νέας συσκευασίας προτού η προηγούμενη εξαντληθεί.

Algorithm 20 Notify the patient for the Remaining pills in the device

```
void printRestOf()
{
  lcd.noDisplay();
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print(F("Remaining Pills"));
  lcd.setCursor(5, 1);
  lcd.print(noPills);
  lcd.display();
}
```

Ο συνάρτηση 21 εμφανίζει στην οθόνη τον υπολειπόμενο αριθμό χαπιών στη συσκευή. Η λειτουργία αυτή εκτελείται είτε μετά από επανεκκίνηση της συσκευής είτε μετά από κάποια ρουτίνα διακοπής. Συγκεκριμένα, λαμβάνει την υπολειπόμενη ώρα μέσω της συνάρτησης *getTimeforPill()* σε λεπτά. Στη συνέχεια, τη μετατρέπει σε φιλική μορφή ως προς τον χρήστη, δηλαδή σε HH:MM, μέσω της ακέραιας διαίρεσης και της εξαγωγής υπολοίπου. Έτσι, ο ασθενής γνωρίζει σε πόση ώρα θα πρέπει να είναι διαθέσιμος για να λάβει τη φαρμακευτική του αγωγή.

Algorithm 21 Show in the screen the remaining time until the next pill

```
void printTimeForThePill()
{
  lcd.begin(16, 2);
  unsigned long timeForPill = getTimeForPill();
  int hh = timeForPill / 60;
  int mm = timeForPill % 60;
  lcd.setCursor(2, 0);
  lcd.print(F("Alarm Clock in: "));
  lcd.setCursor(0, 1);
  lcd.print(hh);
  lcd.setCursor(2, 1);
  lcd.print("h and ");
  lcd.setCursor(11, 1);
  lcd.print(mm);
  lcd.setCursor(13, 1);
  lcd.print(F("min"));
  lcd.display();
  delay(5000);
}
```

Η συνάρτηση 22 καλείται μόλις ο ασθενής λάβει κάποιο χάπι και τον ενημερώνει μέσω της LCD οθόνης για το διάστημα στο οποίο η συσκευή θα βρίσκεται σε αδράνεια. Το μήνυμα αυτό διατηρείται για πέντε δευτερόλεπτα στην οθόνη μέσω της εντολής *delay(5000)*, προκειμένου ο ασθενής/χρήστης να προλάβει να το δει.

Algorithm 22 Show in LCD the interval that the device will be inactive

```
void printTheSleepTime(int timeForPill)
{
  lcd.begin(16, 2);
  int hh = timeForPill / 60;
  int mm = timeForPill
  lcd.setCursor(2, 0);
  lcd.print(F("Sleep for: "));
  lcd.setCursor(0, 1);
  lcd.print(hh);
  lcd.setCursor(2, 1);
  lcd.print("h and ");
  lcd.setCursor(11, 1);
  lcd.print(mm);
  lcd.setCursor(13, 1);
  lcd.print(F("min"));
  lcd.display();
  delay(5000);
}
```

Η συνάρτηση 23 είναι τύπου *Boolean* και επομένως η τιμή που θα επιστρέψει θα είναι είτε αληθής είτε ψευδής. Η λειτουργία της βασίζεται στην ενημέρωση της τρέχουσας ώρας μέσω της συνάρτησης *isTimeForPill()* και στη σύγκρισή της με τις δύο προβλεπόμενες ώρες λήψης χαπιών. Αν η διαφορά της τρέχουσας ώρας με κάποια από τις υπόλοιπες δύο είναι μικρότερη των δέκα λεπτών, τότε η συνάρτηση θα επιστρέψει *true*, διαφορετικά θα επιστρέψει *false*.

Algorithm 23 Is time for Pill?

```
boolean isTimeForPill()
{
unsigned long currentTime = findCurrentTime();
if (difference(currentTime,time1)) < 10 or difference(currentTime,time2) < 10)
then
 | return true
else
 | return false
end
}
```

Ο λόγος προσθήκης του αλγορίθμου 24 είναι ότι οι *Unsigned long* μεταβλητές δεν αποθηκεύουν αρνητικούς αριθμούς. Συνεπώς αν επιδιώκαμε να χρησιμοποιήσουμε τη συνάρτηση *abs(absolute)* για να βρούμε τον υπολειπόμενο χρόνο για το επόμενο χάπι, τα αποτελέσματα μας θα ήταν λανθασμένα. Οι *unsigned long* μεταβλητές αποθηκεύουν 32bits (4bytes) και δέχονται αριθμούς από 0 έως $4,294,967,295(2^{32}-1)$.

Algorithm 24 Find the absolute difference between two values

```
unsigned long difference(unsigned long a, unsigned long b)
{
if (a > b) then
 | return a-b
else
 | return b-a
end
}
```

Η συσκευή θα εισέλθει στη συνάρτηση *getTimeForPill()* (Αλγόριθμος 26) είτε μετά από κάποια επανεκκίνηση είτε μετά από διακοπή σε Sleep Mode. Και στις δύο περιπτώσεις, η ώρα που θα πρέπει να εξάγουμε είναι η πιο κοντινή στην τρέχουσα ώρα. Η λογική λοιπόν βασίζεται στον έλεγχο της πιο κοντινής ώρας και στην επιστροφή της στην κύρια συνάρτηση.

Algorithm 25 Get the the remaining time until the next pill

```
unsigned long getTimeForPill();
{
unsigned long currentTime = findCurrentTime();
unsigned long timeForPill;
if (time1 > currentTime && time2 > currentTime) then
    if ((difference(currentTime, time1) > difference(currentTime, time2))) then
        timeForPill = difference(currentTime, time2);
    else
        timeForPill = difference(currentTime, time1);
    end
else if (time1 < currentTime && time2 > currentTime) then
    timeForPill = difference(currentTime, time2);
else if (time2 < currentTime && time1 > currentTime) then
    timeForPill = difference(currentTime, time1);
else if (time1 < currentTime && time2 < currentTime) then
    if (time1 < time2) then
        timeForPill = 24*60 - currentTime + time1;
    else
        timeForPill = 24*60 - currentTime + time2;
    end
return timeForPill;
}
```

Η συσκευή θα εισέλθει σε λειτουργία αδράνειας στα διαστήματα που δε θα πλησιάζει η στιγμή λήψης κάποιου χαπιού. Συνεπώς, εκμεταλλευόμενοι τη συνάρτηση 26 μπορούμε να αποκτήσουμε το διάστημα που θα πρέπει η συσκευή να είναι σε αδράνεια. Στη συνάρτηση αυτή θα εισέλθουμε ακριβώς μετά την επιτυχημένη λήψη κάποιου χαπιού. Οπότε, η λογική των συνθηκών της συνάρτησης είναι αντίστροφοι με προηγουμένως. Αναλυτικότερα, ο ασθενής μόλις έχει λάβει το χάπι του και η συσκευή θα αδρανοποιηθεί για το διάστημα μέχρι την ώρα λήψης του επόμενου χαπιού. Έτσι, η μεταβλητή *timeForPill* που θα επιστραφεί είναι αυτή που έχει τη μεγαλύτερη απόκλιση από την τρέχουσα ώρα.

Algorithm 26 Get the time that the device will be inactive

```
unsigned long getTimeForNextPill()
{
unsigned long currentTime = findCurrentTime();
unsigned long timeForPill;
if (difference(currentTime, time1) > difference(currentTime, time2)) then
    if (time1 ⩾ currentTime) then
        timeForPill = difference(currentTime, time1);
    else
        timeForPill = 24*60 - currentTime + time1;
    end
else
    if (time2 ⩾ currentTime) then
        timeForPill = difference(currentTime, time2);
    else
        timeForPill = 24*60 - currentTime + time2;
    end
end
return timeForPill;
}
```

Το *Real Time Clock* λειτουργεί ανεξάρτητα από την υπόλοιπη συσκευή. Επίσης, συνεχίζει τη λειτουργία του κατά τη διάρκεια που το Arduino είναι σε αδράνεια. Συνεπώς, είναι εφικτό να λάβουμε την τρέχουσα ώρα μέσω της περιφερειακής αυτής συσκευής οποιαδήποτε στιγμή, χρησιμοποιώντας τον αλγόριθμο 27.

Algorithm 27 Get the current Time from RTC

```
unsigned long findCurrentTime()
{
Date now = rtc.now();
return (now.hour() * 60 + now.minute()); //Return time in minutes
}
```

Ο αλγόριθμος 28 υλοποιεί τη μέρτηση της τάσεις των αγώγιμων σειρών του blister. Η τάση που μετριέται σε κάθε μία από τις έξι σειρές χαπιών είναι ασφαλώς αναλογική. Συνεπώς, για να λάβουμε την πραγματική τιμή της χρησιμοποιήσαμε την εντολή *analogRead()* και στη συνέχεια τη μετατρέψαμε στην τελική της μορφή (Κεφάλαιο 5.3.1.3: Analog To Digital Conversion (ADC)). Σύμφωνα με τον πίνακα 7.1, μπορούμε να εξάγουμε τον υπολειπόμενο αριθμό χαπιών στη συγκεκριμένη σειρά. Τη διαδικασία αυτή την υλοποιεί η συνάρτηση *measurePinsInLine*. Η περίπτωση που το αποτέλεσμα της συνάρτησης ισούται με -1 υποδεικνύει πρόβλημα του συστήματος, καθώς μόνο οι τιμές του 7.1 πίνακα είναι αποδεκτές. Οπότε η συνάρτηση θα

επιστρέψει κι αυτή τον αριθμό -1, με αποτέλεσμα να υποδείξει στη συνάρτηση που την καλεί το εσωτερικό πρόβλημα. Η *parent* συνάρτηση λοιπόν θα χειριστεί το ζήτημα, λαμβάνοντας το -1 από τη *measureVoltage*. Με το πέρας του προηγούμενου ελέγχου *(error handling)*, η συσκευή ελέγχει αν όντως ο ασθενής έλαβε το χάπι του. Στην περίπτωση αυτή, θα ανανεώσει τον τρέχον αριθμό χαπιών και εν τέλει θα τον επιστρέψει στη συνάρτηση που κάλεσε τη *measureVoltage*.

Algorithm 28 Measure Voltage in each line of Blister

```
int measureVotage(int line)
{
int initValue = analogRead(line);
float voltage = initValue * (5.0 / 1023.0);
int pinsInLine = measurePinsInLine(voltage);
if (pinsInLine == -1) then
 | return-1
if (abs(pinsInLine – currentPins[line]) > 0) then
 |  currentPins[line] = pinsInLine;
return pinsInLine;
}
```

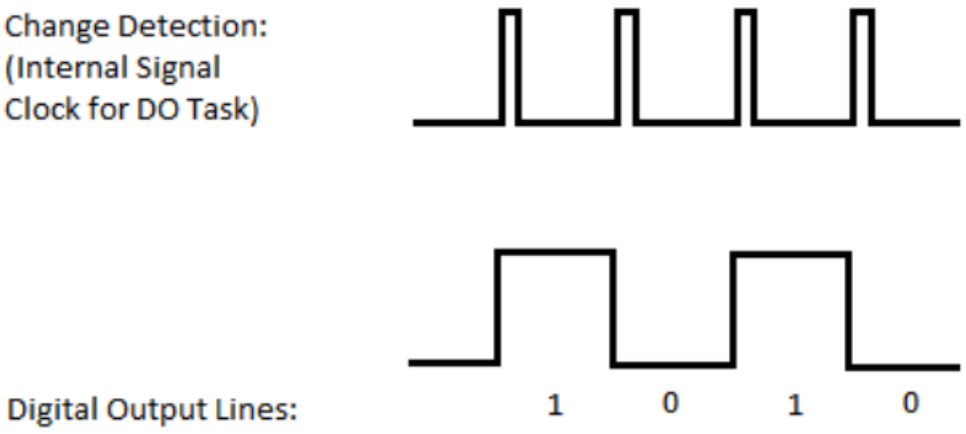
7.5 Εναλλακτική Μέθοδος Σχεδίαση Συσκευής με Χρήση Πολυπλέκτη

Στη διάρκεια της εκπόνησης της διπλωματικής εργασίας διερευνήθηκαν αρκετοί τρόποι για τη συναρμολόγηση των επιμέρους στοιχείων του έργου και τη βέλτιστη αποδοτικότητα αυτών. Μία πρώτη λύση που είχε προσεγγιστεί περιλαμβάνει δύο επιπλέον ηλεκτρονικά στοιχεία:

1. Έναν μετρητή 4-bit
2. Έναν πολυπλέκτη 8 σε 1.

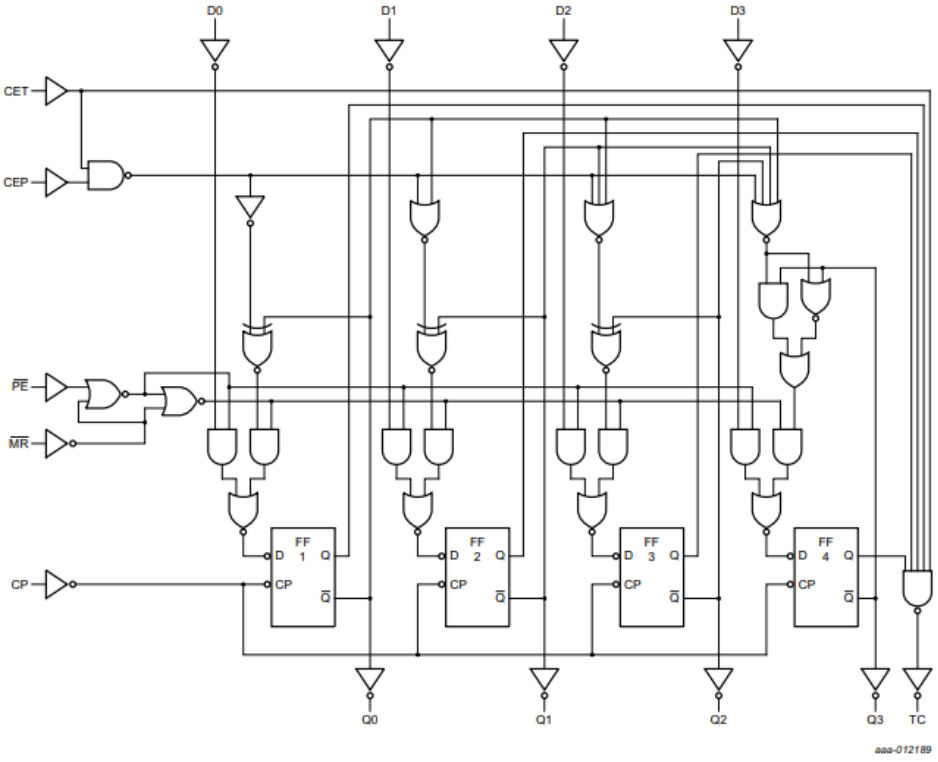
7.5.1 Μετρητής 4-bit

Στην παρούσα μελέτη θα αναλύσουμε τη λειτουργία του μετρητή μέσω του ολοκληρωμένου 74HC163. Παρόμοια χρήση έχει και το πιο διάσημο ολοκληρωμένο 74LS193. Πρόκειται για ένα σύγχρονο μετρητή με ρυθμισμένη τη λειτουργία της επαναφοράς. Αποτελείται από τέσσερα flip-flops. Η σύγχρονη λειτουργία του επιτυγχάνεται έχοντας όλα τα ακμοπυροδότητα (θετικές ακμής) flip-flop χρονομετρημένα ταυτόχρονα. Με την έννοια *ακμοπυροδότητα* εννοούμε ότι η μεταβολή του σήματος από 0 σε 1 και αντίστροφα, πραγματοποιείται τη στιγμή που ο παλμός φτάσει στην θετική ακμή (positive edge) και όχι στην αρνητική (negative edge). Όπως φαίνεται και στο σχήμα 7.6, η έξοδος μεταβάλλεται με τη θετική ακμή του ρολογιού.



Σχήμα 7.6: Ακμοπυροδότητος Παλμός Μετρητή

Στη συνέχεια, θα παρουσιαστεί αναλυτικά το ψηφιακό κύκλωμα του μετρητή, με τις λογικές πύλες που περιλαμβάνει και τα flip-flops (Σχήμα 7.7).

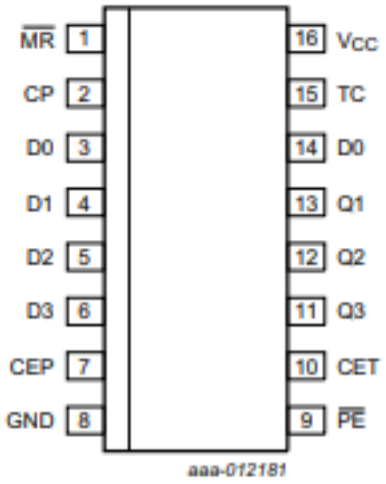


Σχήμα 7.7: Ψηφιακό Κύκλωμα Μετρητή 4-bit

Το ρολόι του κυκλώματος συμβολίζεται ως *CP* (Clock). Οι έξοδοι του κυκλώματος είναι οι Q0-Q3. Η είσοδος $\overline{PE}$ (Parallel Enable Input) όταν βρίσκεται σε χαμηλή κατάσταση απενεργοποιεί τη λειτουργία της μέτρησης και έχει ως αποτέλεσμα τη φόρτωση των εισόδων

(D0 – D3) στις αντίστοιχες εξόδους. Η είσοδος$\overline{MR}$ (Master Reset Input) όταν βρίσκεται σε χαμηλό επίπεδο επαναφέρει όλες τις εξόδους σε 0 με τον επόμενο παλμό του ρολογιού. Αυτή η λειτουργία γίνεται ανεξάρτητα των υπολοίπων εισόδων. Οι είσοδοι *CEP* (Count Enable Input) και *CET* (Count Enable Carry Input) αντιστοιχίζονται στην ενεργοποίηση της λειτουργίας της μέτρησης, οπότε θα πρέπει να είναι θετικοί. Η *TC* (Terminal Count Outpout) ενεργοποιείται από την *CET* είσοδο και παράγει έναν θετικό παλμό, ο οποίος διαρκεί όσο περίπου η έξοδος Q0 είναι θετική.

Το ολοκληρωμένο που χρησιμοποιήθηκε φαίνεται στο σχήμα 7.8 και περιλαμβάνει τις εισόδους/ εξόδους που αναλύθηκαν.



Σχήμα 7.8: Ολοκληρωμένο Μετρητή 4-bit

7.5.2 Πολυπλέκτης 8 σε 1

Στην παρούσα μελέτη θα αναλύσουμε τη λειτουργία του αναλογικού πολυπλέκτη, όπως για παράδειγμα του *74HT4051*. Ο πολυπλέκτης είναι ένα συνδυαστικό κύκλωμα, το οποίο επιλέγει τη δυαδική πληροφορία μιας από πολλές γραμμές εισόδου (κανάλια εισόδου) και την κατευθύνει σε μία μοναδική γραμμή εξόδου. Η επιλογή της συγκεκριμένης γραμμής εξόδου ελέγχεται από ειδικές γραμμές επιλογής. Κανονικά, υπάρχουν 2^n γραμμές εισόδου και n γραμμές επιλογής, των οποίων οι τιμές καθορίζουν ποια είσοδος επιλέγεται. Στην περίπτωση μας οι γραμμές εισόδου συμβολίζονται με D0-D7, ενώ οι γραμμές επιλογής είναι οι A, B και C. Η έξοδος του κυκλώματος είναι η Y, ενώ ως W συμβολίζεται η αντίστροφη τιμή της εξόδου. Η είσοδος $\overline{G}$ ενεργοποιεί ή απενεργοποιεί τη λειτουργία του πολυπλέκτη. Όταν βρίσκεται σε χαμηλή κατάσταση (LOW), η έξοδος του πολυπλέκτη είναι μηδέν, ενώ όταν είναι ένα (HIGH) ο πολυπλέκτης λειτουργεί κανονικά. Η λειτουργία του συνοψίζεται στον πίνακα αληθείας 7.9.

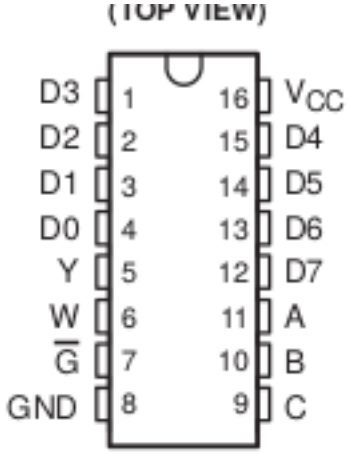
FUNCTION TABLE

INPUTS				OUTPUTS	
SELECT			STROBE $\overline{G}$	Y	W
C	B	A			
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

D0, D1 . . . D7 = the level of the respective D input

Σχήμα 7.9: Πίνακας Αληθείας Αναλογικού Πολυπλέκτη 8 σε 1

Το ολοκληρωμένο που χρησιμοποιήθηκε φαίνεται στο σχήμα 7.10 και περιλαμβάνει τις εισόδους/ εξόδους που αναλύθηκαν.

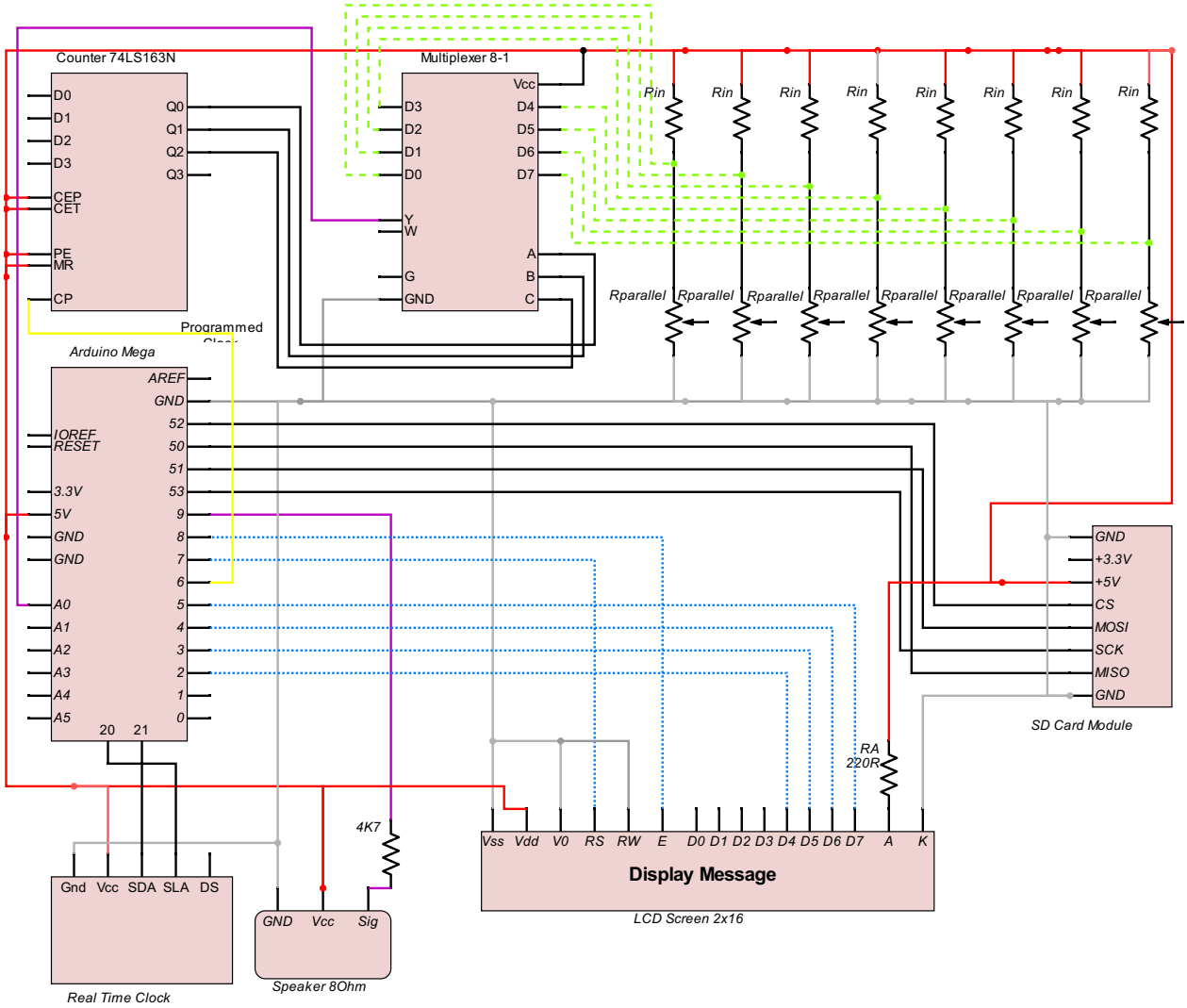


Σχήμα 7.10: Ολοκληρωμένο Πολυπλέκτη 8 σε 1

7.5.3 Σχεδιασμός Εναλλακτικής Λύσης

Η εκμετάλλευση λοιπόν του μετρητή και του πολυπλέκτη για τη μέτρηση της διαφοράς τάσης στο blister χαπιών πραγματοποιήθηκε ως εξής. Ένας παλμός που προερχόταν από το pin6 του Arduino Mega κατέληγε στο ρολόι του μετρητή. Με κάθε παλμό, λοιπόν, οι έξοδοι Q0, Q1 και Q2 μετρούσαν δυαδικά προς τα πάνω. Οι έξοδοι αυτοί συνδεόντουσαν άμεσα με τις γραμμές επιλογής A, B και C του πολυπλέκτη. Οι γραμμές εισόδου D0 – D7 μετρούσαν κάθε στιγμή την τάση από οχτώ ανεξάρτητες σειρές χαπιών. Θεωρήθηκε ότι η δομή των χαπιών στο blister αποτελούνταν από οχτώ όμοιες σειρές χαπιών, κάθε μία εκ των οποίων περιλάμβανε από μηδέν έως τρία χάπια. Συνεπώς, η έξοδος Y του πολυπλέκτη θα περιλαμβάνει τις τιμές των D0 – D7 με κάθε παλμό του ρολογιού και θα συνδέεται με την αναλογική θύρα A0 του Arduino. Οπότε σε οχτώ διαδοχικούς παλμούς του ρολογιού, η θύρα A0 θα έχει υπολογίσει την τάση σε κάθε μία σειρά χαπιών και θα εξάγει τον συνολικό αριθμό χαπιών.

Το συνολικό κύκλωμα παρουσιάζεται στο σχήμα 7.11.


Counter 74LS163N
D0
D1
D2
D3
CEP
CET
PE
MR
CP
Q0
Q1
Q2
Q3
Multiplexer 8-1
D3
D2
D1
D0
Y
W
G
GND
Vcc
D4
D5
D6
D7
A
B
C
Rin
Rparallel
Programmed
Arduino Mega
AREF
GND
IOREF
RESET
3.3V
5V
GND
GND
A0
A1
A2
A3
A4
A5
20
21
52
50
51
53
9
8
7
6
5
4
3
2
1
0
GND
+3.3V
+5V
CS
MOSI
SCK
MISO
GND
SD Card Module
RA
220R
4K7
Vss Vdd V0 RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
Display Message
LCD Screen 2x16
Gnd Vcc SDA SLA DS
Real Time Clock
GND Vcc Sig
Speaker 8Ohm

7.5.4 Μειονεκτήματα και Πλεονεκτήματα

Η κατασκευή της συσκευής με χρήση μετρητή και πολυπλέκτη διαθέτει κάποια πλεονεκτήματα και μειονεκτήματα σε σχέση με τη λύση που επιλέχθηκε. Ως πλεονεκτήματα αναφέρονται τα παρακάτω:

1. Ο αριθμός των χαπιών του Blister μπορεί να αυξηθεί ανάλογα με το μέγεθος του μετρητή και του πολυπλέκτη. Για παράδειγμα, στην παραπάνω περίπτωση θα μπορούσαμε να εκμεταλλευτούμε και την έξοδο Q3 του μετρητή, χρησιμοποιώντας έναν πολυπλέκτη 16 σε 1. Έτσι, θα είχαμε τη δυνατότητα να συμπεριλάβουμε πολύ μεγαλύτερες συσκευασίες χαπιών. Συγκεκριμένα, θα μπορούσαμε να έχουμε διπλάσιο αριθμό χαπιών με τις ίδιες συνθήκες.

2. Βελτιωμένη διακριτική ικανότητα. Εφόσον υπάρχει η δυνατότητα για πολλές διαφορετικές σειρές χαπιών (π.χ. 16), ο αριθμός των παράλληλων χαπιών σε κάθε σειρά θα μπορεί να μειωθεί ακόμα και σε δύο μόνο χάπια. Έτσι, η διαφορά τάσης που προκύπτει από την εξαγωγή ενός χαπιού θα είναι μεγαλύτερη.

3. Οικονομικότερη Συσκευή, αφού το Arduino Mega θα μπορούσε να αντικατασταθεί με ένα απλό Arduino Uno. Ο λόγος που αυτό δεν είναι εφικτό στη συσκευή μας είναι ότι η χρησιμότητα όλων των αναλογικών θυρών A0-A5 έρχεται σε σύγκρουση με τη λειτουργία του I^2C πρωτοκόλλου, το οποίο απαιτεί επίσης τις αναλογικές θύρες A4 και A5.

Αντίθετα, τα μειονεκτήματα είναι τα εξής:

1. Αύξηση της περιπλοκότητας του κυκλώματος εξαιτίας των δύο επιπλεόν κυκλωμάτων. Το κύκλωμα είναι πιο ευάλωτο σε εσωτερικά βραχυκυκλώματα.

2. Ανάλογα με το μέγεθος του πολυπλέκτη, θα απαιτείται τύπωμα πιο πολλών αγώγιμων γραμμών στο Blister. Στη λύση που επιλέχθηκε απαιτείται τύπωμα οχτώ διαφορετικών αγώγιμων γραμμών, ενώ στο κύκλωμα με χρήση πολυπλέκτη θα χρειάζονται τουλάχιστον δέκα (οχτώ έξοδοι του πολυπλέκτη, η γείωση και η τροφοδοσία).

3. Μεγαλύτερη Κατανάλωση λόγω της χρήσης των δύο επιπλέον ολοκληρωμένων.

Συνολικά, λαμβάνοντας υπόψη τη σπουδαιότητα κατασκευής μιας συσκευής χαμηλής πολυπλοκότητας και κατανάλωσης απορρίφθηκε η συγκεκριμένη ιδέα.

Κεφάλαιο 8

Πειραματική Αξιολόγηση

Η λειτουργία της εξειδικευμένης συσκευής συστηματικής παρακολούθησης και βελτίωσης της φαρμακευτικής συμμόρφωσης αξιολογήθηκε πειραματικά με τη διεξαγωγή ποικίλων σεναρίων.

8.1 Οργάνωση πειραμάτων

Τα σενάρια, τα οποία ελέγθηκαν πειραματικά είναι τα εξής:

1. Το blister περιλαμβάνει αριθμό χαπιών μεγαλύτερο του ενός. Η ώρα λήψης χαπιού δεν έχει φτάσει.
2. Ο αριθμός των χαπιών παραμένει θετικός, αλλά ο ασθενής έχει ήδη λαάβει το χάπι του πρωτού να χτυπήσει η ειδοποίηση.
3. Αντίστοιχα με το προηγούμενο σενάριο, η ώρα λήψης χαπιού έχει φτάσει και ο ασθενής ειδοποιείται από τη συσκευή και στη συνέχεια λαμβάνει την αγωγή του.
4. Το blister περιλαμβάνει ακριβώς ένα χάπι.
5. Το blister είναι άδειο.

8.2 Σύστημα αξιολόγησης

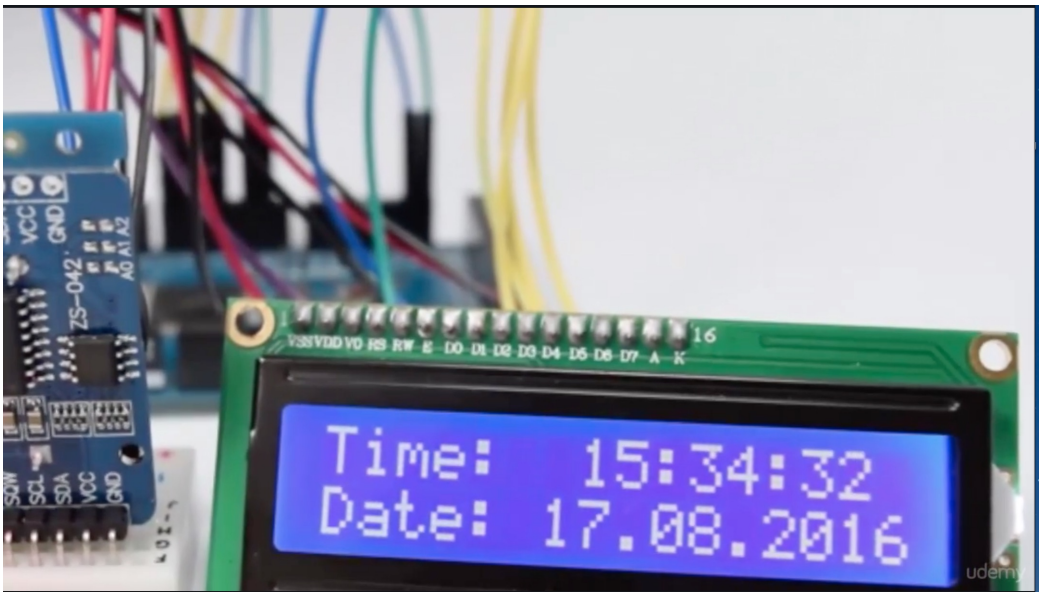
Η συσκευή αξιολογήθηκε ως προς τη λειτουργικότητά της σε σχέση με το σενάριο που θα πρέπει να υλοποιήσει. Σε όλα τα παραπάνω σενάρια, η ειδοποίηση του ασθενή επιτεύχθηκε τις κατάλληλες χρονικές στιγμές. Ακόμα, τα μηνύματα που προβλήθηκαν στην LCD οθόνη αντικατοπτρίζουν ακριβώς τις λειτουργίες της συσκευής και ο χρήστης θα γνωρίζει τις χρονικές στιγμές λήψης των φαρμάκων του.

8.3 Ανάλυση πειραμάτων

Αρχικά με την τροφοδοσία της συσκευής το μήνυμα αρχικοποίησης θα εμφανιστεί στην οθόνη (Σχήμα 8.1.)

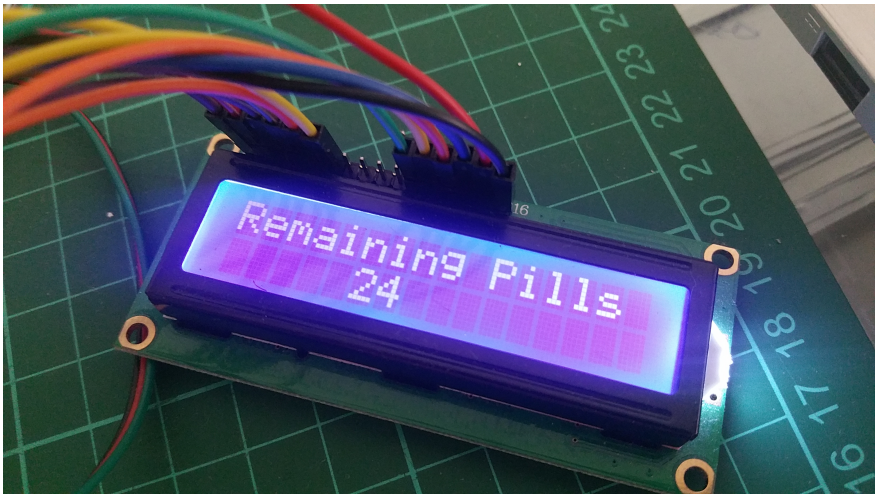
Σχήμα 8.1: Αρχικοποίηση Συσκευής

Ο επιτυχημένος συγχρονισμός του Arduino με το RTC θα αναπαρασταθεί στην οθόνη της συσκευής με μήνυμα την τρέχουσα ημερομηνία. (Σχήμα 8.2).



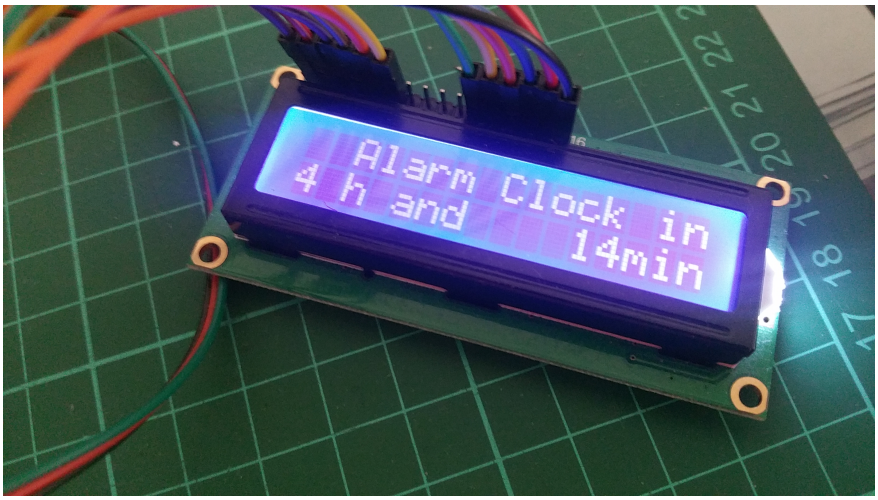
Σχήμα 8.2: Εμφάνιση Ημερομηνίας και Ώρας στην LCD Οθόνη

Στη συνέχεια, θα εμφανιστεί ο υπολειπόμενος αριθμός χαπιών (Σχήμα 8.3).



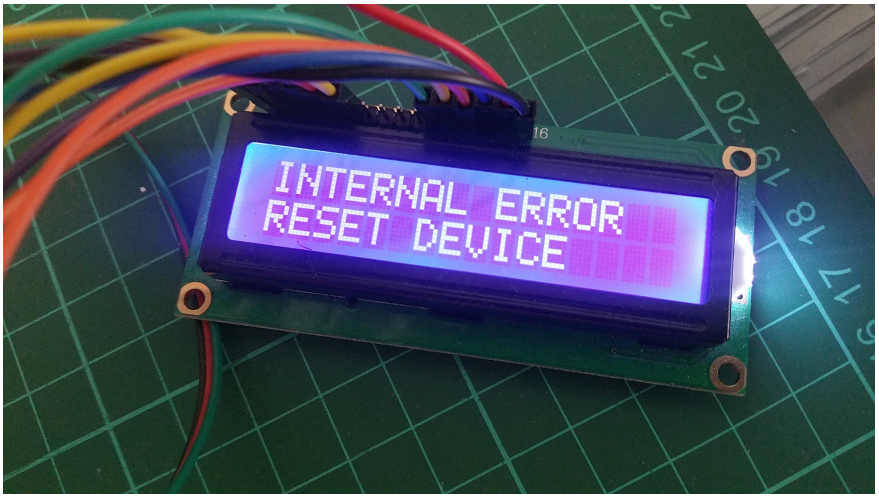
Σχήμα 8.3: Εμφάνιση Υπολειπόμενου Αριθμού Χαπιών στην LCD

Ακόμα, ο χρήστης θα πρέπει να γνωρίζει την ώρα μέχρι τη λήψη της αγωγής του. Για το λόγο αυτό, θα εμφανιστεί αντίστοιχο μήνυμα στην οθόνη (Σχήμα 8.4).



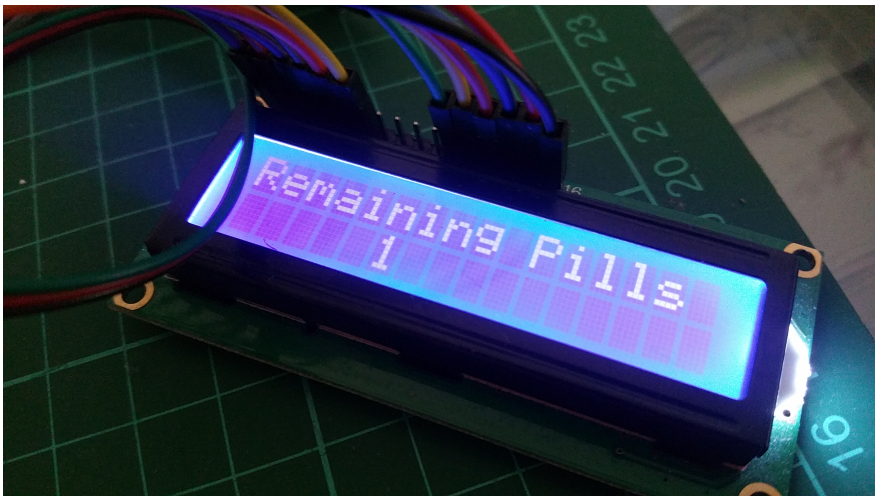
Σχήμα 8.4: Εμφάνιση Ώρας μέχρι τη λήψη χαπιού

Σε οποιαδήποτε περίπτωση εσωτερικής βλάβης, το μήνυμα 8.5 θα εμφανιστεί στην οθόνη.

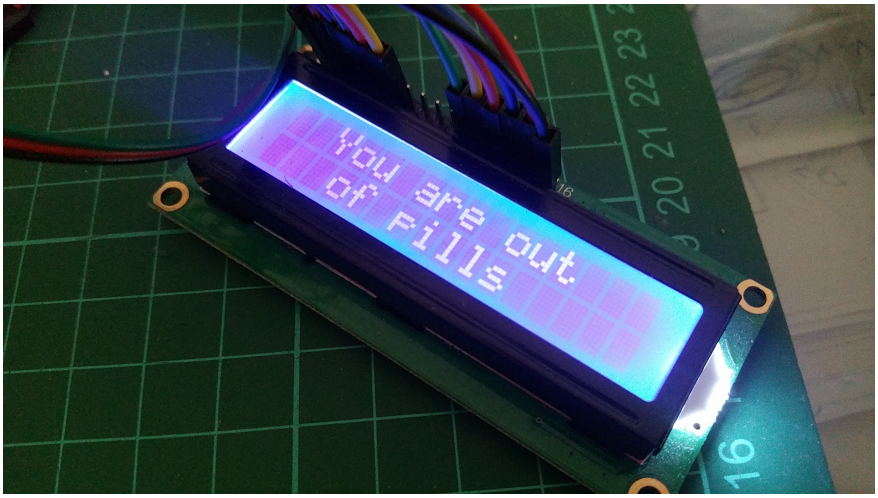


Σχήμα 8.5: Εμφάνιση Μηνύματος Εσωτερικού Προβλήματος Συσκευής

Σε περίπτωση που ο ασθενής έχει μόνο ένα περισσευούμενο χάπι (Σχήμα 8.6), τότε η συσκευή θα τον ειδοποιήσει ότι τα χάπια του τελειώνουν (Σχήμα 8.7).

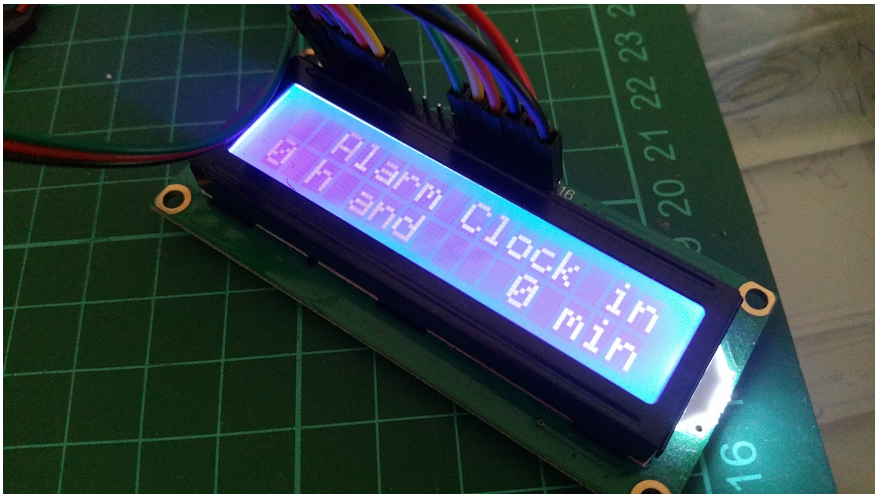


Σχήμα 8.6: Ένα υπολειπόμενο Χάπι



Σχήμα 8.7: Τελείωμα φαρμάκων στο blister

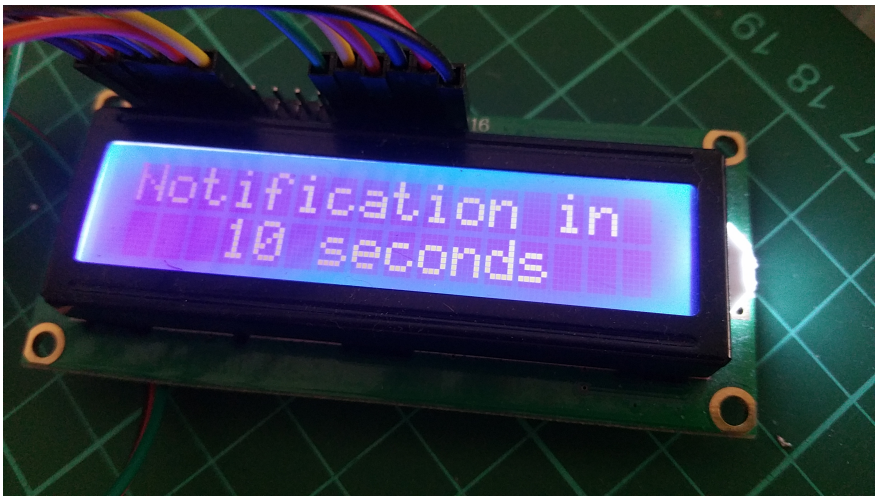
Όταν φτάσει η στιγμή λήψης κάποιου χαπιού από τον ασθενή, η συσκευή θα εμφανίσει αρχικά ότι περισσεύουν *0hours* και *0minutes* για την έναρξη της ειδοποίησης (Σχήμα 8.8) και στη συνέχεια το μήνυμα *"Time to take your pills"* θα εμφανιστεί (Σχήμα 8.9).



Σχήμα 8.8: Άμεση Ειδοποίηση Ασθενή

Σχήμα 8.9: Εμφάνιση Μηνύματος Ειδοποίησης Ασθενή

Εφόσον, ο ασθενής δε λάβει το χάπι του, η συσκευή θα επαναλάβει τη διαδικασία αφύπνισης σε δέκα δευτερόλεπτα. Ακόμα, αντίστοιχο μήνυμα θα εμφανιστεί και στην οθόνη (Σχήμα 8.10).

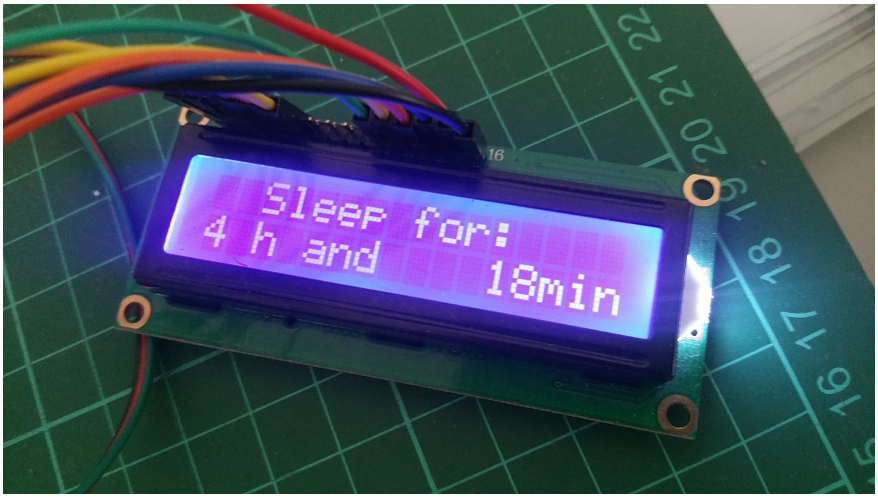


Σχήμα 8.10: Επανάληψη ειδοποίησης κάθε 10 δευτερόλεπτα

Τη στιγμή που θα ασθενής θα λάβει το χάπι του, η συσκευή θα εμφανίσει μήνυμα επιβράβευσης (Σχήμα 8.11).

Σχήμα 8.11: Συγχαρητήριο Μήνυμα

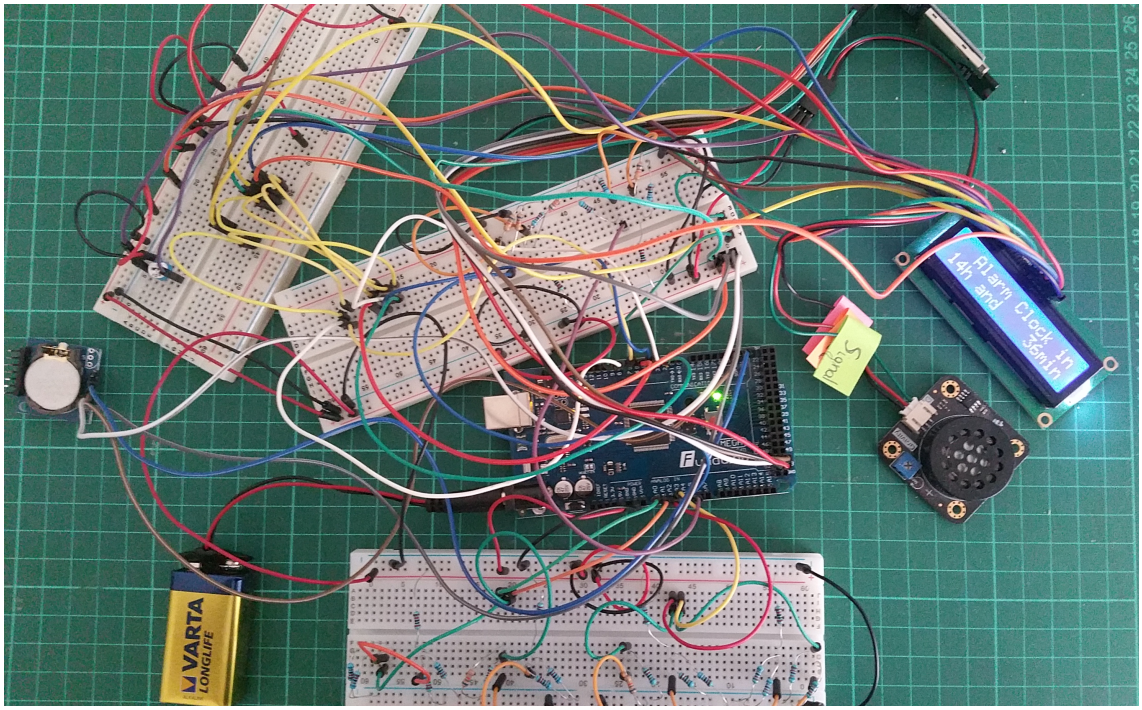
Τέλος, εφόσον η επόμενη ειδοποίηση είναι σε χρονικό διάστημα μεγαλύτερο των δέκα λεπτών, η συσκευή θα εισέλθει σε *Sleep Mode*. Ο χρόνος αδράνειας της συσκευής θα εμφανιστεί στην οθόνη (Σχήμα 8.12).



Σχήμα 8.12: Εμφάνιση Χρόνου Αδράνειας της Συσκευής

Κεφάλαιο 9

Επίλογος



Σχήμα 9.1: Εικόνα Πρότυπης Συσκευής

9.1 Σύνοψη και συμπεράσματα

Η εξειδικευμένη συσκευή αποτελεί μια πρωτοπόρα λύση στη εξάλειψη του προβλήματος της φαρμακευτικής συμμόρφωσης. Η χρήση των δυνατοτήτων του Arduino, σε συνδυασμό με το ραγδαία αναπτυσσόμενο κόσμο του IoT(Internet of Things) μπορούν να οδηγήσουν στην εξολοκλήρου κατασκευή εξαρτημάτων χαμηλού κόστους, που θα συνεισφέρουν άμεσα στη ζωή των ασθενών. Ακόμα, η προτεινόμενη πλατφόρμα μπορεί να προωθηθεί άμεσα σε μαζική παραγωγή, καθώς είναι η μοναδική συσκευή που συνδυάζει τόσο το χαμηλό κόστος όσο και τις πολλαπλές λειτουργίες αφύπνισης και μεταφοράς δεδομένων σε πραγματικό χρόνο. Ακόμα, οι δυνατότητες επέκτασης των δυνατοτήτων της συσκευής είναι ποικίλες καθώς ο τομέας του

IoT αναπτύσσεται μέρα με τη μέρα.

Συνολικά, οι συνεισφορά της συσκευής σε κοινωνικό και οικονομικό τομέα συνοψίζονται παρακάτω.

- Η προγραμματισμένη και με συνέπεια λήψη της φαρμακευτικής αγωγής οδηγεί στη γρηγορότερη ανάρρωση των ασθενών
- Ο ασθενής θα έχει προσαρτημένη κάθε συσκευασία χαπιών σε μία συσκευή. Συνεπώς, η κάθε συσκευή θα τον ενημερώνει για την ώρα λήψης των χαπιών του blister που αυτή είναι συνδεδεμένη. Έτσι, θα αποφεύγεται ο κίνδυνος λανθασμένης λήψης φαρμάκων ή πιθανής σύγχυσης του ασθενή. Η ακούσια λήψη λανθασμένης αγωγής αποτελεί ένα συχνό φαινόμενο, ιδιαίτερα στους ηλικιωμένους.
- Ο γιατρός θα γνωρίζει απευθείας αν η συγκεκριμένα φαρμακευτική θεραπεία είναι αποτελεσματική στον ασθενή. Έτσι, θα αποφεύγεται τυχόν επιβάρυνση της υγείας του με επιπλέον χάπια λόγω της μη φαρμακευτικής συμμόρφωσης του ασθενή.

9.2 Μελλοντικές επεκτάσεις

Το κύριο μέρος της συσκευής που χρειάζεται υλοποίηση, αποτελεί την προθήκη ενός WiFi Module, που θα επιτρέπει την ασύρματη επικοινωνία της με τον server στον οποίο θα αποστέλλονται οι ώρες λήψης των φαρμάκων σε πραγματικό χρόνο. Η λειτουργικότητα αυτή μπορεί εύκολα να επιτευχθεί με την προσθήκη του ESP Module. Οι προσχεδιασμένες βιβλιοθήκες που υπάρχουν, μπορούν να προσφέρουν άμεσα αυτή τη λειτουργικότητα.

Ακόμα, είναι εφικτό να επεκταθούν οι δυνατότητες του φαρμακοποιού ώστε να μπορεί να συμπληρώνει περισσότερες δοσολογίες τη μέρα, όπως και να επιλέγει συγκεκριμένες ημέρες την εβδομάδα.

Τέλος, το ήδη υπάρχον οπτικοακουστικό υλικό που χρησιμοποιείται θα μπορούσε να εξελιχθεί τόσο ως προς τις δυνατότητες της LCD οθόνης όσο και ως προς ανάλυση του ηχείου.

9.3 Σκέψεις/Ιδέες για Βελτίωση της Συσκευής

Η προτεινόμενη συσκευή δεν αρκεί να είναι μόνο χρηστική. Οφείλει να έχει και ενδιαφέρουσα εξαρτήματα που θα προκαλούν το χρήστη να την προτιμήσει. Για το λόγο αυτό θα προταθούν πιθανές ιδέες βελτίωσης της συσκευής.

1. Η κάρτα μνήμης μπορεί να συμπεριλαμβάνει διάφορα αρχεία μουσικής/ αφύπνισης. Με την προσθήκη λοιπόν ενός κουμπιού θα μπορούσε ο ασθενής να επιλέξει τον ήχο ειδοποίησης που προτιμάει. Έτσι, κατά το στάδιο της αρχικοποίησης, θα ακουγόντουσαν ένας ένας οι διάφοροι ήχοι και ο ασθενής με το πάτημα του κουμπιού θα μπορούσε να επιλέξει. Ασφαλώς, σε περίπτωση που ο χρήστης δεν πατούσε το κουμπί, θα υπήρχε ήχος προεπιλογής.

2. Η συσκευή μπορεί να επεκταθεί ώστε να αποστέλλει άμεσα e-mails τόσο στο γιατρό όσο και σε οικογενειακά πρόσωπα του ασθενή σε τακτά χρονικά διαστήματα.

3. Η προσθήκη ενός led μεγάλων διαστάσεων, το οποίο αναβοσβήνει τις στιγμές λήψης χαπιών των ασθενών θα συνέβαλε στη βελτίωση του οπτικοακουστικού ερεθίσματος.

4. Η εργασία της διακοπής που επιτελεί ο Watchdog Timer θα μπορούσε να αντικατασταθεί από την ενσωματωμένη λειτουργικότητα ειδοποιήσεων του Real Time Clock. Έτσι, η περίοδος συνεχόμενης αδράνειας της συσκευής θα μπορούσε να είναι ολόκληρες ώρες και όχι μόνο οχτώ δευτερόλεπτα.

5. Σημαντική προσθήκη στη συσκευή θα αποτελούσε η δυνατότητα αλληλεπίδρασής της με το χρήστη/ασθενή. Αναλυτικότερα, ο ασθενής θα μπορούσε να μεταβάλει χειροκίνητα τις ώρες λήψης των χαπιών του σε περιπτώσεις που δεν θα μπορούσε να αναβάλει. Με την προσθήκη, λοιπόν, επιπλέον κουμπιών και τη χρήση της υπάρχουσας οθόνης είναι εφικτό να διαμορφώνει ο χρήστης όλες τις default αρχικοποιήσεις.

6. Οι φορητές συσκευές λόγω του μικρού μεγέθους τους συχνά χάνονται. Η λύση σε αυτό το πρόβλημα είναι η τοποθέτηση ενός στίγματος στη συσκευή, ώστε ο χρήστης να γνωρίζει πάντα την τοποθεσία της. Η λειτουργικότητα αυτή μπορεί να υλοποιηθεί με την προσθήκη μιας κάρτας sim και ενός module, το οποίο θα εξυπηρετεί την GPS τεχνολογία.

7. Τέλος, η ασύρματη επικοινωνία της συσκευής θα μπορούσε να επεκταθεί και στο κινητό του χρήστη μέσω μιας αντίστοιχης εφαρμογής. Έτσι, ο χρήστης θα μπορούσε να διατηρεί το προσωπικό του ημερολόγιο και να γνωρίζει το ποσοστό βελτίωσης της φαρμακευτικής του συμμόρφωσης.

Κεφάλαιο 10

Παράρτημα 1

Algorithm 29 *Επιλογή Κατάλληλης Τιμής Αντιστάσεων*

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np
from numpy import unravel_index
import matplotlib.ticker as mtick
from mpl_toolkits.mplot3d import proj3d

def fun(X, Y):
Df = abs(5.0 * (X / 4) / (Y + X / 4) - 5.0 * (X / 3) / (Y + X / 3))
return np.array(Df)

X = np.linspace(220, 100000, 6500)
Y = np.linspace(220, 100000, 6500)
X,Y=np.meshgrid(X,Y)
Df = fun(X, Y)
fig = plt.figure()
ax = fig.gca(projection='3d')
surf = ax.plot_surface(X, Y, Df, cmap=cm.coolwarm, linewidth=0, antialiased=False)
ax.set_zlim(0, 0.4)
ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))
ax.set_xlim([220, 100000])
ax.set_ylim([220, 100000])
ax.get_xaxis().set_major_formatter(mtick.FuncFormatter(lambda x, p: format(int(x)/1000)))
ax.get_yaxis().set_major_formatter(mtick.FuncFormatter(lambda x, p: format(int(x)/1000)))

fig.colorbar(surf, shrink=0.5, aspect=5)
max_i = np.amax(Df)
ind = np.argmax(Df)
index = np.unravel_index(Df.argmax(), Df.shape)
indexX = index[0]
indexY = index[1]
print(index)
Rin = "{:10.2f}".format(X[0,indexX])
Rp = "{:10.2f}".format(X[0,indexY])
max_xIndex = "{:10.2f}".format(float(Rin)/1000)
max_yIndex = "{:10.2f}".format(float(Rp)/1000)
plt.title("Εὑρεση Ιδανικῶν Τιμῶν Αντιστσεων",style='italic',fontsize=17)
ax.set_xlabel('Αντίσταση Εισόδου(KVolts)')
ax.set_ylabel('Αντίσταση Ϙαπιοὐ(KVolts)')
ax.set_zlabel('Μέγιστη Διαφορὰ Τάσης')
plt.show()
```

Βιβλιογραφία

[1] Van Onzenoort et al, Determining the feasibility of objective adherence measurement with blister packaging smart technology. In *Am J Health-Syst Pharm69* pp872-879, 2012.

[2] Sabate E, ed. Adherence to Long-Term Therapies: Evidence for Action. Geneva, Switzerland. In *World Health Organization* 2003..

[3] Lee JK, Grace KA, Taylor AJ. Effect of a pharmacy care program on medication adherence and persistence, blood pressure, and low-density lipoprotein.

[4] Cholesterol: a randomized controlled trial In *JAMA*, 2006;296(21):2563-2571.

[5] Osterberg L, Blaschke T. Adherence to medication. In *N Engl J Med*, 2005;353(5):487-497.

[6] Huth EJ, Murray TJ, eds. .Medicine in Quoteations: Views of health and Disease Through the Ages. 2nd ed. Philadelphia. In *PA: American College of Physicians*, 2006

[7] Winkler A, Teuscher AU, Mueller B, Diem P. Monitoring adherence to prescribed medication in type 2 diabetic patients treated with sulfonylureas. In *Swiss Med Wkly*, 2002;132(27-28):379-385.

[8] Timmreck TC, Randolph JF. Smoking cessation: clinical steps to improve compliance. Geriatrics, 1993, 48:63-66.

[9] Norell SE. Accuracy of patient interviews and estimates by clinical staff in determining medication compliance. In *Social Science & Medicine - Part E, Medical Psychology*, 1981,15:57-61

[10] Cramer JA, Mattson RH. Monitoring compliance with antiepileptic drug therapy. In *Cramer JA, Spilker B, eds. Patient compliance in medical practice andclinical trials. New York, Raven Press*, 1991:123-137.

[11] Spector SL et al. Compliance of patients with asthma with an experimental aerosolized medication: implications for controlled clinical trials. In *Journal of Allergy & Clinical Immunology*, 1986, 77:65-70.

[12] Morisky DE, Green LW, Levine DM. Concurrent and predictive validity of a self-reported measure of medication adherence.Medical Care, 1986, 24:67-74.

[13] Freudenheim JL.A review of study designs and methods of dietary assessment in nutritional epidemiology of chronic disease. In *Journal of Nutrition*, 1993, 123:401-405..

[14] Farmer KC. Methods for measuring and monitoring medication regimen adherence in clinical trials and clinical practice. In *Clinical Therapeutics*, 1999, 21:1074-1090.

[15] DiMatteo MR, DiNicola DD.Achieving patient compliance In *New York, Pergamon*, 1982.

[16] Vitolins MZ et al. Measuring adherence to behavioral and medical interventions. Controlled Clinical Trials, 2000, 21:188S-194S..

[17] Kripalani S, Yao X, Haynes RB. Interventions to enhance medication adherence in chronic medical conditions: a systematic review. In *Archives of Internal Medicine*, 2007;167:540–50.

[18] Garfield S, Barber N, Walley P, Willson A, Eliasson L.Quality of medication use in primary care—mapping the problem, working to a solution: a systematic review of the literature. In *BMC Medicine*, 2009;7:50..

[19] Saini SD, Schoenfeld P, Kaulback K, Dubinsky MC. Effect of medication dosing frequency on adherence in chronic diseases. In *American Journal of Managed Care*, 2009;15:22–33.

[20] Goundrey-Smith SJ. Principles of electronic prescribing. In *Springer*, 2008, p142.

[21] Matsuyama JR, Mason BJ, Jue SG. Pharmacists' interventions using an electronic medication monitoring device's adherence data versus pill counts. In *Annals of Pharmacotherapy*, 1993;27:851–5.

[22] Windows 10 for IoT. In *Raspberry Pi Foundation*, 30 April 2015.

[23] Raspberry Pi 3 is out now! Specs, Benchmarks & More. In The MagPi Magazine, 1 April 2016.

[24] Raspberry Pi FAQs – Frequently Asked Questions, Retrieved 8 April 2017.

[25] Raspberry Pi downloads". Retrieved 12 August 2016.

[26] Usage – Raspberry Pi Documentation, raspberrypi.org. Retrieved 12 August 2016..

[27] Blobless Linux on Raspberry Pi (rpi-open-firmware). Retrieved 20 July 2017..

[28] Synchronous counter, Digital, Play hookey.

[29] Singh, Arun Kumar (2006). Digital Principles Foundation of Circuit Design and Application. In New Age Publishers.,ISBN 81-224-1759-0.

[30] Horowitz, Paul; Hill, Winfield (1989). The Art of Electronics. In Cambridge University Press., ISBN 0-521-37095-7.

[31] Graf, Rudolf F (1999). Modern Dictionary of Electronics. In Newnes, ISBN 0-7506-9866-7.

[32] Dean, Tamara (2010). Network+ Guide to Networks. In Delmar, pp. 82–85.

[33] Debashis, De (2010). Basic Electronics. In Dorling Kindersley, p.557.

[34] Lipták, Béla (2002). Instrument engineers' handbook: Process software and digital networks. In CRC Press, p. 343.

[35] Harris, David (2007). Digital Design and Computer Architecture In Penrose, p. 79.

[36] Crowe, John and Barrie Hayes-Gill (1998) Introduction to Digital Electronics pp. 111-113.

[37] What is Serial Synchronous Interface (SSI)? Retrieved 2015-01-28.

[38] SPI Block Guide v3.06 In Motorola/Freescale/NXP, 2003.

[39] Better SPI Bus Design in 3 Steps, dorkbot pdx. Retrieved 3 September 2015.

[40] Soroka, W. Illustrated Glossary of Packaging Terminology (Second ed.). In Institute of Packaging Professionals.

[41] Pilchik, R (Nov 2000), Pharmaceutical Blister Packaging, Part 1, Rationale and Materials. In Pharmaceutical Technology, 68–77, retrieved 26 June 2017.

[42] Pilchik, R (Dec 2000), Pharmaceutical Blister Packaging, Part 2, Machinery and Assembly. In Pharmaceutical Technology, 56–60, retrieved 26 June 2017.

[43] https://www.baldengineer.com/arduino-f-macro.html.

[44] Sagar Khillar. Difference between Von Neumann and Harvard Architecture. In DifferenceBetween.net, March 26, 2018.