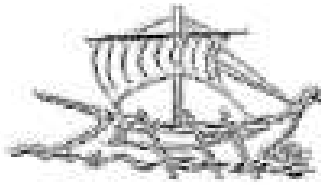


ΤΕΙ ΠΕΙΡΑΙΑ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Π.Μ.Σ. "ΕΦΑΡΜΟΣΜΕΝΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ελεγκτής συσκευών με τεχνολογία GSM

Παναγιώτης Β. Καραγκούνης
Α.Μ : ais0075

Εισηγητής: Έλληνας Ιωάννης, Καθηγητής

ΑΙΓΑΛΕΩ
ΙΑΝΟΥΑΡΙΟΣ 2017

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ελεγκτής συσκευών με τεχνολογία GSM

Παναγιώτης Β. Καραγκούνης
Α.Μ. ais0075

Εισηγητής:

Έλληνας Ιωάννης , Καθηγητής

Εξεταστική Επιτροπή:

Ημερομηνία εξέτασης

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/Η κάτωθι υπογεγραμμένος Παναγιώτης Καραγκούνης,
του Βασιλείου Καραγκούνη, με αριθμό μητρώου ais0075 φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

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

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

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

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία πραγματοποιήθηκε έπειτα από επίπονες προσπάθειες και αδιάκοπη μελέτη των επί μέρους τεχνολογιών οι οποίες και χρησιμοποιήθηκαν για την περάτωση της.

Ευχαριστώ ιδιαιτέρως την γυναίκα μου και την κόρη μου για την αμέριστη συμπαράσταση και υπομονή την οποία έδειξαν απέναντι μου με δεδομένο τις πολλές ώρες απουσίας μου από κοντά τους για τις ανάγκες ολοκλήρωσης του μεταπτυχιακού προγράμματος αλλά και για την περάτωση της διπλωματικής μου εργασίας. Η συμβολή τους ήταν καθοριστική και για αυτό τους αφιερώνω όλη την εργασία μου με αγάπη.

Επίσης θα ήθελα να ευχαριστήσω θερμά τον Δρ. Σαπουντζάκη Δημήτριο ,ο οποίος είναι συνάδελφος μου αλλά και πολύ καλός μου φίλος. Είναι επίσης ο λόγος για τον οποίο ξεκίνησα την παρακολούθηση του μεταπτυχιακού προγράμματος διότι είναι ο άνθρωπος ο οποίος με ενέπνευσε και με έκανε να αγαπήσω την επιστήμη της πληροφορικής. Η διάθεση του για την πρόοδο των επιστημών γενικότερα είναι μοναδική και με κάνει να νιώθω τυχερός που τον γνώρισα διότι με έκανε και καλύτερο επιστήμονα αλλά και άνθρωπο.

Τέλος θα ήθελα να ευχαριστήσω τον Κο Έλληνα Ιωάννη για τις πολύτιμες συμβουλές του καθ’ όλη την διάρκεια του μεταπτυχιακού προγράμματος αλλά και την βοήθεια του για την περάτωση της διπλωματικής εργασίας.

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία ασχολείται με την ανάπτυξη ενός προϊόντος το οποίο θα χρησιμοποιεί το δίκτυο κινητής τηλεφωνίας ώστε να μπορεί να ελέγξει απομακρυσμένα ηλεκτρικές συσκευές εντός του χώρου όπου πρόκειται να εγκατασταθεί. Το προϊόν θα διαθέτει επίσης την δυνατότητα για τοπικό έλεγχο των συσκευών και επικοινωνίας σε περίπτωση έκτακτης ανάγκης.

Η επικοινωνία με την συσκευή θα πραγματοποιείτε με τους παρακάτω τρόπους:

1. Μέσω γραπτού μηνύματος (SMS)
2. Μέσω τηλεφωνικής κλήσης.
3. Μέσω ηλεκτρονικού ταχυδρομείου (e-mail).

Η συγκεκριμένη συσκευή αναπτύχθηκε με τέτοιο τρόπο ώστε να μπορεί να απευθύνετε σε όσο το δυνατόν ευρύ κοινό λαμβάνοντας υπόψη τις σύγχρονες ανάγκες που έχουμε όλοι μας για αυτοματοποιημένες λειτουργίες και απομακρυσμένη διαχείριση με χαμηλό κόστος αγοράς. Επίσης είναι ένα προϊόν το οποίο έχει μεγάλη χρηστικότητα σε ομάδες ανθρώπων με κινητικές δυσκολίες καθώς και ηλικιωμένους διότι δίνει την δυνατότητα της επικοινωνίας σε περίπτωση ανάγκης.

Επίσης έχει χαμηλή κατανάλωση σε ηλεκτρική ενέργεια αλλά και διακριτική εμφάνιση για τοποθέτηση της σε εσωτερικούς χώρους.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Μικρουπολογιστικά Συστήματα.
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Mobile Networks, Arduino, GSM, SIM900, RF, Embedded hardware.

ABSTRACT

This thesis deals with the development of a product that uses the mobile phone network in order to remotely control electrical devices within the space where they are installed. The product will also have the ability for local control and communication in case of emergency for the user.

The communication with the device will be performed in the following ways:

1. Via text message (SMS)
2. Via telephone call.
3. Via e-mail (e-mail).

This device was developed in such a way that it can be apply in a wide range of people taking into account the current needs that we all have for automated functions and remote management with low purchase cost. It is a product which has high usability in groups of people with disabilities and the elderly because it gives the possibility of communication in an emergency case.

It also has a low consumption of electricity and discreet appearance for indoor placement.

SCIENTIFIC AREA: Microcomputer Systems.

KEYWORDS: Mobile Networks, Arduino, GSM, SIM900, RF, Embedded hardware.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

GSM	Global System for Mobile Communications
GPRS	General Packet Radio Service
SMS	Short Message Service
A.C	Alternative Current
D.C	Directive Current
SIM	Subscriber Identity Module
UMTS	Universal Mobile Telecommunications System
HSDPA	High Speed Downlink Packet Access
LTE	Long Term Evolution
MMS	Multimedia Messaging Service
TDMA	Time Division Multiple Access
BTS	Base Transceiver Station
BSC	Base Station Controller
MSC	Mobile Switching Center
EIR	Equipment Identity Register
AUC	Authentication Center
HLR	Home Location Register
VLR	Visitor Location Register
GMSC	Gateway Mobile Switching Centre
SGSN	Serving GPRS Support Node
GGSN	Gateway GPRS Support Node
RAM	Random Access Memory
EEPROM	Electrically Erasable Programmable Read Only Memory
UART	Universal Asynchronous Receiver/Transmitter
I/O	Input / Output
LED	Light Emitting Diode
ROM	Read Only Memory
SRAM	Static Random Access Memory
IDE	Integrated Development Environment

ΚΕΦΑΛΑΙΟ 1
ΕΙΣΑΓΩΓΗ

Η εξέλιξη της τεχνολογίας τα τελευταία χρόνια τόσο στις τηλεπικοινωνίες όσο και στα ηλεκτρονικά έχει οδηγήσει στην ανάπτυξη μιας βιομηχανίας η οποία παράγει συσκευές και εφαρμογές οι οποίες είναι σε όλους μας πλέον απαραίτητες στην καθημερινότητα μας. Καθημερινά όλοι μας πλέον χρησιμοποιούμε κινητές συσκευές οι οποίες μας δίνουν την δυνατότητα να επικοινωνούμε μεταξύ μας με διαφόρους τρόπους όπως με κλήση φωνής, γραπτά μηνύματα (SMS), e-mail αλλά και την δυνατότητα του να μπορούμε να συνδεθούμε με το πάτημα ενός πλήκτρου στο διαδίκτυο και να έχουμε πρόσβαση σε οτιδήποτε.

Σε αυτό το κεφάλαιο παρουσιάζετε το αντικείμενο της παρούσας εργασίας καθώς και μια ιστορική αναδρομή με την εξέλιξη αντιστοίχων συσκευών στον χώρο. Επίσης παρουσιάζετε συνολικά η δομή της εργασίας ανά κεφάλαιο.

1.1 Αντικείμενο διπλωματικής εργασίας

Η παρούσα διπλωματική καλείτε να μελετήσει και να αναπτύξει ένα πρωτότυπο συσκευής με την οποία ο χρήστης του θα μπορεί να ελέγξει απομακρυσμένα άλλες συσκευές εντός της οικίας του (π.χ. τον θερμοσίφωνα στο σπίτι του) με χρήση της τεχνολογίας GSM. Επίσης θα υπάρχει η δυνατότητα επικοινωνίας του χρήστη με προκαθορισμένα άτομα σε περίπτωση ανάγκης.

Πιο συγκεκριμένα ο χρήστης θα μπορεί να ελέγχει απομακρυσμένα συσκευές με την αποστολή γραπτών μηνυμάτων (SMS). Οι τρόποι με τους οποίους ο χρήστης θα μπορεί να επικοινωνεί με άλλους σε περιπτώσεις ανάγκης είναι οι ακόλουθοι:

1. Αποστολή γραπτού μηνύματος (SMS).
2. Τηλεφωνική κλήση με χρήση προ-ηχογραφημένου μηνύματος.
3. Αποστολή e-mail.

Αν και υπάρχουν στο εμπόριο αντίστοιχοι ελεγκτές το βασικό τους μειονέκτημα είναι ότι είναι υλοποιημένοι έτσι ώστε μόνο να δέχονται γραπτά μηνύματα και να εκτελούν τις αντίστοιχες εντολές από τον χρήστη, χωρίς όμως να έχουν την δυνατότητα να λειτουργήσουν και ως συσκευές επικοινωνίας. Επίσης το κόστος τέτοιων συσκευών κυμαίνεται από 100 έως 800 ευρώ πράγμα το οποίο τις

καθιστά ακριβές στην αγορά τους και η χρήση τους περιορίζετε σε εξειδικευμένες εφαρμογές κυρίως στον βιομηχανικό κλάδο.

1.2 Δομή της διπλωματικής εργασίας

Στα κεφάλαια που ακολουθούν γίνετε αναλυτική παρουσίαση του υλικού (hardware) που χρησιμοποιήθηκε αλλά και του λογισμικού (software) το οποίο αναπτύχθηκε για την παρούσα διπλωματική εργασία.

Η διπλωματική εργασία απαρτίζετε από έξι κεφάλαια, καθένα από τα οποία άπτεται ζητήματα τα οποία αφορούν την παρούσα διπλωματική εργασία.

Στο παρών κεφάλαιο γίνεται η παρουσίαση του αντικειμένου της διπλωματικής εργασίας καθώς και μια σύνοψη της δομής της ανά κεφάλαιο.

Στο δεύτερο κεφάλαιο αναλύεται το θεωρητικό υπόβαθρο από το οποίο αντλήθηκαν όλες οι πληροφορίες και η γνώση για την κατανόηση και την ανάπτυξη της εργασίας. Πιο συγκεκριμένα γίνετε αναφορά στην βασική θεωρεία ραδιοσυχνοτήτων, στην τεχνολογία GSM και τέλος στους μικροελεγκτές.

Στο τρίτο κεφάλαιο αναλύουμε την αρχιτεκτονική του συστήματος με τα επί μέρους τμήματα από τα οποία αποτελείτε και τους λόγους για τα οποία επιλέχθηκαν.

Στο τέταρτο κεφάλαιο παρουσιάζετε η μεθοδολογία η οποία ακολουθήθηκε βήμα προς βήμα για την ανάπτυξη της συσκευής.

Στο πέμπτο κεφάλαιο παρουσιάζεται η σχεδίαση της πρωτότυπης πλακέτας (pcb) με βάση την γνώση η οποία αποκτήθηκε από το προηγούμενο κεφάλαιο.

Στο έκτο και τελευταίο κεφάλαιο παρουσιάζονται τα συμπεράσματα από την συνολική μελέτη και ανάπτυξη, οι δυσκολίες επίτευξης του καθώς και μελλοντικές επεκτάσεις που μπορούν να γίνουν ώστε να γίνει καλύτερη η συσκευή.

ΚΕΦΑΛΑΙΟ 2
ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1 Εισαγωγή

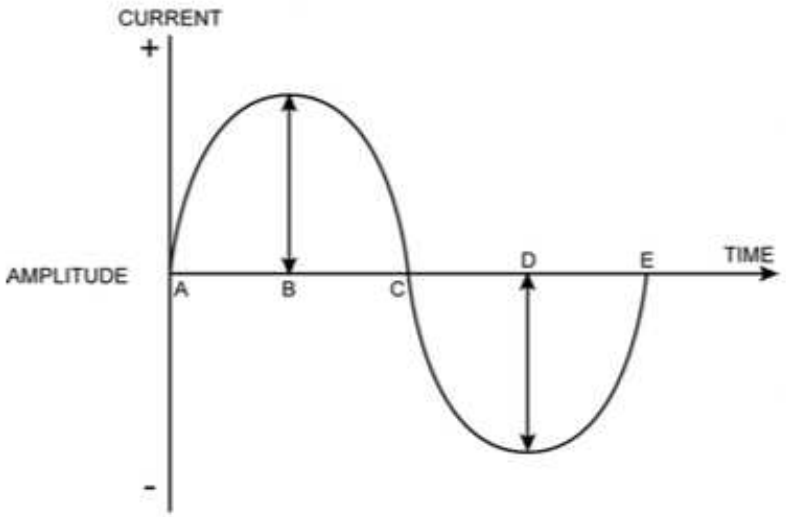
Σε αυτό το κεφάλαιο γίνεται αναφορά στο απαραίτητο θεωρητικό υπόβαθρο το οποίο είναι απαραίτητο για την κατανόηση και την ανάπτυξη της εργασίας.

Στην αρχή δίνονται οι βασικές αρχές και θεωρία των ραδιοσυχνοτήτων η οποία είναι απαραίτητη για να κατανοήσουμε την επόμενη ενότητα η οποία ασχολείται με την τεχνολογία GSM και τον τρόπο με τον οποίο λειτουργεί.

Στην συνέχεια γίνετε αναφορά στους μικροελεγκτές και τον τρόπο με τον οποίο λειτουργούν και τι εφαρμογές μπορούμε να αναπτύξουμε με την χρήση τους.

2.2 Βασική θεωρία ραδιοσυχνοτήτων

Το εναλλασσόμενο ρεύμα (A.C) μας παρέχει έναν πολύ αποδοτικό τρόπο μεταφοράς της ηλεκτρικής ενέργειας σε σχέση με το συνεχές ρεύμα (D.C). Με το εναλλασσόμενο ρεύμα τα ηλεκτρόνια ρέουν τον μισό χρόνο της περιόδου προς την μια κατεύθυνση και τον άλλο μισό χρόνο προς την αντίθετη κατεύθυνση. Το πρότυπο της εναλλαγής αυτής παρουσιάζετε στο παρακάτω σχήμα 1:



Σχήμα 1 Κυματομορφή εναλλασσομένου ρεύματος.

Ο χρόνος ο οποίος απαιτείται από το A στο E αναπαριστά έναν πλήρη κύκλο. Ο όρος συχνότητα χρησιμοποιείτε για να υποδηλώσει των αριθμό των κύκλων που πραγματοποιούνται σε χρόνο ενός δευτερολέπτου. Σύμφωνα με το ελληνικό δίκτυο ηλεκτροδότησης ο χρόνος που χρειάζεται για κάθε κύκλο είναι το ένα πεντηκοστό του δευτερολέπτου, το οποίο μας δίνει συχνότητα 50 κύκλων ανά δευτερόλεπτο ή 50 Hz.

Οι συχνότητες που χρησιμοποιούνται για μεταφορά ραδιοσυχνοτήτων είναι πολύ μεγαλύτερες από τα 50 Hz και είναι της τάξης των KHz , MHz και GHz.

Παρακάτω δίνεται πίνακας με τα εύρη των συχνοτήτων:

Band	Frequency Range
VLF Very Low Frequency	3 –30 KHz
LF Low Frequency	30 – 300 KHz
MF Medium Frequency	300 KHz – 3 MHz
HF High Frequency	3 – 30 MHz
VHF Very High Frequency	30 – 300 MHz
UHF Ultra High Frequency	300 MHz – 3 GHz
SHF Super High Frequency	3 – 30 GHz
EHF Extremely High Frequency	30 – 300 GHz

Πίνακας 1 Εύρη συχνοτήτων

Εάν ένα καλώδιο (ή κεραία) τροφοδοτηθεί με εναλλασσόμενο ρεύμα, κάποιο μέρος από την ενέργεια αυτή ακτινοβολείτε στο διάστημα ως ηλεκτρομαγνητική ενέργεια. Ένα παρόμοιο καλώδιο (κεραία) παράλληλο στο πρώτο θα έχει ένα μικρό εναλλασσόμενο ρεύμα το οποίο προκαλείτε από την εκπεμπόμενη ηλεκτρομαγνητική ενέργεια η οποία και το διαπερνά. Τα χαρακτηριστικά του εναλλασσομένου ρεύματος είναι πανομοιότυπα με αυτά του καλωδίου το οποίο μετέδωσε. Αυτή είναι και η βασική αρχή για όλα τα συστήματα μετάδοσης και λήψης ραδιοσήματος.

2.2.1 Μήκος κύματος

Μόλις το ηλεκτρομαγνητικό κύμα μεταδοθεί αέρια ταξιδεύει με την ταχύτητα του φωτός η οποία είναι 300.000.000 μέτρα το δευτερόλεπτο ή 3x10**8 m/sec.

Το κύμα θα ταξιδέψει μια δεδομένη απόσταση στον μικρό αλλά πεπερασμένο χρόνο που χρειάζεται για να ολοκληρώσει έναν πλήρη κύκλο ενέργειας. Η απόσταση αυτή είναι γνωστή ως μήκος κύματος (λ).

Η σχέση της συχνότητας και του μήκους κύματος δίνεται από την σχέση:

$$f = C/\lambda \quad \text{ή} \quad \lambda = C/f$$

<u>όπου:</u> f = συχνότητα σε Hertz

λ = το μήκος κύματος σε μέτρα

C = είναι η ταχύτητα διάδοσης και είναι ίση με την ταχύτητα του φωτός

Παρακάτω φαίνεται ξανά ο προηγούμενος πίνακας, μόνο που τώρα περιέχει και τα μήκη κύματος για κάθε εύρος συχνοτήτων:

Band	Frequency Range	Wavelength Range	Wavelength Denomination
VLF Very Low Frequency	3 – 30 KHz	100 km – 10 km	Myriametric
LF Low Frequency	30 – 300 KHz	10 km – 1 km	Kilometric
MF Medium Frequency	300 KHz – 3 MHz	1 km – 100 m	Hectometric
HF High Frequency	3 –30 MHz	100 m – 10 m	Decametric
VHF Very High Frequency	30 – 300 MHz	10 m – 1 m	Metric
UHF Ultra High Frequency	300 MHz – 3 GHz	1 m – 10 cm	Decimetric
SHF Super High Frequency	3 – 30 GHz	10 cm – 1 cm	Centimetric (Microwave)
EHF Extremely High Frequency	30 – 300 GHz	1 cm – 1 mm	Millimetric

Πίνακας 2 Εύρη συχνοτήτων με μήκη κύματος

2.2.2 Εύρος ζώνης

Στις ραδιοεπικοινωνίες το εύρος ζώνης είναι το φάσμα συχνοτήτων που καταλαμβάνεται από ένα διαμορφωμένο φέρον σήμα. Πιο συγκεκριμένα είναι η διαφορά μεταξύ των ανώτερων και κατώτερων συχνοτήτων σε μια συνεχή σειρά συχνοτήτων. Η μονάδα μέτρησης είναι τα Hertz και μερικές φορές αναφέρετε και ως ζώνη διέλευσης εύρους ζώνης (passband bandwidth).

Η ζώνη διέλευσης του εύρους ζώνης είναι η διαφορά μεταξύ των ανωτέρων και των κατωτέρων συχνοτήτων αποκοπής. Για παράδειγμα, ένα ζωνοπερατό φίλτρο, ένα κανάλι επικοινωνίας, ή ένα φάσμα σήματος. Στην περίπτωση ενός σήματος με φίλτρο χαμηλής διέλευσης ή βασική ζώνη, του εύρους ζώνης είναι ίσο με την ανώτερη συχνότητα αποκοπής του.

2.3 Τεχνολογία GSM

Το GSM (Global System for Mobile communications), είναι ένα κοινό Ευρωπαϊκό ψηφιακό σύστημα κινητής τηλεφωνίας. Το Ευρωπαϊκό Τηλεπικοινωνιακό Συμβούλιο (European Telecommunications Standards Institute) το 1982, άρχισε την μελέτη για την δημιουργία ενός κοινού Ευρωπαϊκού ψηφιακού συστήματος κινητής τηλεφωνίας δεύτερης γενιάς (2G). Αυτό το σύστημα ονομάστηκε αρχικά Group Special Mobile (GSM).

Το GSM είναι ένα κυψελοειδές ψηφιακό σύστημα κινητής τηλεφωνίας δεύτερης γενιάς (2G), το οποίο χρησιμοποιεί ηλεκτρομαγνητικά σήματα και την τεχνική πολλαπλής πρόσβασης με διαχωρισμό του διαθέσιμου φάσματος συχνοτήτων σε ένα αριθμό καναλιών και την διαίρεση αυτών σε χρονοθυρίδες για την μετάδοση σημάτων.

Το 1989 η ευθύνη του GSM ανατέθηκε στο Ευρωπαϊκό Τηλεπικοινωνιακό Ινστιτούτο Προτύπων (ETSI) και το 1990 ανακοινώθηκαν επίσημα για πρώτη φορά το πρότυπο και τα χαρακτηριστικά του GSM. Το 1991 άρχισε η εμπορική του διάθεση στην Ευρώπη, ενώ στην Ελλάδα το σύστημα χρησιμοποιήθηκε το 1993 από την Wind (πρώην TIM ή πρώην TELESTET).

2.3.1 Ιστορική αναδρομή

Το 1982 άρχισαν οι εργασίες για την ανάπτυξη ενός ευρωπαϊκού προτύπου για την ψηφιακή κυψελοειδή φωνητική τηλεφωνία, όταν η Ευρωπαϊκή Διάσκεψη των Διοικήσεων των Ταχυδρομείων και Τηλεπικοινωνιών (CEPT) συγκροτεί την επιτροπή Groupe Special Mobile.Αργότερα παρείχε μια μόνιμη ομάδα τεχνικής υποστήριξης με έδρα το Παρίσι. Πέντε χρόνια αργότερα, το 1987, 15 εκπρόσωποι από 13 ευρωπαϊκές χώρες υπέγραψαν ένα μνημόνιο στην Κοπεγχάγη για την χρήση ενός κοινού συστήματος κυψελοειδούς τηλεφωνίας σε όλη την Ευρώπη, και θέσπισαν κανόνες της ΕΕ ώστε να γίνει το GSM ένα υποχρεωτικό πρότυπο. Η απόφαση για την ανάπτυξη ενός ευρωπαϊκού προτύπου τελικά κατέληξε σε ένα ενιαίο, ανοιχτό, δίκτυο πρότυπο το οποίο ήταν μεγαλύτερο από ότι στις Ηνωμένες Πολιτείες.

Τον Φεβρουάριο του 1987 στην Ευρώπη παράγεται η πρώτη τεχνική προδιαγραφή για το GSM. Υπουργοί από τις τέσσερις μεγάλες χώρες της ΕΕ εδραίωσαν την πολιτική τους υποστήριξη για το GSM με τη Διακήρυξη της Βόννης για τα Παγκόσμια Δίκτυα Πληροφοριών. Το Μνημόνιο έδωσε κίνητρο στους φορείς

εκμετάλλευσης κινητής τηλεφωνίας από όλη την Ευρώπη να δεσμευτούν να επενδύσουν σε νέα δίκτυα GSM.

Παράλληλα, η Γαλλία και η Γερμανία υπέγραψαν κοινή συμφωνία ανάπτυξης το 1984 και ενώθηκαν με την Ιταλία και το Ηνωμένο Βασίλειο το 1986. Το 1986 η Ευρωπαϊκή Επιτροπή πρότεινε δέσμευση της ζώνης των 900 MHz για το GSM. Ο πρώην πρωθυπουργός της Φινλανδίας Χάρι Χόλκερι πραγματοποίησε την πρώτη κλήση GSM στον κόσμο την 1η Ιουλίου 1991, καλώντας την Kaarina Suonio (δήμαρχος της πόλης του Τάμπερε) χρησιμοποιώντας ένα δίκτυο που χτίστηκε από την Telenokia και τη Siemens. Κατά το επόμενο έτος, το 1992, είδε την αποστολή του πρώτου γραπτού μηνύματος (SMS ή «μήνυμα»), και η Vodafone UK και η Telecom Φινλανδίας υπέγραψαν την πρώτη διεθνή συμφωνία περιαγωγής (roaming).

Οι εργασίες άρχισαν το 1991 για επέκταση του προτύπου GSM στη ζώνη συχνοτήτων 1800 MHz και το πρώτο δίκτυο των 1800 MHz άρχισε να λειτουργεί στο Ηνωμένο Βασίλειο από το 1993. Επίσης εκείνη τη χρονιά, η Telecom Αυστραλίας έγινε ο πρώτος φορέας εκμετάλλευσης του δικτύου για την ανάπτυξη ενός δικτύου GSM έξω από την Ευρώπη και το πρώτο κινητό τηλέφωνο χειρός έγινε διαθέσιμο.

Το 1995 το φαξ, τα δεδομένα και οι υπηρεσίες μηνυμάτων SMS ξεκίνησαν εμπορικά. Το πρώτο λειτουργικό δίκτυο στα 1900 MHz GSM άρχισε να λειτουργεί στις Ηνωμένες Πολιτείες και οι GSM συνδρομητές σε όλο τον κόσμο ξεπέρασαν τα 10 εκατομμύρια. Κατά το ίδιο έτος, σχηματίζεται η Ένωση GSM. Οι προπληρωμένες κάρτες GSM SIM ξεκίνησαν το 1996 και σε όλο τον κόσμο οι συνδρομητές GSM ξεπέρασαν τα 100 εκατομμύρια το 1998.

Το 2000 οι πρώτες εμπορικές υπηρεσίες GPRS ξεκίνησαν και οι πρώτες GPRS-συμβατές συσκευές έγιναν διαθέσιμες προς πώληση.

Το 2001 τα πρώτα UMTS (W-CDMA) του δικτύου ξεκίνησε, μια τεχνολογία 3G που δεν αποτελεί μέρος του GSM. Σε όλο τον κόσμο οι συνδρομητές GSM ξεπέρασαν τα 500 εκατομμύρια.

Το 2002 η υπηρεσία MMS (Multimedia Messaging Service) ξεκίνησε και το πρώτο δίκτυο GSM στη ζώνη συχνοτήτων των 800 MHz άρχισε να λειτουργεί. Οι υπηρεσίες EDGE δόθηκαν για πρώτη φορά σε λειτουργία στο δίκτυο το 2003 και ο αριθμός των ανά τον κόσμο συνδρομητών GSM ξεπέρασε το 1 δις το 2004.

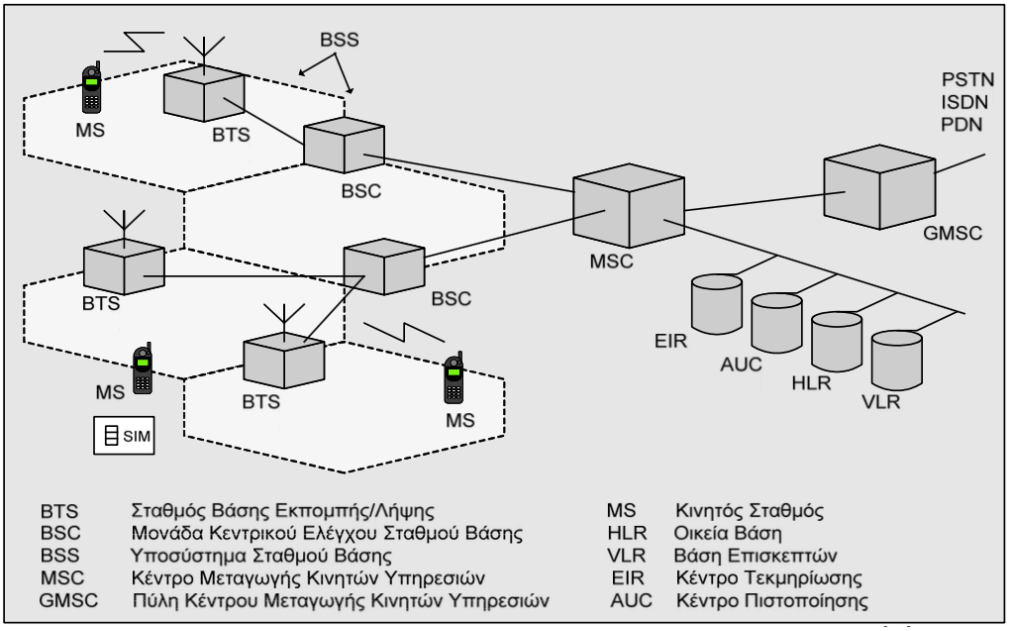
Μέχρι το 2005, τα δίκτυα GSM αντιπροσώπευαν πάνω από το 75% της παγκόσμιας αγοράς των κυψελοειδών δικτύων, που εξυπηρετούν 1,5 δισ συνδρομητές. Το 2005 το πρώτο HSDPA με δυνατότητα δικτύου, επίσης, έγινε λειτουργικό.

Το πρώτο δίκτυο HSUPA ξεκίνησε το 2007. Το HSPA (High-Speed Packet Access) και οι εκδόσεις του (uplink and downlink versions) είναι τεχνολογίες 3G και δεν αποτελούν μέρος του GSM.

Να σημειώσουμε ότι το GSM είναι ένα δεύτερης γενιάς (2G) πρότυπο το οποίο χρησιμοποιεί την τεχνική Time-Division Multiple-Access (TDMA) μερισμού του φάσματος, που εκδίδετε από το Ευρωπαϊκό Ινστιτούτο Τηλεπικοινωνιακών Προτύπων (ETSI). Το πρότυπο GSM δεν περιλαμβάνει την τεχνολογία 3G UMTS CDMA ούτε τα πρότυπα της τεχνολογίας 4G LTE.

2.3.2 Αρχιτεκτονική δικτύου GSM

Στο παρακάτω σχήμα φαίνεται η αρχιτεκτονική ενός δικτύου GSM και στην συνέχεια εξηγούμε την λειτουργία των σημαντικότερων τμημάτων του.



Σχήμα 2. Αρχιτεκτονική Δικτύου GSM

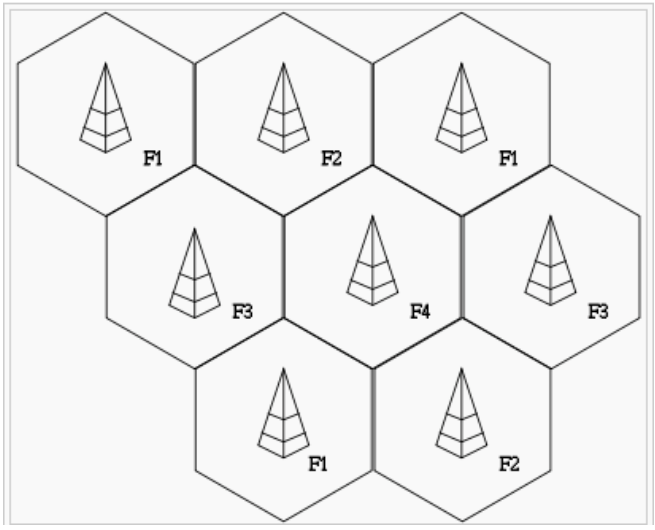
MS: Το ακρωνύμιο προέρχεται από το Mobile Station (κινητός σταθμός) και είναι η συσκευή μας στην οποία τοποθετούμε την SIM κάρτα μας.

BTS: Base Transceiver Station (Σταθμός Βάσης Εκπομπής/Λήψης) είναι η κεραία με την οποία συνδέεται το κινητό μας με το δίκτυο και σκοπός της είναι να καλύπτει την περιοχή μιας κυψέλης.

Κυψέλη είναι μια γεωγραφική περιοχή την οποία και καλύπτει ο σταθμός βάσης BTS. Η πλέον κατάλληλη μορφή της κυψέλης είναι το κανονικό εξάγωνο γιατί προσεγγίζει την επιφάνεια του κύκλου, δεν εμφανίζει κενά και επικαλυπτόμενες περιοχές και απαιτεί μικρότερο αριθμό κυψελών κατά τον σχεδιασμό κάλυψης του δικτύου. Το σημαντικότερο πλεονέκτημα ενός κυψελωτού δικτύου είναι η επαναχρησιμοποίηση συχνοτήτων όπου οι ίδιες συχνότητες του φορέα επαναχρησιμοποιούνται. Η λογική του συστήματος επικοινωνίας είναι η διαίρεση καλυπτόμενων περιοχών σε υποπεριοχές-κυψέλες, των οποίων το μέγεθος της διαμέτρου κυμαίνεται από λίγα χιλιόμετρα μέχρι 50 έως χιλιόμετρα.

Σε κάθε μία κυψέλη παραχωρείται ένα σύνολο συχνοτήτων. Δύο κυψέλες μπορούν να χρησιμοποιούν τις ίδιες συχνότητες, εφόσον βρίσκονται σε απόσταση τέτοια που να αποφεύγονται τα προβλήματα παρεμβολών.

Η χωρητικότητα του κυψελοειδούς συστήματος, όσον αφορά των αριθμό των χρηστών, εξαρτάται από τον αριθμό των διαθέσιμων συχνοτήτων σε κάθε κυψέλη, καθώς και από τον αριθμό των κυψελών.



Σχήμα 3. Κυψελοειδές δίκτυο με επαναχρησιμοποίηση συχνότητας

BSC: Base Station Controller (Μονάδα κεντρικού ελέγχου) είναι συνήθως μαζί με ένα BTS και εκτελεί λειτουργίες διαχείρισης, όπως το να καθορίζει και να

απελευθερώνει συχνότητες για τους χρήστες που ανήκουν στην περιοχή κάλυψης του και να επικοινωνεί με το MSC για δρομολόγηση κλήσεων σε χρήστες που δεν ανήκουν στην περιοχή του.

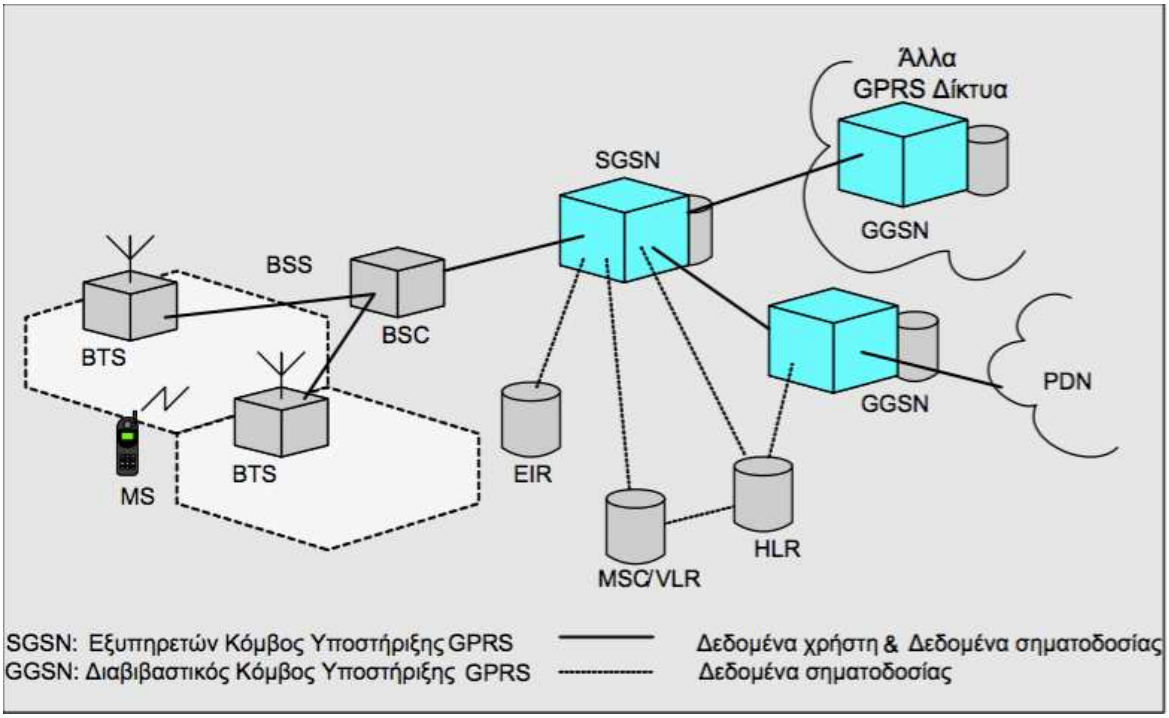
Σε ένα BSC μπορεί να ανήκουν πολλά BTS.

MSC: Mobile Switching Center (Κέντρο μεταγωγής κινητών υπηρεσιών) είναι η καρδιά του GSM. Έχει τον αντίστοιχο ρόλο που έχει και το τηλεφωνικό κέντρο ενός δικτύου σταθερής τηλεφωνίας. Συνδέεται με έναν αριθμό από BSC και χειρίζεται τις κλήσεις που γίνονται στην περιοχή την οποία καλύπτουν τα BSC τα οποία έχουν συνδεθεί σε αυτό.

2.3.3 Δίκτυο κινητών επικοινωνιών GPRS

Το δίκτυο GPRS (2,5G) αποτελεί την εξέλιξη του δικτύου GSM και ο κύριος σκοπός του είναι η μετάδοση δεδομένων παράλληλα με την μετάδοση της φωνής.

Χρησιμοποιεί την υπάρχουσα υποδομή του δικτύου GSM και χρησιμοποίει μεταγωγή πακέτου στο δίκτυο κορμού του, έναντι της μεταγωγής κυκλώματος που χρησιμοποιούσε το GSM δίκτυο. Στο παρακάτω σχήμα φαίνεται η αρχιτεκτονική ενός δικτύου GPRS:



Σχήμα 4. Αρχιτεκτονική δικτύου GPRS

Το GPRS εισάγει ουσιαστικά 2 νέους κόμβους στο υπάρχον δίκτυο κορμού (Core Network) του GSM για την επίτευξη της μεταγωγής πακέτου.

Οι κόμβοι αυτοί είναι:

1. Εξυπηρετούν Κόμβος Υποστήριξης GPRS γνωστός και ως SGSN (Serving GPRS Support Node). Είναι υπεύθυνος για την μεταφορά πακέτων δεδομένων προς και από την περιοχή που εξυπηρετεί. Αναλαμβάνει δηλαδή τον αντίστοιχο ρόλο του MSC στο GSM.

2. Ο Διαβιβαστικός Κόμβος Υποστήριξης GPRS γνωστός και ως GGSN (Gateway GPRS Support Node). Έχει το ρόλο του δρομολογητή των πακέτων δεδομένων προς και από τα εξωτερικά δίκτυα πακέτων δεδομένων. Η μεταφορά των δεδομένων μεταξύ του δικτύου κορμού του GPRS προς και από τα εξωτερικά δίκτυα δεδομένων γίνετε με την χρήση του πρωτοκόλλου σήραγγας GPRS (GPRS Tunneling Protocol, GTP).

2.4 Μικροελεγκτές

Ο μικροελεγκτής (microcontroller) είναι ένας τύπος επεξεργαστή, ουσιαστικά μια παραλλαγή μικροεπεξεργαστή, ο οποίος μπορεί να λειτουργήσει με ελάχιστα εξωτερικά εξαρτήματα, λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει. Χρησιμοποιείται ευρύτατα σε όλα τα ενσωματωμένα συστήματα (embedded systems) ελέγχου χαμηλού και μεσαίου κόστους, όπως αυτά που χρησιμοποιούνται σε αυτοματισμούς, ηλεκτρονικά καταναλωτικά προϊόντα (από ψηφιακές φωτογραφικές μηχανές έως παιχνίδια), ηλεκτρικές συσκευές και σε κάθε είδους αυτοκινούμενα τροχοφόρα οχήματα.

2.4.1 Δομή των μικροελεγκτών

Ο μικροελεγκτής όπως κάθε υπολογιστικό σύστημα περιέχει κεντρική μονάδα επεξεργασίας, έναν αριθμό καταχωρητών, κυκλώματα μνήμης και κυκλώματα έλεγχου περιφερειακών συσκευών. Κάθε μικροελεγκτής είναι λοιπόν ικανός να ανταλλάξει σήματα με το εξωτερικό περιβάλλον, να εκτελέσει πράξεις ανάμεσα σε μεταβλητές και να καταχωρήσει κάποιες τιμές στη μνήμη RAM που διαθέτει.

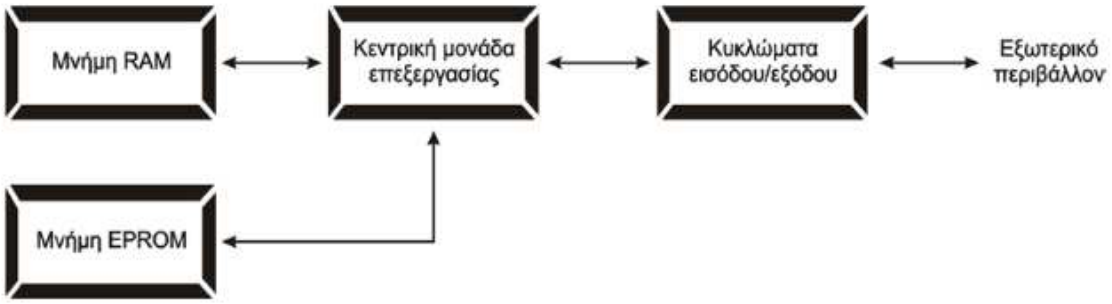
Κάθε μικροελεγκτής περιέχει μέσα σε ένα και μοναδικό ολοκληρωμένο κύκλωμα τα παρακάτω στοιχεία:

- έναν αριθμό από καταχωρητές ειδικού σκοπού (συσσωρευτή, καταχωρητή κατάστασης, μετρητή προγράμματος, καταχωρητή εντολών, καταχωρητή δείκτη).
- εσωτερικούς χρονιστές - απαριθμητές.
- αριθμητική και λογική μονάδα (ALU).
- μονάδα αποκωδικοποίησης εντολών.

Επίσης βασικά στοιχεία ενός μικροελεγκτή αποτελούν:

- η μνήμη προγράμματος (ROM η EPROM) και
- η μνήμη καταχωρητών / μεταβλητών (RAM).
- κυκλώματα χρονισμού και ελέγχου.

Παρακάτω φαίνεται σχήμα με την μορφή block diagram το οποίο δείχνει την δομή ενός μικροελεγκτή.



Σχήμα 5. Βασική δομή ενός μικροελεγκτή

Επίσης βασικά μέρη ενός μικροελεγκτή είναι και οι παράλληλες θύρες εισόδου και εξόδου (I/O ports) και περιφερειακά κυκλώματα όπως UART, A/D converters κ.α. Μέσα από τις θύρες I/O ένας μικροελεγκτής μπορεί να δέχεται σήματα εισόδου με τη μορφή λογικών ψηφιακών καταστάσεων, χαρακτήρες η bytes δεδομένων με την τεχνική της ασύγχρονης ή της σύγχρονης σειριακής επικοινωνίας, σήματα διακοπών, η σε ορισμένες περιπτώσεις και αναλογικά σήματα, τα οποία στη συνέχεια μετατρέπονται σε ψηφιακά. Επίσης μπορεί να αποστέλλει σήματα σε άλλες συσκευές μέσα από θύρες εξόδου, να οδηγεί ηλεκτρονόμους, διόδους LED και άλλα κατάλληλα κυκλώματα, που συνήθως περιλαμβάνονται σε κάθε μορφής αυτοματισμό.

Οι μικροελεγκτές χαρακτηρίζονται από ένα περιορισμένο ρεπερτόριο εντολών, που μπορούν να γραφούν σε συμβολική μορφή (*assembly*), με τη βοήθεια μνημονικών ονομάτων.

Σε τι διαφέρει ένας μικροελεγκτής από έναν συνηθισμένο μικροεπεξεργαστή; Ο μικροελεγκτής είναι ένα μικρό αυτόνομο υπολογιστικό σύστημα, προγραμματισμένο να εκτελεί μια συγκεκριμένη λογική ακολουθία εντολών οι οποίες έχουν καταχωρηθεί στην προγραμματιζόμενη μόνιμη μνήμη του. Κάθε φορά που θα επανεκκινείται ο μικροελεγκτής, θα εκτελεί την ίδια λογική. Θα ανακαλεί τα δεδομένα, θα τα επεξεργάζεται και με βάση τα αποτελέσματα της επεξεργασίας θα ελέγχει το περιβάλλον του. Πρόκειται, δηλαδή, για σύστημα ειδικού σκοπού, αφιερωμένο (*dedicated*) στον έλεγχο και την εξυπηρέτηση ενός συγκεκριμένου αυτοματισμού.

Αντίθετα, ένας μικροεπεξεργαστής μετά την εκκίνηση του δεν είναι από μόνος του σε θέση να εκτελέσει κάποια λογική ακολουθία. Αν και μπορεί να συνδεθεί με μνήμες RAM και ROM, αυτές αποτελούν ξεχωριστές μονάδες, που συνήθως δεν ολοκληρώνονται μέσα στον ίδιο τον μικροεπεξεργαστή.

2.4.2 Γλώσσες προγραμματισμού και εργαλεία ανάπτυξης

Η επιτυχία μιας οικογένειας μικροελεγκτών καθορίζεται σε μεγάλο βαθμό από τη διαθεσιμότητα και την ευχρηστία των σχετικών εργαλείων ανάπτυξης, όπως μεταφραστές από γλώσσες υψηλού επιπέδου σε γλώσσα κατανοητή από τον μικροελεγκτή (assembly), προγραμματιστές της εσωτερικής μνήμης και εργαλεία εκσφαλμάτωσης (debuggers). Στους μικροελεγκτές, τα εργαλεία αυτά δεν αποτελούνται ποτέ μόνο από λογισμικό, καθώς δεν υπάρχει τυποποιημένος τρόπος επικοινωνίας με αυτούς. Στον τομέα των εργαλείων ανάπτυξης, δραστηριοποιούνται όχι μόνο οι ίδιοι οι κατασκευαστές μικροελεγκτών αλλά και εξειδικευμένες εταιρείες. Η πιο διαδεδομένη γλώσσα προγραμματισμού των μικροελεγκτών είναι η C, η C++ και οι παραλλαγές τους. Σε τμήματα του λογισμικού όπου απαιτείται ταχύτητα η μικρό μέγεθος χρησιμοποιούμενης μνήμης, μπορεί να χρησιμοποιείται η Assembly. Όμως οι μεγαλύτερες απαιτήσεις σε λειτουργικότητα και η ευκολία προγραμματισμού της C έναντι της assembly, σε συνδυασμό με την επάρκεια μνήμης των σύγχρονων μικροελεγκτών, έχουν γενικά εκτοπίσει την Assembly από τις περισσότερες εφαρμογές.

ΚΕΦΑΛΑΙΟ 3
ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναπτύξουμε την αρχιτεκτονική του συστήματος και θα εξηγήσουμε τα επί μέρους κομμάτια τα οποία το απαρτίζουν. Επίσης θα αιτιολογήσουμε γιατί επιλέξαμε την συγκεκριμένη αρχιτεκτονική όπως επίσης και τα πλεονεκτήματα και τα μειονεκτήματα της επιλογής αυτής.

3.2 Η πλατφόρμα Arduino

Επιλέξαμε να αναπτύξουμε την συσκευή μας με την πλατφόρμα Arduino διότι είναι μια πλατφόρμα ανάπτυξης ηλεκτρονικών πρωτοτύπων ανοιχτού κώδικα (open source) και μας παρέχει έναν εύκολο τρόπο ανάπτυξης τόσο σε επίπεδο υλικού (hardware) αλλά και λογισμικού (software). Στο παρακάτω σχήμα φαίνεται το Arduino Uno το οποίο και επιλέξαμε για την ανάπτυξη μας.

Σχήμα 6. Όψη Arduino Uno.

Οι κύριοι λόγοι για τους οποίους επιλέξαμε να αναπτύξουμε την συσκευή μας με το Arduino Uno είναι οι παρακάτω:

- Έχει χαμηλό κόστος σε σχέση με άλλες αναπτυξιακές πλατφόρμες και μπορεί κανείς να το συναρμολογήσει και μόνος του. Αν επιλέξει να πάρει όμως έτοιμο υλικό κοστίζει λιγότερο από 30 ευρώ.

- Το περιβάλλον προγραμματισμού μπορεί να τρέξει σε όλα τα λειτουργικά συστήματα (Windows, Mac, Linux), ενώ άλλα αναπτυξιακά περιβάλλοντα περιορίζονται μόνο σε λειτουργικό Windows.
- Απλό και εύχρηστο περιβάλλον προγραμματισμού (IDE).
- Ανοιχτού κώδικα λογισμικό (open source software) το οποίο είναι και αναβαθμίσιμο.
- Ανοιχτού κώδικα και επεκτάσιμο υλικό (open source hardware) με την χρήση shields.

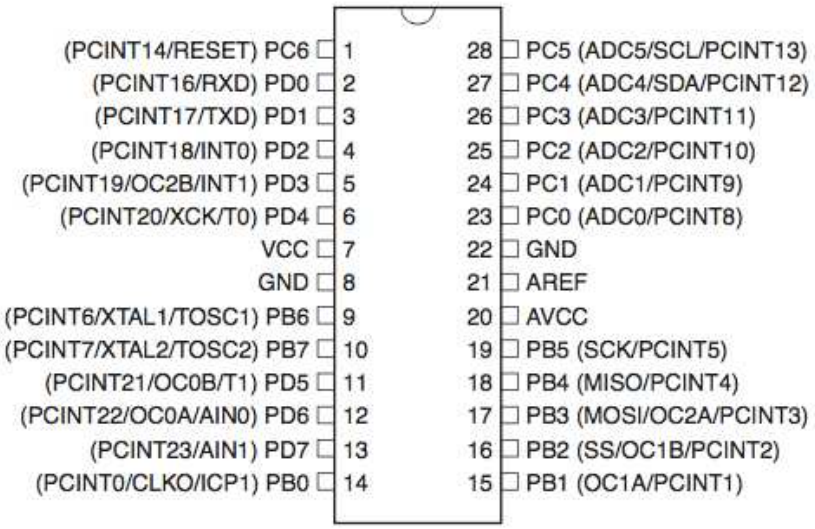
3.2.1 Ανάλυση υλικού (Hardware)

Στην συνέχεια ακολουθεί μια ανάλυση των τμημάτων που απαρτίζουν την αναπτυξιακή πλατφόρμα Arduino.

3.2.1.1 Μικροελεγκτής

Το Arduino Uno είναι βασισμένο στον μικροελεγκτή ATMEGA328P της εταιρείας ATMEL. Ο συγκεκριμένος μικροελεγκτής, όπου φαίνεται στο παρακάτω σχήμα, αποτελείτε από:

- 32KB μνήμη flash με δυνατότητα ανάγνωσης και εγγραφής
- 1024B μνήμη EEPROM
- 2KB SRAM
- 23 ψηφιακές εισόδους/εξόδους (digital I/O)
- 32 γενικής χρήσης καταχωρητές εργασίας (working registers)
- 3 χρονόμετρα/μετρητές (counter/timers) με λειτουργία σύγκρισης
- εσωτερικές και εξωτερικές διακοπές (interrupts)
- σειριακή προγραμματιζόμενη USART
- SPI σειριακή θύρα
- 6 κανάλια των 10bit A/D μετατροπέα (Analog to digital converter)
- 2 καλωδίων σειριακή διεπαφή (interface)
- Προγραμματιζόμενο Watchdog timer με εσωτερικό ταλαντωτή
- 5 προγραμματιστικές επιλεγόμενες καταστάσεις για μείωση της κατανάλωσης



Σχήμα 7. Μικροελεγκτής ATMEGA328P

Παρακάτω φαίνεται και πίνακας με την σύγκριση των μικροελεγκτών όλης της συγκεκριμένης οικογένειας.

Device	Flash	EEPROM	RAM	Interrupt Vector Size
ATmega48A	4KBytes	256Bytes	512Bytes	1 instruction word/vector
ATmega48PA	4KBytes	256Bytes	512Bytes	1 instruction word/vector
ATmega88A	8KBytes	512Bytes	1KBytes	1 instruction word/vector
ATmega88PA	8KBytes	512Bytes	1KBytes	1 instruction word/vector
ATmega168A	16KBytes	512Bytes	1KBytes	2 instruction words/vector
ATmega168PA	16KBytes	512Bytes	1KBytes	2 instruction words/vector
ATmega328	32KBytes	1KBytes	2KBytes	2 instruction words/vector
ATmega328P	32KBytes	1KBytes	2KBytes	2 instruction words/vector

Πίνακας 3. Σύγκριση μικροελεγκτών της οικογένειας P/A

3.2.1.2 Τροφοδοσία

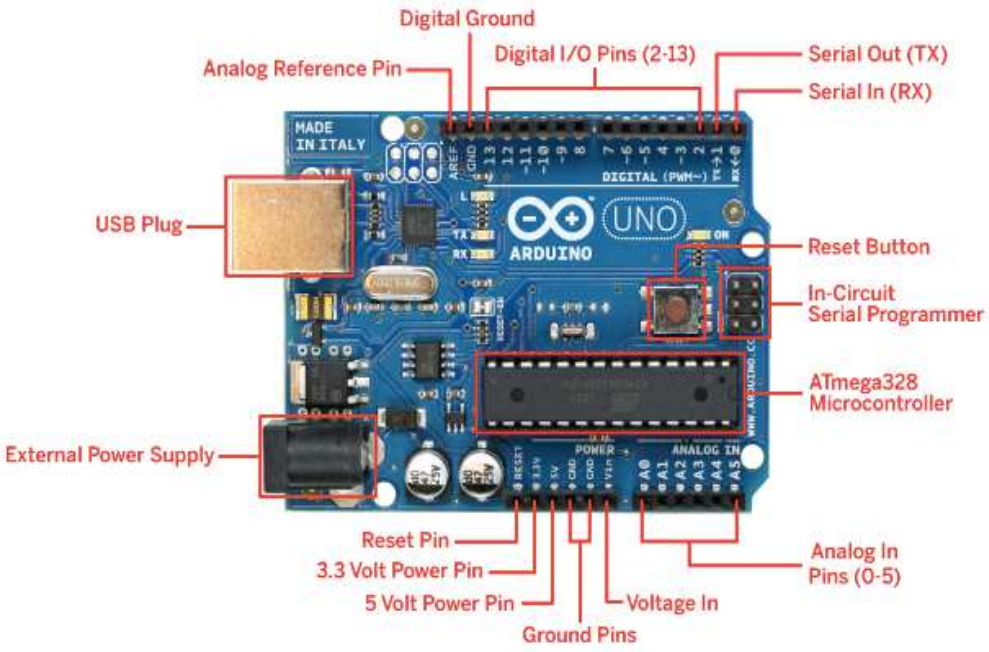
Το Arduino μπορεί να τροφοδοτηθεί από την θύρα USB την οποία διαθέτει ή με εξωτερική τροφοδοσία. Η τροφοδοσία επιλέγεται αυτόματα ανάλογα με τι τροφοδοτούμε την πλατφόρμα. Το Arduino μπορεί να λειτουργήσει με εξωτερική τροφοδοσία από 6 έως 20 Volt. Οι περιορισμοί που υπάρχουν είναι πως εάν τροφοδοτηθεί με λιγότερο από 7 Volt τότε μπορεί να γίνει ασταθές. Επίσης αν τροφοδοτηθεί με περισσότερα από 17 Volt τότε ο ρυθμιστής τάσης μπορεί να

υπερθερμανθεί και να καταστραφεί. Το επιθυμητό εύρος τροφοδοσίας είναι από 7V έως 12V.

Οι επαφές τροφοδοσίας του είναι οι ακόλουθες:

- Vin: Η τάση εισόδου στο Arduino όταν χρησιμοποιεί εξωτερική τροφοδοσία.
- 5V: Αυτή η επαφή μας προσφέρει σταθεροποιημένη τάση 5V από τον ρυθμιστή τάση. Δεν μπορούμε να τροφοδοτήσουμε το arduino από αυτή την επαφή ή την επαφή των 3,3V διότι παρακάμπτεται ο ρυθμιστής τάσης και μπορεί να προκαλέσουμε βλάβη.
- 3V3: Τάση 3,3V η οποία παράγεται και αυτή από τον ρυθμιστή τάσης.
- GND: Επαφές γείωσης.
- IOREF: Αυτή η επαφή μας παρέχει την τάση αναφοράς στην οποία εργάζεται ο μικροελεγκτής και είναι χρήσιμη όταν συνδέσουμε κάποια επέκταση να αντιληφθεί την τάση λειτουργίας και με την σειρά της να επιλέξει την δική της τάση λειτουργίας.

Παρακάτω φαίνεται ένα σχήμα με την τοπολογία των επαφών των οποίων και περιγράψαμε την λειτουργία:



Σχήμα 8. Τοπολογία επαφών Arduino Uno

3.2.1.3 Ψηφιακές είσοδοι - έξοδοι

Οι θύρες του Arduino μπορούν να ρυθμιστούν προγραμματιστικά και να είναι είτε είσοδοι είτε έξοδοι. Από το datasheet του μικροελεγκτή βλέπουμε ότι δεν χρειάζεται να δηλώσουμε τις θύρες ως εισόδους διότι εξ' ορισμού είναι είσοδοι. Αυτό συμβαίνει ώστε να βρίσκονται σε κατάσταση υψηλής εμπέδησης.

Επίσης έχουν πολύ μικρές απαιτήσεις από το κύκλωμα στο οποίο συμμετέχουν και ισοδυναμεί με μια αντίσταση 100ΜΩ μπροστά από την θύρα. Αυτό μεταφράζετε στο ότι χρειάζεται πολύ μικρό ρεύμα για να αλλάξει την κατάσταση της θύρας και τις κάνει κατάλληλες για περιπτώσεις όπως είναι η εφαρμογή ενός χωρητικού αισθητήρα αφής.

3.2.1.4 Αναλογικές είσοδοι - έξοδοι

Όπως αναφέραμε και στα χαρακτηριστικά του επεξεργαστή, ο ATMEGA328P είναι εξοπλισμένος με έξι κανάλια Analog to Digital converter (μετατροπέα από αναλογικό σε ψηφιακό σήμα). Η ανάλυση του μετατροπέα είναι 10 bit και μας επιστρέφει ακέραιους αριθμούς από το 0 έως το 1023. Αν και ο προορισμός των θυρών αυτών είναι να λειτουργούν ως αναλογικοί είσοδοι και έξοδοι, μπορούμε να τις χρησιμοποιήσουμε και σαν ψηφιακές εισόδους / εξόδους γενικής χρήσης, όπως αυτές που αναφέραμε παραπάνω.

Οι αναλογικές θύρες έχουν επίσης αντιστάσεις pull-up οι οποίες συμπεριφέρονται ακριβώς το ίδιο όπως και στις ψηφιακές. Από το datasheet του μικροελεγκτή βλέπουμε επίσης ότι υπάρχει η προειδοποίηση για ηλεκτρικό θόρυβο όταν μεταβαίνουμε σε πολύ στενά χρονικά όρια από την μια αναλογική είσοδο σε μια άλλη. Το πρόβλημα αυτό μπορεί να λυθεί εισάγοντας μια μικρή καθυστέρηση από την μια ανάγνωση στην άλλη.

3.2.1.5 Μνήμες

Υπάρχουν 3 είδη μνήμης στον μικροελεγκτή και είναι οι ακόλουθες:

- Μνήμη Flash στην οποία αποθηκεύεται το πρόγραμμα μας.
- Μνήμη SRAM (Static Random Access Memory) στην οποία στο πρόγραμμα μας δημιουργεί διαχειρίζεται τις μεταβλητές σε πραγματικό χρόνο.

- Μνήμη EEPROM όπου μπορούμε να αποθηκεύσουμε πληροφορία για μεγάλο χρονικό διάστημα.

Στην μνήμη Flash και EEPROM η πληροφορία παραμένει μετά την απενεργοποίηση της συσκευής ενώ στην μνήμη SRAM όλες οι πληροφορίες χάνονται μετά την απενεργοποίηση της συσκευής.

3.2.2 Γλώσσα προγραμματισμού

Η γλώσσα προγραμματισμού η οποία χρησιμοποιείτε είναι η γλώσσα Wiring. Η γλώσσα αυτή είναι ανοιχτού κώδικα για κατασκευή πρωτοτύπων στα ηλεκτρονικά. Επίσης μπορεί να λειτουργήσει σε όλα τα λειτουργικά συστήματα. Τα προγράμματα όλα γράφονται σε γλώσσα προγραμματισμού C/C++ αλλά η ευκολία της είναι πως ο χρήστης χρειάζεται μόνο να καθορίσει 2 συναρτήσεις ώστε να καταφέρει να φτιάξει ένα λειτουργικό πρόγραμμα.

Οι συναρτήσεις αυτές είναι:

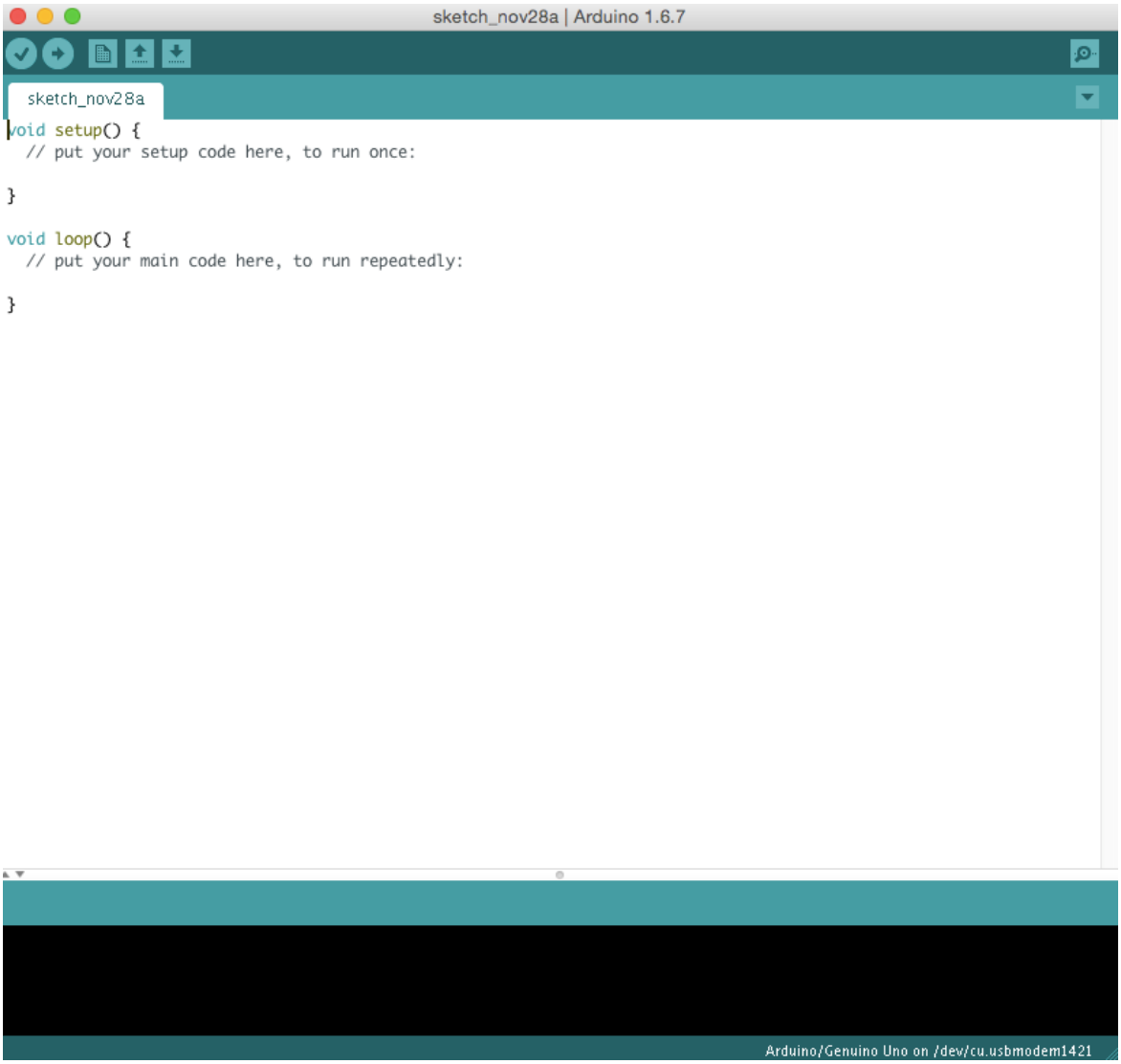
- setup() - είναι μια συνάρτηση όπου τρέχει μια φορά στην αρχή κάθε προγράμματος και χρησιμοποιείτε για να καθορίσει τις αρχικές ρυθμίσεις του συστήματος.
- loop() - είναι η συνάρτηση η οποία καλείτε επαναλαμβανόμενα μέχρι το σύστημα να απενεργοποιηθεί.

Όταν ο χρήστης πατήσει το πλήκτρο για να φορτώσει το πρόγραμμα στον μικροελεγκτή, ένα αντίγραφο του κώδικα γράφεται σε έναν προσωρινό φάκελο με μια έξτρα επικεφαλίδα (include header) στην κορυφή του προγράμματος και μια απλή main() συνάρτηση στο τέλος του προγράμματος ώστε να γίνει ένα έγκυρο πρόγραμμα C++.

3.2.3 Περιβάλλον ανάπτυξης (IDE)

Το περιβάλλον ανάπτυξης του Arduino περιλαμβάνει έναν κειμενογράφο για την συγγραφή του κώδικα, μια περιοχή μηνυμάτων, μια κονσόλα κειμένου, μια μπάρα εργαλείων με κουμπιά για κοινές λειτουργίες και μια σειρά από μενού.

Συνδέεται με το υλικό (hardware) και μπορούμε να φορτώσουμε τα προγράμματα μας στον μικροελεγκτή όπως επίσης και να επικοινωνήσουμε μαζί του. Τα προγράμματα τα οποία γράφονται σε αυτό το περιβάλλον ονομάζονται sketches και αποθηκεύονται με την κατάληξη .ino. Παρακάτω φαίνεται σε σχήμα το περιβάλλον ανάπτυξης.



Σχήμα 9. Περιβάλλον ανάπτυξης Arduino (IDE)

3.3 GPRS/GSM επέκταση (shield)

Στην ενότητα αυτή παρουσιάζουμε την επέκταση (shield) την οποία επιλέξαμε και η οποία συνδέεται στο Arduino Uno ώστε να μπορέσουμε να συνδεθούμε στο δίκτυο της κινητής τηλεφωνίας. Επιλέξαμε την συγκεκριμένη επέκταση διότι μας προσφέρει όλα εκείνα τα χαρακτηριστικά τα οποία χρειαζόμαστε για την ανάπτυξη μας.

Πιο συγκεκριμένα:

- Επικοινωνία με μορφή γραπτών μηνυμάτων
- Επικοινωνία με κλήση φωνής
- Υπηρεσίες GPRS

Το κόστος της συγκεκριμένης επέκτασης είναι προσιτό σε σχέση με άλλες επεκτάσεις του εμπορίου και επίσης είναι και αξιόπιστο (με βάση πάντα τις αναφορές χρηστών από το διαδίκτυο που την έχουν δοκιμάσει). Η συγκεκριμένη επέκταση είναι της εταιρείας Tinysine και είναι βασισμένη στο SIM900 ολοκληρωμένο το οποίο και θα αναλύσουμε στην συνέχεια. Πιο κάτω φαίνεται σχήμα με την όψη της συγκεκριμένης επέκτασης.

Σχήμα 10. Όψη επέκτασης GPRS/GSM

3.3.1 SIM900 module

Το SIM900 module είναι ένα ολοκληρωμένο κύκλωμα - τηλέφωνο το οποίο λειτουργεί στις συχνότητες 850/900/1800/1900Mhz και μπορεί να χρησιμοποιηθεί για επικοινωνία φωνής, μέσω του ενσωματωμένου μικροφώνου και ηχείου, για πρόσβαση στο διαδίκτυο και για αποστολή γραπτών μηνυμάτων. Εσωτερικά ελέγχεται από τον επεξεργαστή ARM926EJ-S. Ο συγκεκριμένος επεξεργαστής ελέγχει την επικοινωνία του τηλεφώνου, την επικοινωνία δεδομένων (μέσω μιας ολοκληρωμένης στοίβας TCP/IP) και την επικοινωνία με το κύκλωμα το οποίο επιδρά στο ίδιο το τηλέφωνο (από μια UART και μια TTL σειριακή διεπαφή).

Σχήμα 11. Όψη SIM900 module

Ο επεξεργαστής είναι επίσης υπεύθυνος και για την κάρτα SIM η οποία λειτουργεί με 3 ή 1.8 V και συνδέεται εξωτερικά με το module. Επίσης το module ενσωματώνει μια αναλογική διεπαφή, έναν μετατροπέα αναλογικού σήματος σε ψηφιακό (A/D converter), ένα ρολόι πραγματικού χρόνου (RTC Real Time Clock), SPI bus, I2C, και PWM λειτουργία. Το τμήμα της ραδιοεπικοινωνίας είναι το GSM 2/2+ και είτε κατηγορίας 4 (2W) στα 850/900 Mhz είτε κατηγορίας 1 (1W) στα 1800/1900 Mhz.

Η σειριακή διεπαφή TTL είναι υπεύθυνη όχι μόνο για την επικοινωνία όλων των δεδομένων σε σχέση με τα γραπτά μηνύματα που έχουν ήδη ληφθεί ή αυτά που αναμένονται να ληφθούν κατά την διάρκεια μιας TCP/IP συνεδρίας στο GPRS αλλά και για την λήψη των εντολών του κυκλώματος οι οποίες είναι εντολές AT (AT commands). Τέλος η μονάδα τροφοδοτείτε με συνεχή τάση (3,4 έως 4,5 V) και απορροφά 0,8 A κατά την διάρκεια των μεταδόσεων.

3.3.2 Προγραμματισμός SIM900

Ο προγραμματισμός του SIM900 ,όπως είπαμε και παραπάνω, γίνετε με τις εντολές AT. Οι εντολές αυτές χρησιμοποιούνται για να ελέγξουν ένα modem και το AT είναι η συντομογραφία της αγγλικής λέξης attention. Έτσι και στην δική μας περίπτωση το GSM modem ελέγχεται από τέτοιες εντολές οι οποίες είναι εξειδικευμένες για την GSM τεχνολογία.

Το AT στην αρχή κάθε εντολής ενημερώνει το modem σχετικά με την αρχή κάποιας νέας εντολής.

Υπάρχουν 2 βασικές κατηγορίες εντολών και αυτές είναι οι παρακάτω:

- Βασικές εντολές: Είναι οι εντολές οι δεν αρχίζουν με το "+". Για παράδειγμα το D (Dial) ATD, A (Answer) ATA είναι βασικές εντολές.
- Εκτεταμένες εντολές: Είναι οι εντολές οι οποίες ξεκινούν με το "+". Όλες οι εντολές του GSM είναι εκτεταμένες εντολές. Για παράδειγμα η εντολή +CMGS είναι μια εκτεταμένη εντολή η οποία χρησιμοποιείτε για να στείλουμε ένα μήνυμα.

Η σύνταξη των AT εντολών έχει ως εξής:

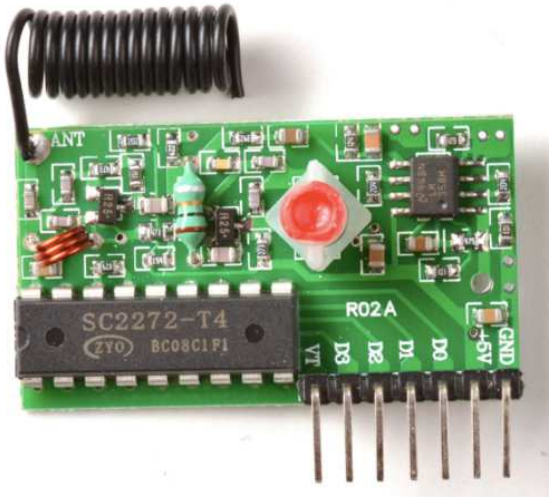
Κάθε εντολή ξεκινά με το "AT" ή "at", στην συνέχεια ακολουθεί η εντολή και μετά οι παράμετροι της εντολής.

"AT<x><n>"

όπου: χ είναι η εντολή και n οι παράμετροι.

3.4 Επέκταση RF

Η προς ανάπτυξη συσκευή μας θα δίνει την δυνατότητα στον χρήστη να μπορεί να ελέγχει κάποιες συσκευές τοπικά με ασύρματο χειρισμό, όπως επίσης και να στείλει ένα γραπτό μήνυμα ή ένα e-mail με το πάτημα ενός πλήκτρου. Για να καταφέρουμε να πραγματοποιήσουμε αυτές τις λειτουργίες θα χρησιμοποιήσουμε την επέκταση RF module την οποία θα συνδέσουμε στο Arduino και στην συνέχεια με προγραμματιστικό τρόπο θα εκτελούμε τις παραπάνω λειτουργίες. Η επέκταση αυτή αποτελείτε από 2 μέρη, τον πομπό και τον δέκτη οι οποίοι φαίνονται στα παρακάτω σχήματα.



Σχήμα 12. Όψη δέκτη RF

Σχήμα 13. Όψη πομπού RF

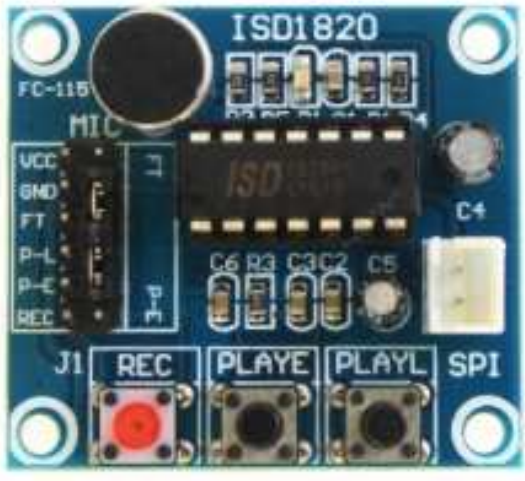
Η λειτουργία του δέκτη είναι βασισμένη στο ολοκληρωμένο κύκλωμα PT2272 το οποίο είναι ένας αποκωδικοποιητής απομακρυσμένου ελέγχου ο οποίος διαθέτει θύρες τριών καταστάσεων των 12 bit η κάθε μια και μας δίνει συνολικά 531.441 διαθέσιμες διευθύνσεις.

Το module λειτουργεί στα 433,92 Mhz και χρειάζεται τροφοδοσία 5V την οποία την παρέχουμε απ' ευθείας από το Arduino. Η λειτουργία του είναι πάρα πολύ απλή, όταν το module έχει τροφοδοτηθεί με την τάση των 5V και πατηθεί κάποιο πλήκτρο από τον πομπό τότε ενεργοποιείτε και η αντίστοιχη έξοδος.

3.5 Επέκταση εγγραφής φωνής

Για την λειτουργία της κλήσης φωνής σε περίπτωση ανάγκης με χρήση προ-ηχογραφημένου μηνύματος θα χρησιμοποιήσουμε την επέκταση ISD1820. Η επέκταση αυτή είναι βασισμένη στο ολοκληρωμένο ISD1820 το οποίο διαθέτει εσωτερικό ταλαντωτή με εξωτερικό έλεγχο, προενισχυτή μικροφώνου, αυτόματο έλεγχο κέρδους, περιοχή μνήμης και προενισχυτή για ηχεία.

Οι εγγραφές αποθηκεύονται σε μη πτητικές περιοχές της μνήμης του και έτσι το καθιστούν χαμηλό σε κατανάλωση ενέργειας. Επίσης όλα τα δείγματα φωνής αποθηκεύονται κατευθείαν στην μνήμη στην φυσική τους μορφή παρέχοντας έτσι υψηλής ποιότητας δείγμα φωνής.



Σχήμα 14. Όψη επέκτασης εγγραφής φωνής

Αναλυτικά η συγκεκριμένη επέκταση διαθέτει:

- Περιοχή με πλήκτρα για έλεγχο των λειτουργιών.
- Αυτόματη λειτουργία κατά την οποία απενεργοποιείτε (μετά από κάποιο χρόνο).
- Ενσωματωμένη προενίσχυση για οδήγηση ηχείου 8Ω.
- Μπορεί να ελεγχθεί τοπικά ή από μικροελεγκτή.
- Ο ρυθμός της δειγματοληψίας και η διάρκεια του δείγματος αλλάζουν εύκολα με την χρήση μιας αντίστασης.
- Μπορεί να πραγματοποιήσει ηχογράφηση έως 20 sec.

Παρακάτω φαίνεται πίνακας με τις τιμές των αντιστάσεων που μπορούν να τοποθετηθούν και την αντίστοιχη διάρκεια εγγραφής που προκύπτει.

ROSC	Duration	Sample Rate	Bandwidth
80KΩ	8 secs	8.0KHz	3.4KHz
100KΩ	10 secs	6.4KHz	2.6KHz
120KΩ	12 secs	5.3KHz	2.3KHz
160KΩ	16 secs	4.0KHz	1.7KHz
200KΩ	20 secs	3.2KHz	1.3KHz

Πίνακας 4. Τιμές αντιστάσεων και διάρκεια εγγραφής φωνής

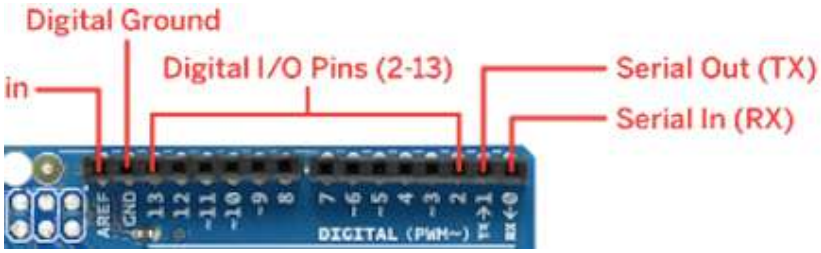
ΚΕΦΑΛΑΙΟ 4
ΜΕΘΟΔΟΛΟΓΙΑ ΑΝΑΠΤΥΞΗΣ

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιάσουμε αναλυτικά την μεθοδολογία την οποία ακολουθήσαμε για την ανάπτυξη της συσκευής μας. Θα παρουσιάσουμε πως τα υλικά τα οποία επιλέξαμε από το προηγούμενο κεφάλαιο συντίθεται ώστε τελικά να έχουμε το επιθυμητό αποτέλεσμα. Επίσης θα παρουσιαστούν οι επί μέρους κώδικες των δοκιμών, καθώς και ο τελικός κώδικας της συσκευής.

4.2 Επικοινωνία Arduino και SIM900

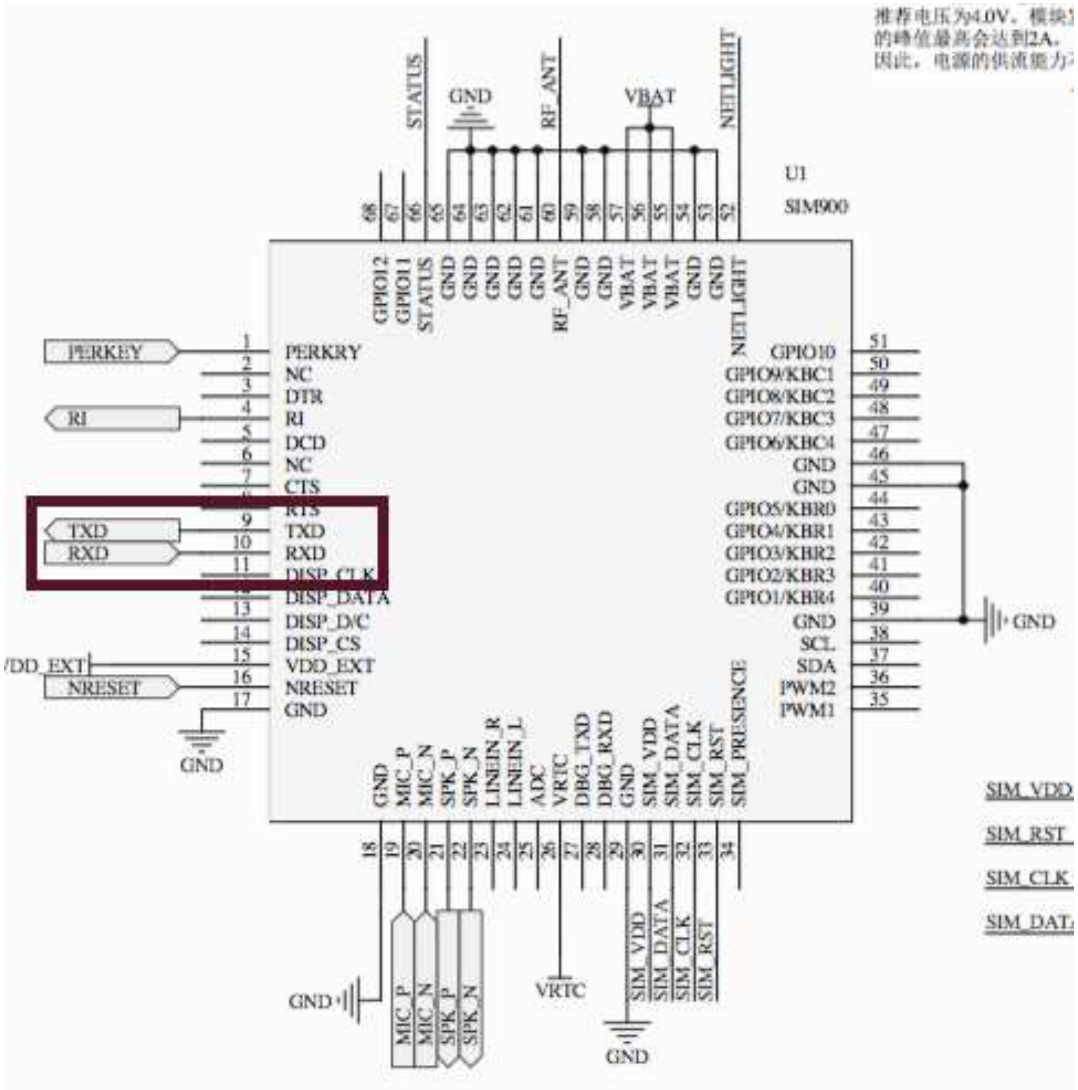
Ξεκινώντας την ανάπτυξη μας ασχοληθήκαμε πρώτα με την επικοινωνία του Arduino και του SIM900. Αρχικά δημιουργήσαμε ένα πρόγραμμα μέσω του οποίου μπορούμε να στείλουμε AT εντολές στο SIM900 και στην συνέχεια να δούμε τις απαντήσεις που αποστέλλει μέσω της φυσικής σειριακής θύρας την οποία διαθέτει. Η σειριακή θύρα του Arduino υλοποιείτε από το UART ολοκληρωμένο το οποίο υπάρχει μέσα στον μικροελεγκτή (ATMEGA328P) και καταλήγει στα pins 0 (Serial ιn) και 1(Serial out).



Σχήμα 15. Σειριακή θύρα Arduino

Για να μπορέσουμε να δούμε τις απαντήσεις των εντολών που θα αποστέλλουμε θα πρέπει να έχουμε σειριακή επικοινωνία και με το SIM900. Λόγω του ότι το Arduino Uno διαθέτει μόνο μια φυσική σειριακή θύρα θα πρέπει να δημιουργήσουμε άλλη μια προγραμματιστικά. Αυτό επιτυγχάνετε με την χρήση της βιβλιοθήκης **SoftwareSerial.h** με την χρήση της οποίας μπορούμε να χρησιμοποιήσουμε οποιοδήποτε pin του Arduino ώστε να αναπαραστήσουμε την

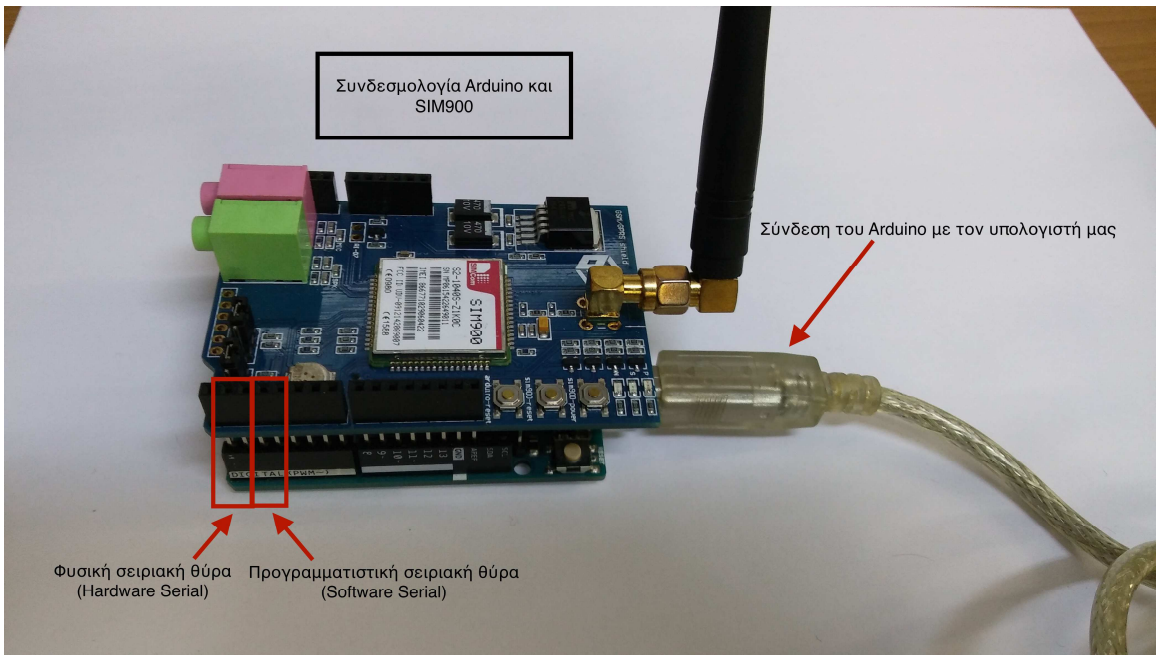
λειτουργία της φυσικής σειριακής θύρας (εξ ου και το όνομα της). Η προγραμματιστική σειριακή θύρα μας στο Arduino δημιουργήθηκε στα pins 2 και 3 διότι σε αυτά καταλήγει η φυσική σειριακή θύρα από το SIM900. Παρακάτω φαίνεται σχήμα από το σχέδιο του SIM900 αναπτυξιακού όπου στα pins 9 και 10 φαίνεται η φυσική σειριακή του θύρα.



Σχήμα 16. Σειριακή θύρα του SIM900

4.2.1 Συνδεσμολογία Arduino και SIM900

Στην συνέχεια φαίνεται η συνδεσμολογία του Arduino με το GSM shield η οποία επιτυγχάνετε με την ένωση του SIM900 shield με το Arduino. Έπειτα με το USB καλώδιο συνδεόμαστε από το Arduino στον υπολογιστή μας και μέσω του προγράμματος ανάπτυξης αποκτούμε πρόσβαση στην σειριακή του θύρα.



Σχήμα 17. Συνδεσμολογία Arduino - SIM900 - Υπολογιστής

Στην επόμενη ενότητα θα δούμε το πρόγραμμα το οποίο δημιουργήσαμε ώστε να μπορούμε να στείλουμε εντολές στο SIM900 και να δούμε τις απαντήσεις τις οποίες μας στέλνει.

4.2.2 Πρόγραμμα και επεξήγηση λειτουργίας

Παρακάτω φαίνεται ο κώδικας και η επεξήγηση των επί μέρους λειτουργιών του.

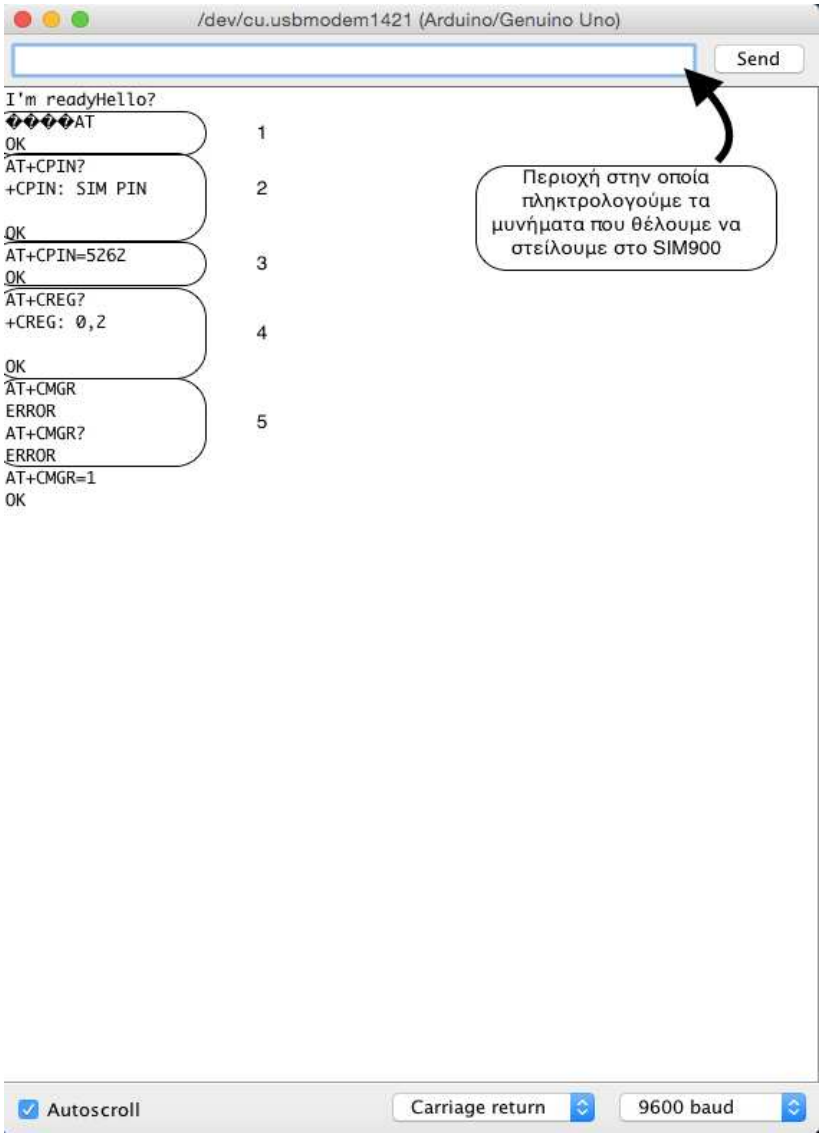
```
#include <SoftwareSerial.h>

SoftwareSerial GPRS(2, 3);
char buffer[64];

void setup(){
```

```
    GPRS.begin(9600);              // the GPRS baud rate
    Serial.begin(9600);            // the Serial port of Arduino baud rate.
    Serial.print("I'm ready");
    Serial.println("Hello?");
  }
  void loop(){
    if (GPRS.available()){         // if data is coming from softwareserial port ==>
data is comming from gprs shield
      while(GPRS.available()){          // reading data into char array
        buffer[count++]=GPRS.read();    // writing data into array
        if(count == 30)break;
      }
    Serial.write(buffer,count);       // if no data transmission ends, write buffer to
hardware serial port
    clearBufferArray();               // call clearBufferArray function to clear the
storaged data from the array
    count = 0;                        // set counter of while loop to zero
  }
  if (Serial.available())             // if data is available on hardwareserial port ==>
data is comming from PC or notebook
    GPRS.write(Serial.read());        // write it to the GPRS shield
  }
  void clearBufferArray(){            // function to clear buffer array
    for (int i=0; i<count;i++){
      buffer[i]=NULL;                 // clear all index of array with command NULL
    }
  }
```

Αρχικά εισάγουμε την βιβλιοθήκη που μας επιτρέπει να δημιουργήσουμε προγραμματιστικές σειριακές θύρες. Στην συνέχεια δημιουργούμε προγραμματιστική σειριακή επικοινωνία στα pins 2 και 3 του Arduino και έναν buffer (πίνακας από char) ώστε να μπορούμε να αποθηκεύουμε τις απαντήσεις του SIM900 και να τις προβάλουμε στην οθόνη μας.

Στην setup() συνάρτηση του κώδικα θέτουμε σε λειτουργία και τις δύο σειριακές θύρες μας και τυπώνουμε ένα μήνυμα ετοιμότητας (για λόγους debuging).

Στην συνάρτηση loop() του κώδικα, η οποία "τρέχει" συνέχεια, όταν υπάρχουν δεδομένα από την σειριακή θύρα του SIM900 τα αποθηκεύουμε στην μεταβλητή buffer την οποία αναφέραμε και νωρίτερα. Έπειτα τα δεδομένα αυτά τα γράφουμε στην σειριακή θύρα του Arduino και τα προβάλουμε την οθόνη του υπολογιστή μας μέσω του Serial monitor που διαθέτει το περιβάλλον προγραμματισμού. Στην συνέχεια δείχνουμε τα αποτελέσματα από τις εντολές τις οποίες στείλαμε στο SIM900 και επεξηγούμε τα αποτελέσματα.



Σχήμα 18. Αποτελέσματα εντολών από το GSM shield

Από τα αποτελέσματα τα οποία λάβαμε και φαίνονται στην παραπάνω εικόνα παρατηρούμε ότι η επικοινωνία μας είναι επιτυχής και λαμβάνουμε απαντήσεις. Οι απαντήσεις από τις δοκιμαστικές εντολές που στείλαμε είναι οι ακόλουθες:

1. Το SIM900 μας απάντησε θετικά στην εντολή ετοιμότητας που του αποστείλαμε (AT). Είναι δηλαδή έτοιμο να δεχθεί περαιτέρω εντολές.
2. Στην ερώτηση εάν η κάρτα SIM που χρησιμοποιούμε έχει κωδικό μας απάντησε θετικά (AT+CPIN?).
3. Αποστείλαμε εντολή με την οποία του γνωστοποιούμε τον κωδικό pin της SIM κάρτας και λαμβάνουμε επιβεβαίωση οκ που σημαίνει πως ο κωδικός είναι σωστός (AT+CPIN=5262).
4. Στην ερώτηση εάν έχει γίνει εγγραφή στο δίκτυο κινητής τηλεφωνίας το SIM900 μας απάντησε θετικά και είμαστε πλέον έτοιμοι να πραγματοποιήσουμε κλήσεις και να στείλουμε μηνύματα (AT+CREG?).
5. Σε αυτό το σημείο βλέπουμε ότι εάν δώσουμε λάθος εντολές στο SIM900 αυτό μας απαντά με ERROR.

4.2.3 Συμπεράσματα ενότητας

Με το πέρας αυτής της ενότητας καταφέραμε να έχουμε επικοινωνία με το Arduino και το SIM900 και στο εξής μπορούμε να στείλουμε οποιαδήποτε εντολή και να πραγματοποιήσουμε τις λειτουργίες τις οποίες έχουμε προδιαγράψει. Στην συνέχεια θα δείξουμε αναλυτικά πως επιτυγχάνουμε αυτές τις λειτουργίες.

Μετά από εκτεταμένες δοκιμές σε αυτό το σημείο παρατηρήσαμε πως η συγκεκριμένη επέκταση (shield) την οποία χρησιμοποιούμε παρουσιάζει αστάθειες και έχουμε συνεχείς συνδέσεις και αποσυνδέσεις από το δίκτυο κινητής τηλεφωνίας. Επίσης πολλές φορές το shied δεν ανταποκρινόταν στις AT εντολές τις οποίες του αποστέλλαμε. Το συγκεκριμένο πρόβλημα λύνετε με επανεκκίνηση του Arduino (cycle power).

Ακόμα παρατηρήσαμε πως το SIM900 shield πρέπει να το ενεργοποιούμε χειροκίνητα από τα πλήκτρα τα οποία διαθέτει και τα οποία φαίνονται στην παρακάτω εικόνα 15. Επομένως θα πρέπει να δημιουργήσουμε προγραμματιστικά έναν μηχανισμό ο οποίος θα ενεργοποιεί το shield με το που λάβει τάση από το Arduino ώστε η συσκευή μας να είναι αυτόνομη και να μην χρειάζεται ανθρώπινη παρουσία για να την ενεργοποιήσει.

Σχήμα 19. Πλήκτρα λειτουργίας του SIM900

Από το σχέδιο του shield είδαμε πως το pin το οποίο ενεργοποιεί το ολοκληρωμένο και στο οποίο βρίσκεται και το power button είναι το pin 8. Για να ενεργοποιήσουμε το ολοκληρωμένο θα πρέπει να θέσουμε σε κατάσταση HIGH το pin για 1.2 sec και στην συνέχεια σε κατάσταση LOW για οχτώ δευτερόλεπτα.

Από την στιγμή που αυτό το pin χρησιμοποιείται από το SIM900 δεν μπορούμε να το χρησιμοποιήσουμε για τίποτα άλλο διότι θα δημιουργήσουμε πρόβλημα στην λειτουργία του shield ανοιγοκλείνοντας το.

Συνάρτηση για ενεργοποίηση του shield:

```
void powerOn()
{
  uint8_t answer = 0;
  answer = sendATcommand("AT", "OK", 300);     // checks if the module is started
  if (answer == 0)
  {
    // power on pulse
    digitalWrite(GSM_pin,HIGH);
```

```
        delay(1200);
        digitalWrite(GSM_pin,LOW);
        delay(8000);
        // waits for an answer from the module
        while(answer == 0){     // Send AT every two seconds and wait for the
answer
         answer = sendATcommand("AT", "OK", 5000);
        }
      }
      Serial.println(F("Module enabled..."));
   }
```

Στην παραπάνω συνάρτηση χρησιμοποιείτε και μια άλλη συνάρτηση με το όνομα sendATcommand() με την οποία στέλνουμε τις AT εντολές μας στο SIM900. Η λειτουργία της φαίνεται στην επόμενη ενότητα.

4.3 Μέθοδος αποστολής εντολών στο SIM900

Συνεχίζοντας την ανάπτυξη μας δημιουργήσαμε μια συνάρτηση με την οποία όταν την καλούμε μπορούμε να στέλνουμε AT εντολές στο SIM900. Πιο συγκεκριμένα η συνάρτηση ονομάζετε sendATcommand() και δέχεται τρία ορίσματα. Ως πρώτο όρισμα δέχεται την εντολή την οποία θέλουμε να αποστείλουμε στο SIM900, ο δεύτερος όρος είναι η απάντηση την οποία περιμένουμε να λάβουμε και ως τρίτο όρο τον χρόνο κατά τον οποίο θα περιμένει για να λάβει απάντηση. Η συνάρτηση φαίνεται παρακάτω:

Συνάρτηση αποστολής εντολών AT:

```
int8_t sendATcommand(char* ATcommand, char* expected_answer, unsigned int
timeout)
   {
      uint8_t x=0,  answer=0;
      char response[100];
      unsigned long previous;
      memset(response, '\0', 100);    // Initialize the string
      delay(100);
```

```
    while( GSM.available() > 0) GSM.read();    // Clean the input buffer
    GSM.println(ATcommand);    // Send the AT command
    x = 0;
    previous = millis();
    // this loop waits for the answer
    do{
      // if there are data in the UART input buffer, reads it and checks for the
asnwer
      if(GSM.available() != 0){
        response[x] = GSM.read();
        x++;
        // check if the desired answer  is in the response of the module
        if (strstr(response, expected_answer) != NULL)
        {
          answer = 1;
        }
      }
      // Waits for the asnwer with time out
    }while((answer == 0) && ((millis() - previous) < timeout));
    return answer;
  }
```

Με αυτόν τον τρόπο μπορούμε να ελέγχουμε εάν σε κάθε εντολή που στέλνουμε στο SIM900 λαμβάνουμε και την αναμενόμενη απάντηση. Έχοντας πλέον τον τρόπο με τον οποίο θα αποστέλλουμε τις εντολές μας αλλά και να ελέγχουμε την επιθυμητή απάντηση μπορούμε να προχωρήσουμε στις λειτουργίες της διαχείρισης των γραπτών μηνυμάτων.

4.4 Διαχείριση γραπτών μηνυμάτων (SMS)

Εφόσον στην προηγούμενη ενότητα καταφέραμε να έχουμε επικοινωνία με το SIM900 αλλά και να στέλνουμε και να ελέγχουμε τις εντολές που αποστέλλουμε προχωράμε στο να χτίσουμε τις λειτουργίες τις οποίες προδιαγράψαμε ότι θα πραγματοποιεί η συσκευή μας. Σε αυτή την ενότητα θα ασχοληθούμε με την λειτουργία της αποστολής και λήψης γραπτών μηνυμάτων και με τις εντολές με τις οποίες το πραγματοποιούμε. Η συνδεσμολογία παραμένει ως έχει με την προηγούμενη ενότητα καθώς προς το παρόν δεν χρειαζόμαστε κάτι άλλο.

4.4.1 Αποστολή γραπτού μηνύματος

Για να αποστείλουμε ένα γραπτό μήνυμα πρέπει να χρησιμοποιήσουμε τις κατάλληλες εντολές AT με τις οποίες θα ρυθμίσουμε το SIM900.

Η πρώτη από αυτές τις εντολές είναι η **AT+CMGF** με την οποία ρυθμίζουμε το GSM modem να λειτουργήσει σε TEXT ή PDU mode. Πιο συγκεκριμένα στο TEXT mode τα μηνύματα αναπαρίστανται ως κανονικό κείμενο το οποίο μπορούμε να διαβάσουμε και στο PDU mode όλα τα μηνύματα αναπαρίστανται ως δυαδικές συμβολοσειρές κωδικοποιημένες σε δεκαεξαδικούς χαρακτήρες, π.χ. 31020B911326880736F40000A900. Εμείς χρησιμοποιούμε το TEXT mode διότι είναι πιο εύκολο στην χρήση του. Η σύνταξη της εν λόγω εντολής φαίνεται στο παρακάτω σχήμα:

Command	**Positive Response**
AT+CMGF=<mode><CR>	OK

Parameters
<mode>: **0 = PDU Mode**, 1 = Text Mode
<CR> = ASCII character 13

Σχήμα 20. Σύνταξη εντολής AT+CMGF

Η εντολή που θα στείλουμε εμείς είναι η: **AT+CMGF = 1** για λειτουργία TEXT MODE.

Η επόμενη εντολή την οποία θα χρειαστούμε είναι η **AT+CMGS** με την οποία αποστέλλουμε το γραπτό μας μήνυμα. Η σύνταξη της εν λόγω εντολής φαίνεται στο παρακάτω σχήμα:

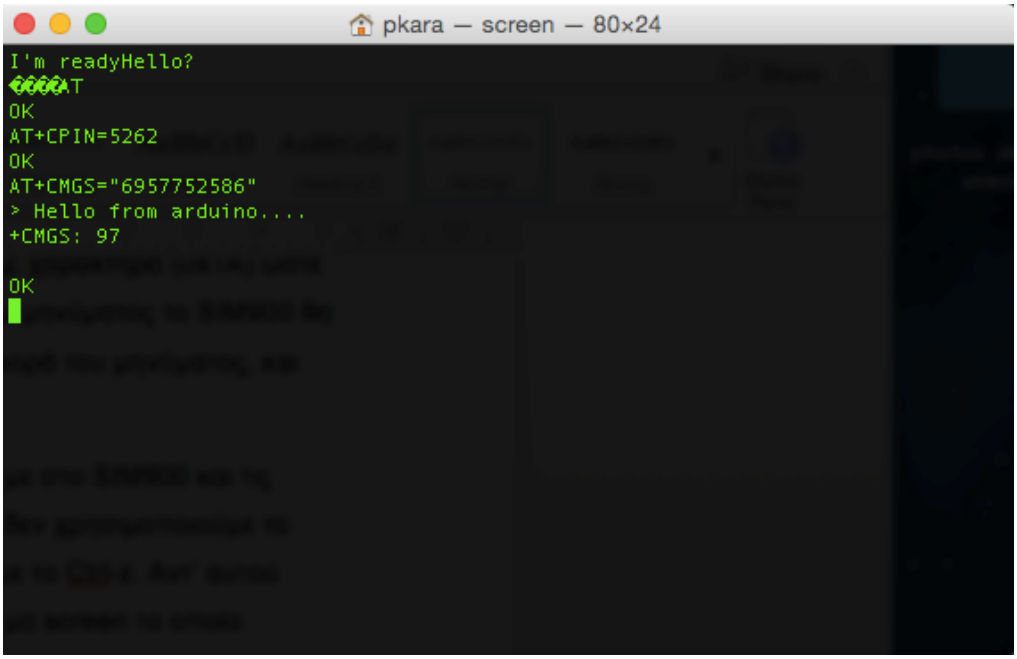
Command	Response
AT+CMGS=<number><CR> <message><CTRL-Z>	+CMGS:<mr> OK

Parameters
<CR> = ASCII character 13
<CTRL-Z> = ASCII character 26
<mr> = Message Reference

Σχήμα 21. Σύνταξη εντολής AT+CMGS

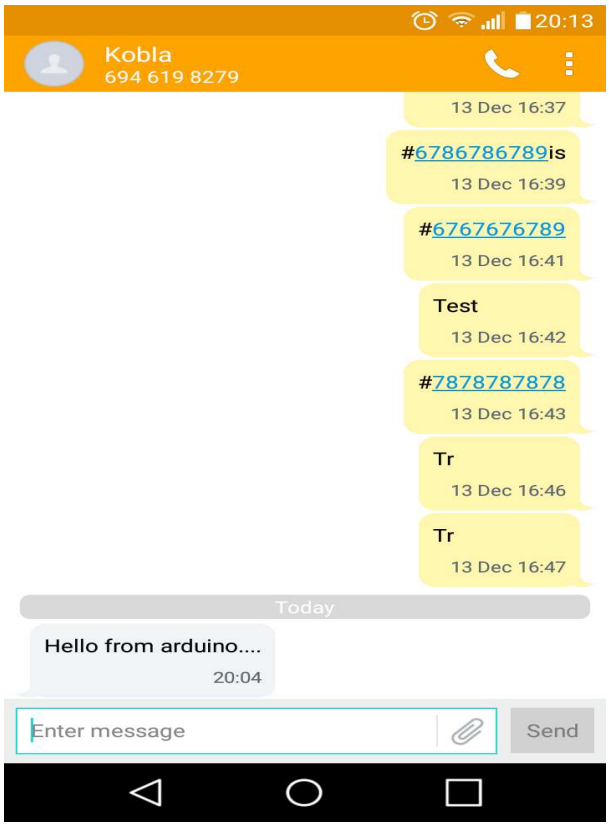
Όπως βλέπουμε και στο παραπάνω σχήμα για να αποστείλουμε ένα γραπτό μήνυμα στον αριθμό 695 7752586 θα πρέπει να αποστείλουμε την εντολή **AT+CMGS="+0306957752586"**. Με το που σταλεί η συγκεκριμένη εντολή μας απαντά το SIM900 με τον χαρακτήρα " > " και περιμένει να πληκτρολογήσουμε το μήνυμα μας. Εφόσον πληκτρολογήσουμε το μήνυμα μας θα πρέπει μετά να εισάγουμε το CTRL-Z χαρακτήρα (0x1A) ώστε να αποσταλεί το μήνυμα μας. Σε επιτυχή αποστολή του μηνύματος το SIM900 θα μας απαντήσει με +CMGS:<mr>, όπου mr είναι η αναφορά του μηνύματος, και στο τέλος ok.

Αρχικά δείχνουμε τις εντολές τις οποίες αποστέλλουμε στο SIM900 και τις απαντήσεις τις οποίες λαμβάνουμε. Σε αυτό το σημείο δεν χρησιμοποιούμε το serial monitor διότι δεν μπορούμε να εισάγουμε τον χαρακτήρα Ctrl-z. Αντ' αυτού χρησιμοποιούμε το πρόγραμμα screen από την κονσόλα εντολών του λειτουργικού της Apple.



Σχήμα 22. Αποστολή γραπτού μηνύματος από κονσόλα.

Παρατηρούμε ότι το μήνυμα στάλθηκε επιτυχώς εφόσον βλέπουμε την απάντηση +CMGS:97. Παρακάτω βλέπουμε και το μήνυμα το οποίο λάβαμε στο κινητό μας.



Σχήμα 23. Παραλαβή μηνύματος στο κινητό μας.

Εφόσον παρατηρήσαμε ότι μπορούμε να στείλουμε μήνυμα με τις AT εντολές από την κονσόλα μας δημιουργούμε μια συνάρτηση που θα χρησιμοποιήσουμε στον τελικό μας κώδικα και η οποία όταν καλείτε θα αποστέλλει ένα γραπτό μήνυμα με κείμενο το οποίο έχουμε προδιαγράψει. Παρακάτω φαίνεται η συνάρτηση την οποία δημιουργήσαμε και η οποία δέχεται όρισμα τον αριθμό του τηλεφώνου στον οποίο και αποστείλει το γραπτό μήνυμα.

Συνάρτηση - κώδικας για αποστολή γραπτού μηνύματος:

```
void sendHelpSms(char* tel)
{
   GSM.print("AT+CMGF=1\r");
   delay(1000);
   sprintf(aux_str,"AT+CMGS=\"%s\"\r", tel);
   Serial.println(aux_str);
   GSM.print(aux_str);
   delay(1000);
   GSM.println(strcpy_P(buffer,(char*)pgm_read_word(&(string_table[4]))));
   GSM.print("\r");
   delay(1000);
   GSM.println((char)26);
   GSM.println();
   delay(3000);
}
```

Στην παραπάνω συνάρτηση χρησιμοποιείτε η εντολή strcpy_P διότι έχουμε αποθηκεύσει όλα μας τα λεκτικά σε πίνακες στην Flash μνήμη του συστήματος. Αυτό το κάναμε διότι τα λεκτικά καταλάμβαναν το μεγαλύτερο μέρος της μνήμης RAM με αποτέλεσμα το σύστημα να μην έχει επαρκή μνήμη και να καταρρέει. Στην συνέχεια θα δείξουμε ολοκληρωμένα πως διαχειριστήκαμε το εν λόγω πρόβλημα. Επίσης βλέπουμε και την υλοποίηση του χαρακτήρα Ctrl-Z με τον αριθμό 26 από τον πίνακα ASCI.

4.4.2 Λήψη γραπτού μηνύματος

Όπως και στην αποστολή γραπτού μηνύματος έτσι και στην λήψη θα πρέπει να χρησιμοποιήσουμε AT εντολές.

Ο κώδικας μας έχει υλοποιηθεί με την λογική της μηχανής πεπερασμένων καταστάσεων (state machine) και έτσι θέλουμε να γνωρίζουμε εάν έχει έρθει κάποιο μήνυμα ώστε να προβούμε στις απαραίτητες ενέργειες. Για να το επιτύχουμε αυτό χρησιμοποιούμε την εντολή **AT+CNMI** με την οποία ρυθμίζουμε την ειδοποίηση ενός νέου μηνύματος. Παρακάτω φαίνεται η σύνταξη της εντολής όπως υπάρχει και στο manual:

Write Command	Response
AT+CNMI=<mode>[,<mt>[,<bm>[,<ds>[,<bfr>]]]]	TA selects the procedure for how the receiving of new messages from the network is indicated to the TE when TE is active, e.g. DTR signal is ON. If TE is inactive (e.g. DTR signal is OFF), message receiving should be done as specified in GSM 03.38. **OK** **ERROR**

Σχήμα 24. Σύνταξη εντολής AT+CNMI

Η εντολή την οποία αποστέλλουμε εμείς είναι η **AT+CNMI = 1,1,0,0,0**.

Παρακάτω φαίνεται η συνάρτηση την οποία δημιουργήσαμε και η οποία καλείτε στην αρχή του κώδικα μας στην συνάρτηση void setup().

```
void setSmsIndicationOn()
{
if(sendATcommand(strcpy_P(buffer,(char*)pgm_read_word(&(sim_commands[
4]))), "OK", 2000))
  {
    Serial.println(F("Sms indication is on..."));
  }
  else Serial.println(F("Sms indication did not set..."));
  clearMemBuffer();
}
```

Η συνάρτηση clearMemBuffer() χρησιμοποιείτε για να καθαρίζει την μεταβλητή buffer και να είναι έτοιμη για την επόμενη φορά που θα χρησιμοποιηθεί.

Η επόμενη εντολή AT την οποία χρειαζόμαστε είναι η **AT+CPMS** με την οποία επιλέγουμε την περιοχή μνήμης από την οποία θα διαβάζουμε τα εισερχόμενα γραπτά μηνύματα στην συσκευή μας. Η συγκεκριμένη εντολή χρησιμοποιείτε επίσης και για να βρούμε και τον αριθμό τον μηνυμάτων που είναι αποθηκευμένα στην συγκεκριμένη περιοχή μνήμης αλλά και για να βρούμε και τον μέγιστο αριθμό μηνυμάτων που μπορούν να αποθηκευτούν. Παρακάτω φαίνεται η σύνταξη της εντολής:

Write Command	Response
AT+CPMS=	TA selects memory storages <mem1>, <mem2> and <mem3> to be used for
<mem1>	reading, writing, etc.
[,<mem2>	**+CPMS: <used1>,<total1>,<used2>,<total2>,<used3>,<total3>**
[,<mem3>]]	
	OK
	ERROR

Parameters	
<mem1>	Messages to be read and deleted from this memory storage "SM" SIM message storage
<mem2>	Messages will be written and sent to this memory storage "SM" SIM message storage
<mem3>	Received messages will be placed in this memory storage if routing to PC is not set ("+CNMI") "SM" SIM message storage
<usedx>	integer type; Number of messages currently in <memx>
<totalx>	integer type; Number of messages storable in <memx>

Σχήμα 25. Σύνταξη εντολής AT+CPMS

Η εντολή που αποστέλλουμε εμείς είναι η **AT+CPMS=\"SM\",\"SM\",\"SM\";** με την οποία επιλέγουμε την μνήμη της κάρτας SIM και στις τρείς παραπάνω περιπτώσεις. Στην συνέχεια φαίνεται η συνάρτηση με την οποία στέλνουμε την παραπάνω εντολή.

Συνάρτηση επιλογής μνήμης γραπτών μηνυμάτων

```
void smsMemory()
{
  if(sendATcommand(AT+CPMS=\"SM\",\"SM\",\"SM\", "OK", 1000))   // selects the memory
  {
    Serial.println(F("SIM memory selected"));
  }
}
```

Το επόμενο βήμα είναι να διαβάσουμε το γραπτό μήνυμα το οποίο θα λάβουμε και στην συνέχεια να προβούμε στις απαραίτητες ενέργειες. Η AT εντολή για να διαβάσουμε ένα γραπτό μήνυμα είναι η **AT+CMGR** και η σύνταξη της φαίνεται παρακάτω:

AT+CMGR Read SMS Message	
Test Command **AT+CMGR=?**	Response **OK**
Write Command **AT+CMGR=<index>[,<mode>]**	Parameters **<index>** integer type; value in the range of location numbers supported by the associated memory **<mode>** 0 normal 1 not change status of the specified SMS record
	Response TA returns SMS message with location value <index> from message storage <mem1> to the TE. If status of the message is 'received unread', status in the storage changes to 'received read'. 1) If text mode (+CMGF=1) and Command successful: for SMS-DELIVER: **+CMGR:** **<stat>,<oa>,[<alpha>],<scts>[,<tooa>,<fo>,<pid>,<dcs>,<sca>,<tosca>,<length>]<CR><LF><data>** for SMS-SUBMIT: **+CMGR:** **<stat>,<da>,[<alpha>][,<toda>,<fo>,<pid>,<dcs>,[<vp>],<sca>,<tosca>,<length>]<CR><LF><data>** for SMS-STATUS-REPORTs: **+CMGR: <stat>,<fo>,<mr>,[<ra>],[<tora>],<scts>,<dt>,<st>** for SMS-COMMANDs: **+CMGR:**

Σχήμα 26. Σύνταξη εντολής AT+CMGR

Η εντολή την οποία αποστέλλουμε εμείς είναι η **AT+CMGR=1**. Με αυτή την εντολή διαβάζουμε κάθε φορά το μήνυμα το οποίο βρίσκεται στην θέση 1 της κάρτας SIM. Ο λόγος που το κάνουμε αυτό είναι πως κάθε φορά που λαμβάνουμε ένα μήνυμα από κάποιον χρήστη το διαβάζουμε και αφού το επεξεργαστούμε και κάνουμε τις απαραίτητες ενέργειες το διαγράφουμε από την μνήμη SIM. Παρακάτω φαίνεται η συνάρτηση με την οποία διαβάζουμε ένα γραπτό μήνυμα και αν έχουμε το επιθυμητό μήνυμα (OPEN ή CLOSE) ανοίγουμε ή κλείνουμε μια επαφή ενός ρελέ.

Συνάρτηση για ανάγνωση ενός γραπτού μηνύματος :

```
void readSms()
{
  x = 0;
  answer = sendATcommand("AT+CMGR=1","+CMGR:", 300);    // reads the first SMS
    if (answer == 1)
    {
      answer = 0;
      while(GSM.available() == 0);
      // this loop reads the data of the SMS
      do{
      // if there are data in the UART input buffer, reads it and checks for the asnwer
        if(GSM.available() > 0)
        {
          SMS[x] = GSM.read();
          x++;
          // check if the desired answer (OK) is in the response of the module
          if (strstr(SMS, "OK") != NULL)
          {
            answer = 1;
          }
          if(strstr(SMS,"OPEN") != NULL)
          {
            arm_flag = 1;
```

```
            event = 1;
          }
          if(strstr(SMS,"CLOSE") != NULL)
          {
            disarm_flag = 1;
            event = 0;
          }
        }
        }while(answer == 0);    // Waits for the asnwer with time out
        SMS[x] = '\0';
        Serial.println(SMS);
        //clearSmsBuffer();
    }
    else
    {
        Serial.print(F("error "));
        Serial.println(answer, DEC);
}
```

4.4.3 Διαγραφή γραπτού μηνύματος

Για να μπορέσουμε να διαγράψουμε ένα μήνυμα από την μνήμη SIM χρησιμοποιούμε την AT εντολή **AT+CMGDA** της οποίας η σύνταξη φαίνεται παρακάτω:

Write Command	Response:
AT+CMGDA=<type>	**OK** **ERROR** **+CMS ERROR: <err>**
	Parameter **<type>** 1) If text mode: "DEL READ" delete all read messages "DEL UNREAD" delete all unread messages "DEL SENT" delete all sent SMS "DEL UNSENT" delete all unsent SMS "DEL INBOX" delete all received SMS "DEL ALL" delete all SMS 2) If PDU mode: 1 delete all read messages

Σχήμα 27. Σύνταξη εντολής AT+CMGDA

Η εντολή την οποία αποστέλλουμε είναι η **AT+CMGDA=\"DEL ALL\";** με την οποία διαγράφουμε όλα τα μηνύματα από την μνήμη. Στην συνέχεια φαίνεται και η συνάρτηση που την υλοποιεί.

Συνάρτηση για διαγραφή ενός γραπτού μηνύματος :

```
void deleteSms()
{
  if(sendATcommand(AT+CMGDA=\"DEL ALL\", "OK", 1000))    // deletes all
the messages
   {
    Serial.println(F("All sms deleted !!!"));
    del_flag = 1;
   } }
```

4.4.4 Συμπεράσματα ενότητας

Με το πέρας αυτής της ενότητας δείξαμε πως δομήσαμε τις λειτουργίες αποστολής, λήψης αλλά και διαγραφής γραπτών μηνυμάτων. Παρατηρήσαμε επίσης πως εάν στέλνουμε πολλές εντολές στο SIM900 ,χωρίς χρονική διαφορά η μια από την άλλη, τότε σταματάει να λειτουργεί εντελώς και επανεκκινείται. Επίσης

όπως αναφέραμε και παραπάνω αντιμετωπίσαμε πρόβλημα με την χαμηλή μνήμη RAM διότι τα λεκτικά τα οποία στέλνουμε στο SIM900 είναι πολλά αλλά και μεγάλα σε μήκος. Η αντιμετώπιση αυτού του προβλήματος φαίνεται στην επόμενη ενότητα που ακολουθεί.

4.5 Διαχείριση μνήμης RAM και FLASH

Όπως αναφέραμε και παραπάνω με την χρήση των πολλών arrays τύπου char είχαμε ως αποτέλεσμα την υπερφόρτωση της μνήμης RAM. Η μνήμη μας είχε φτάσει το 85% με αποτέλεσμα το πρόγραμμα να είναι ασταθές λόγω έλλειψης μνήμης. Για να αντιμετωπίσουμε αυτό το πρόβλημα αποφασίσαμε να αποθηκεύσουμε όλες αυτές τις μεταβλητές στην μνήμη flash του επεξεργαστή και όχι στην RAM όπως γίνεται εξ 'ορισμού. Για να το καταφέρουμε εισάγουμε αρχικά την βιβλιοθήκη **#include <avr/pgmspace.h>** και στην συνέχεια δηλώνουμε τις μεταβλητές μας ως εξής :

π.χ. const char sms_text_1[] PROGMEM = "Relay is open.";

Εφόσον δηλώσουμε τα λεκτικά μας στην συνέχεια δημιουργούμε έναν πίνακα όπου μέσα του βάζουμε όλα μας τα λεκτικά.

```
const dataType variableName[] PROGMEM = {data0, data1, data3...};
```

Στην συνέχεια όταν θέλουμε να χρησιμοποιήσουμε κάποιο από αυτά απλά αναφερόμαστε στον πίνακα με το όνομα του και στην θέση την οποία υπάρχει το λεκτικό που θέλουμε.

π.χ. pgm_read_word(&(sim_commands[4])

Επίσης είχαμε μεγάλη κατανάλωση της μνήμης RAM από τα πολλά Serial.println() τα οποία είχαμε δημιουργήσει για λόγους debugging και τα οποία και αυτά με την σειρά τους δημιουργούσαν το πρόβλημα το οποίο περιγράψαμε παραπάνω. Για να το αντιμετωπίσουμε χρησιμοποιήσαμε την μακροεντολή F() με την οποία το περιεχόμενο της Serial.println() αποθηκεύεται και αυτό στην μνήμη flash. Παρακάτω φαίνεται πως συντάσσουμε την εντολή ώστε ο compiler στην συνέχεια να την αποθηκεύσει στην μνήμη FLASH:

```
Serial.print(F("Write something on the Serial Monitor that is stored in FLASH"));
```

4.6 Αποστολή e-mail με χρήση GPRS

Η επόμενη λειτουργία της οποία και έχουμε προδιαγράψει ότι θα έχει η συσκευή μας είναι η αποστολή ενός e-mail με χρήση της τεχνολογίας GPRS.

Για να το καταφέρουμε χρησιμοποιούμε μια σειρά από πολλές εντολές AT και στην συνέχεια τις κλείνουμε όλες σε μια συνάρτηση η οποία και θα τις εκτελεί σειριακά όταν καλείτε.

Η πρώτη από τις εντολές αυτές είναι η **AT+SAPBR** με την οποία θέτουμε τον τύπο της σύνδεσης μας σε GPRS. Η εντολή που αποστέλλουμε είναι η **AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\".**

Στην συνέχεια ρυθμίζουμε το APN (Access Point Name) το οποίο στην περίπτωση μας εφόσον χρησιμοποιούμε κάρτα SIM της Vodafone είναι **AT+SAPBR=3,1,\"APN\",\"internet.vodafone.gr\".**

Αποστέλλουμε εντολή για ενεργοποίηση του GPRS **AT+SAPBR=1,1** και έπειτα ελέγχουμε εάν η σύνδεση στο δίκτυο GPRS έχει πραγματοποιηθεί με επιτυχία. Αυτό επιτυγχάνετε με την εντολή **AT+SAPBR=2,1** και μόλις λάβουμε απάντηση σαν την ακόλουθη σημαίνει πως η σύνδεση έχει πραγματοποιηθεί. Η απάντηση είναι : +SAPBR: 1,1,"100.120.204.132". Εφόσον έχουμε καταφέρει να έχουμε σύνδεση στο δίκτυο GPRS αποστέλλουμε τις παρακάτω εντολές με τις οποίες δίνουμε τις απαραίτητες πληροφορίες για τον SMTP server που χρησιμοποιούμε όπως και τον αποστολέα, τον παραλήπτη του e-mail κ.τ.λ. Οι εντολές είναι οι ακόλουθες:

- AT+SAPBR=3,1,\"USER\",\"\" Το όνομα χρήστη για τον λογαριασμό e-mail που θα χρησιμοποιήσουμε.
- AT+SAPBR=3,1,\"PWD\",\"\" Ο κωδικός χρήστη για το e-mail.

- AT+EMAILSSL=1 Ενημερώνουμε για ασφαλή σύνδεση με SSL
- AT+SMTPSRV=\"smtp.mail.yahoo.gr\" Ο SMTP server που θα χρησιμοποιήσουμε.
- AT+SMTPAUTH=1,\"panagiotis_karagounis@yahoo.gr\" Ο χρήστης του λογαριασμού.
- AT+SMTPFROM=\"panagiotis_karagounis@yahoo.gr\" Από ποιόν θα σταλεί το μήνυμα.
- AT+SMTPRCPT=0,0,\"panagiotis_karagounis@yahoo.gr\" Ο παραλήπτης του μηνύματος.
- AT+SMTPSUB=\"TEST E-MAIL\" Το θέμα του μηνύματος.
- AT+SMTPSEND Αποστολή του μηνύματος.

Όλες τις παραπάνω εντολές τις έχουμε αποθηκεύσει σε έναν πίνακα στην flash μνήμη μας ως ακολούθως :

```
const char contype[] PROGMEM = "AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"";
const char apn[] PROGMEM =
"AT+SAPBR=3,1,\"APN\",\"internet.vodafone.gr\"";
const char user[] PROGMEM = "AT+SAPBR=3,1,\"USER\",\"\"";
const char pwd[] PROGMEM = "AT+SAPBR=3,1,\"PWD\",\"\"";
const char gprs[] PROGMEM= "AT+SAPBR=1,1";
const char chk_ip[] PROGMEM = "AT+SAPBR=2,1";
const char ssl[] PROGMEM = "AT+EMAILSSL=1";
const char smtp[] PROGMEM = "AT+SMTPSRV=\"smtp.mail.yahoo.gr\"";
const char auth[] PROGMEM =
"AT+SMTPAUTH=1,\"panagiotis_karagounis@yahoo.gr\"";
const char from[] PROGMEM =
"AT+SMTPFROM=\"panagiotis_karagounis@yahoo.gr\"";
const char to[] PROGMEM =
"AT+SMTPRCPT=0,0,\"panagiotis_karagounis@yahoo.gr\"";
const char subj[] PROGMEM = "AT+SMTPSUB=\"TEST E-MAIL\"";
const char send_email[] PROGMEM = "AT+SMTPSEND";
```

Ο πίνακας τον οποίο δημιουργήσαμε φαίνεται παρακάτω:

const char* const sim_commands[] PROGMEM = {del_sms, sim_reg, sms_mode, sms_mem, sms_indic, read_sms, contype, apn, user, pwd, gprs, chk_ip, ssl, smtp, auth, from, to, subj, send_email};

Παρακάτω φαίνεται η συνάρτηση με την οποία ρυθμίζουμε την σύνδεση GPRS και η συνάρτηση με την αποστέλλουμε το e-mail.

Συνάρτηση σύνδεσης στο δίκτυο GPRS:

```
void setBearer()     // Function to sing in to GPRS network and get an IP.
{
  sendATcommand(strcpy_P(buffer,
      (char*)pgm_read_word(&(sim_commands[6]))), "OK", 500);
  sendATcommand(strcpy_P(buffer,
      (char*)pgm_read_word(&(sim_commands[7]))), "OK", 500);
  sendATcommand(strcpy_P(buffer,
      (char*)pgm_read_word(&(sim_commands[8]))), "OK", 500);
  sendATcommand(strcpy_P(buffer, (
      char*)pgm_read_word(&(sim_commands[9]))), "OK", 500);
  sendATcommand(strcpy_P(buffer,
      (char*)pgm_read_word(&(sim_commands[10]))), "OK", 500);
  if(sendATcommand(strcpy_P(buffer,
      (char*)pgm_read_word(&(sim_commands[11]))), "+SAPBR: 1,1", 500))
  {
    Serial.println(F("got ip..."));
  }
  else {
    Serial.println(F("did not get ip"));
    return;
  }
}
```

Συνάρτηση αποστολής e-mail:

```
void sendEmail()
{
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[12]))), "OK", 500))
  {
    Serial.println(F("ssl ok"));
  }
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[13]))), "OK", 500))
  {
    Serial.println(F("server ok"));
  }
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[14]))), "OK", 500))
  {
    Serial.println(F("auth ok"));
  }
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[15]))), "OK", 500))
  {
    Serial.println(F("from ok"));
  }
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[16]))), "OK", 500))
  {
    Serial.println(F("to ok"));
  }
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[17]))), "OK", 500))
  {
    Serial.println(F("subj ok"));
  }
```

```
    if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[18]))), "OK", 1000))
    {
      Serial.println(F("E-mail sent ok!!!"));
    }
    else Serial.println(F("E-mail didn't sent...."));
    if(sendATcommand("", "+SMTPSEND: 62", 500))
    {
      Serial.println(F("Network error..."));
    }
    if(sendATcommand("", "+SMTPSEND: 1", 30000))
    {
      Serial.println(F("E-mail sent ok!!!"));
    }
  }
```

4.7 Προσθήκη επαφών για αποστολή γραπτού μηνύματος

Σε αυτή την ενότητα θα δείξουμε τον τρόπο με τον οποίο καταφέραμε να προσθέτουμε επαφές τις οποίες θα τις χρησιμοποιούμε για να αποστείλουμε γραπτά μηνύματα σε περίπτωση ανάγκης. Οι εν λόγω επαφές αποθηκεύονται στην μνήμη EEPROM του μικροελεγκτή ώστε να μην χάνονται σε περίπτωση απουσίας τάσης από το σύστημα. Για να μπορέσουμε να γράψουμε στην μνήμη EEPROM χρησιμοποιούμε την βιβλιοθήκη **#include <EEPROM.h>**.

Στην συνέχεια δημιουργήσαμε μια δομή δεδομένων η οποία έχει το τηλέφωνο της κάθε επαφής, έναν χαρακτηριστικό αριθμό id και ένα flag με το οποίο θα βλέπουμε εάν η εγγραφή την οποία πραγματοποιήσαμε είναι έγκυρη. Η δομή φαίνεται παρακάτω:

```
struct Record {
char validRecord; //flag 0 or 1
char tel[10];
char id;
};
```

Για να αποθηκευτεί ένας αριθμός θα πρέπει να σταλεί με γραπτό μήνυμα και να χρησιμοποιεί το πρωτόκολλο επικοινωνίας το οποίο έχουμε προδιαγράψει ώστε να μην μπορεί να εισάγει αριθμούς ο καθένας, παρά μόνο ο διαχειριστής της συσκευής. Το πρωτόκολλο για την εισαγωγή ενός χρήστη ξεκινά με το σύμβολο # και στην συνέχεια ακολουθεί ο τηλεφωνικός αριθμός που θέλουμε να εισάγουμε.

π.χ. #6954567890

Για να μπορέσουμε να αναγνωρίσουμε το συγκεκριμένο πρωτόκολλο δημιουργήσαμε την συνάρτηση findNewUser(). Η συγκεκριμένη συνάρτηση χρησιμοποιεί την εντολή strcpn() η οποία βρίσκει εάν μέσα στο εισερχόμενο μήνυμα βρίσκετε ο χαρακτήρας # με τον οποίο ξεκινά το πρωτόκολλο και μας επιστρέφει την θέση του. Στην συνέχεια αποθηκεύουμε τους επόμενους δέκα χαρακτήρες, που είναι ο τηλεφωνικός αριθμός σε έναν buffer με το όνομα num και κλείνουμε το λεκτικό με τον χαρακτήρα \0. Παρακάτω φαίνεται η συνάρτηση:

Συνάρτηση εύρεσης νέου χρήστη :

```
int findNewUser()
{
  counter=0;
  index = strcspn(SMS,"#");
  if(index == (sizeof(SMS)-1))
  {
      return 0;
  }
  if(index != (sizeof(SMS)-1))
  for(int i=(index+1); i<(index+11); i++)
  {
    num[counter] = SMS[i];
    counter++;
  }
  num[10] = '\0';
  Serial.println(num);
  return 1;
  //clearSmsBuffer();
}
```

Στην συνέχεια και εφόσον έχουμε μήνυμα με πρωτόκολλο για νέο χρήστη πρέπει να αποθηκεύσουμε τον τηλεφωνικό αριθμό που λάβαμε στην μνήμη EEPROM. Πριν όμως αποθηκεύσουμε τον αριθμό θα πρέπει κάθε φορά να βρίσκουμε την θέση στην μνήμη EEPROM που θα πρέπει να κάνουμε εγγραφή. Αυτό το πραγματοποιούμε με την συνάρτηση fetchNextId() η οποία φαίνεται παρακάτω:

Συνάρτηση εύρεσης επόμενης θέσης στην EEPROM :

```
int fetchNextID()
{
  int adr,i;
  Record rec;
  adr = EEPROM_MAP_ADR; // έχει δηλωθεί με define στην αρχή του προγράμματος και είναι μηδέν.
  i=0;
  while(i < MAX_RECORDS){
    EEPROM.get(adr,rec);
    if(rec.validRecord != VALID_RECORD){
      return i;
    }
    adr += sizeof(Record);
    i++;
  }
  return FULL;
}
```

Βάση της προδιαγραφής που έχουμε θέσει μπορούν να εισαχθούν έως 4 επαφές. Όταν λοιπόν τις συμπληρώσουμε και προσπαθήσουμε να εισάγουμε και πέμπτη η παραπάνω συνάρτηση μας επιστρέφει τον αριθμό 10 (είναι το FULL το οποίο έχει δηλωθεί στην αρχή του προγράμματος) με τον οποίο καταλαβαίνουμε ότι δεν μπορούμε να εισάγουμε άλλες επαφές.

Η επόμενη συνάρτηση storeRecord() που δημιουργήσαμε αποθηκεύει την επαφή στην θέση της μνήμης που έχουμε βρει από την προηγούμενη συνάρτηση.

Η εγγραφή στην μνήμη EEPROM πραγματοποιείτε με την μέθοδο EEPROM.put της οποίας η σύνταξη είναι η παρακάτω :

EEPROM.put (address, data);

Συνάρτηση αποθήκευσης επαφής στην EEPROM :

```
int storeRecord (Record rec)
{
  int id,z;
  z = 0;
  id = fetchNextID();
  strcpy(rec.tel, num);
  rec.validRecord = VALID_RECORD;
  rec.id = z++;
  if(id == FULL)
  {
    return 0; //mem is full.
  }
  EEPROM.put(EEPROM_MAP_ADR+id*sizeof(Record), rec);
  return 1;
}
```

Έχοντας λοιπόν τους χρήστες μας στην μνήμη δημιουργήσαμε άλλη μια συνάρτηση με την οποία όταν την καλούμε διαβάζουμε τους τηλεφωνικούς αριθμούς που έχουμε αποθηκεύσει στην μνήμη EEPROM. Η μέθοδος η οποία χρησιμοποιείτε για να διαβάσουμε τα στοιχεία τα οποία έχουμε αποθηκεύσει στην μνήμη EEPROM είναι η EEPROM.get και η σύνταξη της φαίνεται παρακάτω :

EEPROM.get (address, data);

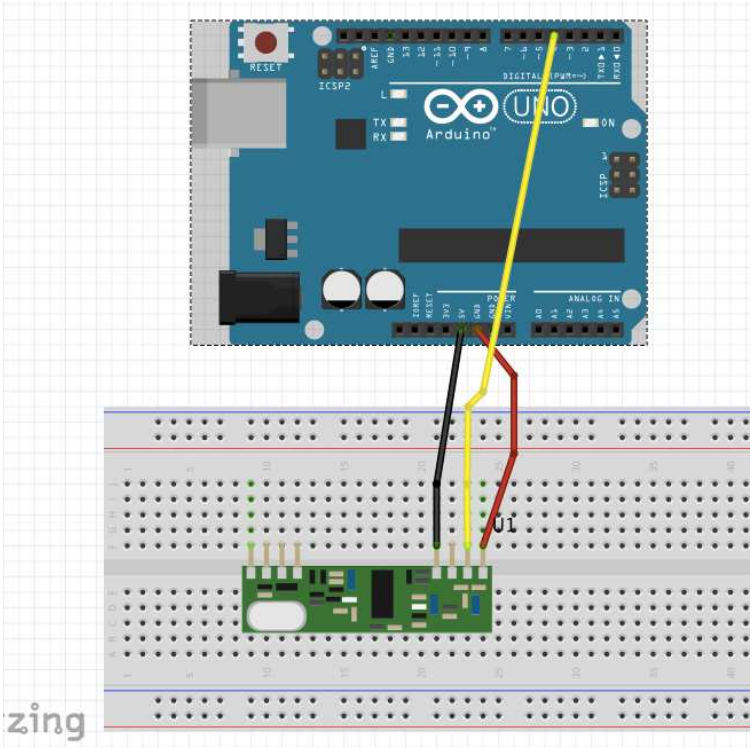
Η συνάρτηση την οποία δημιουργήσαμε ώστε να διαβάζουμε τα αποθηκευμένα φαίνεται παρακάτω :

```
void sendSMSToUsers ()
{
  int i;
  Record rec;

  for (i=0; i < MAX_RECORDS; i++ ){
    EEPROM.get(EEPROM_MAP_ADR+sizeof(rec)*i,rec );
    if (rec.validRecord  ==  VALID_RECORD){
      sendHelpSms ((char*) rec.tel);
    }
  }
}
```

Η εν λόγω συνάρτηση χρησιμοποιεί και την συνάρτηση sendHelpSMSToUsers, την οποία περιγράψαμε παραπάνω, ώστε να στέλνουμε απ' ευθείας το γραπτό μήνυμα σε κάθε έναν αποθηκευμένο αριθμό.

4.8 Διαχείριση ασύρματων εντολών RF

Σε αυτή την ενότητα θα δείξουμε πως θα εντάξουμε την λειτουργία των ασύρματων εντολών με την επέκταση RF για την οποία έχουμε κάνει αναφορά. Η επέκταση RF είναι πολύ απλή στην συνδεσμολογία της και το μόνο που χρειάζεται είναι τροφοδοσία είναι τροφοδοσία 5V και γείωση (GRND). Τα υπόλοιπα pin της επέκτασης (D0, D1, D2, D3) ενεργοποιούνται (γίνονται HIGH) όταν πατηθεί το αντίστοιχο πλήκτρο από το χειριστήριο. Παρακάτω φαίνεται και σχήμα με την συνδεσμολογία της εν λόγω επέκτασης.



Σχήμα 28. Συνδεσμολογία RF module

Το πρόγραμμα το οποίο ακολουθεί δημιουργήθηκε ώστε να δοκιμάσουμε και να επαληθεύσουμε τα όσα είπαμε παραπάνω. Η λειτουργία του προγράμματος είναι πολύ απλή και μας εκτυπώνει κάθε φορά στο serial monitor του περιβάλλοντος του arduino ποιο πλήκτρο έχει πατηθεί από το χειριστήριο.

Πρόγραμμα για έλεγχο λειτουργίας επέκτασης RF :

```
#define RF_1        4
#define RF_2        5
#define RF_3        6
#define RF_4        12
void setup ()
{
  pinMode (RF_1, INPUT);
  pinMode (RF_2, INPUT);
  pinMode (RF_3, INPUT);
  pinMode (RF_4, INPUT);
}
```

```
char scanRf1()
{
  char port = digitalRead(RF_1);
  delay (20);
  if(port) return 1;
  else return 0;
}
char scanRf2()
{
  char port = digitalRead(RF_2);
  delay (20);
  if(port) return 1;
  else return 0;
}
char scanRf3()
{
  char port = digitalRead(RF_3);
  delay (20);
  if(port) return 1;
  else return 0;
}
char scanRf4()
{
  char port = digitalRead(RF_4);
  delay(20);
  if(port) return 1;
  else return 0;
}
void loop()
{
  char rf1 = scanRf1();
  char rf2 = scanRf2();
  char rf3 = scanRf3();
  char rf4 = scanRf4();
```

```
  if(rf1 == 1)
  {
    Serial.println(F("1 pressed..."));
  }
  if(rf2 == 1)
  {
    Serial.println(F("2 pressed..."));
  }
  if(rf3 == 1)
  {
    Serial.println(F("3 pressed..."));
  }
  if(rf4 == 1)
  {
    Serial.println(F("4 pressed..."));
  }
}
```

Η χρήση του κάθε πλήκτρου στον τελικό μας κώδικα φαίνεται στον παρακάτω πίνακα:

Arduino pins

Arduino pin	Program name	Work to do
4	Button rf1	Send e-mail
5	Button rf2	Open - Close the relay
6	Button rf3	Send Sms to users
12	Button rf4	Make a phone call

Πίνακας 5. Πίνακας λειτουργία πλήκτρων

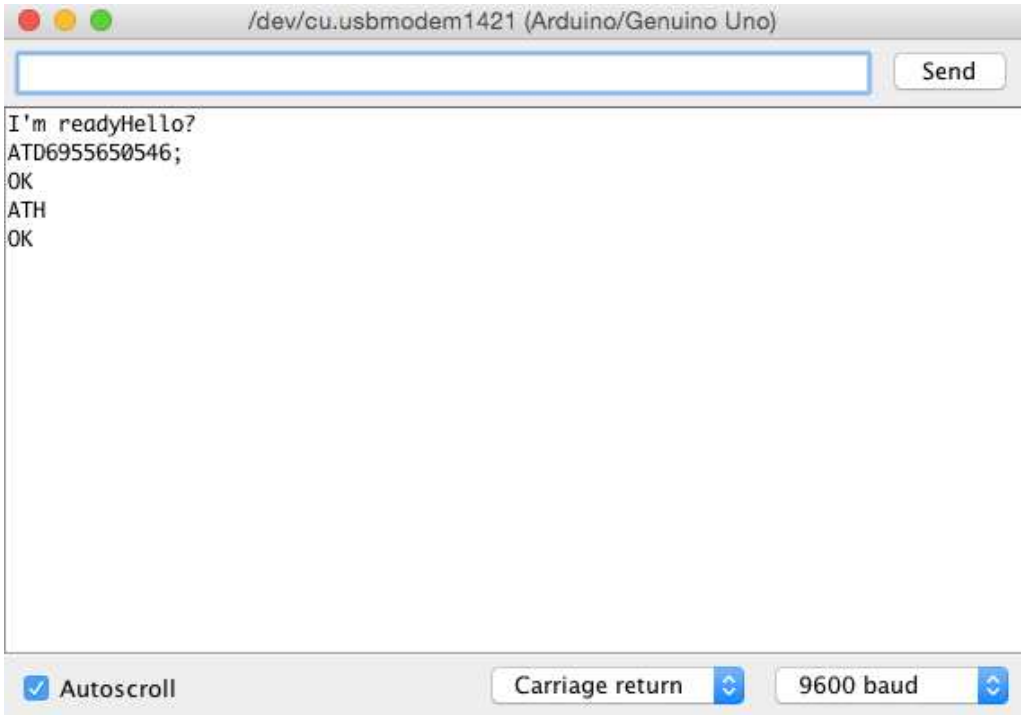
4.9 Τηλεφωνική κλήση ανάγκης

Για να πραγματοποιήσουμε μια τηλεφωνική κλήση με το SIM900 αρχικά θα πρέπει να έχουμε συνδέσει το Arduino με μια εξωτερική πηγή (τροφοδοτικό) το οποίο να έχει την ικανότητα να τροφοδοτήσει με ρεύμα έως και 2A. Ο λόγος είναι ότι το SIM900 παρουσιάζει μεγάλη κατανάλωση όταν προσπαθεί να πραγματοποιήσει την κλήση και η θύρα USB από τον υπολογιστή μας δεν είναι ικανή να παρέχει τόσο μεγάλο ρεύμα.

Στην συνέχεια και εφόσον έχουμε εξασφαλίσει την σωστή τροφοδοσία της συσκευής αποστέλλουμε από την κονσόλα μας την εντολή **ATD** της οποίας η σύνταξη φαίνεται στην παρακάτω εικόνα:

ATD Mobile Originated Call to Dial A Number	
Execution Command **ATD<n>[<mgsm>][;]**	Response This Command can be used to set up outgoing *voice, data or fax calls*. It also serves to control *supplementary services*. Note: This Command may be aborted generally by receiving an **ATH** Command or a character during execution. The aborting is not possible during some states of connection establishment such as handshaking. If no dial tone and (parameter setting **ATX2** or **ATX4**) **NO DIALTONE** If busy and (parameter setting **ATX3** or **ATX4**) **BUSY** If a connection cannot be established **NO CARRIER** If the remote station does not answer **NO ANSWER**

Σχήμα 29. Σύνταξη εντολής ATD



Σχήμα 30. Αποστολή εντολής κλήσης από το Serial monitor

Εφόσον επαληθεύσαμε την λειτουργία της κλήσης με την λειτουργία της εντολής ATD δημιουργήσαμε μια συνάρτηση με την οποία κάθε φορά που την καλούμε πραγματοποιούμε μια κλήση σε έναν προκαθορισμένο αριθμό ο οποίος είναι αποθηκευμένος στον πίνακα string_table στην μνήμη flash του μικροελεγκτή. Η συνάρτηση φαίνεται παρακάτω :

Συνάρτηση για έναρξη τηλεφωνικής κλήσης :

```
void call_for_help()
{
  sprintf (aux_str, "ATD%s;", strcpy_P
(buffer,(char*)pgm_read_word(&(string_table[0]))));
  //Serial.println(aux_str);
  if (sendATcommand (aux_str, "OK", 10000))
  {
    Serial.println(F ("calling for help..."));
  }
}
```

Για να διακόψουμε την τηλεφωνική κλήση αποστέλλουμε την AT εντολή **ATH** χωρίς καμία άλλη παράμετρο. Η αποστολή της εντολής φαίνεται παραπάνω στην εικόνα 23.

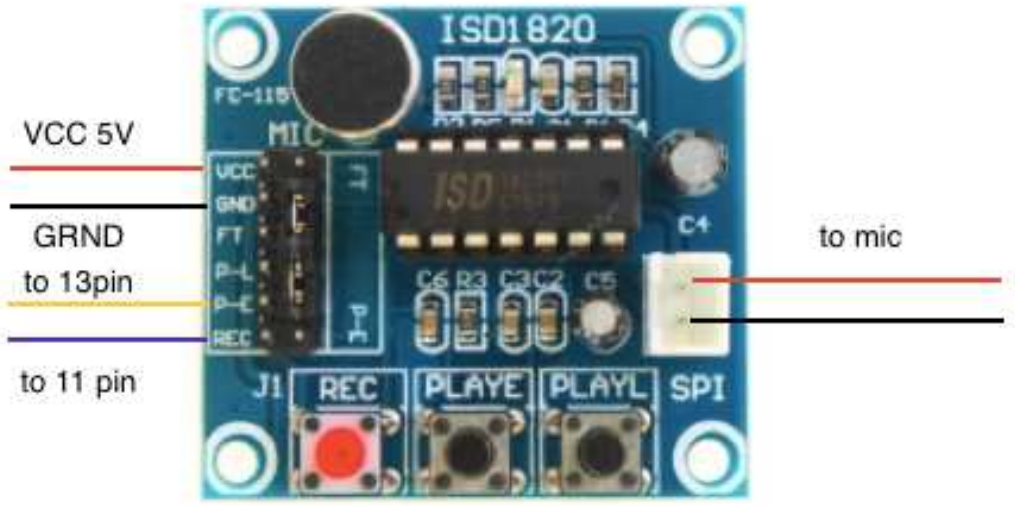
Όπως και στις προηγούμενες λειτουργίες έχουμε εντάξει αυτή την εντολή σε μια συνάρτηση ώστε όταν την καλούμε να τερματίζει την τρέχουσα τηλεφωνική κλήση. Η συνάρτηση φαίνεται παρακάτω :

Συνάρτηση για τερματισμό τηλεφωνικής κλήσης :

```
void end_call()
{
  sendATcommand("ATH", "OK", 500);
  Serial.println(F("call ended..."));
}
```

4.10 Ηχογράφηση φωνητικού μηνύματος

Σε αυτή την ενότητα και με την χρήση της επέκτασης εγγραφής φωνής θα δείξουμε πως πραγματοποιήσαμε την εγγραφή ενός φωνητικού μηνύματος και στην συνέχεια πως αυτό φωνητικό το μήνυμα το οδηγήσαμε στην είσοδο του μικροφώνου την οποία διαθέτει το SIM900. Αρχικά δείχνουμε παρακάτω την συνδεσμολογία την οποία πραγματοποιήσαμε ώστε να λειτουργήσει η επέκταση.



Σχήμα 31. Συνδεσμολογία RF επέκτασης εγγραφής φωνής

Παρατηρούμε ότι η επέκταση χρειάζεται τροφοδοσία 5V. Για να κάνουμε χειροκίνητα εγγραφή ενός μηνύματος κρατάμε πατημένο το πλήκτρο rec και λέμε το μήνυμα μας το οποίο έχει μέγιστη διάρκεια 10sec. Για να αναπαράγουμε το ηχογραφημένο μήνυμα και να το ακούσουμε θα πρέπει πρώτα να έχουμε συνδέσει στις επαφές SPI ένα ηχείο 8Ω και στην συνέχεια να πατήσουμε το πλήκτρο PLAYE. Με το πλήκτρο PLAYL γίνεται αναπαραγωγή του μηνύματος για όσο χρόνο το έχουμε πατημένο. Για να μπορέσουμε να ελέγξουμε προγραμματιστικά τα πλήκτρα REC και PLAYE συνδέουμε το pin P-E , το οποίο αντιστοιχεί στο πλήκτρο PLAY, στο pin 13 του arduino και το pin REC στο pin 11 του arduino. Παρακάτω φαίνεται ο κώδικας τον οποίο δημιουργήσαμε για να δοκιμάσουμε τις εν λόγω λειτουργίες.

Κώδικας για έλεγχο της επέκτασης εγγραφής φωνής :

```
int Rec = 11;
int Play = 13;

void setup ()
{
  pinMode (Rec, OUTPUT);
  pinMode (Play, OUTPUT);
}

void loop ()
{
  digitalWrite (Rec, HIGH);
  delay (10000);
  digitalWrite (Rec, LOW);
  delay (5000);
  digitalWrite (Play, HIGH);
  delay (100);
  digitalWrite (Play, LOW);
  delay(10000);
}
```

Για να μπορέσουμε να χρησιμοποιήσουμε κατά βούληση τις λειτουργίες αυτές δημιουργήσαμε συναρτήσεις τις οποίες θα καλούμε στον τελικό μας κώδικα. Οι συναρτήσεις φαίνονται παρακάτω :

Συνάρτηση αναπαραγωγής ηχογραφημένου μηνύματος :

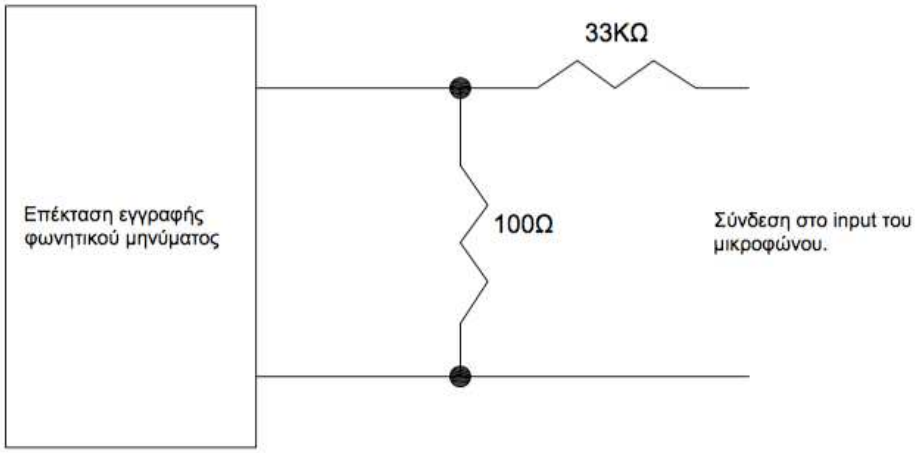
```
void play_rec ()
{
  digitalWrite (PLAY, HIGH);
  delay (100);
  digitalWrite (PLAY, LOW);
}
```

Συνάρτηση εγγραφής ηχητικού μηνύματος :

```
void rec ()
{
  digitalWrite (Rec, HIGH);
  delay (10000);
  digitalWrite (Rec, LOW);
}
```

Στην συνέχεια αντιμετωπίσαμε το πρόβλημα της οδήγησης του ηχογραφημένου μηνύματος από την έξοδο της επέκτασης στην είσοδο του μικροφώνου την οποία διαθέτει το SIM900. Ο λόγος είναι ότι η επέκταση φωνής περιμένει στην έξοδο του ηχείο 8Ω (low impedance) ενώ η είσοδος του μικροφώνου περιμένει φορτίο υψηλής εμπέδησης (high impedance). Για να λυθεί το εν λόγω πρόβλημα πρέπει να κατασκευαστεί ένα κύκλωμα matching με τελεστικό ενισχυτή (opamp) διότι δεν θα έχουμε καθόλου ήχο και υπάρχει κίνδυνος να καταστρέψουμε την είσοδο του μικροφώνου. Λόγω της πολυπλοκότητας του συγκεκριμένου κυκλώματος με τον τελεστικό ενισχυτή επιλέξαμε μια ενδιάμεση λύση η οποία μας προσφέρει προστασία στην είσοδο του μικροφώνου αλλά και καλή σχετικά ποιότητα ήχου.

Η συνδεσμολογία φαίνεται παρακάτω :



Σχήμα 32. Συνδεσμολογία επέκτασης εγγραφής φωνής με το μικρόφωνο του SIM900

4.11 Τελική εφαρμογή

Σε αυτή την ενότητα θα παρουσιάσουμε τον τελικό κώδικα της εφαρμογής με την σύνθεση όλων των παραπάνω συναρτήσεων καθώς και στιγμιότυπα (screenshot's) από το serial monitor του περιβάλλοντος προγραμματισμού.

4.11.1 Κώδικας εφαρμογής

Παρακάτω φαίνεται ο τελικός κώδικας της εφαρμογής μας ο οποίος είναι και αυτός που έχει αποθηκευτεί στο Arduino.

```
/*
 * GSM CONTROLLER
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * Title:          GSM controller
 * Version:        3.0
 * Design:         Karagounis Panagiotis
```

```
 *  Implementation:     Karagounis Panagiotis
 *
 *  Based on Dr. John Ellinas previous work
 */

#include <SoftwareSerial.h>
#include <SoftReset.h>
#include <avr/pgmspace.h>
#include <EEPROM.h>

#define GSM_pin         8     /*Pin that power up the gsm module without pressing
the key*/
#define RELAY_PIN       7     /*Enable a relay here*/
#define RF_1            4     /*when pressed sends an e-mail for help to multiple
recepients*/
#define RF_2            5     /*when pressed enable or disable the relay pin above*/
#define RF_3            6     /*when pressed sends an sms for help to multiple
recepients*/
#define RF_4            12    /*when pressed makes a call for help*/
#define VALID_RECORD    1     /*magic number anything beside 0xff*/
#define FULL            10
#define INVALID_RECORD  0xFF;
#define MAX_RECORDS     5
#define EEPROM_MAP_ADR  0
#define REC             11  /*rec pin*/
#define PLAY            13  /*pe pin*/

/*2 = TX, 3 = RX are the software serial pin that sim900 listen's to*/
SoftwareSerial GSM (2, 3);
int8_t answer;
char aux_str[30];
const char phone_number[] PROGMEM ="6955650546";     /*The number to text
or call*/
const char pin[] PROGMEM = "5262";
```

```
const char sms_text_1[] PROGMEM = "Relay is open.";
const char sms_text_2[] PROGMEM = "Relay is closed.";
const char sms_text_3[] PROGMEM = "I am Karagounis Panagiotis and i need
help at Plotarhou Blessa 5.";
const char* const string_table[] PROGMEM = {phone_number, pin, sms_text_1,
sms_text_2, sms_text_3};
const char del_sms[] PROGMEM = "AT+CMGDA=\"DEL ALL\"";
const char sim_reg[] PROGMEM = "AT+CREG?";
const char sms_mode[] PROGMEM = "AT+CMGF=1";
const char sms_mem [] PROGMEM = "AT+CPMS=\"SM\",\"SM\",\"SM\"";
const char sms_indic[] PROGMEM = "AT+CNMI=1,1,0,0,0";
const char read_sms[] PROGMEM = "AT+CMGR=1";
const char contype[] PROGMEM = "AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"";
const char apn[] PROGMEM = "AT+SAPBR=3,1,\"APN\",\"internet.vodafone.gr\"";
const char user[] PROGMEM = "AT+SAPBR=3,1,\"USER\",\"\"";
const char pwd[] PROGMEM = "AT+SAPBR=3,1,\"PWD\",\"\"";
const char gprs[] PROGMEM= "AT+SAPBR=1,1";
const char chk_ip[] PROGMEM = "AT+SAPBR=2,1";
const char ssl[] PROGMEM = "AT+EMAILSSL=1";
const char smtp[] PROGMEM = "AT+SMTPSRV=\"smtp.mail.yahoo.gr\"";
const char auth[] PROGMEM =
"AT+SMTPAUTH=1,\"panagiotis_karagounis@yahoo.gr\"";
const char from[] PROGMEM =
"AT+SMTPFROM=\"panagiotis_karagounis@yahoo.gr\"";
const char to[] PROGMEM =
"AT+SMTPRCPT=0,0,\"panagiotis_karagounis@yahoo.gr\"";
const char subj[] PROGMEM = "AT+SMTPSUB=\"TEST E-MAIL\"";
const char send_email[] PROGMEM = "AT+SMTPSEND";
const char* const sim_commands[] PROGMEM = {del_sms, sim_reg, sms_mode,
sms_mem, sms_indic, read_sms, contype, apn, user, pwd, gprs, chk_ip, ssl,
smtp, auth, from, to, subj, send_email};
char buffer[70];
char SMS[100];
int x;
```

```
char indic_flag = 0;
char del_flag = 0;
char send_flag = 0;
char arm_flag = 0;
char disarm_flag = 0;
char state;
char num[11];     /*variable to hold the phone number for the new user to be
added*/
int index;        /*var that holds the position of # for the user*/
int counter;      /*for the copy of the phone number for a new user*/
char tmp;

struct Record {
  char validRecord; /*flag 0 or 1*/
  char tel[10];
  char id;
};

Record rec;

enum{
  INIT,
  CHECK_FOR_SMS,
  CHECK_FOR_RF1,
  CHECK_FOR_RF2,
  CHECK_FOR_RF3,
  CHECK_FOR_RF4,
  CALL_FOR_HELP,
  CHECK_FOR_HUNG_UP,
  END_CALL_HELP,
  ACT_ON_SMS,
  MAN_ARM,
  MAN_DISARM
};
```

```
void setup()
{
  pinMode(RELAY_PIN,OUTPUT);
  pinMode(RF_1,INPUT);
  pinMode(RF_2,INPUT);
  pinMode(RF_3,INPUT);
  pinMode(RF_4,INPUT);
  pinMode(REC,OUTPUT);
  pinMode(PLAY,OUTPUT);
  Serial.begin(9600);
  GSM.begin(9600);
  powerOn();
  delay(1000);
  enterPin();
  setSmsIndicationOn();
  setSmsMode();
  smsMemory();
  setBearer();
}

char event = 0;

void loop()
{
  char rf1 = scanRf1();
  char rf2 = scanRf2();
  char rf3 = scanRf3();
  char rf4 = scanRf4();

  switch(state)
  {
    case INIT:
      deleteSms();
```

```
    if(del_flag == 1)     /*The del flag becomes 1 from the delete function*/
    {
      state = CHECK_FOR_SMS;
      del_flag = 0;       /*Reset the flag here*/
    }
  break;

  case CHECK_FOR_SMS:
    if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[5]))), "+CMGR:", 300))
    {
      state = ACT_ON_SMS;
    }
    else state = CHECK_FOR_RF1;
  break;

  case CHECK_FOR_RF1:
    if(rf1 == 1)
    {
      Serial.println(F("1 presed..."));
      sendEmail();
      state = CHECK_FOR_SMS;
    }
    else state = CHECK_FOR_RF2;
  break;

  case CHECK_FOR_RF2:
    if(rf2 == 1)
    {
      Serial.println(F("2 presed..."));
      if(event == 1)
      {
        state = MAN_DISARM;
      }
```

```
    else state = MAN_ARM;
   }
   else state = CHECK_FOR_RF3;
  break;

  case CHECK_FOR_RF3:
   if(rf3 == 1)
   {
    Serial.println(F("3 presed..."));
    sendSMSToUsers();
    state = CHECK_FOR_SMS;
   }
   else state = CHECK_FOR_RF4;
  break;

  case CHECK_FOR_RF4:
   if(rf4 == 1)
   {
    state = CALL_FOR_HELP;
   }
   else state = CHECK_FOR_SMS;
  break;

  case CALL_FOR_HELP:
   call_for_help();
   state = CHECK_FOR_HUNG_UP;
  break;

  case CHECK_FOR_HUNG_UP:
   rf2 = scanRf2();
   if(rf2 == 1)
   {
    Serial.println(F("ending call..."));
    state = END_CALL_HELP;
```

```
    }
    play_rec();
   break;

   case END_CALL_HELP:
    end_call();
    state = CHECK_FOR_SMS;
   break;

   case ACT_ON_SMS:
    readSms();
    /*if we find a new user we are going to store it*/
    if(findNewUser() == 1)
    {
     if(storeRecord(rec)==0)
     {
      Serial.println(F("memory is full"));
     }
     storeRecord(rec);
    }
    Serial.println(index); //i thesi pou vriskete to #
    for(int i=0;i<100;i++)
    {
    EEPROM.get(EEPROM_MAP_ADR+i,tmp);
    Serial.print((char)tmp); Serial.print(",");
    }
    if(arm_flag == 1) //it becomes 1 when we read the message and we have
open.
    {
     //here we are going to arm a relay!
     arm_flag = 0;
     send_flag = 1;
     Serial.println(F("armed..."));
     relayOn();
```

```
      }
      if(disarm_flag == 1)  //it becomes 1 when we read the message and we have
close.
      {
        disarm_flag = 0;
        send_flag = 2;
        Serial.println(F("disarmed..."));
        relayOff();
      }
      deleteSms();
      state = CHECK_FOR_SMS;
    break;

    case MAN_ARM:
      relayOn();
      rf2 = 0;
      event = 1;
      state = CHECK_FOR_SMS;
    break;

    case MAN_DISARM:
      relayOff();
      event = 0;
      state = CHECK_FOR_SMS;
    break;
  }//end of switch
}//end of main

/***********functions for storing data to eeprom*****************/

/* brings the next position in eeprom */
int fetchNextID()
{
  int adr,i;
```

```
  Record rec;

  adr = EEPROM_MAP_ADR;
  i=0;
  while(i < MAX_RECORDS){
    EEPROM.get(adr,rec);
    if(rec.validRecord != VALID_RECORD){
      return i;
    }
    adr += sizeof(Record);
    i++;
  }
  return FULL;
}

int storeRecord(Record rec)
{
  int id,z;

  z = 0;
  id = fetchNextID();

  strcpy(rec.tel, num);
  rec.validRecord = VALID_RECORD;
  rec.id = z++;
  if(id == FULL)
  {
    return 0; //mem is full.
  }
  EEPROM.put(EEPROM_MAP_ADR+id*sizeof(Record), rec);
  return 1;
}

void clearAllRecords()
```

```
{
  int i;

  for(i=0;i<sizeof(Record)*MAX_RECORDS;i++){
    EEPROM.write(i,0xff);
  }
}

void sendSMSToUsers()
{
  int i;
  Record rec;

  for(i=0;i<MAX_RECORDS;i++){
    EEPROM.get(EEPROM_MAP_ADR+sizeof(rec)*i,rec);
    if(rec.validRecord == VALID_RECORD){
      sendHelpSms((char*)rec.tel);
    }
  }
}

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

/*-------- Function's for checking and debouncing the keys input from the rf ---------*/

char scanRf1()
{
  char port = digitalRead(RF_1);
  delay(20);
  if(port) return 1;
  else return 0;
}

char scanRf2()
```

```
{
  char port = digitalRead(RF_2);
  delay(20);
  if(port) return 1;
  else return 0;
}

char scanRf3()
{
  char port = digitalRead(RF_3);
  delay(20);
  if(port) return 1;
  else return 0;
}

char scanRf4()
{
  char port = digitalRead(RF_4);
  delay(20);
  if(port) return 1;
  else return 0;
}



/*-------------------------------------------------------------*/

void powerOn()
{
   uint8_t answer=0;
   answer = sendATcommand("AT", "OK", 300);     // checks if the module is
started
   if (answer == 0)
   {
     // power on pulse
```

```
    digitalWrite(GSM_pin,HIGH);
    delay(1200);
    digitalWrite(GSM_pin,LOW);
    delay(8000);
    // waits for an answer from the module
    while(answer == 0){     // Send AT every two seconds and wait for the answer
     answer = sendATcommand("AT", "OK", 5000);
    }
  }
  Serial.println(F("Module enabled..."));
}

void enterPin()
{
  sprintf(aux_str, "AT+CPIN=%s", strcpy_P(buffer,
(char*)pgm_read_word(&(string_table[1]))));
  if(sendATcommand(aux_str, "OK", 5000))
  {
    Serial.println(F("pin entered"));
  }
  clearMemBuffer();
  Serial.println(F("Connecting to the network..."));
  if((sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[1]))), "+CREG: 0,1", 4000)))
  {
    Serial.println(F("Connected to network!"));
  }
  else Serial.println(F("NOT Connected to network!"));
  clearMemBuffer();
}

void clearMemBuffer()
{
  for(int i=0; i < sizeof(buffer); i++)
```

```
  {
    SMS[i] = (char)0;
  }
  Serial.println(F("Memory Buffer cleared."));
}

int8_t sendATcommand(char* ATcommand, char* expected_answer, unsigned int
timeout)
{
    uint8_t x=0,  answer=0;
    char response[100];
    unsigned long previous;
    memset(response, '\0', 100);    // Initialize the string
    delay(100);
    while( GSM.available() > 0) GSM.read();    // Clean the input buffer
    GSM.println(ATcommand);    // Send the AT command
    x = 0;
    previous = millis();
    // this loop waits for the answer
    do{
        // if there are data in the UART input buffer, reads it and checks for the
asnwer
        if(GSM.available() != 0){
            response[x] = GSM.read();
            x++;
            // check if the desired answer  is in the response of the module
            if (strstr(response, expected_answer) != NULL)
            {
                answer = 1;
            }
        }
        // Waits for the asnwer with time out
    }while((answer == 0) && ((millis() - previous) < timeout));
    return answer;
```

```
}

void sendHelpSms(char* tel)
{
   GSM.print("AT+CMGF=1\r");
   delay(1000);
   sprintf(aux_str,"AT+CMGS=\"%s\"\r", tel);
   Serial.println(aux_str);
   GSM.print(aux_str);
   //GSM.print("AT+CMGS=\"6957752586\"\r");
   delay(1000);
   GSM.println(strcpy_P(buffer,(char*)pgm_read_word(&(string_table[4]))));
   GSM.print("\r");
   delay(1000);
   GSM.println((char)26);
   GSM.println();
   delay(3000);
}

void setSmsMode()
{
  Serial.println(F("Setting SMS mode..."));
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[2]))), "OK", 200));    // sets the SMS
mode to text
  {
    Serial.println(F("sms mode set ok"));
    clearMemBuffer();
  }
}

void smsMemory()
{
```

```
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[3]))), "OK", 1000))   // selects the
memory
  {
    Serial.println(F("SIM memory selected"));
    //delay(2000);
    clearMemBuffer();
  }
}



void readSms()
{
  x = 0;
  answer = sendATcommand("AT+CMGR=1","+CMGR:", 300);    // reads the first
SMS
    if (answer == 1)
    {
      answer = 0;
      while(GSM.available() == 0);
      // this loop reads the data of the SMS
      do{
      // if there are data in the UART input buffer, reads it and checks for the asnwer
        if(GSM.available() > 0)
        {
          SMS[x] = GSM.read();
          x++;
          // check if the desired answer (OK) is in the response of the module
          if (strstr(SMS, "OK") != NULL)
          {
            answer = 1;
          }
          /*only one number can arm or disarm the relay from sms*/
          if(strstr(SMS,"6957752586")!=NULL)
```

```
        {
          if(strstr(SMS,"OPEN") != NULL)
          {
            arm_flag = 1;
            event = 1;
          }
          if(strstr(SMS,"CLOSE") != NULL)
          {
            disarm_flag = 1;
            event = 0;
          }
        }
      }
      }while(answer == 0);    // Waits for the asnwer with time out
      SMS[x] = '\0';
      Serial.println(SMS);
   }
   else
   {
      Serial.print(F("error "));
      Serial.println(answer, DEC);
   }

}

/*checks for the protocol sing in the message and parsing the phone numbe if
there is one*/
int findNewUser()
{
  counter=0;
  index = strcspn(SMS,"#");
  if(index == (sizeof(SMS)-1))
  {
    return 0;
```

```
  }
  if(index != (sizeof(SMS)-1))
  for(int i=(index+1); i<(index+11); i++)
  {
    num[counter] = SMS[i];
    counter++;
  }
  num[10] = '\0';
  Serial.println(num);
  return 1;
}

void clearSmsBuffer()
{
  for(int i=0; i < sizeof(SMS); i++)
  {
    SMS[i] = (char)0;
  }
  Serial.println(F("Buffer cleared."));
  Serial.println(SMS);
}

/*Here with this AT command we get an indication when a new message arrives*/
void setSmsIndicationOn()
{
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[4]))), "OK", 2000))
  {
    Serial.println(F("Sms indication is on..."));
  }
  else Serial.println(F("Sms indication did not set..."));
  clearMemBuffer();
}
```

```
void deleteSms()
{
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[0]))), "OK", 1000))    // deletes all the
messages
  {
    Serial.println(F("All sms deleted !!!"));
    del_flag = 1;
  }
  clearMemBuffer();
}

void relayOn()
{
  digitalWrite(RELAY_PIN,HIGH);
}

void relayOff()
{
  digitalWrite(RELAY_PIN,LOW);
}

/*Function to sing in to GPRS network and get an IP*/
void setBearer()
{
  sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[6]))), "OK", 500);
  sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[7]))), "OK", 500);
  sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[8]))), "OK", 500);
  sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[9]))), "OK", 500);
```

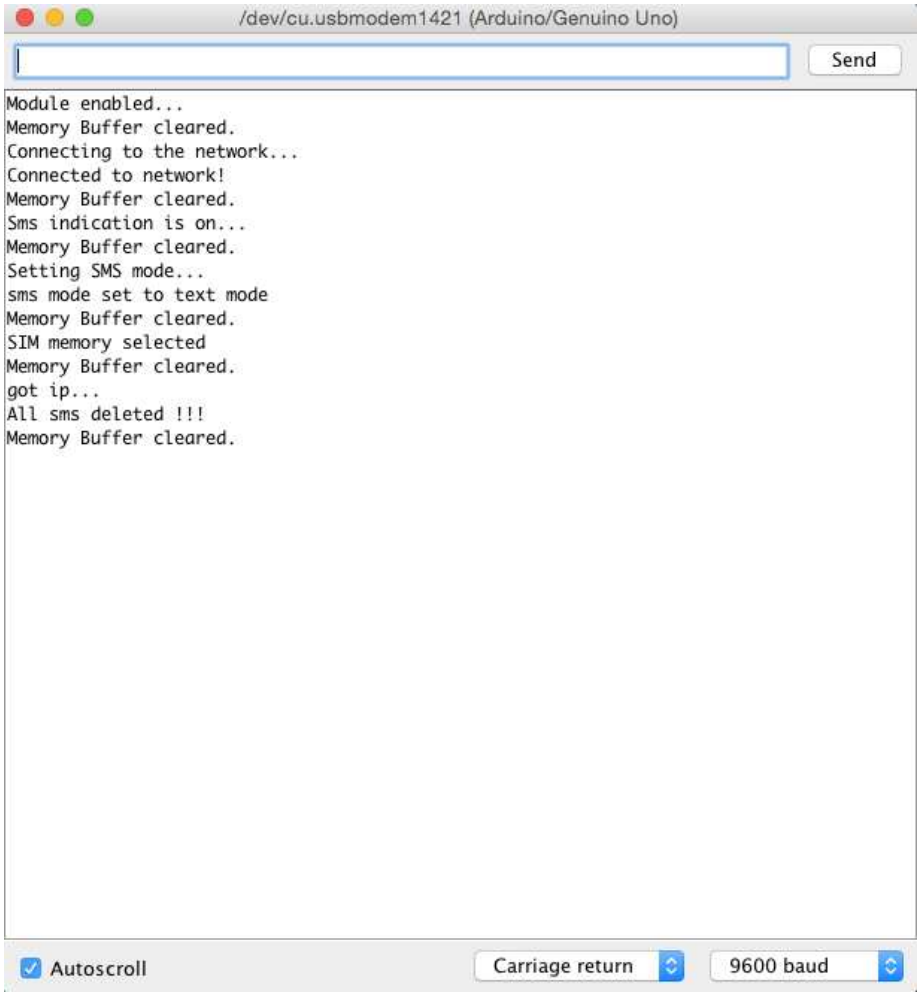
```
  sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[10]))), "OK", 500);
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[11]))), "+SAPBR: 1,1", 500))
  {
    Serial.println(F("got ip..."));
  }
  else {
    Serial.println(F("did not get ip"));
    return;
  }
}

void sendEmail()
{
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[12]))), "OK", 500))
  {
    Serial.println(F("ssl ok"));
  }
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[13]))), "OK", 500))
  {
    Serial.println(F("server ok"));
  }
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[14]))), "OK", 500))
  {
    Serial.println(F("auth ok"));
  }
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[15]))), "OK", 500))
  {
    Serial.println(F("from ok"));
```

```
  }
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[16]))), "OK", 500))
  {
    Serial.println(F("to ok"));
  }
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[17]))), "OK", 500))
  {
    Serial.println(F("subj ok"));
  }
  if(sendATcommand(strcpy_P(buffer,
(char*)pgm_read_word(&(sim_commands[18]))), "OK", 1000))
  {
    Serial.println(F("E-mail sent ok!!!"));
  }
  else Serial.println(F("E-mail didn't sent...."));
  if(sendATcommand("", "+SMTPSEND: 62", 500))
  {
    Serial.println(F("Network error..."));
  }
  if(sendATcommand("", "+SMTPSEND: 1", 30000))
  {
    Serial.println(F("E-mail sent ok!!!"));
  }
}

/*Function to make a call*/
void call_for_help()
{
  sprintf(aux_str, "ATD%s;",
strcpy_P(buffer,(char*)pgm_read_word(&(string_table[0]))));
  //Serial.println(aux_str);
  if(sendATcommand(aux_str, "OK", 10000))
```

```
  {
    Serial.println(F("calling for help..."));
  }
}

/*Function to terminate a call*/
void end_call()
{
  sendATcommand("ATH", "OK", 500);
  Serial.println(F("call ended..."));
}

/*Function to play the recorded message*/
void play_rec()
{
  digitalWrite(PLAY, HIGH);
  delay(100);
  digitalWrite(PLAY, LOW);
}


/*Function to record a message*/
void voice_rec()
{
  digitalWrite(REC, HIGH);
  delay(10000);
  digitalWrite(REC, LOW);
}
```

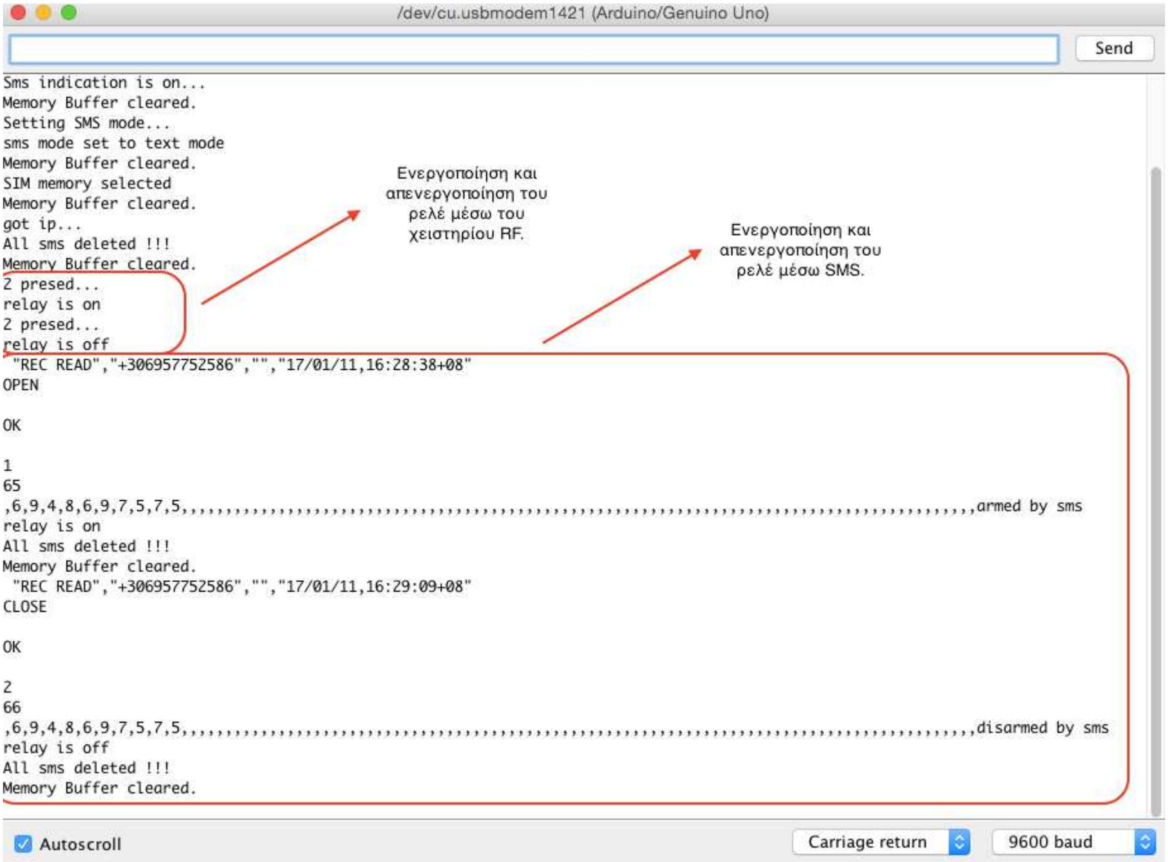
4.11.2 Στιγμιότυπα λειτουργίας της συσκευής

Παρακάτω φαίνονται σχήματα από το serial monitor του Arduino όταν η συσκευή μας είχε τεθεί σε λειτουργία και δοκιμάζαμε τις λειτουργίες της.



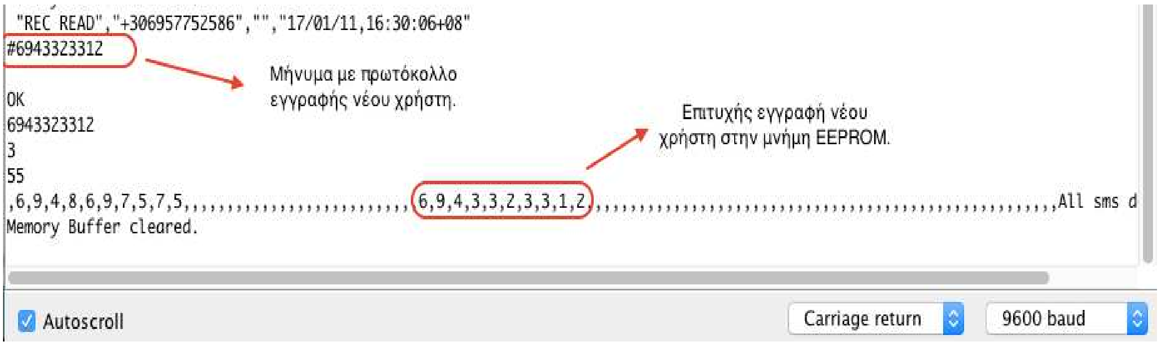
Σχήμα 33. Έναρξη λειτουργίας συσκευής

Στο παραπάνω σχήμα βλέπουμε τα μηνύματα τα οποία τυπώνονται στο serial monitor τα οποία μας ενημερώνουν για τις ενέργειες τις οποίες πραγματοποιούνται κατά την εκκίνηση της συσκευής. Στο τέλος των ενεργειών παρατηρούμε ότι όλα τα μηνύματα έχουν διαγραφεί και η συσκευή είναι έτοιμη να δεχθεί ένα νέο μήνυμα ή εντολή από το χειριστήριο RF.



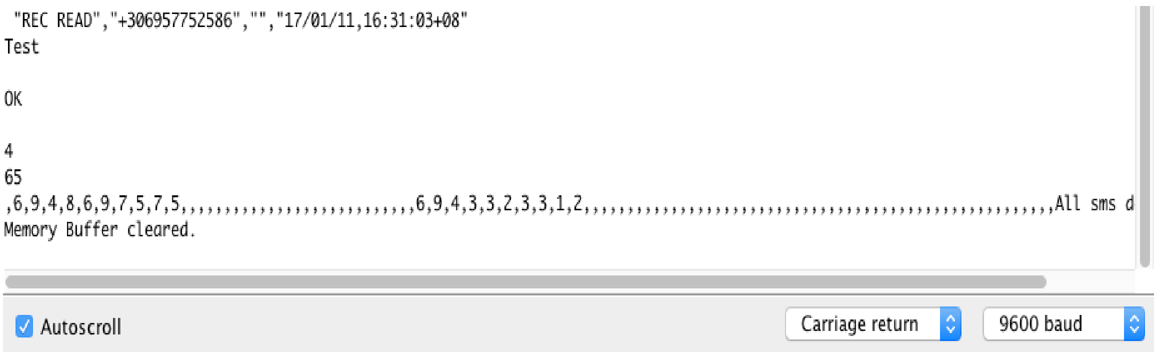
Σχήμα 34. Ενεργοποίηση - απενεργοποίηση ρελέ

Στο παραπάνω σχήμα βλέπουμε τα μηνύματα που παίρνουμε από το serial monitor κατά την ενεργοποίηση και απενεργοποίηση της επαφής που ελέγχει ένα ρελέ. Παρουσιάζονται και οι δυο τρόποι που μπορεί να γίνει η συγκεκριμένη ενέργεια και είναι είτε μέσω του χειριστηρίου RF είτε μέσω γραπτού μηνύματος με συγκεκριμένο λεκτικό μήνυμα. Να υπενθυμίσουμε πως η παραπάνω ενέργεια είναι δυνατή μόνο από τον διαχειριστή της συσκευής. Οποιοδήποτε μήνυμα το οποίο δεν προέρχεται από τον τηλεφωνικό αριθμό του διαχειριστή αγνοείτε και διαγράφετε.



Σχήμα 35. Εγγραφή νέου χρήστη στην μνήμη EEPROM

Στο παραπάνω σχήμα βλέπουμε πως η συσκευή μας λαμβάνει ένα Sms το οποίο προέρχεται από τον διαχειριστή της συσκευής και το οποίο περιέχει πρωτόκολλο για την αποθήκευση του αριθμού στην μνήμη EEPROM. Στην συνέχεια βλέπουμε την επιτυχή αποθήκευση του αριθμού στην μνήμη.

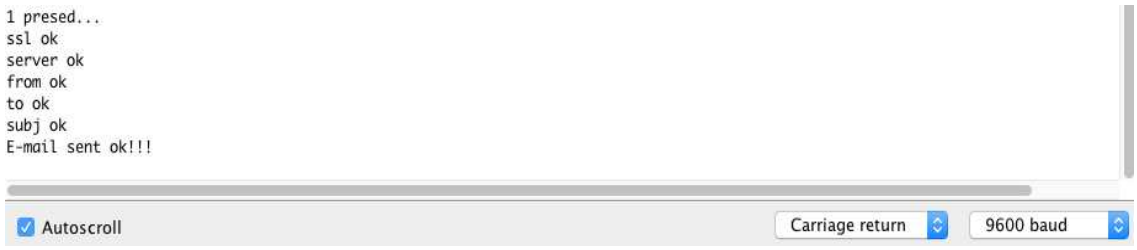


Σχήμα 36. Λήψη μηνύματος με τυχαίο κείμενο

Στο σχήμα 36 παρατηρούμε ότι συσκευή μας εάν λάβει κάποιο μήνυμα το οποίο δεν έχει προκαθοριστεί ώστε να πραγματοποιεί κάποια λειτουργία απλά αγνοείτε και στην συνέχεια διαγράφετε.

Σχήμα 37. Αποστολή SMS από την συσκευή

Στο παραπάνω σχήμα βλέπουμε την αποστολή ενός Sms από συσκευή μας προς έναν χρήστη τον οποίο έχουμε αποθηκεύσει στην μνήμη EEPROM.



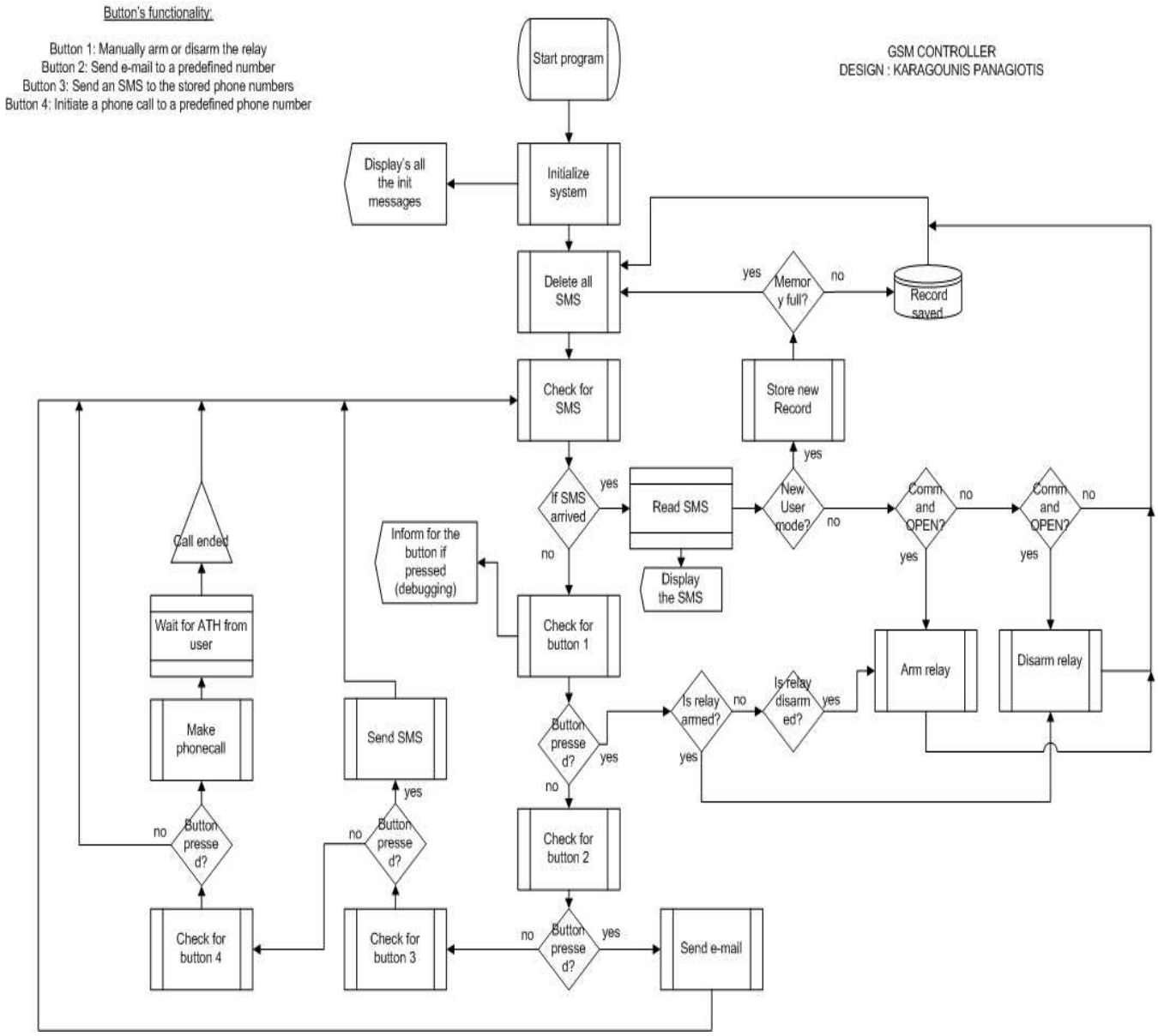
Σχήμα 38. Αποστολή e-mail από την συσκευή

Στο σχήμα 38 βλέπουμε την αποστολή ενός e-mail με το πάτημα του πλήκτρου 3 από το χειριστήριο RF.

Σχήμα 39. Τηλεφωνική κλήση από την συσκευή

Στο σχήμα 39 βλέπουμε το μήνυμα "calling for help" την ώρα που η συσκευή μας πραγματοποιεί τηλεφωνική κλήση προς έναν προκαθορισμένο αριθμό.

4.11.3 Διάγραμμα ροής της εφαρμογής



Στο παραπάνω διάγραμμα ροής φαίνονται όλα τα στάδια λειτουργίας του συστήματος.

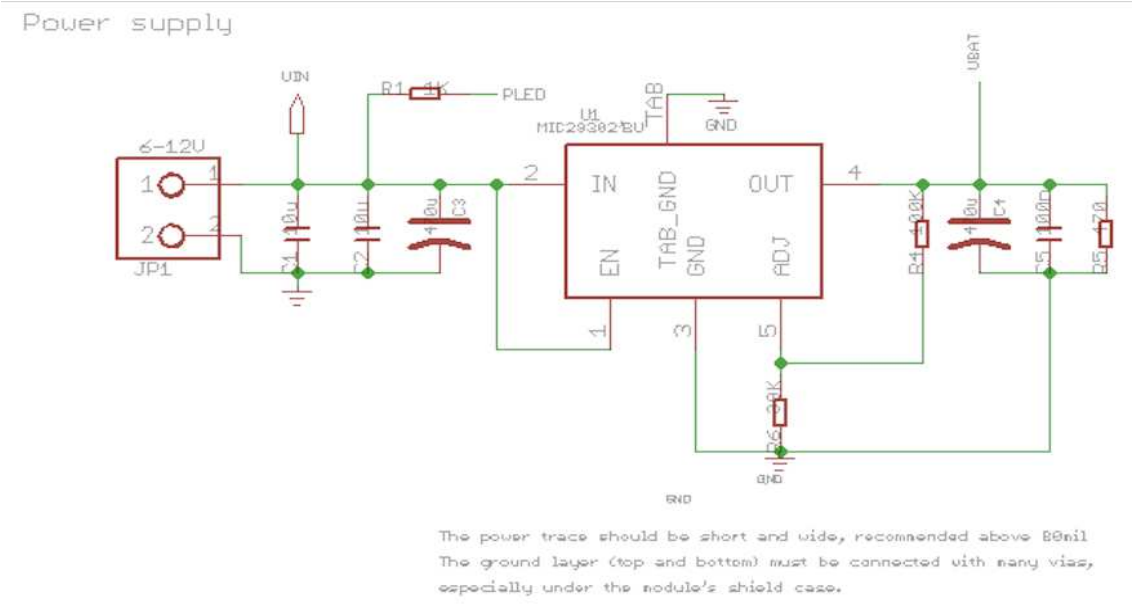
ΚΕΦΑΛΑΙΟ 5
ΑΝΑΠΤΥΞΗ ΠΡΩΤΟΤΥΠΟΥ

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιάσουμε την σχεδίαση της πρωτότυπης πλακέτας (pcb) η οποία και τελικά θα λειτουργεί στην συσκευή μας. Στόχος μας είναι η πλακέτα να είναι όσο το δυνατόν πιο μικρή σε μέγεθος ,ώστε να μπορεί το τελικό προϊόν να είναι κομψό και να μπορεί με ευκολία να τοποθετηθεί σε μια οικία ή σε οποιοδήποτε άλλο χώρο, χωρίς όμως να δημιουργούνται προβλήματα ηλεκτρομαγνητικών παρεμβολών (EMI) από την συσκευή μας σε γειτονικές συσκευές.

5.2 Σχεδίαση τροφοδοτικού.

Η σχεδίαση του τροφοδοτικού έγινε με γνώμονα τις απαιτήσεις του SIM900 στο οποίο όπως έχουμε πει και σε προηγούμενο κεφάλαιο χρειαζόμαστε τουλάχιστον 2.5 A κατά την διαδικασία της πραγματοποίησης τηλεφωνικής κλήσης. Επίσης έχουν υπολογιστεί και οι καταναλώσεις των led τα οποία χρησιμοποιούμε για ενδείξεις καλής λειτουργίας και αλλαγής καταστάσεων του προγράμματος. Παρακάτω φαίνεται το σχέδιο του τροφοδοτικού :

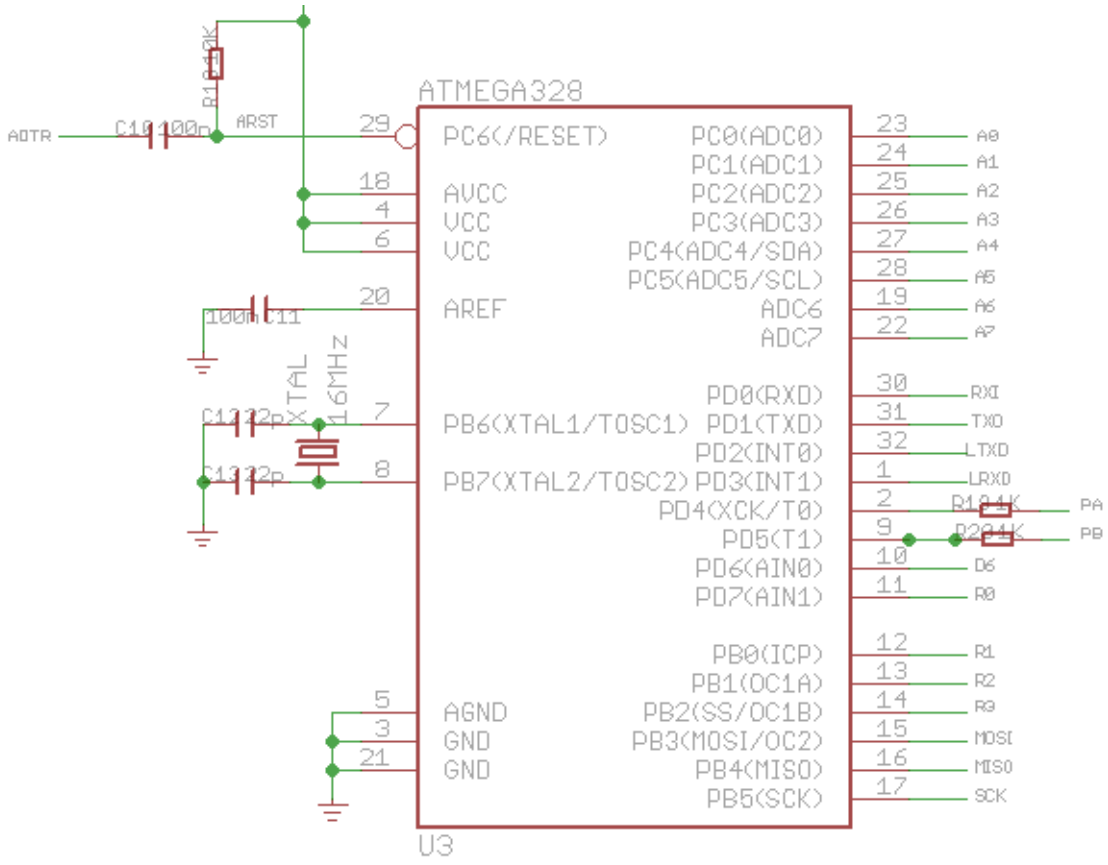


Σχήμα 40. Ηλεκτρονικό σχέδιο τροφοδοτικού

Ο ρυθμιστής τάσης ο οποίος χρησιμοποιούμε είναι ο MIC29302BU ο οποίος είναι ικανός να διαχειριστεί το μεγάλο ρεύμα το οποίο και απαιτεί η συσκευή μας. Πιο συγκεκριμένα ο συγκεκριμένος ρυθμιστής μπορεί να αντέξει έως 3 A. Η τάση στην έξοδο του ρυθμιστή είναι 5 V.

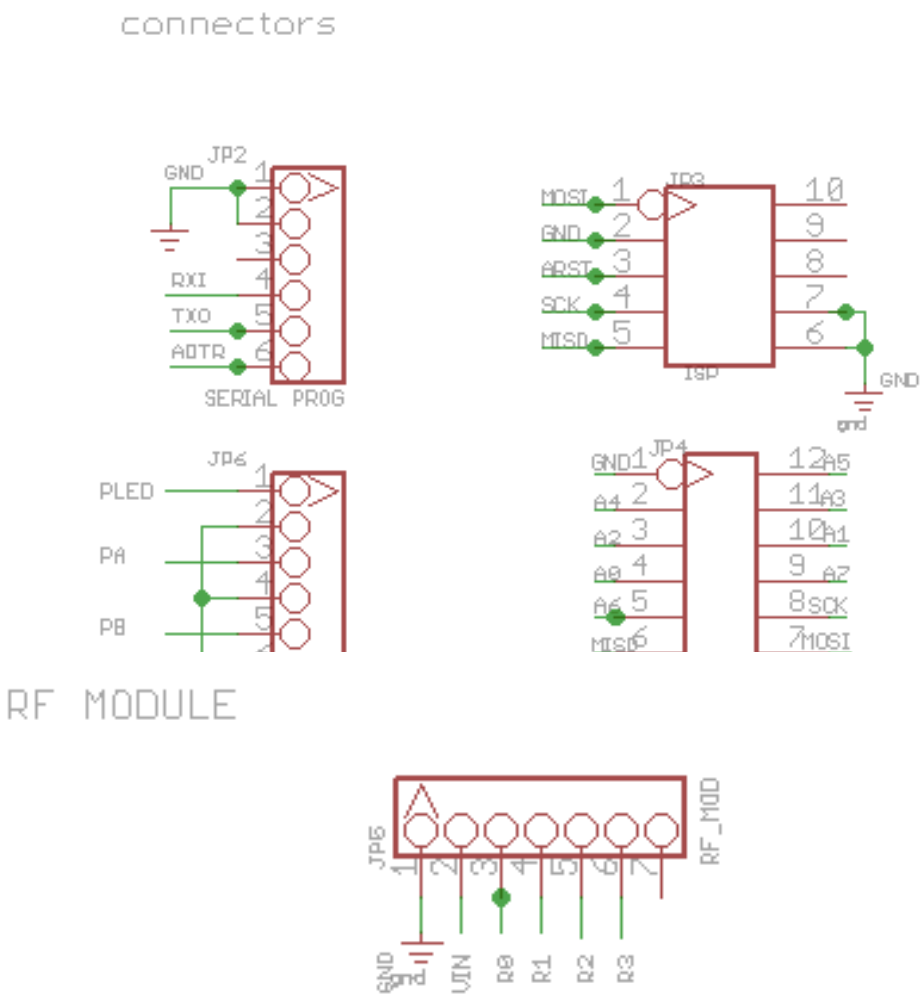
5.3 Σχεδίαση I/O Μικροελεγκτή

Ο μικροελεγκτής τον οποίο χρησιμοποιούμε είναι ο ίδιος με αυτόν που χρησιμοποιεί και το arduino και είναι ο ATMEGA328P. Παρακάτω φαίνεται η σχεδίαση της τροφοδοσίας του επεξεργαστή, καθώς και η χρήση του κάθε pin.



Σχήμα 41. Σχεδίαση της τροφοδοσίας των εισόδων/εξόδων του μικροελεγκτή

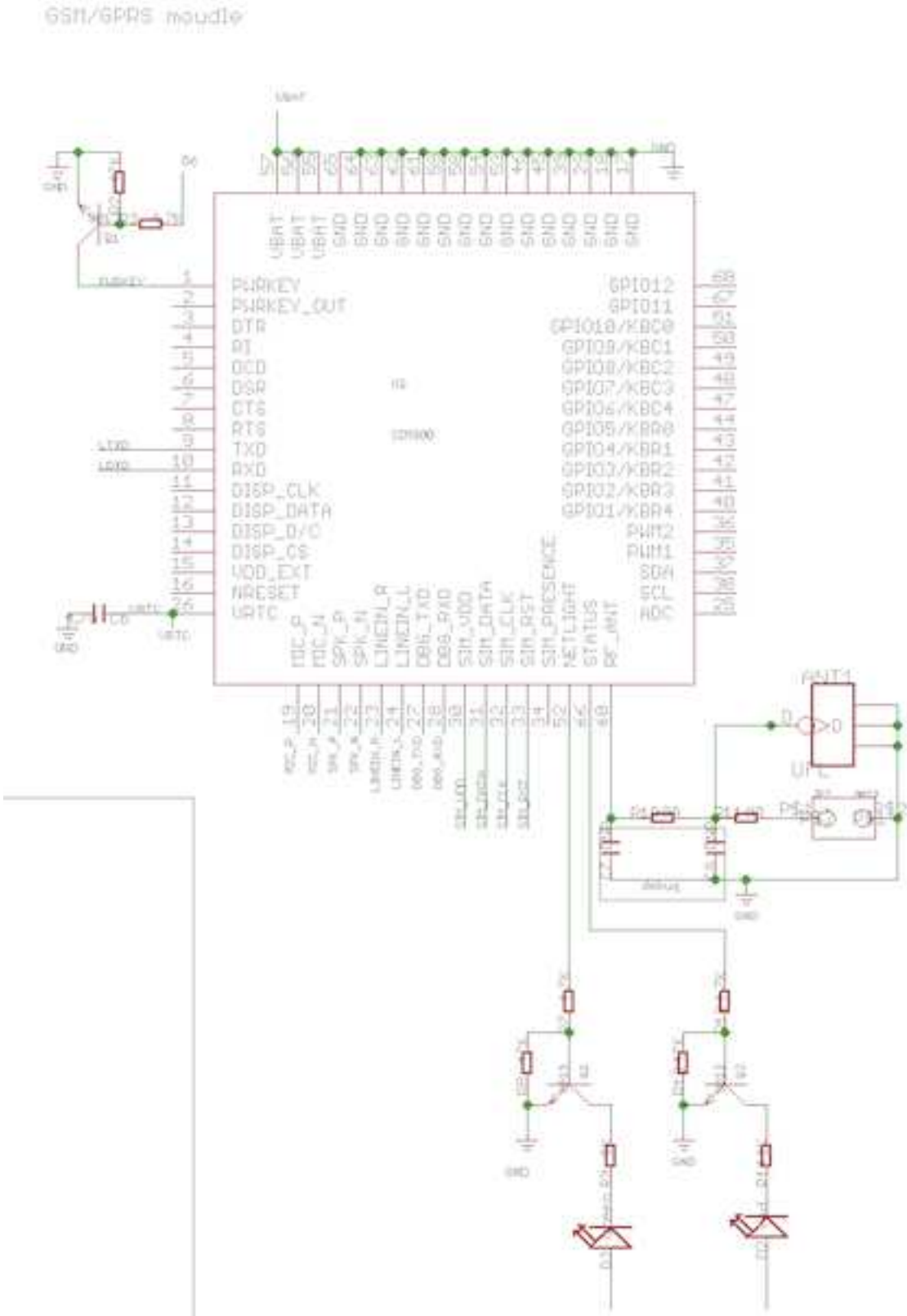
Στην συνέχεια δείχνουμε τις θύρες στις οποίες καταλήγουν τα pin του Arduino και με τα οποία μπορούμε να προγραμματίσουμε σειριακά τον μικροελεγκτή, να συνδέσουμε τα ενδεικτικά led και την επέκταση RF.



Σχήμα 42. Ακροδέκτες προγραμματισμού της συσκευής και επεκτάσεων

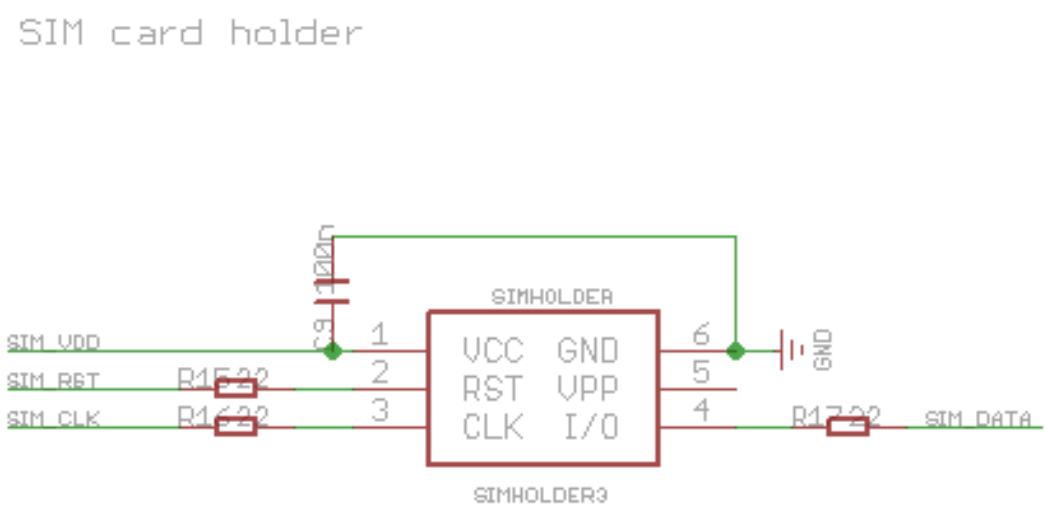
5.4 Σχεδίαση I/O SIM900

Παρακάτω βλέπουμε την σχεδίαση του SIM900 και την χρήση του κάθε pin.



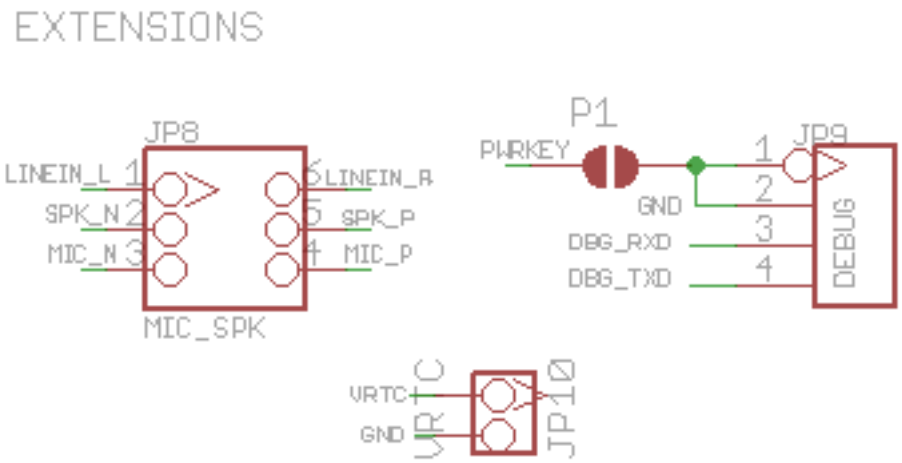
Σχήμα 43. Σχεδίαση εισόδων/εξόδων SIM900

Στο επόμενο σχήμα βλέπουμε το σχέδιο της υποδοχής της κάρτας SIM και τους ακροδέκτες που συνδέονται με το SIM900.



Σχήμα 44. Σχεδίαση υποδοχής κάρτας SIM

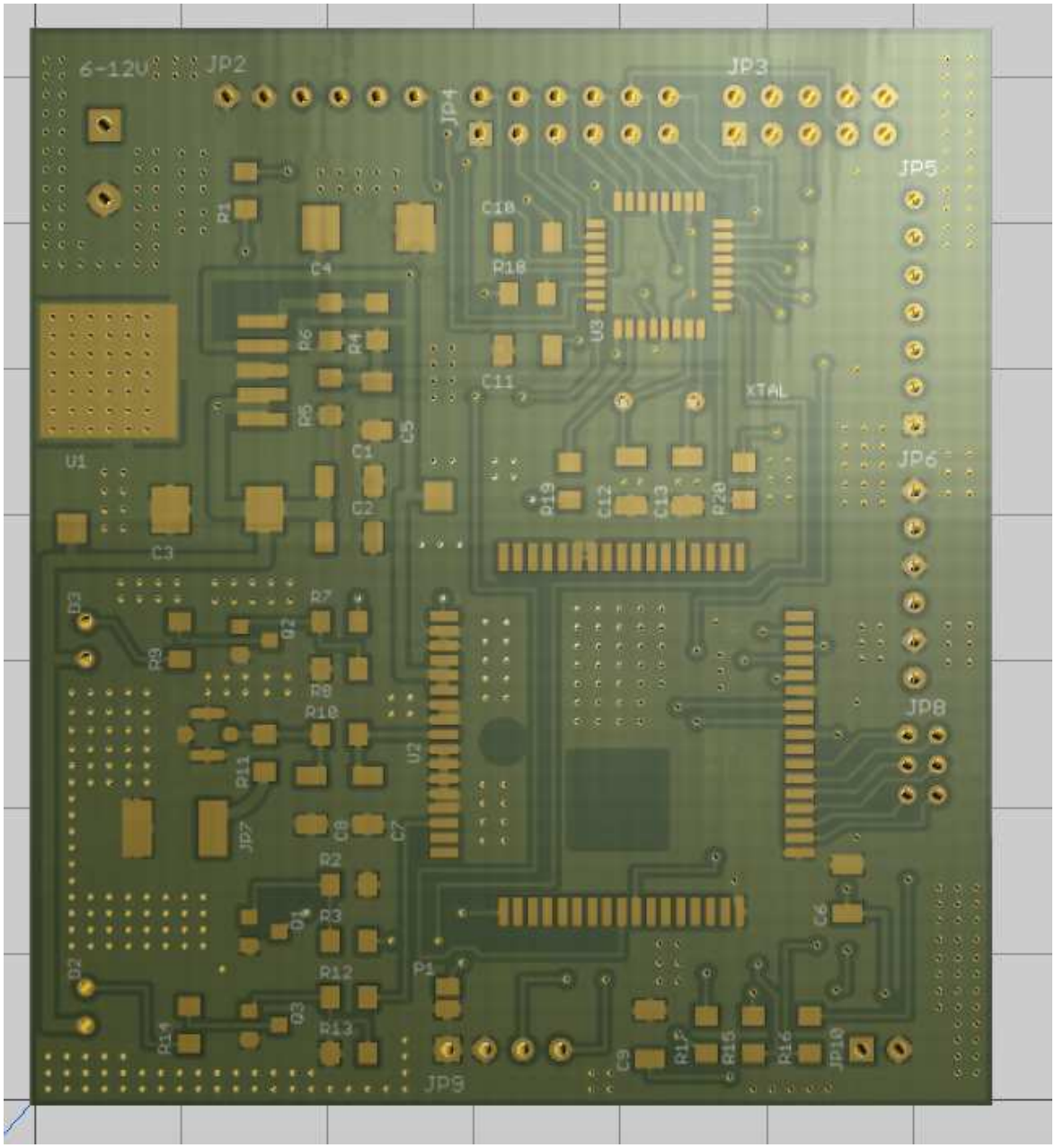
Παρακάτω φαίνονται οι υποδοχές στις οποίες μπορούμε να συνδέσουμε κάποιες επεκτάσεις στο SIM900, όπως ηχείο και μικρόφωνο.



Σχήμα 45. Σχεδίαση επεκτάσεων SIM900

5.5 Τελική όψη πλακέτας

Μετά το ηλεκτρονικό σχέδιο της συσκευής πραγματοποιήσαμε την σχεδίαση των διαδρομών της πλακέτας με γνώμονα τις διαχειριζόμενες τάσεις ανά κύκλωμα και την αποφυγή ηλεκτρομαγνητικών παρεμβολών. Η τελική μορφή της πλακέτας μας φαίνεται στο παρακάτω σχήμα.



Σχήμα 46. Τελική όψη πλακέτας

ΚΕΦΑΛΑΙΟ 6
ΣΥΜΠΕΡΑΣΜΑΤΑ - ΜΕΛΛΟΝΤΙΚΕΣ ΕΠΕΚΤΑΣΕΙΣ

6.1 Εισαγωγή

Σε αυτό το κεφάλαιο καταγράφονται τα συμπεράσματα τα οποία εξάγαμε από το σύνολο της μελέτης μας και της ανάπτυξης της εν λόγω συσκευής. Επίσης παρουσιάζουμε και τις ιδέες μας για μελλοντικές επεκτάσεις που μπορούν να πραγματοποιηθούν ώστε η συσκευή να γίνει καλύτερη και να έχει περισσότερες δυνατότητες.

6.1 Συμπεράσματα

Από την συνολική μελέτη και ανάπτυξη της εν λόγω συσκευής συμπεραίνουμε ότι το τελικό αποτέλεσμα συναντά τις προδιαγραφές τις οποίες περιγράψαμε στην αρχή της διπλωματικής εργασίας. Η συσκευή μας πραγματοποιεί με επιτυχία όλες τις λειτουργίες επικοινωνίας χρησιμοποιώντας το δίκτυο κινητών επικοινωνιών αλλά και τοπικά με την χρήση του ασύρματου δέκτη RF.

Τα προβλήματα τα οποία συναντήσαμε κατά την ανάπτυξη της συσκευής είναι κυρίως προβλήματα ελλιπής τεκμηρίωσης και προδιαγραφών των αναπτυξιακών πλακετών που χρησιμοποιήσαμε. Επίσης κατά την δοκιμή του προϊόντος διαπιστώσαμε πως για να λειτουργεί αποδοτικά η συσκευή θα πρέπει να υπάρχει πολύ καλό σήμα του παρόχου κινητής τηλεφωνίας που χρησιμοποιούμε αλλιώς η συσκευή μας παρουσιάζει συνεχείς συνδέσεις και αποσυνδέσεις με αποτέλεσμα να γίνεται αναξιόπιστη.

6.2 Μελλοντικές επεκτάσεις

Λόγω του παραπάνω προβλήματος αναφορικά με την ποιότητα του σήματος εντός μιας οικίας θα μπορούσε να χρησιμοποιηθεί ως κύριο μέσο επικοινωνίας και ελέγχου των συσκευών το Internet ,με χρήση πλακέτας Wi-Fi, και ως δευτερεύων μέσο το GSM σε περίπτωση που δεν θα λειτουργούσε το πρώτο.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Basic Radio Theory (JAA ATPL theory)

[2] Σημειώσεις μαθήματος Ευρυζωνικών δικτύων, Νικόλαος Δ. Τσελίκας

[3] Σχεδιασμός και υλοποίηση απομακρυσμένου συστήματος αυτομάτου ελέγχου, Χρήστος Κατσάνος (Διπλωματική εργασία)

[4] Ανάπτυξη συστήματος συναγερμού με τεχνολογία GSM, Χρίστος Μακρομανωλάκης (Πτυχιακή εργασία)

[5] Sauter, Martin (21 Nov 2013). "The GSM Logo: The Mystery of the 4 Dots Solved". Retrieved 23 Nov 2013. [...] here's what [Yngve Zetterstrom, rapporteur of the Maketing and Planning (MP) group of the MoU (Memorandum of Understanding group, later to become the GSM Association (GSMA)) in 1989] had to say to solve the mystery: '[The dots symbolize] three [clients] in the home network and one roaming client.' There you go, an answer from the prime source!"

[6] Redl, Siegmund M.; Weber, Matthias K.; Oliphant, Malcolm W (February 1995). An Introduction to GSM. Artech House.

[7] Mouly, Michel; Pautet, Marie-Bernardette (June 2002). The GSM System for Mobile Communications. Telecom Publishing

[8] Atmel-8271J-AVR- ATmega-Datasheet_11/2015

[9] SIM900 AT command manual V 1.03

[10] SIM900 e-mail AT command manual V 1.02

[11] GPRS/GSM Tinyshine shield manual V 1.0

[12] ISD1820 Voice recorder module user manual

[13] Bernhard Walke, Wolf Mende, Georgios Hatziliadis: “CELLPAC: A packet radio protocol applied to the cellular GSM mobile radio network”, Proceedings of 41st IEEE Vehicular Technology Conference, May 1991, 408-413.

[14] Peter Decker, Bernhard Walke: “A General Packet Radio Service proposed for GSM”, ETSI SMG Workshop “GSM in a Future Competitive Environment”, Helsinki, Finland, Oct. 13, 1993, pp. 1-20