ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ

Η υπηρεσία αποστολής SMS μηνυμάτων είναι πολύ δημοφιλής και εύχρηστη σε ιδιώτες και επαγγελματίες. Στόχος της εργασίας είναι η δημιουργία μιας εφαρμογής αποστολής και λήψης τέτοιων μηνυμάτων μέσω Η/Υ καθώς και η αποθήκευση των ληφθέντων μηνυμάτων σε μια βάση δεδομένων, με την βοήθεια ενός GSM/GPRS modem ή ενός κινητού τηλεφώνου. Στα επόμενα κεφάλαια δίνονται λεπτομερείς πληροφορίες σχετικά με την αποστολή και λήψη μηνυμάτων μέσω Η/Υ, τα GSM/GPRS modem, τις AT εντολές και το Hyper Terminal. Γίνεται ανάπτυξη της εφαρμογής και του κώδικα καθώς και οδηγίες χρήσεως.

ΚΕΦΑΛΑΙΟ 1

1.1 Τι είναι το SMS(Sort Message Service);

SMS σημαίνει Sort Message Service δηλαδή Υπηρεσία Σύντομων Μηνυμάτων. Είναι μια τεχνολογία που επιτρέπει την αποστολή και λήψη μηνυμάτων μεταξύ κινητών τηλεφώνων. Το sms εμφανίστηκε στην Ευρώπη το 1992. Περιλήφθηκε αρχικά στα GSM(Global System for Mobile Communications- Παγκόσμιο Σύστημα Κινητών Επικοινωνιών) πρότυπα. Αργότερα μεταφέρθηκε σε ασύρματες τεχνολογίες, όπως CDMA και TDMA. Τα πρότυπα GSM και SMS αναπτύχθηκαν αρχικά από το ETSI(European Telecommunications Standards Institute- Ευρωπαϊκό Ινστιτούτο Τηλεπικοινωνιακών Προτύπων). Τώρα το 3GPP (έργο κοινοπραξίας τρίτης γενιάς) είναι υπεύθυνο για την ανάπτυξη και διατήρηση των προτύπων GSM και SMS.

Όπως προτείνεται και στην ονομασία «Υπηρεσία Σύντομων Μηνυμάτων» τα δεδομένα που κρατάει ένα SMS μήνυμα είναι πολύ περιορισμένα. Ένα SMS μπορεί να περιέχει το μέγιστο 140 bytes(1120 bits) δεδομένων, έτσι ένα SMS μπορεί να περιέχει πάνω από:

- 160 χαρακτήρες εάν χρησιμοποιείται 7-bit κωδικοποίηση χαρακτήρων (η 7-bit κωδικοποίηση χαρακτήρων είναι κατάλληλη για την κωδικοποίηση λατινικών χαρακτήρων όπως τα Αγγλικά αλφάβητα.)

- 70 χαρακτήρες εάν χρησιμοποιείται 16-bit Unicode UC22 κωδικοποίηση χαρακτήρων (SMS μηνύματα κειμένου που περιέχουν μη λατινικούς χαρακτήρες, όπως οι Κινέζικοι χαρακτήρες, θα πρέπει να χρησιμοποιούν 16-bit κωδικοποίηση χαρακτήρων.)

Τα SMS μηνύματα κειμένου υποστηρίζουν γλώσσες διεθνώς. Λειτουργεί άψογα με όλες τις γλώσσες που υποστηρίζονται από το Unicode, όπως Αραβικά, Κινέζικα, Ιαπωνικά και Κορεατικά. Εκτός από το κείμενο, τα μηνύματα SMS μπορούν επίσης να μεταφέρουν δυαδικά δεδομένα. Είναι δυνατή η αποστολή ringtones, εικόνες,

λογότυπα χειριστή, wallpapers, animations, επαγγελματικές κάρτες (π.χ. vCard) και WAP συνθέσεις σε ένα κινητό τηλέφωνο με SMS μηνύματα.

Ένα σημαντικό πλεονέκτημα του SMS είναι ότι υποστηρίζεται 100% από GSM κινητά τηλέφωνα. Σχεδόν όλες οι συνδρομές που παρέχουν ασύρματο σήμα περιλαμβάνουν φθηνή υπηρεσία SMS μηνυμάτων. Σε αντίθεση με το SMS, κινητές τεχνολογίες όπως WAP και mobile Java δεν υποστηρίζονται από πολλά παλιά μοντέλα κινητών τηλεφώνων.

1.1.2 Συνεχόμενα SMS μηνύματα/Μακρά SMS μηνύματα

Ένα μειονέκτημα της τεχνολογίας SMS είναι ότι ένα SMS μήνυμα μπορεί να μεταφέρει μια πολύ περιορισμένη ποσότητα δεδομένων. Για να ξεπεραστεί αυτό το μειονέκτημα, αναπτύχθηκε μια επέκταση που ονομάζεται συνεχόμενα SMS (επίσης γνωστή ως long SMS).Ένα συνεχόμενο SMS μήνυμα κειμένου μπορεί να περιέχει περισσότερους από 160 Αγγλικούς χαρακτήρες. Το συνεχόμενο SMS δουλεύει κάπως έτσι: Το κινητό τηλέφωνο του αποστολέα σπάει το μακρύ μήνυμα σε μικρότερα μέρη και στέλνει το καθένα σ' ένα μόνο SMS μήνυμα. Όταν αυτά τα μηνύματα φτάσουν στον προορισμό, το κινητό τηλέφωνο-αποδέκτης θα τα ενώσει πίσω σ' ένα μακρύ μήνυμα.

Το μειονέκτημα του συνεχόμενου SMS είναι ότι είναι λιγότερο ευρέως υποστηριζόμενο από το SMS στις ασύρματες συσκευές.

1.2 Τι είναι το Κέντρο SMS(SMSC);

Ένα κέντρο SMS (SMSC) είναι υπεύθυνος για τη διαχείριση των SMS λειτουργιών ενός ασύρματου δικτύου. Όταν στέλνεται ένα SMS μήνυμα από ένα κινητό τηλέφωνο, θα φθάσει πρώτα ένα κέντρο SMS. Το κέντρο SMS, στη συνέχεια, προωθεί το SMS μήνυμα στον προορισμό. Το μήνυμα SMS μπορεί να χρειαστεί να περάσει μέσα από περισσότερες από μία οντότητα του δικτύου (π.χ. SMSC και SMS gateway) πριν φθάσει στον προορισμό. Το κύριο καθήκον του SMSC είναι να δρομολογήσει τα

μηνύματα SMS και να ρυθμίζει τη διαδικασία. Εάν ο παραλήπτης δεν είναι διαθέσιμος (για παράδειγμα, όταν το κινητό τηλέφωνο είναι απενεργοποιημένο), το SMSC θα αποθηκεύσει το μήνυμα SMS. Θα προωθήσει το μήνυμα SMS όταν ο αποδέκτης είναι διαθέσιμος.

Πολύ συχνά ένα SMSC αφιερώνεται στο να χειρίζεται την κίνηση των SMS σ' ένα ασύρματο δίκτυο. Ο χειριστής του δικτύου διαχειρίζεται συνήθως το/τα δικό του SMSC και το/τα τοποθετεί μέσα στο δικό του σύστημα ασύρματου δικτύου. Ωστόσο, είναι πιθανόν ένας χειριστής δικτύου να χρησιμοποιήσει ένα SMSC τρίτου μέρους που είναι τοποθετημένο έξω από το σύστημα ασύρματου δικτύου.

Πρέπει να γνωρίζουμε τη διεύθυνση του SMSC του ασύρματου δικτύου, ώστε να χρησιμοποιήσουμε τα μηνύματα SMS με το κινητό μας τηλέφωνο. Συνήθως μια διεύθυνση SMSC είναι ένα συνηθισμένος αριθμός τηλεφώνου σε διεθνή μορφή. Ένα κινητό τηλέφωνο θα πρέπει να έχει μια επιλογή μενού που μπορεί να χρησιμοποιηθεί για να διαμορφωθεί η διεύθυνση του SMSC. Κανονικά, η διεύθυνση SMSC είναι προκαθορισμένη στην κάρτα SIM από τον φορέα του ασύρματου δικτύου, το οποίο σημαίνει ότι δεν χρειάζεται να κάνουμε οποιεσδήποτε αλλαγές σε αυτό.

1.3 Εισαγωγή στα GSM/GPRS Ασύρματα Μόντεμ

1.3.1 Τι είναι το GSM Μόντεμ;

Ένα GSM μόντεμ είναι ένα ασύρματο μόντεμ που δουλεύει με GSM ασύρματο δίκτυο. Ένα ασύρματο μόντεμ συμπεριφέρεται όπως ένα dial-up μόντεμ. Η κυρίως διαφορά μεταξύ τους είναι ότι ένα dial-up μόντεμ στέλνει και λαμβάνει δεδομένα μέσω σταθερής τηλεφωνικής γραμμής, ενώ ένα ασύρματο μόντεμ στέλνει και λαμβάνει δεδομένα μέσω ραδιοκυμάτων.

Ένα GSM μόντεμ μπορεί να είναι μια εξωτερική συσκευή ή μια Κάρτα PC/PCMIA Κάρτα. Τυπικά ένα GSM μόντεμ συνδέεται στον υπολογιστή μέσω ενός σειριακού καλωδίου ή USB καλωδίου. Ένα GSM μόντεμ σε μορφή Κάρτα PC/PCMIA Κάρτα είναι σχεδιασμένο για χρήση με ένα φορητό υπολογιστή. Θα πρέπει να εισαχθεί σε ένα από τους PC Card / PCMCIA Card υποδοχείς του φορητού υπολογιστή. Όπως

το GSM κινητό τηλέφωνο, έτσι και το GSM μόντεμ χρειάζεται μια SIM κάρτα από κάποιο ασύρματο φορέα για να λειτουργήσει. Οι υπολογιστές χρησιμοποιούν AT εντολές για να ελέγχουν τα μόντεμ. Τα GSM μόντεμ και τα dial-up μόντεμ υποστηρίζουν ένα κοινό σετ καθορισμένων AT εντολών. Μπορούμε να χρησιμοποιήσουμε ένα GSM μόντεμ ακριβώς όπως ένα dial-up μόντεμ.

Εκτός από τις βασικές AT εντολές, τα GSM μόντεμ υποστηρίζουν ένα εκτεταμένο σετ AT εντολών. Αυτές οι εκτεταμένες AT εντολές ορίζονται στα πρότυπα GSM. Με τις εκτεταμένες AT εντολές, μπορούμε να κάνουμε πράγματα όπως:

- Ανάγνωση, εγγραφή και διαγραφή SMS μηνυμάτων
- Αποστολή SMS μηνυμάτων
- Εμφάνιση της ισχύς του σήματος
- Εμφάνιση της κατάστασης της φόρτισης και το επίπεδο φόρτισης της μπαταρίας
- Ανάγνωση, εγγραφή και αναζήτηση επαφών του καταλόγου

Ο αριθμός των μηνυμάτων SMS που μπορούν να επεξεργαστούν από ένα GSM μόντεμ ανά λεπτό είναι πολύ χαμηλός - μόνο περίπου έξι έως δέκα SMS μηνύματα ανά λεπτό.

1.3.2 Τι είναι το GPRS μόντεμ

Το GPRS μόντεμ είναι ένα GSM μόντεμ που επιπλέον υποστηρίζει την τεχνολογία GPRS για την μεταφορά δεδομένων. GPRS είναι τα αρχικά από General Packet Radio Service. Είναι μια τεχνολογία μεταγωγής πακέτου που είναι μια επέκταση της GSM (η GSM είναι μια τεχνολογία μεταγωγής κυκλώματος). Ένα βασικό πλεονέκτημα της GPRS από την GSM είναι ότι η GPRS έχει μεγαλύτερη ταχύτητα μεταφοράς δεδομένων.

Η GPRS μπορεί να θεωρηθεί ως φορέας της SMS. Αν χρησιμοποιηθεί η SMS με την GPRS, μπορεί να επιτευχθεί ταχύτητα μεταφοράς SMS των 30 μηνυμάτων περίπου. Αυτό είναι πολύ ταχύτερο από το να χρησιμοποιηθεί η συνηθισμένη SMS με την GSM, της οποίας η ταχύτητα μεταφοράς SMS είναι περίπου της τάξης 6 με 10

μηνύματα το λεπτό. Ένα GPRS μόντεμ χρειάζεται για να στείλουμε και να λάβουμε SMS με την GPRS. Σημειώστε ότι ορισμένοι ασύρματοι φορείς δεν υποστηρίζουν την

αποστολή και λήψη μηνυμάτων SMS μέσω GPRS. Εάν πρέπει να στείλετε ή να λάβετε μηνύματα MMS, ένα μόντεμ GPRS είναι συνήθως απαραίτητο.

1.3.3 Σύγκριση κινητού τηλεφώνου με ένα GSM/GPRS Μόντεμ

Σε γενικές γραμμές, ένα GSM / GPRS μόντεμ συνιστάται για χρήση με υπολογιστή για αποστολή και λήψη μηνυμάτων. Αυτό οφείλεται στο γεγονός ότι ορισμένα κινητά τηλέφωνα έχουν ορισμένους περιορισμούς σε σύγκριση με τα GSM / GPRS μόντεμ. Ορισμένοι από τους περιορισμούς περιγράφονται παρακάτω:

- Μερικά μοντέλα κινητών τηλεφώνων (παράδειγμα: Ericsson R380) δεν μπορούν να χρησιμοποιηθούν με υπολογιστή για λήψη συνεχόμενων SMS μηνυμάτων.

  Ποια είναι η αιτία του προβλήματος;

  Όταν το κινητό τηλέφωνο λαμβάνει τα μηνύματα SMS που αποτελούν μέρη ενός συνεχόμενου SMS μηνύματος, θα τα ενώσει σε ένα μήνυμα αυτόματα. Η σωστή συμπεριφορά πρέπει να είναι: όταν το κινητό τηλέφωνο λαμβάνει τα SMS μηνύματα που αποτελούν μέρη ενός συνεχόμενου SMS μηνύματος, τα προωθεί στον υπολογιστή χωρίς να τα ενώσει.

- Πολλά μοντέλα κινητών τηλεφώνων δεν μπορούν να χρησιμοποιηθούν με υπολογιστή για την λήψη MMS μηνυμάτων. Διότι όταν λαμβάνουν μια ειδοποίηση MMS την χειρίζονται αυτόματα αντί να την προωθούν στον υπολογιστή.

- Ένα κινητό τηλέφωνο μπορεί να μην υποστηρίζει μερικές AT εντολές, παραμέτρους εντολών και τιμές παραμέτρων. Για παράδειγμα, μερικά κινητά

τηλέφωνα δεν υποστηρίζουν την αποστολή και λήψη μηνυμάτων SMS σε κατάσταση κειμένου. Έτσι η AT εντολή «AT+CMGF=1» (δίνει εντολή στο κινητό τηλέφωνο για να χρησιμοποιήσει τη λειτουργία κειμένου) θα επιστρέψει

μήνυμα λάθους. Συνήθως τα GSM / GPRS μόντεμ υποστηρίζουν μια πιο πλήρη σειρά AT εντολών από τα κινητά τηλέφωνα.

- Οι περισσότερες εφαρμογές ανταλλαγής SMS μηνυμάτων πρέπει να είναι διαθέσιμες 24 ώρες την ημέρα. (Για παράδειγμα, μια εφαρμογή ανταλλαγής μηνυμάτων SMS που παρέχει υπηρεσία για “κατέβασμα” ήχων κλήσεων θα πρέπει να τρέχει όλη μέρα έτσι ώστε ο χρήστης να μπορεί να κατεβάσει ήχους κλήσεις οποιαδήποτε στιγμή θελήσει). Εάν οι εφαρμογές ανταλλαγής μηνυμάτων SMS χρησιμοποιήσουν κινητά τηλεφώνα για αποστολή και λήψη μηνυμάτων SMS, τα κινητά τηλέφωνα θα πρέπει να είναι ενεργοποιημένα όλη την ώρα. Ωστόσο, μερικά μοντέλα κινητών τηλεφώνων δεν μπορούν να λειτουργήσουν χωρίς μπαταρία ακόμα κι αν είναι συνδεδεμένα στον φορτιστή, το οποίο σημαίνει ότι η μπαταρία θα πρέπει να φορτίζεται όλη μέρα.

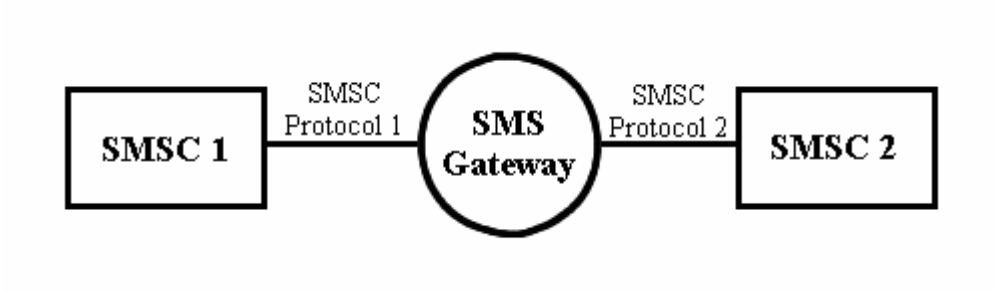
Εκτός από τα παραπάνω ζητήματα, τα κινητά τηλέφωνα και τα GSM / GPRS μόντεμ είναι περισσότερο ή λιγότερο το ίδιο για την αποστολή και λήψη μηνυμάτων SMS από τον υπολογιστή. Για την ακρίβεια μπορούμε να σκεφτούμε ένα κινητό τηλέφωνο που είναι διαθέσιμο για AT εντολές ως «GSM/GPRS μόντεμ+πληκτρολόγιο+οθόνη+..» Δεν υπάρχει μεγάλη διαφορά μεταξύ κινητών τηλεφώνων και GSM / GPRS μόντεμ από άποψη ρυθμού μετάδοσης SMS, δεδομένου ότι ο καθοριστικός παράγοντας για το ρυθμό μετάδοσης SMS είναι το ασύρματο δίκτυο.

1.4 Τι είναι το SMS Gateway;

Ένα πρόβλημα της ανταλλαγής μηνυμάτων SMS είναι ότι τα κέντρα SMS που αναπτύσσονται από διαφορετικές εταιρίες χρησιμοποιούν το δικό τους πρωτόκολλο επικοινωνίας και τα περισσότερα από αυτά τα πρωτόκολλα είναι ιδιόκτητα. Για παράδειγμα, η Nokia διαθέτει ένα SMSC πρωτόκολλο που ονομάζεται CIMD ενώ κάποιος άλλος πωλητής SMSC, ο CMG, έχει ένα πρωτόκολλο SMSC που

ονομάζεται EMI. Δεν μπορούμε να συνδέσουμε δύο κέντρα SMS εάν δεν υποστηρίζουν ένα κοινό SMSC πρωτόκολλο. Για να αντιμετωπίσουμε αυτό το πρόβλημα, ένα SMS gateway τοποθετείται μεταξύ δύο κέντρων SMS. Αυτό φαίνεται στο παρακάτω σχήμα. Το SMS gateway συμπεριφέρεται σαν μια καθυστέρηση μεταξύ των δύο κέντρων SMS. Μεταφράζει ένα SMSC πρωτόκολλο σε κάποιο άλλο. Με αυτό τον τρόπο μπορεί να χρησιμοποιηθεί από δύο διαφορετικούς παρόχους ασύρματης επικοινωνίας για να διασύνδεουν τα SMS κέντρα τους για σκοπούς όπως την δυνατότητα ανταλλαγής μεταξύ φορέων εκμετάλλευσης SMS μηνυμάτων.

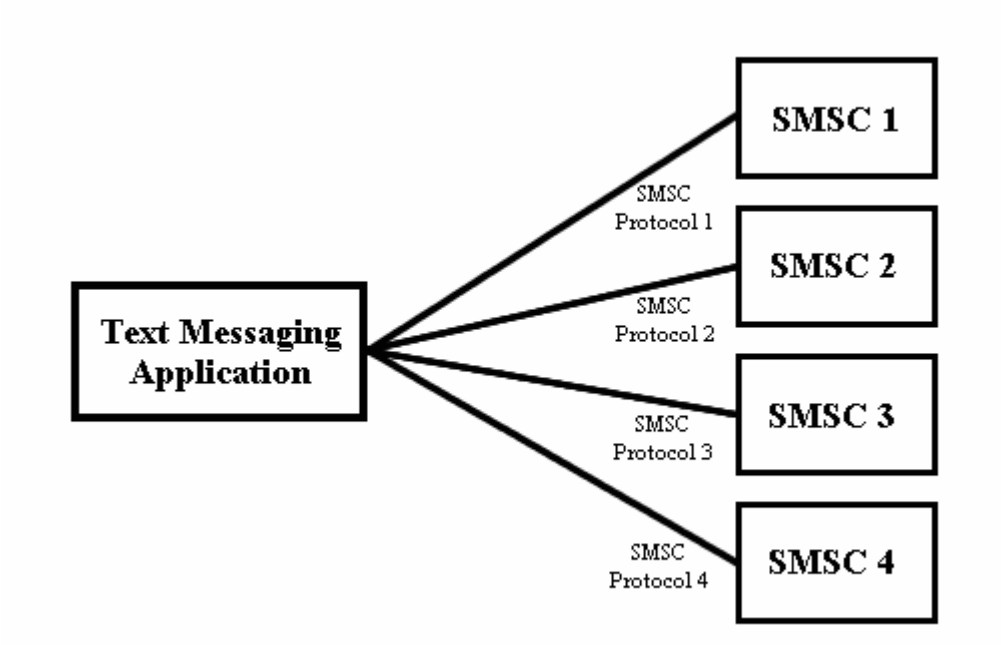
Σχήμα 1. Ένα SMS gateway συμπεριφέρεται σαν μια καθυστέρηση μεταξύ των δύο κέντρων



Εκτός από τους παρόχους ασύρματης επικοινωνίας, τους content providers και τους σχεδιαστές των εφαρμογών SMS μπορούμε να θεωρήσουμε ένα SMS gateway αρκετά χρήσιμο. Ας σκεφτούμε την ακόλουθη περίπτωση: Υποθέτουμε ότι είμαστε ο σχεδιαστής μιας εφαρμογής ανταλλαγής γραπτών μηνυμάτων SMS. Για να στείλουμε

και να λάβουμε μηνύματα SMS στον server μας, ένας τρόπος είναι να συνδεθούμε στο κέντρο SMS των φορέων ασύρματης επικοινωνίας. Διαφορετικοί φορείς μπορεί να χρησιμοποιούν κέντρα SMS από διαφορετικές εταιρίες, το οποίο σημαίνει ότι η εφαρμογή μας μπορεί να πρέπει να υποστηρίξει πολλαπλά συγκεκριμένα SMS πρωτόκολλα. (Αυτό φαίνεται στο παρακάτω σχήμα). Σαν αποτέλεσμα η πολυπλοκότητα και ο χρόνος ανάπτυξης της εφαρμογής αυξάνονται.

Σχήμα 2. Μία εφαρμογή ανταλλαγής μηνυμάτων SMS συνδέεται σε κέντρα SMS χωρίς ένα SMS gateway



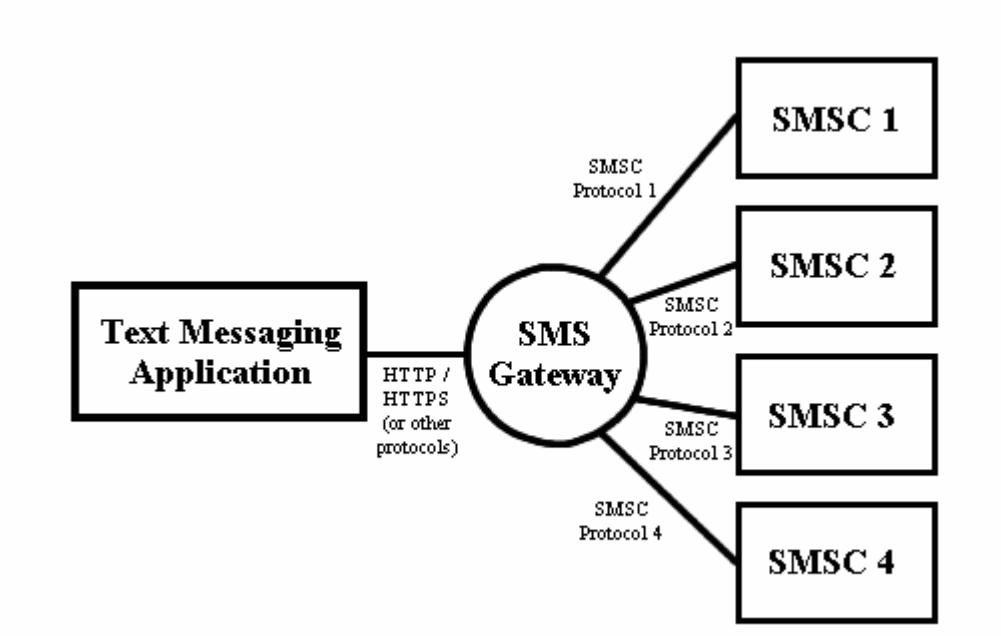
Για να αντιμετωπίσουμε το παραπάνω πρόβλημα, ένα SMS gateway μπορεί να στηθεί για να χειρίζεται τις συνδέσεις προς στα κέντρα SMS. Τώρα η εφαρμογή γραπτού μηνύματος SMS το μόνο που χρειάζεται να γνωρίζει είναι το πώς θα συνδεθεί στο SMS gateway. Για να υποστηρίξουμε περισσότερα κέντρα SMS, αυτό που πρέπει να κάνουμε απλά είναι να τροποποιήσουμε τις ρυθμίσεις του SMS gateway. Δεν απαιτείται καμία αλλαγή στον πηγαίο κώδικα της εφαρμογής.

Η χρήση ενός SMS gateway μπορεί να μειώσει σημαντικά το χρόνο ανάπτυξης της εφαρμογής.

Για να συνδεθούμε σ' ένα SMS gateway, μπορούμε να χρησιμοποιήσουμε ένα SMSC πρωτόκολλο όπως το SMPP και το CIMD. Μερικά SMS gateway υποστηρίζουν HTTP/HTTPS διεπαφή. Είναι πιο εύκολο να χρησιμοποιήσουμε το HTTP/HTTPS από τα SMSC πρωτόκολλα. Το μειονέκτημα είναι ότι μπορεί να υπάρχουν λιγότερες δυνατότητες SMS προς χρήση. Για παράδειγμα, ένα SMS

gateway μπορεί να μην υποστηρίζει την αποστολή μηνυμάτων εικόνας μέσω της HTTP / HTTPS διεπαφής.

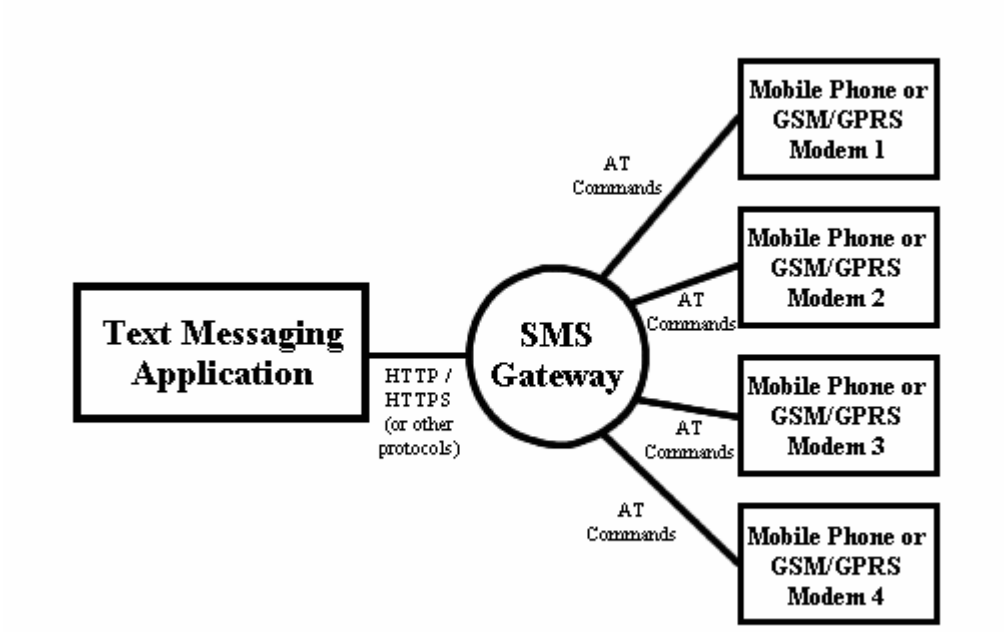
Σχήμα 3. Μια εφαρμογή γραπτών μηνυμάτων SMS συνδέεται με κέντρα SMS μέσω ενός SMS gateway



Εκτός από την χρήση απευθείας σύνδεσης σ' ένα κέντρο SMS ενός φορέα ασύρματης επικοινωνίας, ένας άλλος τρόπος αποστολής και λήψης γραπτών μηνυμάτων SMS σ' έναν υπολογιστή είναι να χρησιμοποιήσουμε ένα κινητό τηλέφωνο ή ένα GSM/GPRS μόντεμ. Για να το κάνουμε αυτό, η εφαρμογή μας θα πρέπει να γνωρίζει πώς να συνδεθεί με το κινητό τηλέφωνο ή το GSM/GPRS μόντεμ χρησιμοποιώντας AT εντολές.

Μερικά SMS gateway είναι ικανά να χειριστούν τις συνδέσεις με κινητά τηλέφωνα ή GSM/GPRS μόντεμ. Για να στείλει και να λάβει ένα γραπτό μήνυμα SMS με ένα κινητό τηλέφωνο ή ένα GSM/GPRS μόντεμ, η εφαρμογή το μόνο που θα πρέπει να γνωρίζει είναι το πώς θα μιλήσει στο SMS gateway και δεν χρειάζεται να γνωρίζει οτιδήποτε σχετικά με AT εντολές.

Σχήμα 4. Μια εφαρμογή γραπτών μηνυμάτων SMS συνδέεται με μια ομάδα κινητών τηλεφώνων και GPRS/GSM μόντεμ μέσω ενός SMS gateway



1.4.1 Λογισμικό ανοικτού κώδικα και δωρεάν SMS gateway

Όπως βλέπουμε στην παραπάνω ενότητα, ένα SMS gateway έχει πολλές ευθύνες σ' ένα σύστημα ανταλλαγής μηνυμάτων SMS. Έτσι το λογισμικό ενός SMS gateway μπορεί να είναι πολύ περίπλοκο και ένα περίπλοκο λογισμικό συνήθως είναι πολύ ακριβό. Ευτυχώς υπάρχουν πακέτα SMS gateway λογισμικού ανοικτού κώδικα που μπορούμε να τα κατεβάσουμε από το internet χωρίς χρέωση. Ένα υψηλής ποιότητας δωρεάν πακέτο SMS gateway λογισμικού είναι το Kannel, το οποίο είναι γραμμένο σε γλώσσα προγραμματισμού C. Το Kannel μπορεί να χειριστεί συνδέσεις με κέντρα SMS, κινητά τηλέφωνα και GSM/GPRS μόντεμ. Έχει μια HTTP/HTTPS διεπαφή για την αποστολή και λήψη

SMS μηνυμάτων. Περισσότερες πληροφορίες σχετικά με το Kannel υπάρχουν στην ιστοσελίδα http://www.kannel.org/ .

1.5 Εισαγωγή στις AT εντολές

Οι AT εντολές είναι οδηγίες που χρησιμοποιούνται για να ελέγχουν ένα μόντεμ. AT είναι η συντομογραφία του Attention (Προσοχή). Κάθε γραμμή εντολών ξεκινά με «AT» ή «at». Γι αυτό οι εντολές μόντεμ λέγονται AT εντολές. Πολλές από τις εντολές που χρησιμοποιούνται για τον έλεγχο ενσύρματου dial-up μόντεμ, όπως οι ATD (Dial), ATA (Answer), ATH (Hook control), ATO (Return to online data state), υποστηρίζονται επίσης από GSM/GPRS μόντεμ και κινητά τηλέφωνα. Εκτός από αυτό το κοινό σετ AT εντολών, τα GSM/GPRS μόντεμ και τα κινητά τηλέφωνα υποστηρίζουν ένα σετ AT εντολών που είναι συγκεκριμένες στην τεχνολογία GSM, το οποίο περιέχει εντολές σχετιζόμενες με τα SMS όπως AT+CMGS (Send SMS message-Στείλε ένα SMS μήνυμα), AT+CMSS (Send SMS message from storage-Στείλε ένα SMS μήνυμα από την μνήμη),

AT+CMGL(List SMS messages-Κάνε μια λίστα των SMS μηνυμάτων) και AT+CMGR (Read SMS messages-Διάβασε τα SMS μηνύματα).

Να σημειώσουμε ότι το αρχικό «AT» είναι το πρόθεμα που ενημερώνει το μόντεμ για την έναρξη μιας γραμμής εντολών. Δεν είναι μέρος του ονόματος της AT εντολής. Για παράδειγμα, το D είναι το πραγματικό όνομα της AT εντολής στην ATD και το +CMGS είναι το πραγματικό όνομα της AT εντολής στην AT+CMGS. Ωστόσο, ορισμένα βιβλία και ιστοσελίδες τα χρησιμοποιούν εναλλακτικά ως το όνομα μιας εντολής AT.

Εδώ είναι μερικές εργασίες που μπορούν να γίνουν χρησιμοποιώντας AT εντολές με ένα GSM/GPRS μόντεμ ή ένα κινητό τηλέφωνο:

- Να πάρουμε βασικές πληροφορίες για το κινητό τηλέφωνο ή το GSM/GPRS μόντεμ. Για παράδειγμα, το όνομα του κατασκευαστή (AT+CMGMI), τον αριθμό IMEI (International Mobile Equipment Identity) (AT+CGSN) και η έκδοση του λογισμικού (AT+CGMR)
- Να πάρουμε βασικές πληροφορίες για τον συνδρομητή. Για παράδειγμα, MSISDN (AT+CNUM) και τον αριθμό IMSI (International Mobile Subscriber Identity)(AT+CIMI).
- Να στείλουμε και να λάβουμε fax (ATD, ATA, AT+F*).
- Να στείλουμε (AT+CMGS, AT+CMSS), να διαβάσουμε (AT+CMGR, AT+CMGL),να γράψουμε (AT+CMGW) ή να διαγράψουμε (AT+CMGD) μηνύματα SMS και να λαμβάνουμε ειδοποιήσεις για νέα ληφθέντα μηνύματα (AT+CNMI)
- Να διαβάσουμε (AT+CPBR), να γράψουμε (AT+CPBW), ή να ψάξουμε μια επαφή στον τηλεφωνικό κατάλογο
- Να εκτελούμε εργασίες σχετικές με την ασφάλεια, όπως το άνοιγμα και το κλείσιμο των κλειδαριών διευκόλυνσης (AT+CLCK),τον έλεγχο αν μια διευκόλυνση είναι κλειδωμένη (AT+CLCK), και την αλλαγή password

(AT+CPWD).(Παραδείγματα κλειδώματος διευκόλυνσης: κλείδωμα SIM [ένα password πρέπει να δίνεται στην κάρτα SIM κάθε φορά που ανοίγει το κινητό τηλέφωνο] και κλείδωμα PH-SIM [μια συγκεκριμένη κάρτα SIM σχετίζεται με το κινητό τηλέφωνο. Για να χρησιμοποιήσουμε άλλες κάρτες SIM με το κινητό τηλέφωνο, θα πρέπει να εισάγουμε ένα password).

- Να αποθηκεύσουμε και να επαναφέρουμε τις διαμορφώσεις του GSM/GPRS μόντεμ ή του κινητού τηλεφώνου. Για παράδειγμα, να αποθηκεύσουμε (AT+CSAS) και να επαναφέρουμε (AT+CRES) ρυθμίσεις σχετικά με την ανταλλαγή μηνυμάτων SMS, όπως την διεύθυνση του κέντρου SMS.

Να σημειώσουμε ότι οι κατασκευαστές κινητών τηλεφώνων συνήθως δεν παρέχουν όλες τις AT εντολές, τις παραμέτρους των εντολών και τις τιμές των παραμέτρων στα κινητά τους τηλέφωνα. Επίσης, η συμπεριφορά των παρεχόμενων AT εντολών μπορεί να είναι διαφορετική από αυτή που ορίζεται στο πρότυπο. Σε γενικές γραμμές, τα GSM/GPRS μόντεμ που είναι

σχεδιασμένα για ασύρματες εφαρμογές υποστηρίζουν καλύτερα τις AT εντολές από τα συνηθισμένα κινητά τηλέφωνα.

Επιπλέον, ορισμένες εντολές AT απαιτούν την υποστήριξη των φορέων κινητών δικτύων. Για παράδειγμα, SMS μέσω GPRS μπορεί να ενεργοποιηθεί σε ορισμένα κινητά τηλέφωνα GPRS και GPRS μόντεμ με την εντολή + CGSMS (όνομα εντολής στο κείμενο:Select Service for MO SMS Messages). Αλλά αν η εταιρία κινητής τηλεφωνίας δεν υποστηρίζει τη μετάδοση των μηνυμάτων SMS μέσω GPRS, δεν μπορούμε να χρησιμοποιήσουμε αυτήν τη δυνατότητα.

1.5.1 Βασικές Εντολές και Επεκτάσιμες Εντολές

Υπάρχουν δύο τύποι AT εντολών:

- Βασικές εντολές. Οι Βασικές εντολές είναι οι AT εντολές που δεν ξεκινάνε με «+». Για παράδειγμα, οι D (Dial), A (Answer), H (Hook control) και O (Return to online data state) είναι βασικές εντολές.

- Επεκτάσιμες εντολές. Οι επεκτάσιμες εντολές είναι οι AT εντολές που ξεκινάνε με «+». Όλες οι GSMAT εντολές είναι επεκτάσιμες εντολές. Για παράδειγμα,οι +CMGS (Send SMS message-Στείλε ένα SMS μήνυμα), +CMSS (Send SMS message from storage-Στείλε ένα SMS μήνυμα από την μνήμη), +CMGL(List SMS messages-Κάνε μια λίστα των SMS μηνυμάτων) και +CMGR (Read SMS messages-Διάβασε τα Στείλε ένα SMS μηνύματα) είναι επεκτάσιμες εντολές.

ΚΕΦΑΛΑΙΟ 2

2.1 Το Microsoft Hyper Terminal

Το Microsoft Hyper Terminal είναι ένα μικρό πρόγραμμα που έρχεται μαζί με τα Microsoft Windows. Μπορούμε να το χρησιμοποιήσουμε για να στείλουμε AT εντολές στο κινητό μας τηλέφωνο ή το GSM/GPRS μόντεμ. Βρίσκεται στο μενού *Έναρξη-> Προγράμματα-> Βοηθήματα-> Επικοινωνίες-> Hyper Terminal.* Αν χρησιμοποιούμε Windows 98 πολύ πιθανόν να μην είναι εγκατεστημένο.

Πριν ξεκινήσουμε να γράφουμε την εφαρμογή μας για ανταλλαγή SMS μηνυμάτων καλό θα ήταν να ελέγξουμε αν το κινητό μας τηλέφωνο ή το GSM/GPRS μόντεμ και η κάρτα SIM δουλεύουν κανονικά. Το MS Hyper Terminal

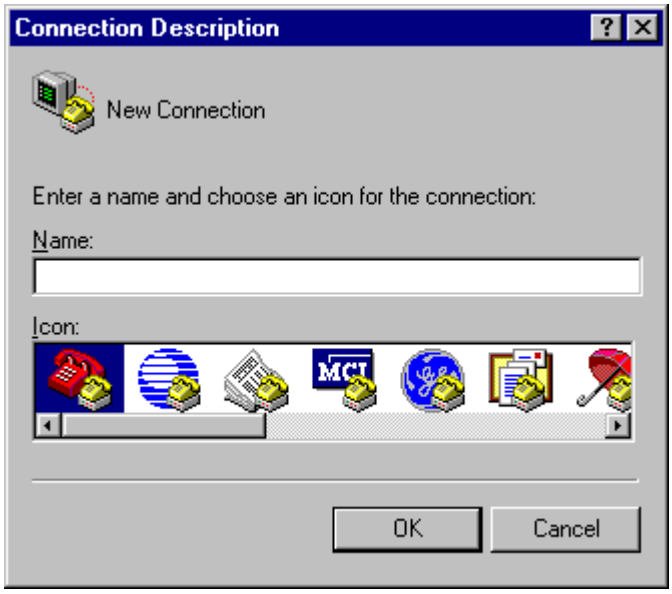
είναι ένα χρήσιμο εργαλείο όσο αφορά την δοκιμή των GSM συσκευών μας. Είναι μια καλή ιδέα να δοκιμάσουμε τις GSM συσκευές μας εκ των προτέρων. Όταν παρουσιαστεί κάποιο πρόβλημα, μερικές φορές είναι δύσκολο να πούμε τι προκαλεί το πρόβλημα. Η αιτία μπορεί να είναι το πρόγραμμά μας, η GSM συσκευή ή η κάρτα SIM. Αν τεστάρουμε την GSM συσκευή μας και την κάρτα SIM με το MS Hyper Terminal και λειτουργούν κανονικά τότε είναι πολύ πιθανόν το πρόβλημα να προκλήθηκε από το πρόγραμμά μας. Για τους χρήστες του Linux, μπορεί να χρησιμοποιηθεί το minicom αντί του Hyper Terminal.

Για να χρησιμοποιήσουμε το MS Hyper Terminal για να στείλουμε AT εντολές στο κινητό μας τηλέφωνο ή το GSM/GPRS μόντεμ, ακολουθούμε την παρακάτω διαδικασία:

1. Τοποθετούμε μια έγκυρη κάρτα SIM στο κινητό τηλέφωνο ή στο GSM/GPRS μόντεμ.
2. Συνδέουμε το κινητό μας τηλέφωνο ή το GSM/GPRS μόντεμ στον υπολογιστή και εγκαθιστούμε το αντίστοιχο driver του μόντεμ το οποίο θα το βρούμε σε CD μαζί με την συσκευή από ή από το internet.

3. Τρέχουμε το MS Hyper Terminal επιλέγοντας *Έναρξη(Start)-> Προγράμματα(Programs)-> Βοηθήματα(Accessories)-> Επικοινωνίες(Communications)-> Hyper Terminal.*
4. Στο dialog box *Περιγραφή Σύνδεσης(Connection Description)* γράφουμε το όνομα της σύνδεσης και το εικονίδιο που μας αρέσει και πατάμε το OK.

Σχήμα 5. Το dialog box *Περιγραφή Σύνδεσης* του MS Hyper Terminal



5. Στο dialog box *Σύνδεση Στο(Connection To)* διαλέγουμε την θύρα COM που είναι συνδεδεμένο το κινητό τηλέφωνο ή το GSM/GPRS μόντεμ στο *Σύνδεση με(Connect using)* combo box. Μπορούμε να ελέγξουμε σε ποια θύρα είναι συνδεδεμένη η συσκευή μας ακολουθώντας την παρακάτω διαδικασία:

Στα Windows 98:

Πηγαίνουμε στον *Πίνακας Ελέγχου(Control Panel)->Μόντεμ(Modem)* κάνουμε κλίκ στην καρτέλα *Διαγνωστικά* (*Diagnostics*)και στην λίστα μπορούμε να δούμε σε ποια θύρα COM είναι συνδεδεμένη η συσκευή μας.

Στα Windows 2000 και Windows XP:

Πηγαίνουμε στον *Πίνακας Ελέγχου(Control Panel)->Επιλογές Τηλεφώνου και Μόντεμ(Telephone and Modem Options)* και κάνουμε κλίκ στην καρτέλα *Μόντεμ(Modem)*. Στην λίστα μπορούμε να δούμε σε ποια θύρα COM είναι συνδεδεμένη η συσκευή μας.

Σχήμα 6. Το dial box *Σύνδεση Στο* του MS Hyper Terminal

6. Το dialog box *Ιδιότητες* εμφανίζεται. Εισάγουμε τις σωστές ρυθμίσεις της θύρας για το κινητό μας τηλέφωνο ή το GSM/GPRS μόντεμ και πατάμε το OK.

Για να τσεκάρουμε τις ρυθμίσεις τις θύρας που χρησιμοποιεί η συσκευή μας στα Windows 98 ακολουθούμε τα εξής βήματα:

a. Πηγαίνουμε στον *Πίνακας Ελέγχου(Control Panel)->Μόντεμ(Modem).*

b. Διαλέγουμε το κινητό μας τηλέφωνο ή το GSM/GPRS μόντεμ από την λίστα

c. Κάνουμε κλικ στην καρτέλα *Ιδιότητες(Properties).*

d. Εμφανίζεται το dialog box *Ιδιότητες(Properties)*. Το πεδίο *Μέγιστες ταχύτητες(Maximum speeds)* στην καρτέλα *Γενικά(General)* αντιστοιχεί στο πεδίο *Bits ανά δευτερόλεπτο(Bits per second)* του Hyper Terminal. Κάνοντας κλίκ στην καρτέλα *Σύνδεση(Connection)* μπορούμε να βρούμε τις ρυθμίσεις για τα bit δεδομένων, την ισοτιμία και τα bit διακοπής. Κάνοντας κλίκ στο κουμπί *Για Προχωρημένους(Advanced)* μπορούμε να βρούμε τις ρυθμίσεις για τον έλεγχο ροής.

Για να τσεκάρουμε τις ρυθμίσεις τις θύρας που χρησιμοποιεί η συσκευή μας στα Windows 2000 και Windows XP ακολουθούμε τα εξής βήματα:

a. Πηγαίνουμε στον *Πίνακας Ελέγχου(Control Panel)->Επιλογές Τηλεφώνου και Μόντεμ(Telephone and Modem Options)* και κάνουμε κλίκ στην καρτέλα *Μόντεμ*(Modem).

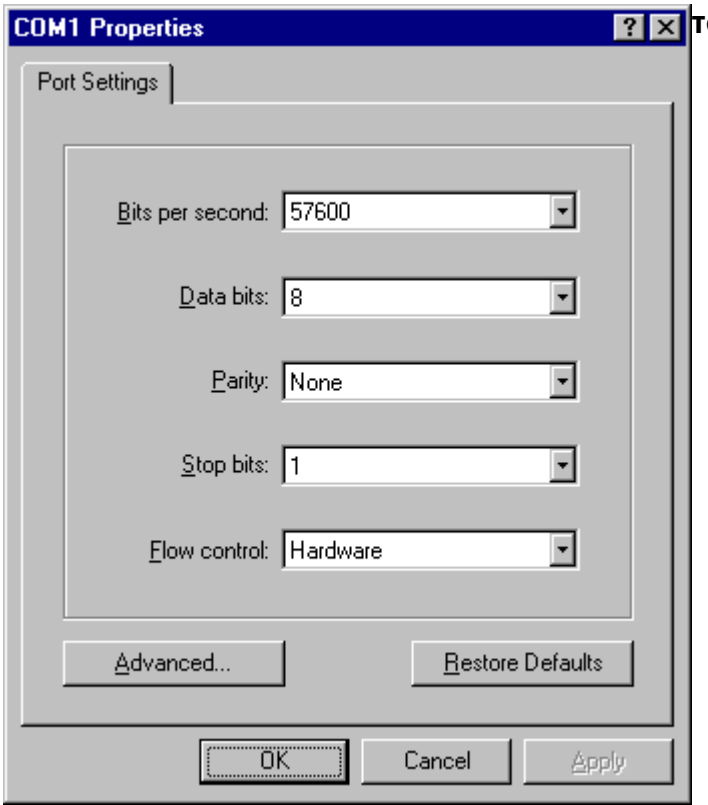
b. Διαλέγουμε το κινητό μας τηλέφωνο ή το GSM/GPRS μόντεμ από την λίστα.

c. Κάνουμε κλικ στην καρτέλα *Ιδιότητες(Properties).*

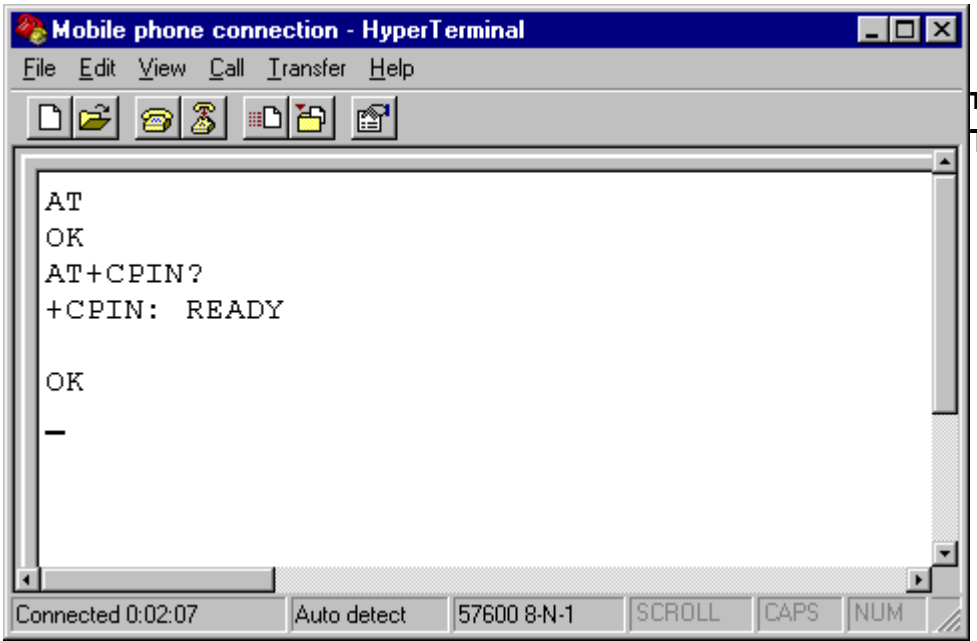
d. Εμφανίζεται το dialog box *Ιδιότητες(Properties)*. Κάνουμε κλίκ στην καρτέλα *Για Προχωρημένους(Advanced)* και κάνουμε κλίκ στο κουμπί *Αλλαγή Προεπιλεγμένων Προτιμήσεων(Change Default Preferences).*

e. Εμφανίζεται το dialog box *Αλλαγή Προεπιλεγμένων Προτιμήσεων(Change Default Preferences).* Το πεδίο *ταχύτητα Θύρας(Port speed)* στην καρτέλα *Γενικά(General)* αντιστοιχεί το πεδίο *Bits ανά δευτερόλεπτο(Bits per second)* του Hyper Terminal. Επίσης στην καρτέλα *Γενικά(General)* μπορούμε να βρούμε και τις ρυθμίσεις για τον έλεγχο ροής. Στην καρτέλα *Για Προχωρημένους(Advanced)* μπορούμε να βρούμε τις ρυθμίσεις για τα bit δεδομένων, την ισοτιμία και τα bit διακοπής.

Σχήμα 7. Το dial box Ιδιότητες του Hyper Terminal



7. Πληκτρολογούμε «AT» στο κυρίως παράθυρο. Μια απάντηση «OK» θα πρέπει να επιστραφεί από το κινητό τηλέφωνο ή το GSM/GPRS μόντεμ. Πληκτρολογούμε «AT+CPIN?» στο κυρίως παράθυρο. Η AT εντολή «AT+CPIN?» χρησιμοποιείται για να εξετάσουμε αν το κινητό τηλέφωνο ή το GSM/GPRS μόντεμ περιμένει κάποιο PIN. Αν η απάντηση είναι «+CPIN:READY» σημαίνει ότι η κάρτα SIM δεν απαιτεί PIN και είναι έτοιμο για χρήση. Αν η κάρτα SIM απαιτεί PIN τότε θα πρέπει να το δώσουμε με την εντολή «AT+CPIN=<PIN>».



Σχήμα 8. Το κυρίως παράθυρο του Hyper Terminal

Αν πάρουμε απάντηση, το κινητό μας τηλέφωνο ή το GSM/GPRS μόντεμ δουλεύει κανονικά και μπορούμε να ξεκινήσουμε να πληκτρολογούμε AT εντολές για να ελέγχουμε το κινητό τηλέφωνο ή το GSM/GPRS μόντεμ.

2.2 Αποστολή μηνύματος SMS με κινητό τηλέφωνο και GSM/GPRS μόντεμ από τον υπολογιστή χρησιμοποιώντας AT εντολές.

Υπάρχουν πολλοί τρόποι για να συνδέσουμε ένα κινητό τηλέφωνο ή GSM/GPRS μόντεμ σ' έναν υπολογιστή. Για παράδειγμα, μπορούν να συνδεθούν μέσω σειριακού καλωδίου, USB καλωδίου, σύνδεση Bluetooth ή σύνδεση με υπέρυθρες αναλόγως με την ικανότητα του τηλεφώνου ή του GSM/GPRS μόντεμ.

Αφού συνδέσουμε το κινητό τηλέφωνο ή το GSM/GPRS μόντεμ στον υπολογιστή μπορούμε να ελέγξουμε με AT εντολές. Τα dial-up μόντεμ, τα κινητά τηλέφωνα, τα GSM/GPRS μόντεμ υποστηρίζουν ένα κοινό σετ σταθερών AT εντολών, τα κινητά τηλέφωνα και τα GSM/GPRS μόντεμ υποστηρίζουν ένα επεκτάσιμο σετ AT εντολών. Μια χρήση των επεκτάσιμων AT εντολών είναι ο έλεγχος της αποστολής και λήψης μηνυμάτων SMS. Ο ακόλουθος πίνακας καταγράφει της AT εντολές που σχετίζονται με το γράψιμο και την αποστολή μηνυμάτων SMS.

AT εντολές	Έννοια
+CMGS	Αποστολή μηνύματος
+CMSS	Αποστολή αποθηκευμένων μηνυμάτων
+CMGW	Εγγραφή μηνύματος στη μνήμη
+CMGD	Διαγραφή μηνύματος
+CMGC	Αποστολή εντολής
+CMMS	Αποστολή περισσότερων μηνυμάτων

Παρακάτω φαίνεται ένα απλό παράδειγμα που δείχνει πώς να χρησιμοποιήσουμε AT εντολές και το πρόγραμμα HyperTerminal για να στείλουμε ένα μήνυμα SMS. Οι γραμμές με τα έντονα είναι οι γραμμές εντολών που πληκτρολογούμε στο HyperTerminal. Οι άλλες γραμμές είναι οι απαντήσεις που επιστρέφουν από το κινητό τηλέφωνο ή το GSM/GPRS μόντεμ.

AT

OK

AT+CMGF=1

OK

AT+CMGW=”6977401908”

>HELLO TO YOU

+CMGW:1

OK

AT+CMSS=1

+CMSS:20

OK

Περιγραφή:

- Γραμμή 1: Στέλνουμε «AT» στο GSM/GPRS μόντεμ ή στο κινητό τηλέφωνο για να δοκιμάσουμε την σύνδεση. Το μόντεμ ή το τηλέφωνο επιστρέφει το κωδικό αποτέλεσμα «OK» (γραμμή 2), το οποίο σημαίνει ότι η σύνδεση μεταξύ της συσκευής και του Hyper Terminal δουλεύει καλά.

- Γραμμή 3: Η AT εντολή +CMGF χρησιμοποιείται για να δώσει οδηγία στο μόντεμ ή στο κινητό τηλέφωνο να λειτουργήσει σε κατάσταση κειμένου SMS. Το κωδικό αποτέλεσμα «OK» (γραμμή 4) επιστρέφεται, το οποίο δείχνει ότι η εντολή «AT+CMGF=1» εκτελέστηκε επιτυχώς. Αν επιστραφεί το κωδικό αποτέλεσμα «ERROR», είναι πιθανόν το μόντεμ ή το τηλέφωνο να μην υποστηρίζει την κατάσταση κειμένου. Για να το επιβεβαιώσουμε, πληκτρολογούμε «AT+CMGF=?» στο Hyper Terminal. Αν η απάντηση είναι «+CMGF:(0,1)» (0=κατάσταση PDU, 1=κατάσταση κειμένου), τότε υποστηρίζεται η κατάσταση κειμένου SMS. Αν η απάντηση είναι «+CMGF:(0)» τότε η κατάσταση κειμένου δεν υποστηρίζεται.

- Γραμμή 5 και 6: Η AT εντολή +CMGW χρησιμοποιείται για να γράψουμε ένα μήνυμα κειμένου SMS στην μνήμη του μόντεμ ή του κινητού τηλεφώνου. «6977401908» είναι ο αριθμός του κινητού τηλεφώνου του παραλήπτη. Αφότου γράψουμε το τηλέφωνο του παραλήπτη πρέπει να πατήσουμε ENTER. Το μόντεμ ή το κινητό τηλέφωνο θα επιστρέψει το πρόθεμα «>» και μπορούμε να γράψουμε το κείμενο «HELLO TO YOU». Όταν τελειώσουμε πατάμε το Ctrl+z στο πληκτρολόγιο.

- Γραμμή 7: Η «+CMGW:1» μας λέει ότι ο δείκτης που τοποθετήθηκε στο μήνυμα κειμένου είναι το 1. Δείχνει την θέση του μηνύματος στην μνήμη.

- Γραμμή 9: Το κωδικό αποτέλεσμα «OK» δείχνει ότι η εκτέλεση της εντολής +CMGW είναι επιτυχής.

- Γραμμή 10: Η AT εντολή +CMSS χρησιμοποιείται για να στείλουμε το μήνυμα κειμένου από την μνήμη του μόντεμ ή του κινητού τηλεφώνου. «1» είναι ο δείκτης του μηνύματος κειμένου που λάβαμε από την γραμμή 7.

- Γραμμή 11: Η «+CMSS:20» μας λέει ότι ο αριθμός που δίνεται στο μήνυμα κειμένου SMS είναι 20.

- Γραμμή 13: Το κωδικό αποτέλεσμα «OK» δείχνει ότι η εκτέλεση της εντολής +CMSS είναι επιτυχής. Υπάρχουν περιπτώσεις όμως που μπορεί να προκύψει κάποιο λάθος όπως να είναι γεμάτη η μνήμη του κινητού τηλεφώνου ή του GSM/GPRS μόντεμ ή να μην είναι έγκυρη η κάρτα SIM. Τα λάθη αυτά εμφανίζονται με κωδικούς, για παράδειγμα «ERROR:320». Ο ακόλουθος πίνακας καταγράφει τους κωδικούς των λαθών και την έννοιά τους.

Κωδικοί λαθών	**Έννοια**
0-127	GSM 04.11 Annex E-2 τιμές
128-255	GSM 03.40 section 9.2.3.22 τιμές
300	Αποτυχία τηλεφώνου ή μόντεμ
301	Αποκλεισμένη υπηρεσία SMS του τηλεφώνου ή του μόντεμ
302	Μη επιτρεπόμενη λειτουργία
303	Μη υποστηριζόμενη λειτουργία
304	Άκυρη PDU mode παράμετρος
305	Άκυρη txt mode παράμετρος
310	Η κάρτα SIM δεν εισήχθη
311	Απαιτείται SIM PIN
312	Απαιτείται PH-SIM PIN
313	Αποτυχία της κάρτας SIM
314	Απασχολημένη κάρτα SIM
315	Λάθος κάρτα SIM
320	Αποτυχία μνήμης
321	Άκυρος δείκτης μνήμης
322	Μνήμη Πλήρης

Κωδικοί λαθών	Έννοια
330	Άγνωστη διεύθυνση κέντρου SMS
331	Καμία υπηρεσία δικτύου
332	Διακοπή δικτύου
500	Άγνωστο λάθος
512	Ειδικός κατασκευαστής

Για να στείλουμε μηνύματα SMS από μια εφαρμογή, πρέπει να γράψουμε τον πηγαίο κώδικα για την σύνδεση και αποστολή AT εντολών στο κινητό τηλέφωνο ή το GSM/GPRS μόντεμ, ακριβώς όπως κάνει ένα τερματικό πρόγραμμα. Μπορούμε να γράψουμε τον πηγαίο κώδικα σε C, C++, Java, Visual Basic, Delphi ή οποιαδήποτε άλλη γλώσσα προγραμματισμού.

2.2.1 Σημαντικό μειονέκτημα της αποστολής μηνυμάτων μέσω Κινητού Τηλεφώνου ή του GSM/GPRS Μόντεμ—Χαμηλός αριθμός μηνυμάτων SMS

Η χρήση κινητού τηλεφώνου ή GSM/GPRS μόντεμ για να στείλουμε μηνύματα SMS έχει ένα σημαντικό μειονέκτημα, το οποίο είναι ότι ο αριθμός μηνυμάτων SMS είναι πολύ χαμηλός. Μόνο 6-10 μηνύματα μπορούν να σταλθούν ανά λεπτό (όταν χρησιμοποιείται η κατάσταση «SMS μέσω GSM». Η απόδοση δεν επηρεάζεται από τη σύνδεση μεταξύ του υπολογιστή και του κινητού τηλεφώνου ή του GSM / GPRS μόντεμ (ο αριθμός μηνυμάτων είναι περίπου ο ίδιος χωρίς να έχει σημασία αν το κινητό τηλέφωνο ή το GSM/GPRS μόντεμ είναι συνδεδεμένο στον υπολογιστή μέσω σειριακού καλωδίου, USB καλωδίου, σύνδεση Bluetooth ή σύνδεση με υπέρυθρες) και δεν εξαρτάται από το αν χρησιμοποιείται κινητό τηλέφωνο ή GSM / GPRS μόντεμ. Ο καθοριστικός παράγοντας για τον αριθμό μηνυμάτων SMS είναι το ασύρματο δίκτυο.

2.3 Λήψη Μηνυμάτων SMS Χρησιμοποιώντας Υπολογιστή

Γενικά υπάρχουν τρεις τρόποι να λάβουμε μηνύματα SMS χρησιμοποιώντας τον υπολογιστή μας:

1. Να συνδέσουμε ένα κινητό τηλέφωνο ή GSM / GPRS μόντεμ σ' έναν υπολογιστή. Μετά χρησιμοποιούμε τον υπολογιστή και τις AT εντολές για να πάρουμε τα ληφθέντα μηνύματα SMS από το κινητό τηλέφωνο ή το GSM / GPRS μόντεμ.
2. Να αποκτήσουμε πρόσβαση στο κέντρο SMS (SMSC) ή το SMS gateway του φορέα ασύρματης επικοινωνίας. Όλα τα μηνύματα SMS που θα ληφθούν θα προωθηθούν στον υπολογιστή μας χρησιμοποιώντας ένα πρωτόκολλο/διεπαφή υποστηριζόμενο από το κέντρο SMS ή SMS gateway.
3. Να αποκτήσουμε πρόσβαση στο SMS gateway ενός παρόχου υπηρεσιών SMS. Όλα τα μηνύματα SMS που θα ληφθούν θα προωθηθούν στον υπολογιστή μας χρησιμοποιώντας ένα πρωτόκολλο/διεπαφή υποστηριζόμενο από το SMS gateway.

2.3.1 Λήψη Μηνυμάτων SMS με Κινητό Τηλέφωνο και GSM/GPRS Μόντεμ Χρησιμοποιώντας τον Υπολογιστή.

Η λήψη μηνυμάτων SMS μέσω κινητού τηλεφώνου ή GSM / GPRS μόντεμ έχει ένα σημαντικό πλεονέκτημα—οι φορείς ασύρματης επικοινωνίας συνήθως δεν χρεώνουν κανένα φόρο για την λήψη εισερχόμενων μηνυμάτων με τις δικιές τους κάρτες SIM. Το μειονέκτημα της λήψης μηνυμάτων SMS με αυτόν τον τρόπο είναι ότι ένα κινητό τηλέφωνο ή GSM / GPRS μόντεμ δεν μπορεί να χειριστεί ένα μεγάλο ποσό κίνησης SMS. Ένας τρόπος για να ξεπεράσουμε αυτό είναι να φορτώσουμε την εξισορρόπηση της κίνησης SMS με μια ομάδα κινητών

τηλεφώνων ή GSM / GPRS μόντεμ. Κάθε κινητό τηλέφωνο ή GSM / GPRS μόντεμ θα έχει την δικιά του κάρτα SIM και νούμερο τηλεφώνου.

Όσον αφορά τον προγραμματισμό, η αποστολή και η λήψη μηνυμάτων SMS μέσω κινητού τηλεφώνου ή GSM / GPRS μόντεμ είναι παρόμοιες. Αυτό που χρειάζεται να κάνουμε είναι να στείλουμε οδηγίες (με την μορφή των AT εντολών) στο κινητό τηλέφωνο ή GSM / GPRS μόντεμ. Ο ακόλουθος πίνακας καταγράφει της AT εντολές που σχετίζονται με την λήψη και την ανάγνωση μηνυμάτων SMS.

AT εντολή	**Έννοια**
+CNMI	Ενδείξεις Νέου Μηνύματος
+CMGL	Κάνε μια Λίστα των Μηνυμάτων
+CMGR	Διάβασε τα μηνύματα
+CNMA	Αναγνώριση Νέου Μηνύματος

Παρακάτω φαίνεται ένα απλό παράδειγμα που δείχνει πώς να χρησιμοποιήσουμε AT εντολές και το πρόγραμμα HyperTerminal για να διαβάσουμε μηνύματα SMS που

λήφθηκαν από ένα GSM/GPRS μόντεμ ή κινητό τηλέφωνο. Οι γραμμές με τα έντονα είναι οι γραμμές εντολών που πληκτρολογούμε στο HyperTerminal. Οι άλλες γραμμές είναι οι απαντήσεις που επιστρέφουν από το κινητό τηλέφωνο ή το GSM/GPRS μόντεμ.

AT

OK

AT+CMGF=1

OK

AT+CMGL=”ALL”

+CMGL: 1,”REC UNREAD”,”+6977401908”, ,”11/5/6,00:30:29+32”

Hello how are you?

+CMGL: 2,”REC UNREAD”,”+6947480898”, ,”11/5/7,00:32:20+32”

Where are you?

OK

Περιγραφή:

- Γραμμή 1: Στέλνουμε «AT» στο GSM/GPRS μόντεμ ή στο κινητό τηλέφωνο για να δοκιμάσουμε την σύνδεση. Το μόντεμ ή το τηλέφωνο επιστρέφει το κωδικό αποτέλεσμα «OK» (γραμμή 2), το οποίο σημαίνει ότι η σύνδεση μεταξύ της συσκευής και του Hyper Terminal δουλεύει καλά.

- Γραμμή 3: Η AT εντολή +CMGF χρησιμοποιείται για να δώσει οδηγία στο μόντεμ ή στο κινητό τηλέφωνο να λειτουργήσει σε κατάσταση κειμένου SMS. Το κωδικό αποτέλεσμα «OK» (γραμμή 4) επιστρέφεται, το οποίο δείχνει ότι η εντολή «AT+CMGF=1» εκτελέστηκε επιτυχώς.

- Γραμμές 5-9: Η AT εντολή +CMGL χρησιμοποιείται για να φτιάξει μια λίστα όλα τα μηνύματα στην μνήμη του μόντεμ ή του τηλεφώνου. Υπάρχουν δύο μηνύματα στην μνήμη: «Hello how are you?» και «Where are you?», «6977401908» και «6947480898» είναι τα νούμερα των αποστολέων. Τα «11/5/6,00:30:29+32» και «11/5/7,00:32:20+32» μας λένε πότε λήφθηκαν τα μηνύματα SMS από το κέντρο SMS. «+32» είναι η ζώνη ώρας. Να

σημειώσουμε εδώ ότι η μονάδα μέτρησης είναι το ένα τέταρτο της ώρας. Έτσι το +32 σημαίνει GMT+8 ώρες, εφόσον 32 τέταρτα μιας ώρας=8 ώρες. Το «REC UNREAD» δείχνει ότι και τα δύο μηνύματα έχουν διαβαστεί πριν.

- Γραμμή 11: Το κωδικό αποτέλεσμα «OK» δείχνει ότι η εκτέλεση της εντολής +CMGL είναι επιτυχής.

Όπως στην αποστολή έτσι και στην λήψη γραπτών μηνυμάτων, για να μπορέσει μια εφαρμογή γραπτών μηνυμάτων SMS να συνδεθεί με και να στείλει AT εντολές σε κινητό τηλέφωνο ή GSM/GPRS μόντεμ, θα πρέπει να γράψουμε πηγαίο κώδικα. Τον κώδικα μπορούμε και σ' αυτήν την περίπτωση να τον γράψουμε σε C, C++, Java, Visual Basic, Delphi ή οποιαδήποτε άλλη γλώσσα προγραμματισμού.

ΚΕΦΑΛΑΙΟ 3

3.1 Σχεδίαση της εφαρμογής.

Για την εφαρμογή χρησιμοποιήσαμε ένα κινητό τηλέφωνο και συγκεκριμένα το Samsung S3650 (μπορούμε να χρησιμοποιήσουμε οποιοδήποτε κινητό τηλέφωνο αρκεί να δουλεύει σαν GPRS/GSM modem) και ένα καλώδιο USB. Τα εργαλεία σχεδίασης που χρησιμοποιήθηκαν είναι ο SQL SERVER για την σχεδίαση της βάσης δεδομένων και η Borland 6 C++ για την σχεδίαση του κώδικα.

3.2 Για να ξεκινήσουμε..

Αρχικά εγκαθιστούμε τον οδηγό του GPRS/GSM modem ή του κινητού τηλεφώνου που θέλουμε να χρησιμοποιήσουμε. Αφού γίνει η εγκατάσταση συνδέουμε το μόντεμ ή το κινητό τηλέφωνο με το USB καλώδιο και ελέγχουμε την θύρα στην οποία είναι συνδεδεμένο και τα bit δεδομένων με τον ίδιο τρόπο που ελέγχουμε και στο Hyper Terminal (βλ. Κεφάλαιο 2).

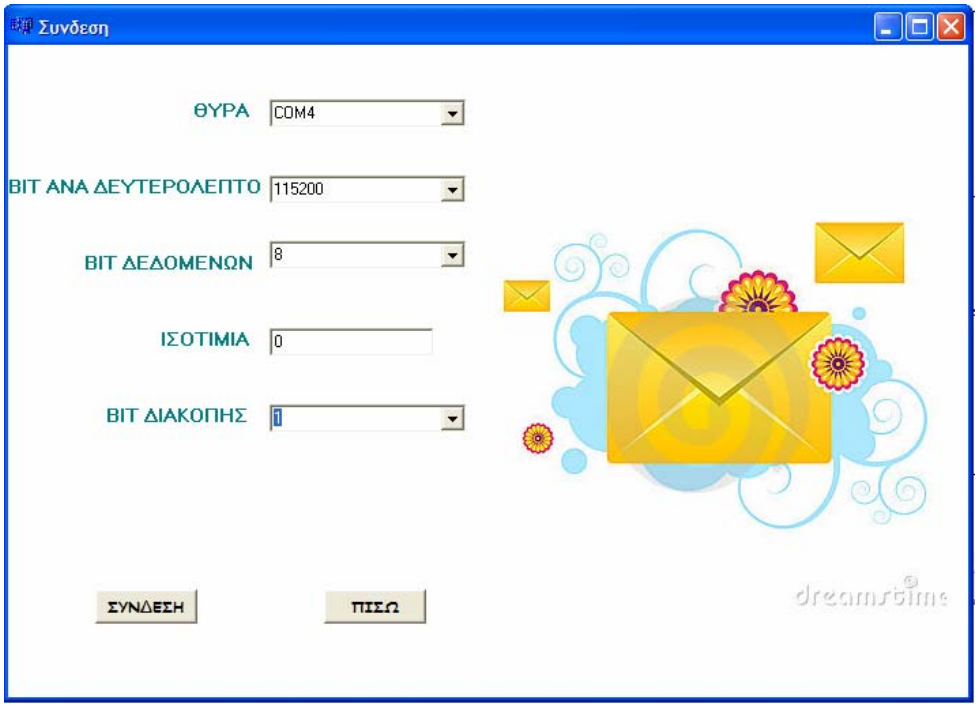
3.3 Οδηγίες χρήσης της εφαρμογής

Ανοίγοντας την εφαρμογή εμφανίζεται η πρώτη φόρμα όπου εισάγουμε το username και το password μας.

Εφόσον γίνει επιτυχής είσοδος ανοίγει η δεύτερη φόρμα που είναι το μενού

ΣΥΝΔΕΣΗ:

Πατώντας το κουμπί της σύνδεσης μας ανοίγει η φόρμα για να συνδέσουμε την συσκευή μας και ελέγχουμε αν συνδέθηκε σωστά. Εισάγουμε την θύρα στην οποία συνδέσαμε με το USB καλώδιο την συσκευή μας, τα bit ανά δευτερόλεπτο, τα bit δεδομένων που στέλνει κάθε φορά, την ισοτιμία δηλαδή το σήμα ελέγχου που χρειάζεται για την μεταφορά των δεδομένων και τα bit διακοπής. Μόλις εισάγουμε όλα τα στοιχεία πατάμε το κουμπί ΣΥΝΔΕΣΗ.

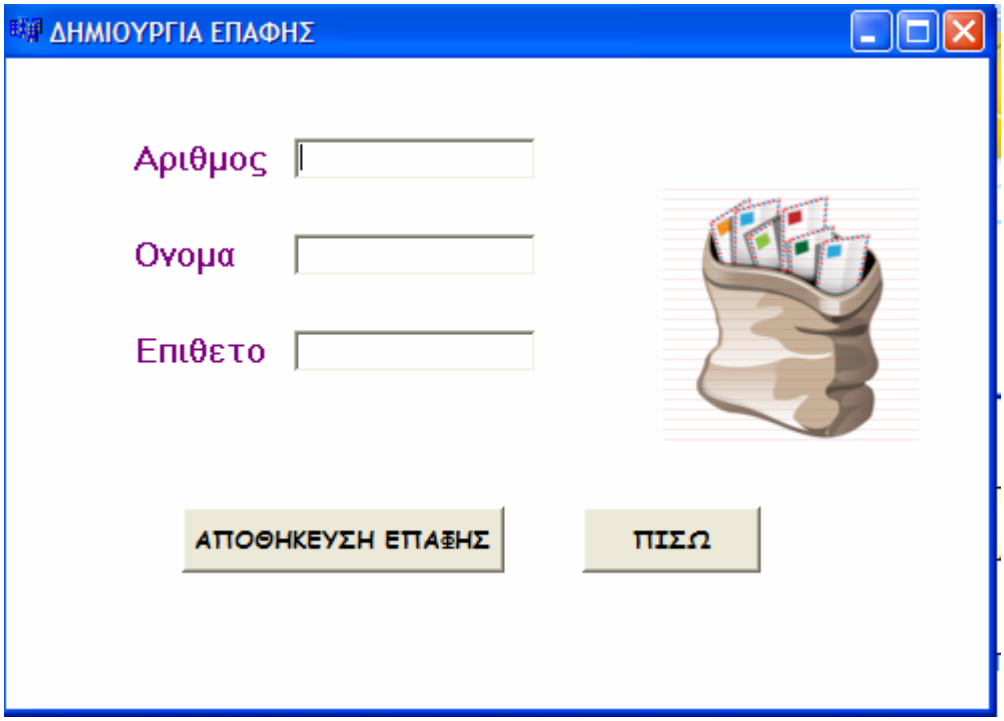


Αν η σύνδεση είναι επιτυχής εμφανίζεται μήνυμα επιτυχίας σύνδεσής αλλιώς σε περίπτωση που δεν συνδέθηκε σωστά η συσκευή μας εμφανίζεται μήνυμα αποτυχίας.

ΣΗΜΕΙΩΣΗ: Εφόσον η σύνδεση είναι επιτυχής ενεργοποιείται το κουμπί ΑΠΟΣΤΟΛΗ ΜΗΝΥΜΑΤΟΣ στην φόρμα του ΜΕΝΟΥ.

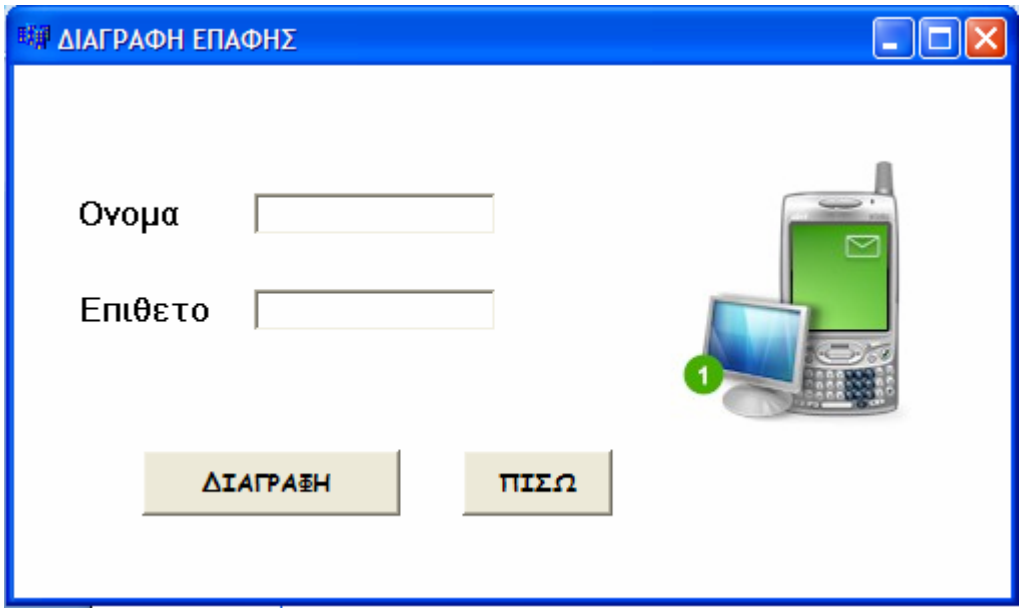
ΔΗΜΙΟΥΡΓΙΑ ΕΠΑΦΗΣ

Πατώντας το κουμπί ΔΗΜΙΟΥΡΓΙΑ ΕΠΑΦΗΣ εμφανίζεται η φόρμα για να εισάγουμε τα στοιχεία για την νέα επαφή που θέλουμε να αποθηκεύσουμε στην βάση μας. Εισάγουμε το όνομα, το επίθετο και τον αριθμό τηλεφώνου της επαφής και πατάμε το κουμπί ΑΠΟΘΗΚΕΥΣΗ ΕΠΑΦΗΣ.



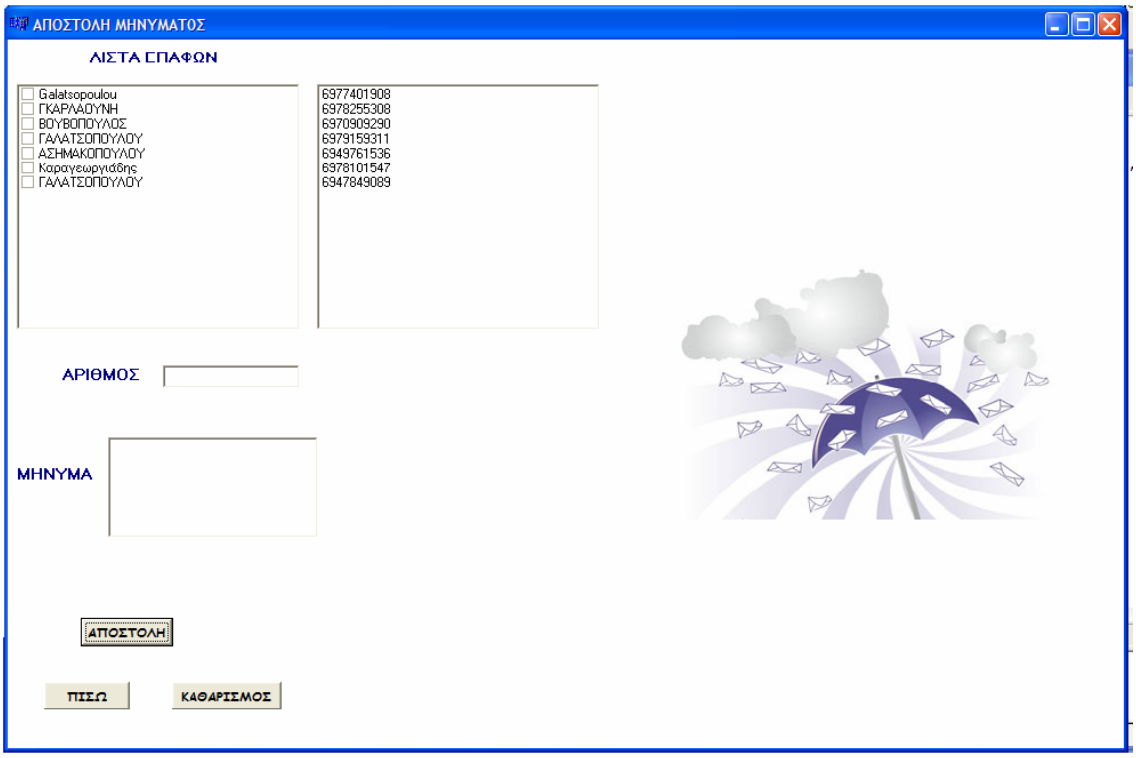
ΔΙΑΓΡΑΦΗ ΕΠΑΦΗΣ

Πατώντας το κουμπί ΔΙΑΓΡΑΦΗ ΕΠΑΦΗΣ εμφανίζεται η φόρμα για να διαγράψουμε μια επαφή δίνοντας το όνομα και το επίθετο της επαφής που θέλουμε να διαγράψουμε.



ΑΠΟΣΤΟΛΗ ΜΗΝΥΜΑΤΟΣ

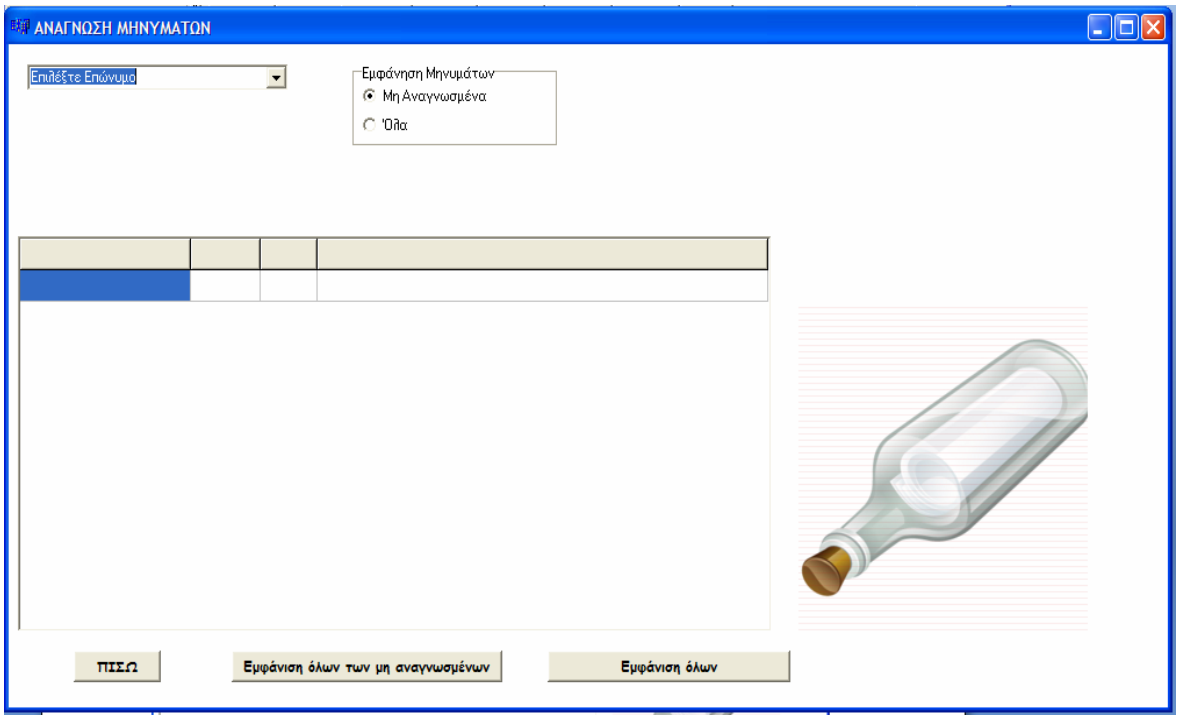
Πατώντας το κουμπί ΑΠΟΣΤΟΛΗ ΜΗΝΥΜΑΤΟΣ εμφανίζεται η φόρμα



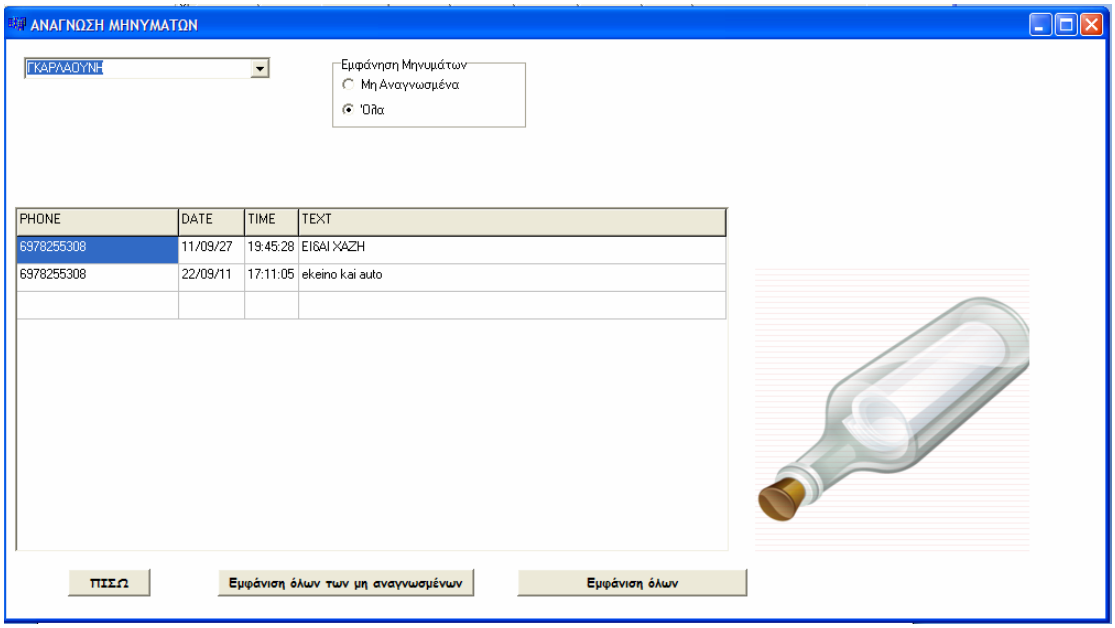
Στην ΛΙΣΤΑ ΕΠΑΦΩΝ εμφανίζονται τα επίθετα των επαφών που έχουμε αποθηκευμένα στην βάση δεδομένων και δίπλα ο αριθμός τηλεφώνου κάθε επαφής. Για να στείλουμε μήνυμα σε μία ή περισσότερες επαφές απλώς τσεκάρουμε στα κουτάκια. Σε περίπτωση που θέλουμε να στείλουμε μήνυμα σ' έναν αριθμό που δεν υπάρχει στις επαφές μας γράφουμε τον αριθμό στο πεδίο ΑΡΙΘΜΟΣ. Έπειτα γράφουμε το κείμενο στο πεδίο ΜΗΝΥΜΑ και πατάμε το κουμπί ΑΠΟΣΤΟΛΗ. Όταν το μήνυμα αποσταλεί εμφανίζεται μήνυμα επιτυχίας.

ΑΝΑΓΝΩΣΗ ΜΗΝΥΜΑΤΩΝ

Πατώντας το κουμπί ΑΝΑΓΝΩΣΗ ΜΗΝΥΜΑΤΩΝ εμφανίζεται η φόρμα



Επιλέγουμε το επώνυμο της επαφής από της οποίας θέλουμε να διαβάσουμε τα μηνύματα, στο πεδίο Επιλέξτε Επώνυμο και από την Εμφάνιση μηνυμάτων διαλέγουμε Μη Αναγνωσμένα αν θέλουμε να διαβάσουμε τα καινούρια μηνύματα ή διαλέγουμε Όλα αν θέλουμε να διαβάσουμε καινούρια και παλιά μηνύματα της επαφής. Στην λίστα εμφανίζονται ο αριθμός της επαφής η ημερομηνία, η ώρα και το μήνυμα.



Πατώντας το κουμπί **Εμφάνιση όλων των μη αναγνωσμένων** εμφανίζονται όλα τα μη αναγνωσμένα μηνύματα από της επαφές μας αλλά και από άλλους αριθμούς που δεν είναι αποθηκευμένοι στη βάση δεδομένων. Το ίδιο και με το κουμπί **Εμφάνιση όλων** με την διαφορά ότι σ' αυτήν την περίπτωση εμφανίζονται και τα διαβασμένα.

ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ 1

ΟΙ ΚΛΑΣΕΙΣ

Η Contact:

```
#pragma hdrstop
#include "Contact.h"


#pragma package(smart_init)

Contact::Contact()
{
  name_="";
  surname_="";
  telNumber_="";
}
```

```
Contact::Contact(string name,string surname,string telNumber)
{
  name_=name;
  surname_=surname;
  telNumber_=telNumber;
}

Contact::Contact(string id,string name,string surname,string telNumber)
{
  id_=id;
  name_=name;
  surname_=surname;
  telNumber_=telNumber;
}

void Contact::setId(string id)
{
    id_=id;
}

string Contact::getId()
{
   return id_;
}

void Contact::setNameContact(string name)
{
    name_=name;
```

```
}

string Contact::getNameContact()
{
  return name_;
}

void Contact::setSurnameContact(string surname)
{
    surname_=surname;
}


string Contact::getSurnameContact()
{
  return surname_;
}

void Contact::setTelNumberContact(string telNumber)
{
   telNumber_=telNumber;
}
 string Contact::getTelNumberContact()
{
  return telNumber_;
}
```

Η Message:

```
#pragma hdrstop
```

```
#include "Message.h"

#pragma package(smart_init)
Message::Message()
{
  tel_="";
  text_="";
  date_="";
  time_="";
}

Message::Message(string tel,string text,string date,string time)

{
    tel_=tel;
    text_=text;
    date_=date;
    time_=time;
    }

void Message::setMessageTel(string tel)
{
   tel_=tel;
}

string Message::getMessageTel()
{
   return tel_;
}
```

```
void Message::setMessageText(string text)
{
  text_=text;
}

string Message::getMessageText()
{
  return text_;
}

void Message::setMessageDate(string date)
{
  date_=date;
}

string Message::getMessageDate()
{
  return date_;
}

void Message::setMessageTime(string time)
{
  time_=time;
}

string Message::getMessageTime()
{
  return time_;
```

```
}
```

Η ContactDb:

```
#pragma hdrstop
#include "ContactDB.h"

#pragma package(smart_init)

bool ContactDb::InsertContact(string telephone,string name,string surname)
{
  SQLHANDLE EnvHandle;

  SQLHANDLE ConHandle;
  SQLHANDLE StmtHandle;
  SQLRETURN rc;

  string dbName_="sms1";
  bool value=false;

  rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &EnvHandle);

    if (rc != SQL_SUCCESS) { //Failed to allocate environment handle
      value=false;
    }

    // Set The ODBC Application Version To 3.x
    rc = SQLSetEnvAttr(EnvHandle, SQL_ATTR_ODBC_VERSION,
                       (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_UINTEGER);
```

```
if (rc == SQL_SUCCESS) { //Environment properties were set succesfully

 rc = SQLAllocHandle(SQL_HANDLE_DBC, EnvHandle, &ConHandle);

 if (rc == SQL_SUCCESS) { //Connection handle allocated succesfully


  rc = SQLConnect(ConHandle, (SQLTCHAR *) "sms1", SQL_NTS, (SQLCHAR *)
"exposyst-qzwk6c", SQL_NTS, (SQLCHAR *) "", SQL_NTS);

  if (rc != SQL_ERROR) { //Connection was made succesfully


 rc = SQLAllocHandle(SQL_HANDLE_STMT, ConHandle, &StmtHandle);

   if (rc != SQL_ERROR) { // Statement handle allocated succesfully

string sqlString=" INSERT INTO CATALOG
(CATALOG_TEL1,CATALOG_NAME,CATALOG_SURNAME) VALUES ('"
   + telephone + "','"
   + name + "','"
   + surname +"');";
rc = SQLExecDirect(StmtHandle, (SQLTCHAR *)sqlString.c_str(), SQL_NTS);

    if ((rc  = SQLFetch(StmtHandle))!= SQL_NO_DATA) {

        value=true;
```

```
        }
        SQLFreeHandle(SQL_HANDLE_STMT, StmtHandle);
       }

       SQLDisconnect(ConHandle);
      }

      SQLFreeHandle(SQL_HANDLE_DBC, ConHandle);
     }
    }
    SQLFreeHandle(SQL_HANDLE_ENV, EnvHandle);
     return value;
  }

bool ContactDb::DeleteContact(string name,string surname)
{

      SQLHANDLE EnvHandle;
      SQLHANDLE ConHandle;
      SQLHANDLE StmtHandle;
      SQLRETURN rc;
       bool value=false;
       string dbName_="sms1";



       // Allocate An Environment Handle
      rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &EnvHandle);
```

```
if (rc != SQL_SUCCESS) { //Failed to allocate environment handle
   value=false;
}

// Set The ODBC Application Version To 3.x
rc = SQLSetEnvAttr(EnvHandle, SQL_ATTR_ODBC_VERSION,
                   (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_UINTEGER);

if (rc == SQL_SUCCESS) { //Environment properties were set succesfully

  rc = SQLAllocHandle(SQL_HANDLE_DBC, EnvHandle, &ConHandle);

  if (rc == SQL_SUCCESS) { //Connection handle allocated succesfully

rc = SQLConnect(ConHandle, (SQLTCHAR *) "sms1", SQL_NTS, (SQLCHAR *)
"exposyst-qzwk6c", SQL_NTS, (SQLCHAR *) "", SQL_NTS);

if (rc != SQL_ERROR) { //Connection was made succesfully

  rc = SQLAllocHandle(SQL_HANDLE_STMT, ConHandle, &StmtHandle);

  if (rc != SQL_ERROR) { // Statement handle allocated succesfully
        string sqlString = "DELETE FROM CATALOG WHERE
CATALOG_NAME='" + name +"' AND CATALOG_SURNAME='" + surname +"'";

        rc = SQLExecDirect(StmtHandle, (SQLTCHAR *)sqlString.c_str(),
SQL_NTS);

        if ((rc  = SQLFetch(StmtHandle))!= SQL_NO_DATA) {
            value=true;
        }
```

```
            SQLFreeHandle(SQL_HANDLE_STMT, StmtHandle); //Free statement handle
        }
        SQLDisconnect(ConHandle); //Drop connection
      }

      SQLFreeHandle(SQL_HANDLE_DBC, ConHandle); //Free connection handle
        }
      }

      SQLFreeHandle(SQL_HANDLE_ENV, EnvHandle); //Free environment handle
         return value;

}
      vector<Contact> ContactDb::read()
{
    vector<Contact> contacts;

    char id[255];
    char name[255];
    char surname[255];
    char telNumber[255];

    SQLHANDLE EnvHandle;
    SQLHANDLE ConHandle;
    SQLHANDLE StmtHandle;
    SQLRETURN rc;

    string dbName_="sms1";
```

```
// Allocate An Environment Handle
rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &EnvHandle);

if (rc != SQL_ERROR) { //Failed to allocate environment handle

}

  // Set The ODBC Application Version To 3.x
rc = SQLSetEnvAttr(EnvHandle, SQL_ATTR_ODBC_VERSION,
                (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_UINTEGER);

if (rc != SQL_ERROR) { //Environment properties were set succesfully

  rc = SQLAllocHandle(SQL_HANDLE_DBC, EnvHandle, &ConHandle);

  if (rc != SQL_ERROR) { //Connection handle allocated succesfully



   rc = SQLConnect(ConHandle, (SQLTCHAR *) "sms1", SQL_NTS, (SQLCHAR *)
"exposyst-qzwk6c", SQL_NTS, (SQLCHAR *) "", SQL_NTS);

   if (rc != SQL_ERROR) { //Connection was made succesfully

     rc = SQLAllocHandle(SQL_HANDLE_STMT, ConHandle, &StmtHandle);

     if (rc != SQL_ERROR) { // Statement handle allocated succesfully

       string sqlString = "SELECT
CATALOG_ID,CATALOG_NAME,CATALOG_SURNAME,CATALOG_TEL1 FROM
CATALOG";
```

```
        rc = SQLExecDirect(StmtHandle, (SQLTCHAR *)sqlString.c_str(), SQL_NTS);


        if (rc == SQL_SUCCESS) {
          //We know that the 1st column of the query we sent is the id
          // the 2nd column is the name
          // the 3rd column is the surname and so on
          SQLBindCol(StmtHandle, 1, SQL_C_CHAR, (SQLPOINTER) id, sizeof(id), NULL);
          SQLBindCol(StmtHandle, 2, SQL_C_CHAR, (SQLPOINTER) name, sizeof(name), NULL);
          SQLBindCol(StmtHandle, 3, SQL_C_CHAR, (SQLPOINTER) surname, sizeof(surname), NULL);
        SQLBindCol(StmtHandle, 4, SQL_C_CHAR, (SQLPOINTER) telNumber, sizeof(telNumber), NULL);
        while((rc  = SQLFetch(StmtHandle))!= SQL_NO_DATA){
              string idString(id);
              string nameString(name);
              string surnameString(surname);
              string telNumberString(telNumber);
              Contact contact(idString,nameString,surnameString,telNumberString);
              contacts.push_back(contact);
          }
        }
        SQLFreeHandle(SQL_HANDLE_STMT, StmtHandle);
      }


      SQLDisconnect(ConHandle);
    }


    SQLFreeHandle(SQL_HANDLE_DBC, ConHandle);
    }
```

```
    }
    SQLFreeHandle(SQL_HANDLE_ENV, EnvHandle);
    return contacts;
}
```

Η MessageDb:

```
#pragma hdrstop
#include "MessageDb.h"

#pragma package(smart_init)


Message *MessageDb::ShowMessage()
 {
       SQLHANDLE EnvHandle;
       SQLHANDLE ConHandle;
       SQLHANDLE StmtHandle;
       SQLRETURN rc;

       char textMessage[255];
       char telMessage[255];
       char dateMessage[255];
       char timeMessage[255];
       Message *message=0;
       string dbName_="sms1";

       // Allocate An Environment Handle
    rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &EnvHandle);
```

```
if (rc != SQL_SUCCESS) { //Failed to allocate environment handle
  return NULL;
}

// Set The ODBC Application Version To 3.x
  rc = SQLSetEnvAttr(EnvHandle, SQL_ATTR_ODBC_VERSION,
                     (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_UINTEGER);

  if (rc == SQL_SUCCESS) { //Environment properties were set succesfully

    rc = SQLAllocHandle(SQL_HANDLE_DBC, EnvHandle, &ConHandle);

    if (rc == SQL_SUCCESS) { //Connection handle allocated succesfully


  rc = SQLConnect(ConHandle, (SQLTCHAR *) "sms1", SQL_NTS, (SQLCHAR *)
"exposyst-qzwk6c", SQL_NTS, (SQLCHAR *) "", SQL_NTS);

    if (rc != SQL_ERROR) { //Connection was made succesfully

      rc = SQLAllocHandle(SQL_HANDLE_STMT, ConHandle, &StmtHandle);

      if (rc != SQL_ERROR) { // Statement handle allocated succesfully
        string sqlString = "SELECT
MESSAGE_TEXT,MESSAGE_DATE,MESSAGE_TIME,MESSAGE_TEL FROM
MESSAGES WHERE MESSAGE_READ='0'";

      rc = SQLExecDirect(StmtHandle, (SQLTCHAR *)sqlString.c_str(), SQL_NTS);

       if (rc != SQL_ERROR) {
```

```
        SQLBindCol(StmtHandle, 1, SQL_C_CHAR, (SQLPOINTER) textMessage, sizeof(textMessage), NULL);

        SQLBindCol(StmtHandle, 2, SQL_C_CHAR, (SQLPOINTER) dateMessage, sizeof(dateMessage), NULL);

        SQLBindCol(StmtHandle, 3, SQL_C_CHAR, (SQLPOINTER) timeMessage, sizeof(timeMessage), NULL);

        SQLBindCol(StmtHandle, 4, SQL_C_CHAR, (SQLPOINTER) telMessage, sizeof(telMessage), NULL);


    if ((rc  = SQLFetch(StmtHandle))!= SQL_NO_DATA){

            string textMessageString(textMessage);




  message = new Message(telMessage,textMessageString,dateMessage,timeMessage);

        }

          }

      SQLFreeHandle(SQL_HANDLE_STMT, StmtHandle);

     }


      SQLDisconnect(ConHandle);

    }


    SQLFreeHandle(SQL_HANDLE_DBC, ConHandle);

   }

  }


  SQLFreeHandle(SQL_HANDLE_ENV, EnvHandle);


  return message;
```

```
}

bool MessageDb::InsertMessage(string smsDate,string smsTime,string
smsNumber,string smsText)
{
  SQLHANDLE EnvHandle;
  SQLHANDLE ConHandle;
  SQLHANDLE StmtHandle;
  SQLRETURN rc;
  string read,time,text,date,number;
  string dbName_="sms1";

 bool value=false;

  rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &EnvHandle);

    if (rc != SQL_SUCCESS) { //Failed to allocate environment handle
      value=false;
    }

    // Set The ODBC Application Version To 3.x
    rc = SQLSetEnvAttr(EnvHandle, SQL_ATTR_ODBC_VERSION,
                    (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_UINTEGER);

    if (rc == SQL_SUCCESS) { //Environment properties were set succesfully

      rc = SQLAllocHandle(SQL_HANDLE_DBC, EnvHandle, &ConHandle);

      if (rc == SQL_SUCCESS) { //Connection handle allocated succesfully
```

```
rc = SQLConnect(ConHandle, (SQLTCHAR *) "sms1", SQL_NTS, (SQLCHAR *) "exposyst-qzwk6c", SQL_NTS, (SQLCHAR *) "", SQL_NTS);

if (rc != SQL_ERROR) { //Connection was made succesfully

 rc = SQLAllocHandle(SQL_HANDLE_STMT, ConHandle, &StmtHandle);

 if (rc != SQL_ERROR) { // Statement handle allocated succesfully
```

```
string sqlString = "INSERT INTO MESSAGES (MESSAGE_TEXT,MESSAGE_DATE,MESSAGE_TIME,MESSAGE_TEL,MESSAGE_READ) VALUES ('"
        + smsText + "','"
        + smsDate + "','"
        + smsTime + "','"
         + smsNumber + "','"
        + "0" + "');";

 rc = SQLExecDirect(StmtHandle, (SQLTCHAR *)sqlString.c_str(), SQL_NTS);

 if ((rc  = SQLFetch(StmtHandle))!= SQL_NO_DATA) {

     value=true;

 }
 SQLFreeHandle(SQL_HANDLE_STMT, StmtHandle);
```

```
        }

        SQLDisconnect(ConHandle);
       }

       SQLFreeHandle(SQL_HANDLE_DBC, ConHandle);
      }
     }
     SQLFreeHandle(SQL_HANDLE_ENV, EnvHandle);
      return value;
}

bool MessageDb::setUnreadRead(string smsNumber)
{
  SQLHANDLE EnvHandle;
  SQLHANDLE ConHandle;
  SQLHANDLE StmtHandle;
  SQLRETURN rc;
  string read,time,text,date,number;
  string dbName_="sms1";
  bool value=false;

  rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &EnvHandle);

    if (rc != SQL_SUCCESS) { //Failed to allocate environment handle
      value=false;
    }

    // Set The ODBC Application Version To 3.x
```

```
rc = SQLSetEnvAttr(EnvHandle, SQL_ATTR_ODBC_VERSION,
                   (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_UINTEGER);

if (rc == SQL_SUCCESS) { //Environment properties were set succesfully

  rc = SQLAllocHandle(SQL_HANDLE_DBC, EnvHandle, &ConHandle);

  if (rc == SQL_SUCCESS) { //Connection handle allocated succesfully



   rc = SQLConnect(ConHandle, (SQLTCHAR *) "sms1", SQL_NTS, (SQLCHAR *)
"exposyst-qzwk6c", SQL_NTS, (SQLCHAR *) "", SQL_NTS);

   if (rc != SQL_ERROR) { //Connection was made succesfully

     rc = SQLAllocHandle(SQL_HANDLE_STMT, ConHandle, &StmtHandle);

     if (rc != SQL_ERROR) { // Statement handle allocated succesfully



      string sqlString = "UPDATE MESSAGES SET MESSAGE_READ='1' WHERE
MESSAGE_TEL='"+ smsNumber +"'";

rc = SQLExecDirect(StmtHandle, (SQLTCHAR *)sqlString.c_str(), SQL_NTS);

      if ((rc  = SQLFetch(StmtHandle))!= SQL_NO_DATA) {

          value=true;
```

```
        }
        SQLFreeHandle(SQL_HANDLE_STMT, StmtHandle);
       }

       SQLDisconnect(ConHandle);
      }

      SQLFreeHandle(SQL_HANDLE_DBC, ConHandle);
     }
    }
    SQLFreeHandle(SQL_HANDLE_ENV, EnvHandle);
     return value;
}

bool MessageDb::setAllUnreadRead()
{
  SQLHANDLE EnvHandle;
  SQLHANDLE ConHandle;
  SQLHANDLE StmtHandle;
  SQLRETURN rc;
  string read,time,text,date,number;
  string dbName_="sms1";
  bool value=false;

  rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &EnvHandle);

   if (rc != SQL_SUCCESS) { //Failed to allocate environment handle
     value=false;
   }
```

```
// Set The ODBC Application Version To 3.x
rc = SQLSetEnvAttr(EnvHandle, SQL_ATTR_ODBC_VERSION,
                   (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_UINTEGER);

if (rc == SQL_SUCCESS) { //Environment properties were set succesfully

  rc = SQLAllocHandle(SQL_HANDLE_DBC, EnvHandle, &ConHandle);

  if (rc == SQL_SUCCESS) { //Connection handle allocated succesfully



rc = SQLConnect(ConHandle, (SQLTCHAR *) "sms1", SQL_NTS, (SQLCHAR *)
"exposyst-qzwk6c", SQL_NTS, (SQLCHAR *) "", SQL_NTS);

   if (rc != SQL_ERROR) { //Connection was made succesfully

     rc = SQLAllocHandle(SQL_HANDLE_STMT, ConHandle, &StmtHandle);

     if (rc != SQL_ERROR) { // Statement handle allocated succesfully



      string sqlString = "UPDATE MESSAGES SET MESSAGE_READ='1' WHERE
MESSAGE_READ='0'";

rc = SQLExecDirect(StmtHandle, (SQLTCHAR *)sqlString.c_str(), SQL_NTS);

      if ((rc  = SQLFetch(StmtHandle))!= SQL_NO_DATA) {
```

```
            value=true;


          }
          SQLFreeHandle(SQL_HANDLE_STMT, StmtHandle);
        }

        SQLDisconnect(ConHandle);
      }

      SQLFreeHandle(SQL_HANDLE_DBC, ConHandle);
     }
   }
   SQLFreeHandle(SQL_HANDLE_ENV, EnvHandle);
    return value;
}


vector<Message> MessageDb::readUnreadMessage(string smsNumber)
{
   vector<Message> messages;

      SQLHANDLE EnvHandle;
      SQLHANDLE ConHandle;
      SQLHANDLE StmtHandle;
      SQLRETURN rc;

      char textMessage[255];
      char telMessage[255];
```

```
char dateMessage[255];
char timeMessage[255];
string dbName_="sms1";

// Allocate An Environment Handle
rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &EnvHandle);

if (rc != SQL_SUCCESS) { //Failed to allocate environment handle

}

// Set The ODBC Application Version To 3.x

rc = SQLSetEnvAttr(EnvHandle, SQL_ATTR_ODBC_VERSION,
                   (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_UINTEGER);

  if (rc == SQL_SUCCESS) { //Environment properties were set succesfully

    rc = SQLAllocHandle(SQL_HANDLE_DBC, EnvHandle, &ConHandle);

    if (rc == SQL_SUCCESS) { //Connection handle allocated succesfully


  rc = SQLConnect(ConHandle, (SQLTCHAR *) "sms1", SQL_NTS, (SQLCHAR *)
"exposyst-qzwk6c", SQL_NTS, (SQLCHAR *) "", SQL_NTS);

  if (rc != SQL_ERROR) { //Connection was made succesfully

    rc = SQLAllocHandle(SQL_HANDLE_STMT, ConHandle, &StmtHandle);
```

```
if (rc != SQL_ERROR) { // Statement handle allocated succesfully

string sqlString = "SELECT MESSAGE_TEXT,MESSAGE_DATE,MESSAGE_TIME,MESSAGE_TEL FROM MESSAGES WHERE MESSAGE_TEL='"+ smsNumber +"' AND MESSAGE_READ='0'";


rc = SQLExecDirect(StmtHandle, (SQLTCHAR *)sqlString.c_str(), SQL_NTS);


if (rc != SQL_ERROR)

{

SQLBindCol(StmtHandle, 1, SQL_C_CHAR, (SQLPOINTER) textMessage, sizeof(textMessage), NULL);

SQLBindCol(StmtHandle, 2, SQL_C_CHAR, (SQLPOINTER) dateMessage, sizeof(dateMessage), NULL);


SQLBindCol(StmtHandle, 3, SQL_C_CHAR, (SQLPOINTER) timeMessage, sizeof(timeMessage), NULL);

SQLBindCol(StmtHandle, 4, SQL_C_CHAR, (SQLPOINTER) telMessage, sizeof(telMessage), NULL);


while((rc = SQLFetch(StmtHandle))!= SQL_NO_DATA)

{

string textMessageString(textMessage);

string dateMessageString(dateMessage);

string timeMessageString(timeMessage);

string telMessageString(telMessage);

Message message(telMessageString,textMessageString,dateMessageString,timeMessageString);

messages.push_back(message);

}

}

SQLFreeHandle(SQL_HANDLE_STMT, StmtHandle);
```

```
        }

        SQLDisconnect(ConHandle);
      }

      SQLFreeHandle(SQL_HANDLE_DBC, ConHandle);
     }
    }

    SQLFreeHandle(SQL_HANDLE_ENV, EnvHandle);

    return messages;

}

vector<Message> MessageDb::readAllMessage(string smsNumber)
{
    vector<Message> messages;

      SQLHANDLE EnvHandle;
      SQLHANDLE ConHandle;
      SQLHANDLE StmtHandle;
      SQLRETURN rc;

      char textMessage[255];
      char telMessage[255];
      char dateMessage[255];
      char timeMessage[255];
      string dbName_="sms1";
```

```
// Allocate An Environment Handle
rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &EnvHandle);

if (rc != SQL_SUCCESS) { //Failed to allocate environment handle

}

// Set The ODBC Application Version To 3.x
  rc = SQLSetEnvAttr(EnvHandle, SQL_ATTR_ODBC_VERSION,
                     (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_UINTEGER);

  if (rc == SQL_SUCCESS) { //Environment properties were set succesfully

    rc = SQLAllocHandle(SQL_HANDLE_DBC, EnvHandle, &ConHandle);

    if (rc == SQL_SUCCESS) { //Connection handle allocated succesfully



  rc = SQLConnect(ConHandle, (SQLTCHAR *) "sms1", SQL_NTS, (SQLCHAR *)
"exposyst-qzwk6c", SQL_NTS, (SQLCHAR *) "", SQL_NTS);

  if (rc != SQL_ERROR) { //Connection was made succesfully

    rc = SQLAllocHandle(SQL_HANDLE_STMT, ConHandle, &StmtHandle);

    if (rc != SQL_ERROR) { // Statement handle allocated succesfully

      string sqlString = "SELECT
MESSAGE_TEXT,MESSAGE_DATE,MESSAGE_TIME,MESSAGE_TEL FROM
MESSAGES WHERE MESSAGE_TEL='"+ smsNumber +"'";
```

```
rc = SQLExecDirect(StmtHandle, (SQLTCHAR *)sqlString.c_str(), SQL_NTS);


if (rc != SQL_ERROR)

 {

    SQLBindCol(StmtHandle, 1, SQL_C_CHAR, (SQLPOINTER) textMessage,
sizeof(textMessage), NULL);

    SQLBindCol(StmtHandle, 2, SQL_C_CHAR, (SQLPOINTER)
dateMessage, sizeof(dateMessage), NULL);

    SQLBindCol(StmtHandle, 3, SQL_C_CHAR, (SQLPOINTER)
timeMessage, sizeof(timeMessage), NULL);

    SQLBindCol(StmtHandle, 4, SQL_C_CHAR, (SQLPOINTER) telMessage,
sizeof(telMessage), NULL);



while((rc  = SQLFetch(StmtHandle))!= SQL_NO_DATA)

    {

        string textMessageString(textMessage);

        string dateMessageString(dateMessage);

        string timeMessageString(timeMessage);

        string telMessageString(telMessage);

        Message
message(telMessageString,textMessageString,dateMessageString,timeMessageStri
ng);

        messages.push_back(message);

    }

 }

 SQLFreeHandle(SQL_HANDLE_STMT, StmtHandle);

}


 SQLDisconnect(ConHandle);

}
```

```
      SQLFreeHandle(SQL_HANDLE_DBC, ConHandle);
     }
    }

    SQLFreeHandle(SQL_HANDLE_ENV, EnvHandle);

    return messages;

}

vector<Message> MessageDb::readAllUnreadMessage()
{
    vector<Message> messages;

      SQLHANDLE EnvHandle;
      SQLHANDLE ConHandle;
      SQLHANDLE StmtHandle;
      SQLRETURN rc;

      char textMessage[255];
      char telMessage[255];
      char dateMessage[255];
      char timeMessage[255];
      string dbName_="sms1";

      // Allocate An Environment Handle
    rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &EnvHandle);

    if (rc != SQL_SUCCESS) { //Failed to allocate environment handle
```

```
}

// Set The ODBC Application Version To 3.x
  rc = SQLSetEnvAttr(EnvHandle, SQL_ATTR_ODBC_VERSION,
                     (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_UINTEGER);

  if (rc == SQL_SUCCESS) { //Environment properties were set succesfully

    rc = SQLAllocHandle(SQL_HANDLE_DBC, EnvHandle, &ConHandle);


if (rc == SQL_SUCCESS) { //Connection handle allocated succesfully

  rc = SQLConnect(ConHandle, (SQLTCHAR *) "sms1", SQL_NTS, (SQLCHAR *)
  "exposyst-qzwk6c", SQL_NTS, (SQLCHAR *) "", SQL_NTS);

  if (rc != SQL_ERROR) { //Connection was made succesfully

    rc = SQLAllocHandle(SQL_HANDLE_STMT, ConHandle, &StmtHandle);

    if (rc != SQL_ERROR) { // Statement handle allocated succesfully

      string sqlString = "SELECT
MESSAGE_TEXT,MESSAGE_DATE,MESSAGE_TIME,MESSAGE_TEL FROM
MESSAGES WHERE MESSAGE_READ='0'";

    rc = SQLExecDirect(StmtHandle, (SQLTCHAR *)sqlString.c_str(), SQL_NTS);

      if (rc != SQL_ERROR)
       {
```

```
SQLBindCol(StmtHandle, 1, SQL_C_CHAR, (SQLPOINTER) textMessage, sizeof(textMessage), NULL);

SQLBindCol(StmtHandle, 2, SQL_C_CHAR, (SQLPOINTER) dateMessage, sizeof(dateMessage), NULL);

SQLBindCol(StmtHandle, 3, SQL_C_CHAR, (SQLPOINTER) timeMessage, sizeof(timeMessage), NULL);

SQLBindCol(StmtHandle, 4, SQL_C_CHAR, (SQLPOINTER) telMessage, sizeof(telMessage), NULL);


while((rc  = SQLFetch(StmtHandle))!= SQL_NO_DATA)

{

string textMessageString(textMessage);

string dateMessageString(dateMessage);


 string timeMessageString(timeMessage)

string telMessageString(telMessage);

Message message(telMessageString,textMessageString,dateMessageString,timeMessageString);

messages.push_back(message);

}


  }


   SQLFreeHandle(SQL_HANDLE_STMT, StmtHandle);

 }


  SQLDisconnect(ConHandle);

}


SQLFreeHandle(SQL_HANDLE_DBC, ConHandle);

}
```

```
    }

    SQLFreeHandle(SQL_HANDLE_ENV, EnvHandle);

    return messages;

}

vector<Message> MessageDb::readAllAllMessage()
{
    vector<Message> messages;

      SQLHANDLE EnvHandle;
      SQLHANDLE ConHandle;
      SQLHANDLE StmtHandle;
      SQLRETURN rc;

      char textMessage[255];
      char telMessage[255];
      char dateMessage[255];
      char timeMessage[255];
      string dbName_="sms1";

      // Allocate An Environment Handle
    rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &EnvHandle);

    if (rc != SQL_SUCCESS) { //Failed to allocate environment handle

    }
```

```
// Set The ODBC Application Version To 3.x
   rc = SQLSetEnvAttr(EnvHandle, SQL_ATTR_ODBC_VERSION,
                      (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_UINTEGER);

   if (rc == SQL_SUCCESS) { //Environment properties were set succesfully

     rc = SQLAllocHandle(SQL_HANDLE_DBC, EnvHandle, &ConHandle);

     if (rc == SQL_SUCCESS) { //Connection handle allocated succesfully


   rc = SQLConnect(ConHandle, (SQLTCHAR *) "sms1", SQL_NTS, (SQLCHAR *)
"exposyst-qzwk6c", SQL_NTS, (SQLCHAR *) "", SQL_NTS);

   if (rc != SQL_ERROR) { //Connection was made succesfully

     rc = SQLAllocHandle(SQL_HANDLE_STMT, ConHandle, &StmtHandle);

     if (rc != SQL_ERROR) { // Statement handle allocated succesfully

       string sqlString = "SELECT
MESSAGE_TEXT,MESSAGE_DATE,MESSAGE_TIME,MESSAGE_TEL FROM
MESSAGES";

     rc = SQLExecDirect(StmtHandle, (SQLTCHAR *)sqlString.c_str(), SQL_NTS);

      if (rc != SQL_ERROR)
       {
          SQLBindCol(StmtHandle, 1, SQL_C_CHAR, (SQLPOINTER) textMessage,
sizeof(textMessage), NULL);
```

```
            SQLBindCol(StmtHandle, 2, SQL_C_CHAR, (SQLPOINTER) dateMessage, sizeof(dateMessage), NULL);
            SQLBindCol(StmtHandle, 3, SQL_C_CHAR, (SQLPOINTER) timeMessage, sizeof(timeMessage), NULL);
            SQLBindCol(StmtHandle, 4, SQL_C_CHAR, (SQLPOINTER) telMessage, sizeof(telMessage), NULL);

            while((rc  = SQLFetch(StmtHandle))!= SQL_NO_DATA)
            {
                  string textMessageString(textMessage);
                  string dateMessageString(dateMessage);
                  string timeMessageString(timeMessage);
                  string telMessageString(telMessage);

Message message(telMessageString,textMessageString,dateMessageString,
timeMessageString);
                  messages.push_back(message);
            }
          }
         SQLFreeHandle(SQL_HANDLE_STMT, StmtHandle);
       }

       SQLDisconnect(ConHandle);
     }

     SQLFreeHandle(SQL_HANDLE_DBC, ConHandle);
     }
   }

   SQLFreeHandle(SQL_HANDLE_ENV, EnvHandle);
```

```
    return messages;

}
```

H LogIn:

```
#pragma hdrstop
#include "LogIn.h"
#pragma package(smart_init)


BOOL LogIn::logIn(string username,string password)
{
    char id[255];
    string dbName_="sms1";
    bool value=false;



    SQLHANDLE EnvHandle;
    SQLHANDLE ConHandle;
    SQLHANDLE StmtHandle;
    SQLRETURN rc;

    // Allocate An Environment Handle
    rc = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &EnvHandle);

    if (rc != SQL_SUCCESS) { //Failed to allocate environment handle
      value=false;
    }
```

```
// Set The ODBC Application Version To 3.x
rc = SQLSetEnvAttr(EnvHandle, SQL_ATTR_ODBC_VERSION,
                (SQLPOINTER) SQL_OV_ODBC3, SQL_IS_UINTEGER);

if (rc == SQL_SUCCESS) { //Environment properties were set succesfully

  rc = SQLAllocHandle(SQL_HANDLE_DBC, EnvHandle, &ConHandle);

  if (rc == SQL_SUCCESS) { //Connection handle allocated succesfully



  rc = SQLConnect(ConHandle, (SQLTCHAR *)"sms1", SQL_NTS, (SQLCHAR *)
"exposyst-qzwk6c", SQL_NTS, (SQLCHAR *) "", SQL_NTS);

  if (rc != SQL_ERROR) { //Connection was made succesfully

    rc = SQLAllocHandle(SQL_HANDLE_STMT, ConHandle, &StmtHandle);

    if (rc != SQL_ERROR) { // Statement handle allocated succesfully

      string sqlString = "SELECT USERS_ID FROM USERS WHERE
USERS_USERNAME = '" + username + "' AND USERS_PASSWORD='" + password
+"'";

      rc = SQLExecDirect(StmtHandle, (SQLTCHAR *)sqlString.c_str(), SQL_NTS);

      if (rc != SQL_ERROR) {
        SQLBindCol(StmtHandle, 1, SQL_C_CHAR, (SQLPOINTER) id, sizeof(id),
NULL);
        if ((rc  = SQLFetch(StmtHandle))!= SQL_NO_DATA){
```

```
            value = true;
            }
         }
         SQLFreeHandle(SQL_HANDLE_STMT, StmtHandle);
        }

        SQLDisconnect(ConHandle);
      }

      SQLFreeHandle(SQL_HANDLE_DBC, ConHandle);
      }
    }

    SQLFreeHandle(SQL_HANDLE_ENV, EnvHandle);
    return value;
}
```

Οι κλάσεις ContactDb, MessageDb και LogIn περιέχουν συναρτήσεις για την σύνδεση με την βάση δεδομένων.

ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ 2

ΟΙ ΦΟΡΜΕΣ

Η φόρμα LOGIN:

```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "LogIn.h"
```

```
#include "Unit2.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TLoginForm *LoginForm;
__fastcall TLoginForm::TLoginForm(TComponent* Owner)
	: TForm(Owner)
{

}

void __fastcall TLoginForm::FormHide(TObject *Sender)

{
	usernameEdit->Text="";
	passwordEdit->Text="";
}

void __fastcall TLoginForm::loginButtonClick(TObject *Sender)
{
	bool isUser;
	LogIn log;
	isUser=log.logIn(usernameEdit->Text.c_str(),passwordEdit->Text.c_str());
	if(isUser)
	{
		LoginForm->Hide();
		MenuForm->Show();
```

```
        }
}
```

Η φόρμα MENU:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit2.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"

#include "Unit7.h"
#include "Unit8.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TMenuForm *MenuForm;

__fastcall TMenuForm::TMenuForm(TComponent* Owner)
        : TForm(Owner)
{
}

void __fastcall TMenuForm::CreateContactClick(TObject *Sender)
{
  CreateContForm->Show();
  MenuForm->Hide();
}
```

```
void __fastcall TMenuForm::DeleteContactClick(TObject *Sender)
{
  DeleteContForm->Show();
  MenuForm->Hide();
}

void __fastcall TMenuForm::ConnectionClick(TObject *Sender)
{
  ConnectForm->Show();
  MenuForm->Hide();
}

void __fastcall TMenuForm::sendMessegeClick(TObject *Sender)
{
   SendMesForm->Show();
   MenuForm->Hide();
}

void __fastcall TMenuForm::Button1Click(TObject *Sender)
{
     ReadMesForm->Show();
     MenuForm->Hide();
}
```

Η φόρμα ΔΗΜΙΟΥΡΓΙΑ ΕΠΑΦΗΣ:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit3.h"
```

```
#include "Unit2.h"
#include "ContactDB.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TCreateContForm *CreateContForm;

__fastcall TCreateContForm::TCreateContForm(TComponent* Owner)
        : TForm(Owner)
{
}

void __fastcall TCreateContForm::InsertContactClick(TObject *Sender)
{
   bool checkInsert;
  ContactDb contactdb;
  checkInsert=contactdb.InsertContact(Edit1->Text.c_str(),Edit2->Text.c_str(),Edit3-
>Text.c_str());
  if(checkInsert)
  {
          ShowMessage("Η ΕΠΑΦΗ ΑΠΟΘΗΚΕΥΤΙΚΕ");
       }
       else
       {
          ShowMessage("ΑΠΟΤΥΧΙΑ ΑΠΟΘΗΚΕΥΣΗΣ");
       }

}
```

```
void __fastcall TCreateContForm::BackClick(TObject *Sender)
{
  MenuForm->Show();
  CreateContForm->Hide();
}
```

Η φόρμα ΔΙΑΓΡΑΦΗ ΕΠΑΦΗΣ:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit4.h"
#include "Unit2.h"
#include "ContactDb.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TDeleteContForm *DeleteContForm;

__fastcall TDeleteContForm::TDeleteContForm(TComponent* Owner)
      : TForm(Owner)
{
}

void __fastcall TDeleteContForm::DeleteContactClick(TObject *Sender)
{
   bool checkDelete;
  ContactDb contactdb1;
  checkDelete=contactdb1.DeleteContact(Edit1->Text.c_str(),Edit2->Text.c_str());
  if (checkDelete)
      {
```

```
        ShowMessage("Η ΕΠΑΦΗ ΔΙΕΓΡΑΦΗ!");
      }
      else
      {
        ShowMessage("ΑΠΟΤΥΧΙΑ ΔΙΑΓΡΑΦΗΣ!");
      }

}

void __fastcall TDeleteContForm::BackClick(TObject *Sender)
{

    MenuForm->Show();
     DeleteContForm->Hide();
}
```

Η φόρμα ΣΥΝΔΕΣΗ:

```
#include <vcl.h>
#pragma hdrstop
#include <string>
#include "Unit5.h"
#include "Unit6.h"
#include "Unit2.h"
#include "Contact.h"
#include "Message.h"
#include "MessageDb.h"
#include "ContactDB.h"

HANDLE hComm = NULL;
```

```
TRead *ReadThread;
COMMTIMEOUTS ctmoNew = {0}, ctmoOld;

using namespace std;
#pragma package(smart_init)
#pragma resource "*.dfm"
TConnectForm *ConnectForm;

__fastcall TConnectForm::TConnectForm(TComponent* Owner)
        : TForm(Owner)
{

}

void __fastcall TConnectForm::ConectionClick(TObject *Sender)
{
     string conf,bitSec,bitData,parity,bitStop;
     bitSec=ComboBox2->Text.c_str();
     bitData=ComboBox3->Text.c_str();
     parity=Edit4->Text.c_str();
     bitStop=ComboBox4->Text.c_str();
     conf="\""+bitSec+","+bitData+","+parity+","+bitStop+"\"";

      DCB dcbCommPort;
       hComm = CreateFile(ComboBox1->Text.c_str(),
                  GENERIC_READ | GENERIC_WRITE,
                  0,
                  0,
                  OPEN_EXISTING,
```

```
            0,
            0);
    if(hComm == INVALID_HANDLE_VALUE)
    {
        ShowMessage("ΑΠΟΤΥΧΙΑ ΣΥΝΔΕΣΗΣ!");
        Application->Terminate();
    }
    else
    {
       ShowMessage("ΣΥΝΔΕΘΗΚΕ!");
        MenuForm->sendMessege->Enabled=true;

    }
    GetCommTimeouts(hComm,&ctmoOld);
    ctmoNew.ReadTotalTimeoutConstant = 100;
    ctmoNew.ReadTotalTimeoutMultiplier = 0;
    ctmoNew.WriteTotalTimeoutMultiplier = 0;
    ctmoNew.WriteTotalTimeoutConstant = 0;
    SetCommTimeouts(hComm, &ctmoNew);

    dcbCommPort.DCBlength = sizeof(DCB);
    GetCommState(hComm, &dcbCommPort);
    BuildCommDCB( conf.c_str(), &dcbCommPort);
    SetCommState(hComm, &dcbCommPort);

    ReadThread = new TRead(false);
}
```

```
void __fastcall TConnectForm::BackClick(TObject *Sender)
{
        MenuForm->Show();
        ConnectForm->Hide();
}
```

Η φόρμα ΑΠΟΣΤΟΛΗ ΜΗΝΥΜΑΤΟΣ:

```
#include <vcl.h>

#pragma hdrstop
#include "Unit5.h"
#include "time.h"
#include "Unit6.h"
#include "Unit7.h"
#include "Unit2.h"
#include "Contact.h"
#include "Message.h"
#include "MessageDb.h"
#include "ContactDB.h"

extern HANDLE hComm;

#pragma package(smart_init)
#pragma resource "*.dfm"
TSendMesForm *SendMesForm;
```

```
__fastcall TSendMesForm::TSendMesForm(TComponent* Owner)
        : TForm(Owner)
{
}

void __fastcall TSendMesForm::sendClick(TObject *Sender)
{
   int i,k;
   time_t start_time,cur_time;
   if(Edit1->Text.IsEmpty()==false)
   {

    char ch[3]="AT\r";
     char chh[10]="AT+CMGF=1\r";
     char ch1[500]="AT+CMGS=\"";
     char ch2[10]="AT+CMGD=4\r";
     strcat(ch1,Edit1->Text.c_str());
     strcat(ch1,"\"\r");
     strcat(ch1,Memo2->Text.c_str());
     strcat(ch1,"\032");

       for(i=0;i<sizeof(ch);i++)
              {
                     TransmitCommChar(hComm,ch[i]);
                     ch[i]=0;
              }
              time(&start_time);
              do
              {
```

```
            time(&cur_time);
        }
        while((cur_time-start_time)<3);

        for(i=0;i<sizeof(chh);i++)
        {
            TransmitCommChar(hComm,chh[i]);
            chh[i]=0;
        }
        time(&start_time);
        do

{
            time(&cur_time);
 }
        while((cur_time-start_time)<3);
        for(i=0;i<sizeof(ch1);i++)
        {
            TransmitCommChar(hComm,ch1[i]);
            ch1[i]=0;
        }
        time(&start_time);
        do
        {
            time(&cur_time);
        }
        while((cur_time-start_time)<3);
}
```

```
for(k=0;k<CheckListBox1->Items->Count;k++)
{

    int i=0;
    if(CheckListBox1->Checked[k]==True)
    {
        char ch[3]="AT\r";
        char chh[10]="AT+CMGF=1\r";
        char ch1[500]="AT+CMGS=\"";
        //char ch2[10]="AT+CMGD=4\r";
        strcat(ch1,ListBox2->Items->Strings[k].c_str());

        strcat(ch1,"\"\r");
        strcat(ch1,Memo2->Text.c_str());
        strcat(ch1,"\032");



        for(i=0;i<sizeof(ch);i++)
        {
            TransmitCommChar(hComm,ch[i]);
            ch[i]=0;
        }
        time(&start_time);
        do
        {
            time(&cur_time);
        }
        while((cur_time-start_time)<3);
```

```
        for(i=0;i<sizeof(chh);i++)
        {
            TransmitCommChar(hComm,chh[i]);
            chh[i]=0;
        }
        time(&start_time);
        do
        {
            time(&cur_time);
        }

        while((cur_time-start_time)<3);
        for(i=0;i<sizeof(ch1);i++)
        {
            TransmitCommChar(hComm,ch1[i]);
            ch1[i]=0;
        }
        time(&start_time);
        do
        {
            time(&cur_time);
        }
        while((cur_time-start_time)<3);


    }
}
    ShowMessage("ΜΗΝΥΜΑ ΕΣΤΑΛΗ");
```

```
 }


void __fastcall TSendMesForm::backClick(TObject *Sender)
{
  MenuForm->Show();
  SendMesForm->Hide();
}

void __fastcall TSendMesForm::Button2Click(TObject *Sender)
{

   Edit1->Clear();
   Memo2->Clear();
}


void __fastcall TSendMesForm::Memo1Change(TObject *Sender)
{
        MessageDb newSms;
       char *fir,*sec,*thi,*fou,*ffir;

 if((strstr(Memo1->Lines->Strings[Memo1->Lines->Count-
1].c_str(),"OK"))&&(strstr(Memo1->Lines->Strings[Memo1->Lines->Count-
4].c_str(),"+CMGR:")))
       {
             Memo3->Lines->Add(Memo1->Lines->Strings[Memo1->Lines->Count-4]);
             Memo3->Lines->Add(Memo1->Lines->Strings[Memo1->Lines->Count-3]);
             AnsiString dok(Memo1->Lines->Strings[Memo1->Lines->Count-4].c_str());
             fir=strtok(dok.c_str(),",");
```

```
            sec=strtok(NULL,",");
            thi=strtok(NULL,",");
            fou=strtok(NULL,",");
            AnsiString ena(fir);
            AnsiString fff(sec);
            AnsiString tthi(thi);
            AnsiString ffou(fou);
            Memo3->Lines->Add(fff.SubString(4,fff.Length()-4).c_str());
            Memo3->Lines->Add(tthi.SubString(2,tthi.Length()).c_str());
            Memo3->Lines->Add(ffou.SubString(1,ffou.Length()-3).c_str());



newSms.InsertMessage(tthi.SubString(2,tthi.Length()).c_str(),ffou.SubString(1,ffou.Le
ngth()-3).c_str(),fff.SubString(4,fff.Length()-4).c_str(),Memo1->Lines-
>Strings[Memo1->Lines->Count-3].c_str());


        }


}


void __fastcall TSendMesForm::FormCreate(TObject *Sender)
{
    vector<Contact>::iterator i;
    vector<Contact> cont;
    Contact contact;
    ContactDb a;
    int vectorSize=0;
    int j=1;


    cont=a.read();
```

```
    vectorSize=cont.size();
    i = cont.begin();
    Contact conta = *i;
    while (i != cont.end())
    {
      Contact conta = *i;
      CheckListBox1->Items->Add(conta.getSurnameContact().c_str());
      ListBox2->Items->Add(conta.getTelNumberContact().c_str());
      i++;
      j++;


  }

}
```

Η φόρμα ΑΝΑΓΝΩΣΗ ΜΗΝΥΜΑΤΩΝ:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit8.h"
#include "MessageDb.h"
#include "Unit2.h"
#include "Contact.h"
#include "Message.h"
#include "MessageDb.h"
#include "ContactDB.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TReadMesForm *ReadMesForm;
```

```
__fastcall TReadMesForm::TReadMesForm(TComponent* Owner)
        : TForm(Owner)
{
}

void __fastcall TReadMesForm::FormCreate(TObject *Sender)
{
    vector<Contact>::iterator i;
    vector<Contact> cont;
    Contact contact;

    ContactDb a;
    int vectorSize=0;
    int j=1;

    cont=a.read();
    vectorSize=cont.size();
    i = cont.begin();
    Contact conta = *i;
    while (i != cont.end())
    {
      Contact conta = *i;
      ComboBox1->Items->Add(conta.getSurnameContact().c_str());
      ListBox1->Items->Add(conta.getTelNumberContact().c_str());
      i++;
      j++;

    }
```

```
}

void __fastcall TReadMesForm::ComboBox1Select(TObject *Sender)
{

        vector<Message>::iterator i;
        vector<Message> mess;
        Message messa;
        MessageDb a;
        int vectorSize=0;
        int maxx,maxy;

 int j=1;

        if(RadioGroup1->ItemIndex==0)
        {
                mess=a.readUnreadMessage(ListBox1->Items->Strings[ComboBox1-
>ItemIndex].c_str());
                if(mess.empty()==false)
                {
                        maxx = StringGrid1->ColCount;
                        maxy = StringGrid1->RowCount;
                        for (int y=0; y<maxy; ++y)
                        {
                                for (int x=0; x<maxx; ++x)
                                {
                                        StringGrid1->Cells[x][y] = "";
                                }
                        }
                        StringGrid1->Cells[0][0]="PHONE";
```

```
            StringGrid1->Cells[1][0]="DATE";
            StringGrid1->Cells[2][0]="TIME";
            StringGrid1->Cells[3][0]="TEXT";
            vectorSize=mess.size();
            StringGrid1->RowCount=vectorSize+1;
            i = mess.end();
            i--;
            Message messa = *i;
             while (i != mess.begin())
            {
                    Message messa = *i;

                  StringGrid1->Cells[0][j]=messa.getMessageTel().c_str();
                  StringGrid1->Cells[1][j]=messa.getMessageDate().c_str();
                  StringGrid1->Cells[2][j]=messa.getMessageTime().c_str();
                  StringGrid1->Cells[3][j]=messa.getMessageText().c_str();
                  i--;
                  j++;
            }

a.setUnreadRead(ListBox1->Items->Strings[ComboBox1->ItemIndex].c_str());
      }
      else
      {
            maxx = StringGrid1->ColCount;
            maxy = StringGrid1->RowCount;
            for (int y=0; y<maxy; ++y)
            {
                  for (int x=0; x<maxx; ++x)
```

```
                    {
                        StringGrid1->Cells[x][y] = "";
                    }
                }
                StringGrid1->RowCount=2;
                StringGrid1->Cells[0][0]="PHONE";
                StringGrid1->Cells[1][0]="DATE";
                StringGrid1->Cells[2][0]="TIME";
                StringGrid1->Cells[3][0]="TEXT";
                StringGrid1->Cells[0][1]="";
                StringGrid1->Cells[1][1]="";

                StringGrid1->Cells[2][1]="";
                StringGrid1->Cells[3][1]="";
            }
        }
        else if(RadioGroup1->ItemIndex==1)
        {
mess=a.readAllMessage(ListBox1->Items->Strings[ComboBox1->ItemIndex].c_str());
            if(mess.empty()==false)
            {
                maxx = StringGrid1->ColCount;
                maxy = StringGrid1->RowCount;
                for (int y=0; y<maxy; ++y)
                {
                    for (int x=0; x<maxx; ++x)
                    {
                        StringGrid1->Cells[x][y] = "";
                    }
```

```
        }
        StringGrid1->Cells[0][0]="PHONE";
        StringGrid1->Cells[1][0]="DATE";
        StringGrid1->Cells[2][0]="TIME";
        StringGrid1->Cells[3][0]="TEXT";
        vectorSize=mess.size();
        StringGrid1->RowCount=vectorSize+1;
        i = mess.end();
        i--;
        Message messa = *i;
                while (i != mess.begin())

    {
                Message messa = *i;
               StringGrid1->Cells[0][j]=messa.getMessageTel().c_str();
               StringGrid1->Cells[1][j]=messa.getMessageDate().c_str();
               StringGrid1->Cells[2][j]=messa.getMessageTime().c_str();
               StringGrid1->Cells[3][j]=messa.getMessageText().c_str();
               i--;
               j++;
        }

}
else
{
        maxx = StringGrid1->ColCount;
        maxy = StringGrid1->RowCount;
        for (int y=0; y<maxy; ++y)
        {
```

```
                    for (int x=0; x<maxx; ++x)
                    {
                        StringGrid1->Cells[x][y] = "";
                    }
                }
                StringGrid1->RowCount=2;
                StringGrid1->Cells[0][0]="PHONE";
                StringGrid1->Cells[1][0]="DATE";
                StringGrid1->Cells[2][0]="TIME";
                StringGrid1->Cells[3][0]="TEXT";
                StringGrid1->Cells[0][1]="";

                StringGrid1->Cells[1][1]="";
                StringGrid1->Cells[2][1]="";
                StringGrid1->Cells[3][1]="";
            }
        }
}

void __fastcall TReadMesForm::RadioGroup1Click(TObject *Sender)
{
        vector<Message>::iterator i;
        vector<Message> mess;
        Message messa;
        MessageDb a;
        int vectorSize=0;
        int maxx,maxy;
        int j=1;
```

```
 if(RadioGroup1->ItemIndex==0)
 {
mess=a.readUnreadMessage(ListBox1->Items->Strings[ComboBox1->ItemIndex].c_str());
        if(mess.empty()==false)
        {
            maxx = StringGrid1->ColCount;
            maxy = StringGrid1->RowCount;
            for (int y=0; y<maxy; ++y)
            {
                for (int x=0; x<maxx; ++x)
                {

                    StringGrid1->Cells[x][y] = "";
                }
            }
            StringGrid1->Cells[0][0]="PHONE";
            StringGrid1->Cells[1][0]="DATE";
            StringGrid1->Cells[2][0]="TIME";
            StringGrid1->Cells[3][0]="TEXT";
            vectorSize=mess.size();
            StringGrid1->RowCount=vectorSize+1;
            i = mess.begin();
            Message messa = *i;
                 while (i != mess.end())
            {
                 Message messa = *i;
                StringGrid1->Cells[0][j]=messa.getMessageTel().c_str();
                StringGrid1->Cells[1][j]=messa.getMessageDate().c_str();
                StringGrid1->Cells[2][j]=messa.getMessageTime().c_str();
```

```
                StringGrid1->Cells[3][j]=messa.getMessageText().c_str();
                i++;
                j++;
            }

            a.setUnreadRead(ListBox1->Items->Strings[ComboBox1->ItemIndex].c_str());
        }
        else
        {
            maxx = StringGrid1->ColCount;
            maxy = StringGrid1->RowCount;

            for (int y=0; y<maxy; ++y)
            {
                for (int x=0; x<maxx; ++x)
                {
                    StringGrid1->Cells[x][y] = "";
                }
            }
            StringGrid1->RowCount=2;
            StringGrid1->Cells[0][0]="PHONE";
            StringGrid1->Cells[1][0]="DATE";
            StringGrid1->Cells[2][0]="TIME";
            StringGrid1->Cells[3][0]="TEXT";
            StringGrid1->Cells[0][1]="";
            StringGrid1->Cells[1][1]="";
            StringGrid1->Cells[2][1]="";
            StringGrid1->Cells[3][1]="";
        }
```

```
    }
    else if(RadioGroup1->ItemIndex==1)
    {
mess=a.readAllMessage(ListBox1->Items->Strings[ComboBox1->ItemIndex].c_str());
        if(mess.empty()==false)
        {
            maxx = StringGrid1->ColCount;
            maxy = StringGrid1->RowCount;
            for (int y=0; y<maxy; ++y)
            {
                for (int x=0; x<maxx; ++x)

               {
                    StringGrid1->Cells[x][y] = "";
                }
            }
            StringGrid1->Cells[0][0]="PHONE";
            StringGrid1->Cells[1][0]="DATE";
            StringGrid1->Cells[2][0]="TIME";
            StringGrid1->Cells[3][0]="TEXT";
            vectorSize=mess.size();
            StringGrid1->RowCount=vectorSize+1;
            i = mess.end();
            i--;
            Message messa = *i;
                while (i != mess.begin())
            {
                Message messa = *i;
               StringGrid1->Cells[0][j]=messa.getMessageTel().c_str();
```

```
                StringGrid1->Cells[1][j]=messa.getMessageDate().c_str();
                StringGrid1->Cells[2][j]=messa.getMessageTime().c_str();
                StringGrid1->Cells[3][j]=messa.getMessageText().c_str();
                i--;
                j++;
        }

}
else
{
        maxx = StringGrid1->ColCount;

        maxy = StringGrid1->RowCount;
        for (int y=0; y<maxy; ++y)
        {
                for (int x=0; x<maxx; ++x)
                {
                        StringGrid1->Cells[x][y] = "";
                }
        }
        StringGrid1->RowCount=2;
        StringGrid1->Cells[0][0]="PHONE";
        StringGrid1->Cells[1][0]="DATE";
        StringGrid1->Cells[2][0]="TIME";
        StringGrid1->Cells[3][0]="TEXT";
        StringGrid1->Cells[0][1]="";
        StringGrid1->Cells[1][1]="";
        StringGrid1->Cells[2][1]="";
        StringGrid1->Cells[3][1]="";
```

```
            }
        }
}

void __fastcall TReadMesForm::Button1Click(TObject *Sender)
{
 MenuForm->Show();
 ReadMesForm->Hide();
}

void __fastcall TReadMesForm::Button2Click(TObject *Sender)

{
        vector<Message>::iterator i;
        vector<Message> mess;
        Message messa;
        MessageDb a;
        int vectorSize=0;
        int maxx,maxy;
        int j=1;

        mess=a.readAllUnreadMessage();
        if(mess.empty()==false)
        {
                maxx = StringGrid1->ColCount;
                maxy = StringGrid1->RowCount;
                for (int y=0; y<maxy; ++y)
                {
                        for (int x=0; x<maxx; ++x)
```

```
            {
                StringGrid1->Cells[x][y] = "";
            }
        }
        StringGrid1->Cells[0][0]="PHONE";
        StringGrid1->Cells[1][0]="DATE";
        StringGrid1->Cells[2][0]="TIME";
        StringGrid1->Cells[3][0]="TEXT";
        vectorSize=mess.size();
        StringGrid1->RowCount=vectorSize+1;
        i = mess.begin();

         Message messa = *i;
        while (i != mess.end())
        {
            Message messa = *i;
            StringGrid1->Cells[0][j]=messa.getMessageTel().c_str();
            StringGrid1->Cells[1][j]=messa.getMessageDate().c_str();
            StringGrid1->Cells[2][j]=messa.getMessageTime().c_str();
            StringGrid1->Cells[3][j]=messa.getMessageText().c_str();
            i--;
            j++;
        }

        a.setAllUnreadRead();
    }
    else
    {
        maxx = StringGrid1->ColCount;
```

```
        maxy = StringGrid1->RowCount;
        for (int y=0; y<maxy; ++y)
        {
            for (int x=0; x<maxx; ++x)
            {
                StringGrid1->Cells[x][y] = "";
            }
        }
        StringGrid1->RowCount=2;
        StringGrid1->Cells[0][0]="PHONE";
        StringGrid1->Cells[1][0]="DATE";

        StringGrid1->Cells[2][0]="TIME";
        StringGrid1->Cells[3][0]="TEXT";
        StringGrid1->Cells[0][1]="";
        StringGrid1->Cells[1][1]="";
        StringGrid1->Cells[2][1]="";
        StringGrid1->Cells[3][1]="";
    }


}


void __fastcall TReadMesForm::ReadMesFormClick(TObject *Sender)
{
    vector<Message>::iterator i;
    vector<Message> mess;
    Message messa;
```

```
MessageDb a;
int vectorSize=0;
int maxx,maxy;
int j=1;

mess=a.readAllAllMessage();
if(mess.empty()==false)
{
        maxx = StringGrid1->ColCount;
        maxy = StringGrid1->RowCount;
        for (int y=0; y<maxy; ++y)

            {
              for (int x=0; x<maxx; ++x)
              {
                    StringGrid1->Cells[x][y] = "";
              }
        }
        StringGrid1->Cells[0][0]="PHONE";
        StringGrid1->Cells[1][0]="DATE";
        StringGrid1->Cells[2][0]="TIME";
        StringGrid1->Cells[3][0]="TEXT";
        vectorSize=mess.size();
        StringGrid1->RowCount=vectorSize+1;
        i = mess.end();
        i--;
        Message messa = *i;
        while (i != mess.begin())
        {
```

```
            Message messa = *i;
            StringGrid1->Cells[0][j]=messa.getMessageTel().c_str();
            StringGrid1->Cells[1][j]=messa.getMessageDate().c_str();
            StringGrid1->Cells[2][j]=messa.getMessageTime().c_str();
            StringGrid1->Cells[3][j]=messa.getMessageText().c_str();
            i--;
            j++;
        }

 }
 else

{
        maxx = StringGrid1->ColCount;
        maxy = StringGrid1->RowCount;
        for (int y=0; y<maxy; ++y)
        {
            for (int x=0; x<maxx; ++x)
            {
                StringGrid1->Cells[x][y] = "";
            }
        }
        StringGrid1->RowCount=2;
        StringGrid1->Cells[0][0]="PHONE";
        StringGrid1->Cells[1][0]="DATE";
        StringGrid1->Cells[2][0]="TIME";
        StringGrid1->Cells[3][0]="TEXT";
        StringGrid1->Cells[0][1]="";
        StringGrid1->Cells[1][1]="";
```

```
            StringGrid1->Cells[2][1]="";
            StringGrid1->Cells[3][1]="";
        }

}
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ευάγγελος Γ. Ούτσιος, ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ-ΣΗΜΕΙΩΣΕΙΣ ΘΕΩΡΙΑΣ, Τμήμα Πληροφορικής και Επικοινωνιών, Σέρρες 2004

Σπύρος Καζαρλής, ΟΠΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ-ΣΗΜΕΙΩΣΕΙΣ ΕΡΓΑΣΤΗΡΙΟΥ, Τμήμα Πληροφορικής και Επικοινωνιών, Σέρρες 2003

Jarrod Hollingworth-Bob Swart-Mark Cashman-Paul Gustavson, «Borland C++Builder 6 Developer's Guide», Εκδόσεις SAMS, Δεκέμβριος 2002, 2η Έκδοση

Διαδικτυακοί τόποι που χρησιμοποιήθηκαν:

http://www.developershome.com/sms/howToSendSMSFromPC.asp