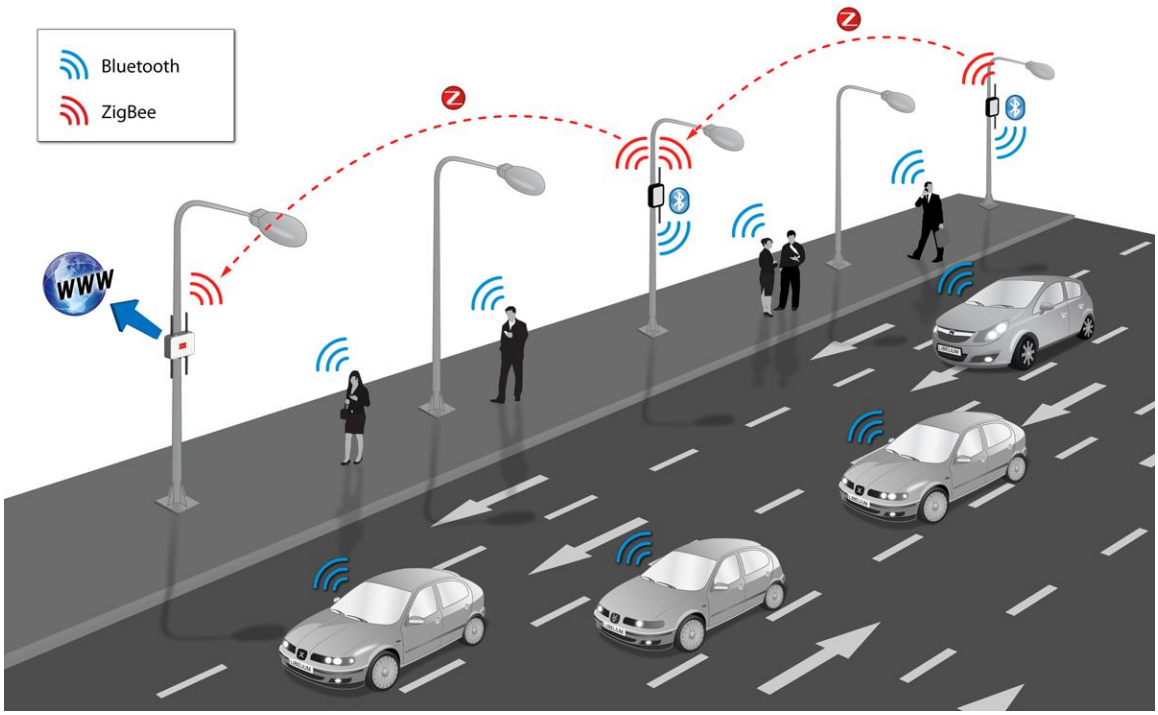
Τμήμα Μηχανικών Πληροφορικής
ΤΕΙ ΚΡΗΤΗΣ



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ασφαλής παρακολούθηση και διαχείριση έξυπνων αυτοκινήτων με χρήση ασύρματων αισθητήρων

ΦΟΙΤΗΤΕΣ:

Ρουμπάκης Κωνσταντίνος ΑΜ:3152

Μανουσάκης Στυλιανός ΑΜ:3224

ΔΙΔΑΣΚΩΝ:

Δρ. Μανιφάβας Χαράλαμπος

Δρ. Φυσαράκης Κωνσταντίνος

ΕΥΧΑΡΙΣΤΙΕΣ

Η πτυχιακή εργασία αυτή δημιουργήθηκε στα πλαίσια του προπτυχιακού προγράμματος σπουδών για την σχολή «Μηχανικών Πληροφορικής»(Πρώην τμήμα Εφαρμοσμένης Πληροφορικής και Πολυμέσων).

Θα θέλαμε πρώτα απ' όλα να ευχαριστήσουμε τους καθηγητές μας Δρ. Μανιφάβα Χαράλαμπο και Δρ. Φυσαράκη Κωνσταντίνο για την όλη υποστήριξη που μας παρείχαν μέχρι την τελική εκπόνηση της πτυχιακής μας εργασίας.

Στην συνέχεια θα θέλαμε να ευχαριστήσουμε τον φοιτητή Χουστουλάκη Νικόλαο για την πολύτιμη βοήθεια την οποία μας πρόσφερε μέσω την εμπειρίας του στην Java.

Φυσικά δεν θα μπορούσαμε να μην ευχαριστήσουμε τις οικογένειες μας οι οποίες μας στήριξαν ηθικά και μας έδωσαν την ώθηση να μην τα παρατάμε στα δύσκολα.

Περιεχόμενα

ΠΕΡΙΛΗΨΗ

Σήμερα ζούμε σε μια εποχή η οποία δοκιμάζεται καθημερινά από νέες τεχνολογικές προκλήσεις. Σε αυτές τις τεχνολογικές προκλήσεις προσπαθούν να ανταπεξέλθουν πολλοί τομείς όπως οι τηλεπικοινωνίες με την ανάπτυξη δικτυών νέας γενιάς(5G, Τηλεόραση μέσω LTE κλπ.), η βιομηχανική πληροφορική (συστήματα αυτομάτου ελέγχου, RFID κλπ), εξελιγμένη σχεδίαση προϊόντων(3D εκτυπωτές κλπ.). Ένας από τους κλάδους ο οποίος δέχεται συνεχώς νέες προκλήσεις είναι ο χώρος της αυτοκινητοβιομηχανίας.

Ας πάρουμε τα πράγματα λίγο από την αρχή. Στην αρχή της ιστορίας του χώρου του αυτοκίνητου είχαμε την ύπαρξη απλά του κινητήρα . Έπειτα είχαμε την προσθήκη των ζωνών ασφάλειας. Μετά τις ζώνες ασφάλειας μπήκε ο αερόσακος. Και μετά τον αερόσακο μπήκε το ABS το ποιο αφορά σε ένα πιο εξελιγμένο σύστημα φρένων. Και έτσι όσο περνούσαν τα χρονιά ολοένα και περισσότερες τεχνολογίες κυρίως για την ασφάλεια του οδηγού αρχίσαν να προστίθενται.

Και φτάσαμε στο σήμερα, οπού βοηθήματα και εξαρτήματα τα οποία ήταν προαιρετικά για ένα όχημα τείνουν να γίνουν υποχρεωτικά και πλήρως απαραίτητα. Παραδείγματα βοηθημάτων είναι ο αισθητήρας παρκαρίσματος και η κάμερα οπισθοπορείας τα οποία προειδοποιούν τον οδηγό κατά το παρκάρισμα . Έτσι θα μπορεί να δει κάποιον ποδηλάτη ο οποίος περνάει εκείνη την ώρα από πίσω του η κάποιον πεζό και έτσι να αποφευχθεί κάποιο ατύχημα. Ένα άλλο βοήθημα είναι το Bluetooth οπού μπορεί ο οδηγός να μιλάει στο τηλέφωνο να διαβάζει τα sms του κλπ, όσο οδηγεί χωρίς να αποσπάται η προσοχή του.

Ένα άλλο βοήθημα το οποίο χρησιμοποιείται τα δυο τελευταία χρονιά και το οποίο τείνει να γίνει βασικό σε όλα τα οχήματα είναι μια κάμερα η οποία τοποθετείται στο μπροστινό παρμπρίζ του οχήματος. Η κάμερα έχει την εξής ιδιότητα όταν δει μπροστά της κάποιο αντικείμενο όπως κάποιο όχημα, κάποιο ζώο κλπ. προειδοποιώντας τον οδηγό με ηχητική ένδειξη. Όσο πιο κοντά πλησιάζει ο οδηγός τόσο πιο δυνατά χτυπάει ο ήχος. Εκτός από την ηχητική προειδοποίηση κάποια συστήματα παίρνουν εντελώς τον έλεγχο του οχήματος μειώνοντας σταδιακά την ταχύτητα του.

Ωραία τα όσα είδαμε παραπάνω. Ας δούμε τώρα το τι εφαρμόσαμε στην δική μας περίπτωση, ανάλογα φυσικά με τον εξοπλισμό τον οποίο διαθέταμε.

Το σύστημα το οποίο δημιουργήσαμε εμείς στην ουσία στηρίζεται στην τεχνολογία του Internet of things. Αυτή η τεχνολογία αποτελείται από ένα σύνολο αισθητήρων οπού ο κάθε αισθητήρας παίζει τον δικό του ρολό και όλοι οι αισθητήρες μαζί έχουν την ιδιότητα να συλλέγουν και να ανταλλάσσουν δεδομένα και πληροφορίες μεταξύ τους.

Για να λειτουργήσουμε το σύστημα χρησιμοποιήσαμε πρώτα απ' όλα έναν αισθητήρα ο οποιος διέθετε πάνω του μετρητή θερμοκρασίας και επιταχυνσιόμετρο, το Waspmote Board μαζί με το Waspmote Gateway. Επίσης χρησιμοποιήσαμε το γνωστό πια σε όλους μας Arduino μαζί με ένα 3G/GPRS/GPS Shield.

Η εφαρμογή μας δημιουργήθηκε με σκοπό την διαχείριση των αισθητήρων και των συστημάτων επικοινωνιών που υπάρχουν πάνω στις διαθέσιμες πλακέτες , καθώς επίσης και για την αποτελεσματικότερη αποστολή, λήψη, αποθήκευση και επεξεργασία των δεδομένων που μεταδίδονται από αυτές, με έναν φιλικό και εύχρηστο προς τον χρηστή τρόπο . Κομβικά σημεία της εφαρμογής μας είναι το μενού επιλογή για την γλώσσας , το σύστημα πρόβλεψης καιρικών συνθηκών, το σύστημα διαχείρισης βάσης για την καταγραφή των δεδομένων, το σύστημα διαχείρισης συντεταγμένων GPS, τα συστήματα διαχείρισης και γραφικής προβολής των δεδομένων από τους αισθητήρες όπως η θερμοκρασία , οι ειδοποιήσεις ασφαλείας τόσο ηχητικές – γραφικές όσο και πάνω στις πλακέτες (leds), καθώς επίσης τα συστήματα διαχείρισης σειριακών καναλιών. Τέλος έχουμε τα συστήματα κρυπτογράφησης αποκρυπτογράφησης και αποστολής και λήψης Email.

ABSTRACT

Today we live in a era which is approved daily in new technological challenges. In these technological challenges it should cope a lot of sectors like the telecommunications with the growth of networks of new generation (5G, Television via LTE etc), the industrial information technology (systems of automatic control, RFID etc), evolved designing of products (3D printers etc). One of the branch which accepts continuously new challenges is the world of automotive industry.

Let's take the things from the start. In the beginning of history of automotive world we had the existence of simply engine only. Then we had the addition of seatbelts. Afterwards the airbag was added. And afterwards the airbag the ABS was added which concerned about in more advanced brakes. And thus as long as years passed continuously more technologies mainly for the safety of the driver began to developing.

And we reached in today, where aids and elements which were optional for a vehicle tend to become obligatorily and completely essentially. Examples of aids are the parking sensor and the integrated back camera which is on the back plate and assist the driver while parking. Thus the driver will have the ability to see a cyclist which passes that hour from behind his certain pedestrian and thus some accidents to be avoided. An another aid is Bluetooth where the driver can speaks to the telephone, to reads sms etc, without losing his attention.

An another aid which is used last two years and which tends to be basic in all vehicles is a camera which is placed in forward windshield of the vehicle. The camera has the following ability when sees some object like vehicles, animals etc and warns the driver with a sound warning. As near as it approaches the driver so louder sounds are playing. Apart from the sound warning certain systems take automatic completely the control of vehicle decreasing progressively his speed.

Very interesting the above details which we see until now. Let's see what we implemented in our interface (depending on the equipment which we had available).

The system which we created we is based Internet Of things. This technology is consisted of sensors where each sensor plays its own role and all sensors together have the ability to collect and to share data and information from and to each other.

In order to our system to function we used first of all sensors of thermometer and accelerometer, which was embedded on WaspmoteBoard together with WaspmoteGateway. We also used the famous board Arduino by using this along with a 3G/GPRS/GPSShield.

Our application was created in order to manage the sensors and the communication systems that are available onboard the platform that were provided to us , and also to make the procedures of sending receiving storing and processing the transferring data in a way more efficient and user friendly . Key elements of our project are the language selection menu, the weather broadcast system, our database system, the system that manages the available GPS data, the managing systems that use a graphical presentation of our onboard sensors like the thermometer, our safety notification system that uses sound, graphical and hardware leds to trigger the awareness of the user and finally our data encryption system and our email platform.

Title of Thesis:

"Safe monitoring and management of intelligent cars using wireless sensors"

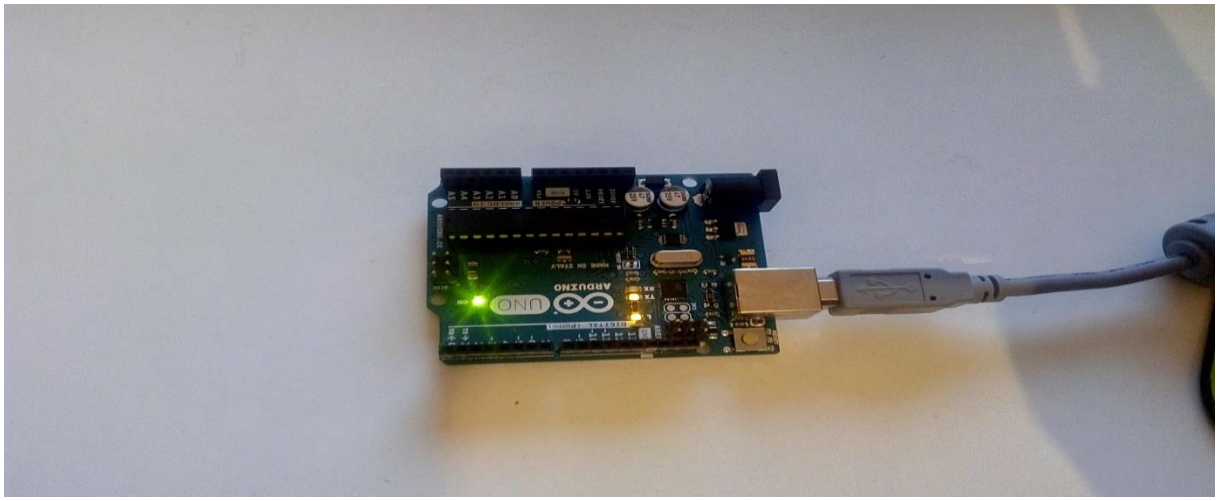
ΚΕΦΑΛΑΙΟ 1

Στο κεφάλαιο αυτό θα αναλυθεί η όλη διαδικασία σχετικά με την εγκατάσταση των προγραμμάτων και του Hardware που χρησιμοποιήθηκε για την υλοποίηση της πτυχιακής, τα οποία είναι συνοπτικά από πλευράς προγραμμάτων τα Netbeans,Putty,Waspmote, και Arduino Ide's, X-Ctu της Digi και από πλευράς Hardware το Arduino Uno, το 3G/GPRS/GPS Shield των Libelium-Cooking Hacks και το **Waspmote - Wireless Sensor Network 802.15.4 ZigBee(η εν συντομία Waspmote Board)** μαζί με το Waspmote Gateway.

1.1 Διασύνδεση Του Arduino Uno Με Το 3G/GPRS/GPS Shield

Το πρώτο πράγμα το οποίο κάναμε ήταν να τεστάρουμε τον εξοπλισμό μας, αν λειτουργεί σωστά ώστε να δούμε ότι υπάρχει μεταξύ μας επικοινωνία(Arduino Uno- 3G/GPRS/GPS Shield).

Αρχική συνδέσαμε το Arduino Uno μεμονωμένα στον υπολογιστή για να αναγνωριστούν Drivers κλπ. (Εικόνα 1.1).



Εικόνα 1.1

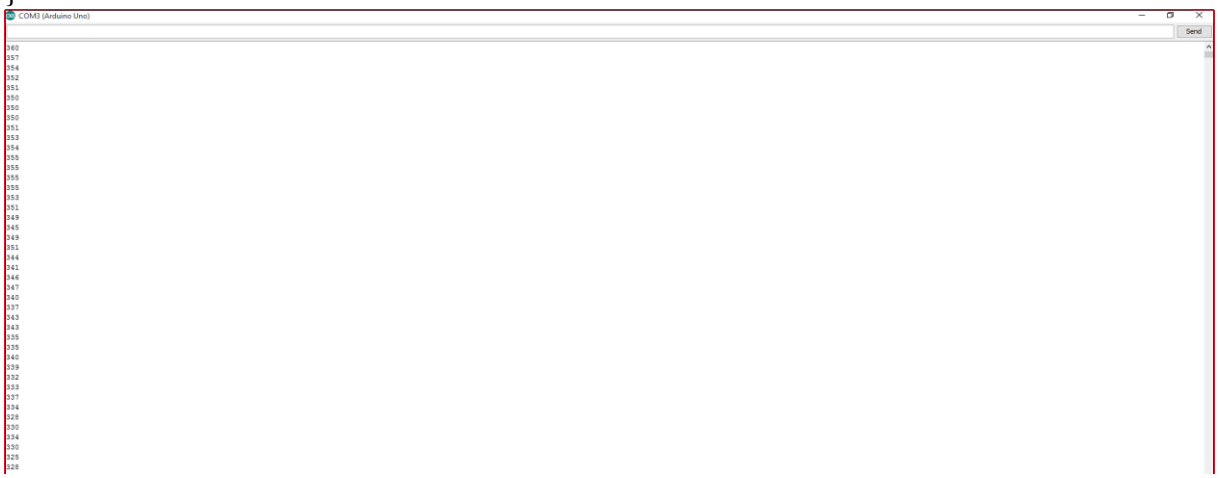
Παράλληλα εγκαταστήσαμε και το Arduino Ide το οποίο περιείχε τα Drivers τα οποία και μας ήταν απαραίτητα για να αναγνωριστεί το Arduino Uno , και να μπορέσουμε να προγραμματίσουμε πάνω σε αυτό.

Έπειτα δοκιμάσαμε από τα έτοιμα examples του Arduino το AnalogReadSerial . Σε αυτό τον κώδικα βλέπουμε το πως γίνεται η σειριακή επικοινωνία , αρχικοποιώντας την στα 9600 bps, και μετά είδαμε στην οθόνη τα δεδομένα που διαβάζει από το input 0 του Arduino και τα εκτυπώνει επ' άπειρον στην οθόνη. Ο κώδικας αυτός χρησιμοποιήθηκε σαν πρώτος κώδικας ώστε να βεβαιωθούμε ότι όλες οι λειτουργίες του Arduino δουλεύουν όπως προβλέπεται(Εικόνα 1.2).

<u>Ο κώδικας του Example ήταν ο εξής:</u>

```
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}
// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
int sensorValue = analogRead(A0);
// print out the value you read:
```

```
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```



Εικόνα 1.2

Στην συνέχεια προχωρήσαμε με την εγκατάσταση και την αναγνώριση του 3G/GPRS/GPS Shield από τον υπολογιστή . Για αν γίνει αυτό η συνδεσμολογία ήταν η εξής(Εικόνα 1.3 -1.4):

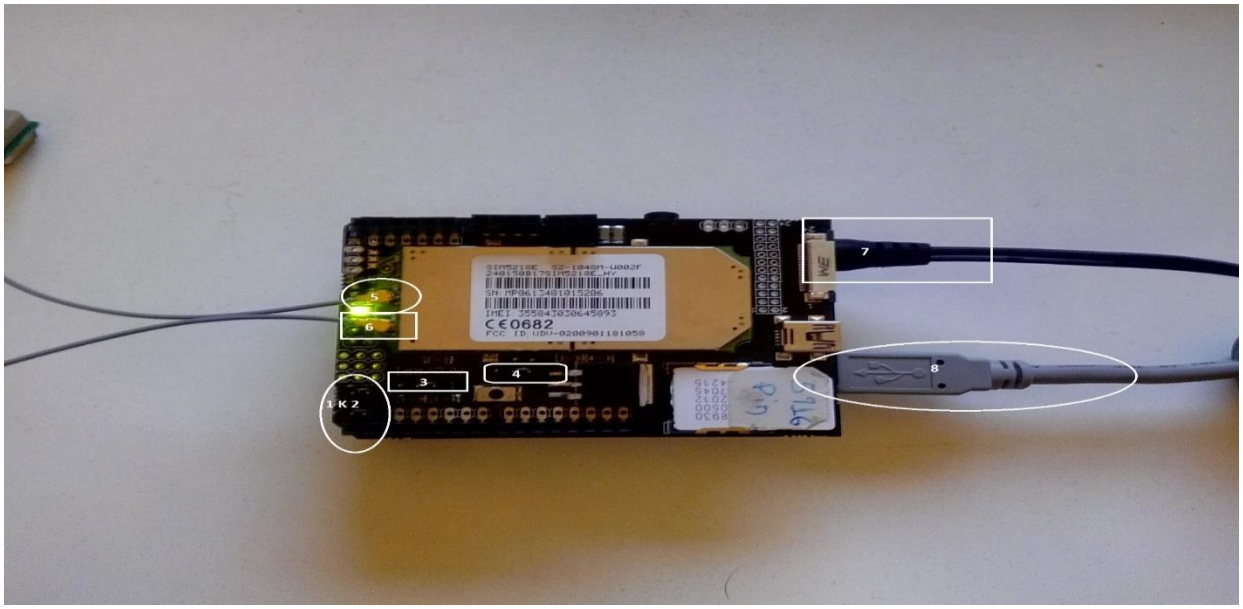
Εικόνα 1.3

Για να αναγνωριστεί από τον υπολογιστή ήταν αναγκαίο να εγκαταστήσουμε τους Drivers του Sim5218.

Εικόνα 1.4

Τέλος είχαμε την διασύνδεση μεταξύ του Arduino και του 3G/GPRS/GPS Shield οπού το μόνο το οποίο προσθέσαμε ήταν μια τροφοδοσία 5 V για να μπορέσει να λειτουργήσει το 3G/GPRS/GPS.

Στην Εικόνα 1.5 επίσης έχουμε τα 1,2 τα οποία εναλλάσσουν την διαδικασία Usb η Ard, οπού Usb είναι το Gateway Mode και Ard είναι χρήση κώδικα Arduino. Επιπλέον έχουμε τα 3 κ΄4 οπού στο 3 έχουμε το jumper για την επιλογή Arduino η Raspberry (Ανάλογα που έχουμε συνδέσει το 3G/GPRS/GPS Shield). Ακολούθως είχαμε τα 5,6 οπού είναι οι κεραίες μας , και πιο συγκεκριμένα η 5 οπού είναι για τον δεκτή του GPS και η 6 οπού είναι για την λήψη του τηλεφωνικού σήματος(3G/GPRS). Αξίζει να αναφέρουμε πως η κεραία μας υποστηρίζει και την λήψη δικτυών (LTE-4G) αλλά δυστυχώς δεν το υποστηρίζει το 3G/GPRS/GPS Shield. Κλείνοντας είχαμε την συνδεσμολογία του υπολογιστή με το Arduino και παράλληλα με το 3G/GPRS/GPS Shield(στην Εικόνα 1.5 το 8), (το οποίο για λογούς συντομίας από εδώ και έπειτα θα το λεμέ απλώς Shield), με την χρήση μετασχηματιστή 5V(στην Εικόνα 1.5 το 7).



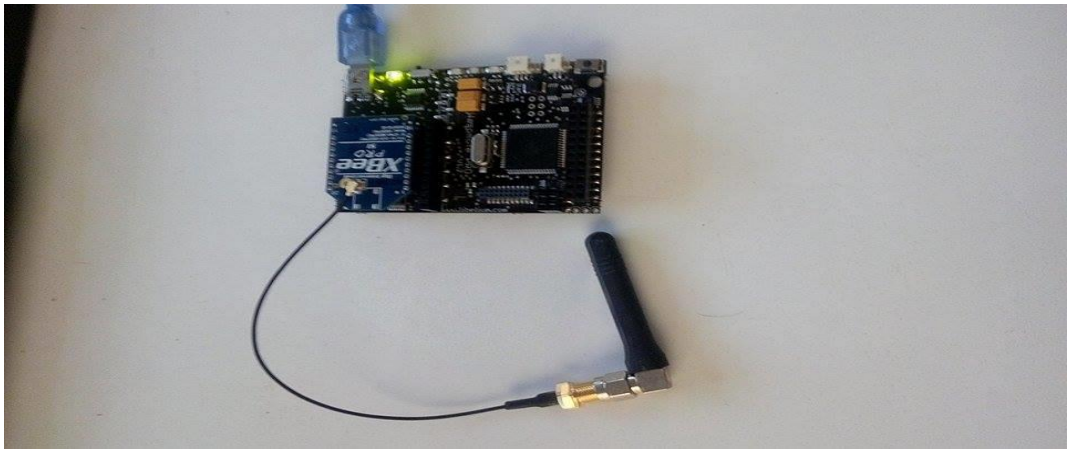
Εικόνα 1.5

Αφού έγιναν όλα τα παραπάνω στην ενότητα 1.1 συνδέσαμε και ένα παραπάνω καλώδιο(σε σχέση με την Εικόνα 1.5, 2 στην Εικόνα 1.6). Επίσης κάτω από το jumper του ON/OFF βάλαμε και αυτό(1 στην Εικόνα 1.6). Σκοπός για όλα αυτά είναι να πάρουμε τις σωστές συντεταγμένες μέσω του GPS.

Εικόνα 1.6

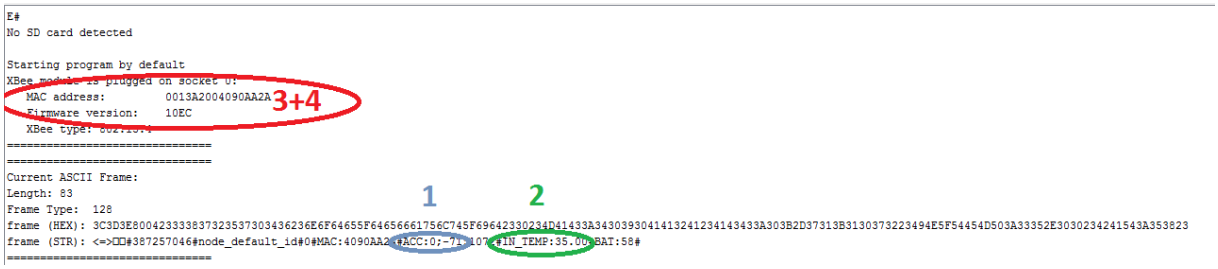
1.2 Η Εγκατάσταση-Χρήση Του Waspmote Ide Σε Συνδυασμό Με Τα Waspmote Gateway Και Waspmote Board.

Το πρώτο πράγμα το οποίο κάναμε ήταν να κατεβάσουμε το Waspmote ide από το επίσημο site της Libelium. Το Waspmote Ide έχει μόνο και μόνο την χρήση του να ανεβάζουμε κώδικα στο Waspmote Board και να βλέπουμε τα δεδομένα ανάλογα τον κώδικα στο Serial Monitor το οποίο διαθέτει. Η διαφορά με το Arduino Ide είναι ότι δεν περιέχει τους Drivers γιατί τα Waspmote Board και Waspmote Gateway αναγνωρίζονται αυτόματα από τον υπολογιστή. Μετά το κατέβασμα του Waspmote Ide βάλαμε το Waspmote Board στον υπολογιστή για να αναγνωριστεί και να αρχίσουμε την χρήση του (Εικόνα 1.7).



Εικόνα 1.7

Για να βεβαιωθούμε ότι λειτουργεί όπως πρέπει το Waspmote Board δοκιμάσαμε από τα έτοιμα examples από την ιστοσελίδα της Libelium το Pro Test Code. Επιλέξαμε αυτό το παράδειγμα για να δούμε ότι μας δείχνει, κυρίως την θερμοκρασία(1 στην Εικόνα 1.8) και την επιτάχυνση(2 στην Εικόνα 1.8), μέσω των ειδικών αισθητήρων που διαθέτει(και οπού τα χαρακτηριστικά τους φαίνονται παρακάτω**), καθώς επίσης μας δείχνει και άλλες διάφορες σημαντικές πληροφορίες σχετικά με το Waspmote όπως την Mac Address του, την έκδοση του Firmware(3+4 στην Εικόνα 1.7) κλπ.



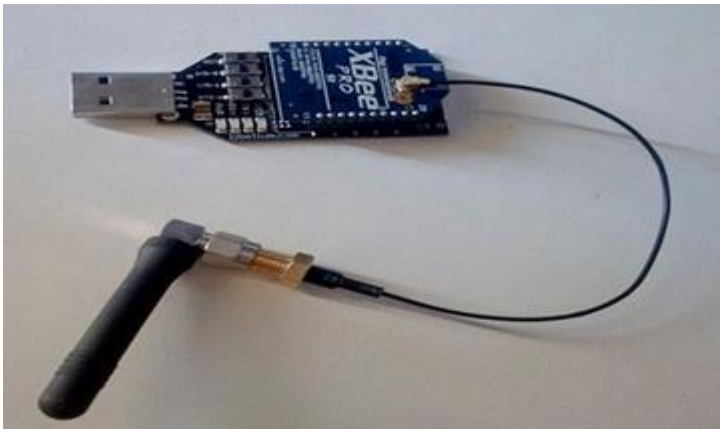
Εικόνα 1.8

Αξίζει να αναφέρουμε πως έπρεπε να επιλέξουμε στο Waspmote Ide την πλακέτα και την θύρα που ήταν συνδεμένο το Waspmote Board. Επίσης τα δεδομένα από την θερμοκρασία και τα χρησιμοποιήσαμε για την υλοποίηση της πτυχιακής μας.

**Temperature (+/-): -40ºC , +85ºC. Accuracy: 0.25ºC.
Accelerometer: ±2g/±4g/

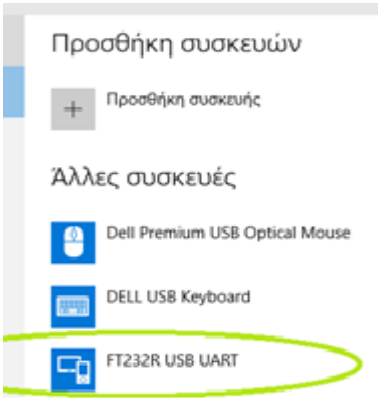
Low power: 0.5 Hz / 1 Hz / 2 Hz / 5 Hz / 10 Hz
Normal mode: 50 Hz / 100 Hz / 400 Hz / 1000 Hz

Ένα από τα πλεονεκτήματα του Waspmote είναι ότι μπορούμε να λαμβάνουμε δεδομένα ασύρματα(θερμοκρασία κλπ.). Αυτό γίνεται με την χρήση ενός αλλού Hardware σε συνδυασμό με το Waspmote Board, το Waspmote Gateway (Εικόνα 1.9).



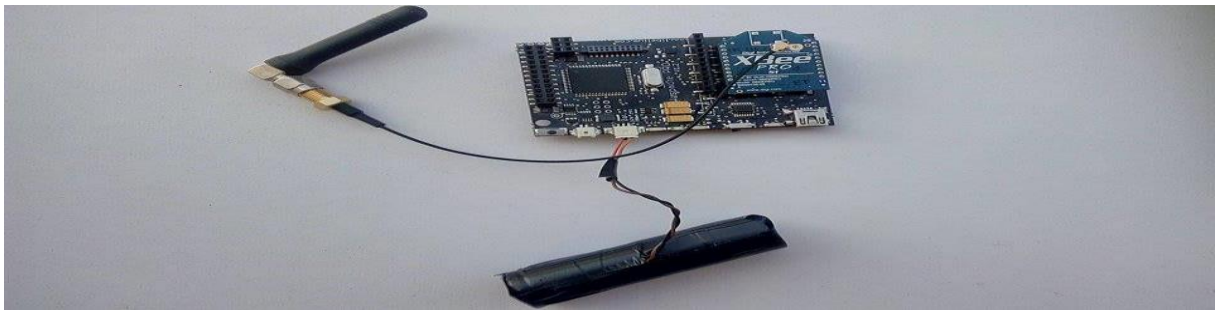
Εικόνα 1.9

Το Waspmote Gateway δεν ήθελε τίποτε ιδιαίτερο σχετικά με την εγκατάσταση του απλά το βάλαμε στον υπολογιστή και αναγνωρίστηκε αυτόματα(Εικόνα 1.10).



Εικόνα 1.10

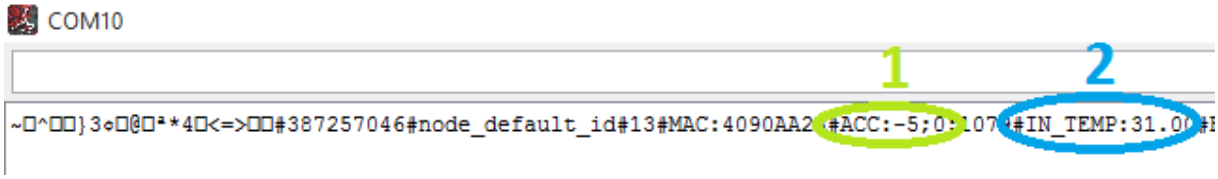
Για να γίνει η επικοινωνία μεταξύ του Waspmote Board και του Waspmote Gateway έπρεπε να είναι το Waspmote Gateway στον υπολογιστή απλά συνδεμένο και η μονή αλλαγή που έπρεπε να γίνει στο Waspmote Board ήταν ότι δεν χρειάζονταν να είναι στον υπολογιστή αλλά έπρεπε να είναι συνδεμένη πάνω του στην ειδική υποδοχή που διαθέτει, μια μπαταριά. Η μπαταριά ήταν απαραίτητη ώστε να έχουμε και την ασύρματη μετάδοση δεδομένων μεταξύ των δυο(Εικόνα 1.11).



Εικόνα 1.11

Και σε αυτήν την περίπτωση όπως και στην προηγουμένη (Απλά με την χρήση του Waspmote Board), χρησιμοποιήσαμε τον κώδικα του Pro Test Code από την ιστοσελίδα της Libelium.

Τα αποτελέσματα της ασύρματης λήψης δεδομένων φαίνονται παρακάτω(Εικόνα 1.12). Και εδώ μπορούμε να δούμε την επιτάχυνση(1 στην Εικόνα 1.12) και την θερμοκρασία(2 στην Εικόνα 1.12).

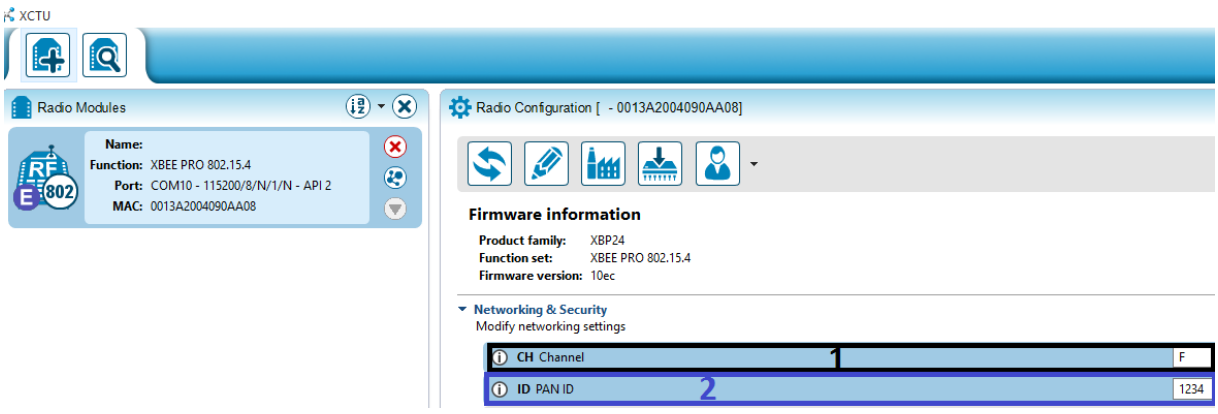


Εικόνα 1.12

Σε αυτό το σημείο αξίζει να αναφέρουμε κάποια σημαντικά στοιχεία τα οποία δεν αναφέρθηκαν παραπάνω.

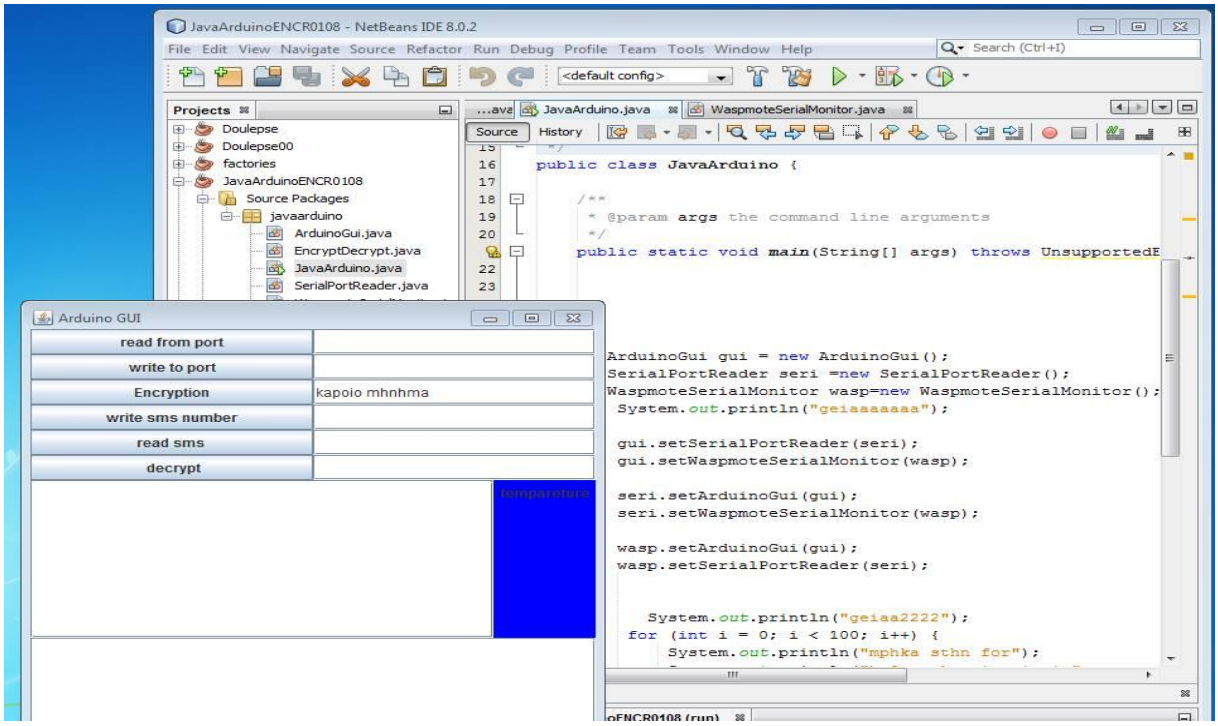
Για να γίνει η επικοινωνία μεταξύ του Waspmote Board και του Waspmote Gateway φορτώσαμε στην πλακέτα του Waspmote από τα έτοιμα examples του Waspmote Ide το 802_01_configure_XBee_parameters. Χρησιμοποιήσαμε αυτό το παράδειγμα για την ρύθμιση των παραμέτρων επικοινωνίας. Συγκεκριμένα ρυθμίζει το κανάλι επικοινωνίας, και το PanID το οποίο σημαίνει Personal Area Network Identifier στο Waspmote.

Οπως έπρεπε να ανεβάσουμε τον κώδικα στο Waspmote Board για να του βάλουμε το Channel και το PanId έτσι έπρεπε να περάσουμε και τα ιδιά στοιχεία στο Waspmote Gateway. Χρειάζεται να έχουν τα ιδιά στοιχεία για να μπορεί να υπάρξει ταυτόχρονη επικοινωνία μεταξύ των Waspmote Boardκαι Waspmote Gateway. Η μονή διαφορά είναι ότι δεν το κάναμε αυτό μέσω κώδικα αλλά μέσω ενός άλλους προγράμματος του X-Ctu της Digi και στην Εικόνα 1.13 παρακάτω φαίνονται οι παράμετροι που αλλάξαμε.



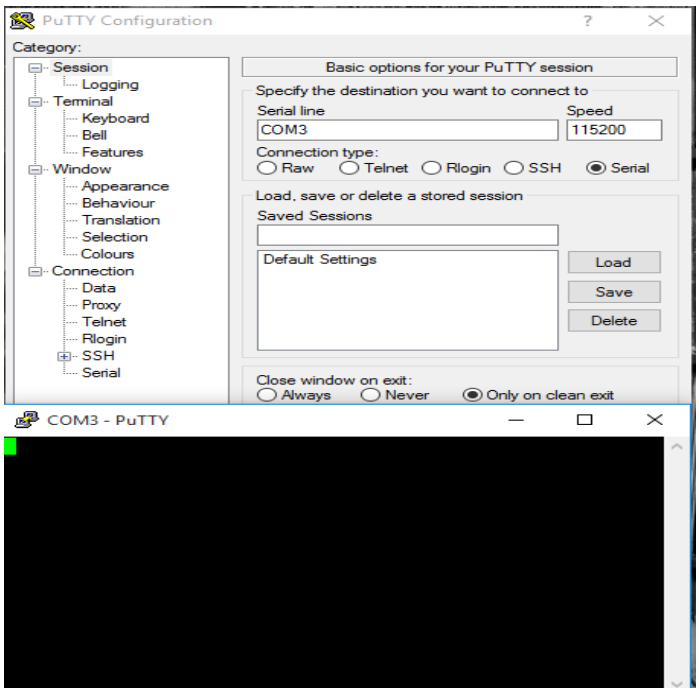
Εικόνα 1.13

Εκτός από τα προαναφερθείσα για την υλοποίηση της πτυχιακής μας χρησιμοποιήθηκε το Netbeans, το οποίο υποστηρίζεται από την Java. Η εγκατάσταση του ήταν απλή , κατεβάσαμε το τελευταίο Jdk από την σελίδα της Oracle, το εγκαταστήσαμε και έπειτα εγκαταστήσαμε. Το Netbeans όπως θα δούμε και παρακάτω χρησιμοποιήθηκε για την δημιουργία ενός Java Intrface, το οποίο θα έχει τον κεντρικό έλεγχο του όλου δημιουργήματος (Εικόνα 1.14).



Εικόνα 1.14

Τέλος το Putty είναι μια δωρεάν υλοποίηση του Telnet και SSH και το οποίο διαθέτει ένα εξομοιωτή τερματικού. Το μόνο που κάναμε ήταν απλά ένα Install(Εικόνα 1.15).



Εικόνα 1.15

ΚΕΦΑΛΑΙΟ 2

Στο παρακάτω κεφάλαιο θα αναλύσουμε την προσπάθεια που κάναμε για να γίνει κρυπτογράφηση και αποκρυπτογράφηση μεταξύ Waspmote και Waspmote Gateway σε συνδυασμό με την Java(με την χρήση του Netbeans).

Πριν ξεκινήσουμε με την περιγραφή της διαδικασίας της κρυπτογράφησης και της αποκρυπτογράφησης ας που με λίγα παραπάνω λογία για το Hardware το οποίο χρησιμοποιήθηκε.

Waspmote Gateway

Το Waspmote Gateway είναι μια συσκευή που επιτρέπει την συνεχόμενη ροή δεδομένων από ένα δίκτυο αισθητήρων στον υπολογιστή μας ή οποιαδήποτε συσκευή που έχει USB ports. Το Waspmote Gateway λειτουργεί σαν γέφυρα δεδομένων ή σαν σημείο πρόσβασης μεταξύ του δικτύου αισθητήρων και του εξοπλισμού λήψης που έχει επίσης την αρμοδιότητα για την αποθήκευση , επεξεργασία και χρήση των δεδομένων αυτών.

Waspmote Board

Το Waspmote Board είναι μια open source πλατφόρμα αισθητήρων ειδικά υλοποιημένη πάνω σε λειτουργία χαμηλής κατανάλωσης το οποίο επιτρέπει στην πλατφόρμα μια πιο αυτόνομη λειτουργία όσων αφορά την κατανάλωση ενεργείας σε τέτοιο βαθμό που με τις κατάλληλες συνθήκες να μπορεί να λειτουργεί από μήνες έως και χρονιά αυτόνομα.

Τα υποσυστήματα επικοινωνίας του Waspmote Board είναι ενσωματωμένα, επιτρέποντας μια end-point σύνδεση με τις διαφορές συσκευές. Τα υποσυστήματα αυτά λειτουργούν βάση του πρωτοκόλλου IEEE 802.15.4 δίνοντας έτσι τη δυνατότητα για ταχύτατη point-to-multipoint η point-to-peer δικτύωση, σχεδιασμένα για υψηλή διακίνηση δεδομένων και χαμηλές απώλειες.

Οι Δυνατότητες Του Waspmote Στην Κρυπτογράφηση

Το Waspmote δίνει την δυνατότητα στον χρηστή με απλές μεθόδους να εφαρμόσει κρυπτογράφηση δεδομένων καθώς επίσης και έλεγχο ακεραιότητας .

Οι μέθοδοι κρυπτογράφησης περιλαμβάνουν RSA, καθώς επίσης και AES 128, 192, 256 με χρήση block κρυπτογράφησης είτε CBC είτε ECB , και με δυνατότητες συμπλήρωσης(Padding) PKCS5 byte padding και Zeros byte padding.

Στις μεθόδους Hash περιλαμβάνονται οι MD5 , SHA-1, SHA-224, SHA-256, SHA-384 καθώς και η SHA- 512

Να αναφέρουμε επίσης ότι οι μέθοδοι κρυπτογράφησης μπορούν να χρησιμοποιηθούν είτε για μερικά δεδομένα (πχ οι τιμές κάποιου αισθητήρα)είτε για ολόκληρο data frame

Παραδείγματα χρήσης συναρτήσεων κρυπτογράφησης:

AES 128 ECB PKCS5 padding

```
AES.encrypt(  AES_128
              , password   // to key
              , message    // the message
              , encrypted_message   // output of the method
              , ECB               // block type
              , PKCS5);           // Padding method
```

AES 192 CBC Zero byte padding

```
AES.encrypt(  AES_192
              , password
              , message
```

```
                , encrypted_message
                , CBC
                , ZEROS
                , IV); // χρήση Initial Vector λόγω CBC
```

AES 128 CBC PKCS5 πάνω σε Frame

```
AES.encrypt(  128
                , password
                , frame.buffer  // dedomena frame
                , frame.length        // mhkos frame
                , encrypted_message    // to output ths me8odou
                , CBC    // block type
                , PKCS5   // Padding method
                , IV);    // χρήση  Initialg Vector λόγω CBC
```

Τώρα θα ξεκινήσουμε την περιγραφή της όλη διαδικασία(κρυπτογράφησης-αποκρυπτογράφησης) με την χρήση των παραπάνω(Waspmote Board κ΄ Waspmote Gateway).

Αρχικά φορτώσαμε τον κώδικα της κρυπτογράφησης δεδομένων στο Waspmote Board. Ο κώδικας είναι ο παρακάτω: (Στις Εικόνες 2.1, 2.2, 2.3)

```
#include "WaspAES.h"
#include <WaspXBee802.h>
#include <WaspSX1272.h>
#include <WaspFrame.h>

// Define a 16-Byte (AES-128) private key to encrypt message
char password[] = "libeliumlibelium";

// original message on which the algorithm will be applied
char message[] = "this is a message";

// Variable for encrypted message's length
uint16_t encrypted_length;

// Declaration of variable encrypted message
uint8_t encrypted_message[300];
uint8_t destination[8]={
  0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFF};
packetXBee* packet;
boolean LoRa_type = false;
uint8_t out0;

// Define Initial Vector
uint8_t IV[16] = { 0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0A,0x0B,0x0C,0x0D,0x0E,0x0F};
void setup()
{
  // init USB port
  USB.ON();
   Utils.setLED(LED0, LED_OFF);
  Utils.setLED(LED1, LED_OFF);
  for (int i = 0 ; i < 4 ; i++)
  {
    Utils.blinkLEDs(100);
  }
}

void loop()
{
```

Εικόνα 2.1

```
39    Utils.setLED(LED1,LED_ON);
40    ////////////////////////////////////////////////////////////////
41    // 1. Encrypt message
42    ////////////////////////////////////////////////////////////////
43    USB.println(F("1. Encrypt message"));
44    // ftiaxnw frame
45    frame.createFrame(ASCII);
46
47    // 1.2. Calculate encrypted message with ECB cipher mode and PKCS5 padding.
48    AES.encrypt(  AES_128
49                , password
50                , message
51                , encrypted_message
52                , ECB
53                , PKCS5
54                 );
55
56    // 1.3. Printing encrypted message
57    USB.print(F("AES Encrypted message:"));
58   //USB.printf(%s,encrypted_message);
59    //AES.printMessage( encrypted_message, encrypted_length);
60    // 1.4. Printing encrypted message's length
61  frame.addSensor(SENSOR_STR, (char*) encrypted_message);
62
63    if( LoRa_type == true )
64    {
65     // out0 = sx1272.sendPacketTimeout(BROADCAST_0, encrypted_message, encrypted_length);
66      out0 = sx1272.sendPacketTimeout(BROADCAST_0, frame.buffer, frame.length);
67    }
68    else
69    {
70
```

Εικόνα 2.2

```
71      // 10.1 set packet to send
72      packet=(packetXBee*) calloc(1,sizeof(packetXBee)); // memory allocation
73      packet->mode=BROADCAST; // set Unicast mode
74      // 10.2 send the packet via the correct object depending on the protocol
75      // case 802.15.4
76
77        // turn XBee on
78        xbee802.ON();
79        // sets Destination parameters
80       // xbee802.setDestinationParams(packet, destination, encrypted_message, encrypted_length);
81       xbee802.setDestinationParams(packet, destination, frame.buffer, frame.length);
82        // send data
83        xbee802.sendXBee(packet);
84        // check TX flag
85
86      free(packet);
87      packet = NULL;
88
89    }
90
91    delay(5000);
92    USB.println(F("***********************"));
93
94  }
```

Εικόνα 2.3

Όπως μπορούμε να διακρίνουμε και από την εικόνα 2.4 έχουμε ορίσει κυρίως το κλειδί της κρυπτογράφησης και το μήνυμα που θέλουμε να κρυπτογραφήσουμε.

```
// Define a 16-Byte (AES-128) private key to encrypt message
char password[] = "libeliumlibelium";

// original message on which the algorithm will be applied
char message[] = "this is a message";
```

Εικόνα 2.4

Και στην Εικόνα 2.5 μπορούμε να δούμε τον τρόπο τον οποίο χρησιμοποιήσαμε για να γίνει επιτυχής η κρυπτογράφηση.

```
  // 1.2. Calculate encrypted message with ECB cipher mode and PKCS5 padding.
  AES.encrypt(  AES_128
              , password
              , message
              , encrypted_message
              , ECB
              , PKCS5
               );

  // 1.3. Printing encrypted message
  USB.print(F("AES Encrypted message:"));
 //USB.printf(%s,encrypted_message);
  //AES.printMessage( encrypted_message, encrypted_length);
  // 1.4. Printing encrypted message's length
frame.addSensor(SENSOR_STR, (char*) encrypted_message);
```

Εικόνα 2.5

Αφού φορτώθηκε ο κώδικας στο Waspmote Board , είχαμε τα παρακάτω αποτελέσματα στο Serial Monitor (Εικόνα 2.6). Φυσικά η όλη διαδικασία η οποία ακολουθήθηκε, έγινε στην ασύρματή λειτουργία(με την χρήση του Waspmote Gateway).

```
~□J□□}3∘□@□ª*3□<=>□□#387257046#node_01#8#STR:ðmüq$□&'□□□e%_}3□ª ó□DJ5V□â.Hµ□Ü□□#á~□J□□}3∘□@□ª*3□<=>□□#387257046#
```

Εικόνα 2.6

Στην Εικόνα 2.7 μπορούμε να δούμε τα αποτελέσματα τα οποία είχαμε στο Serial monitor στην ενσύρματη λειτουργία(μόνο με την χρήση του Waspmote Board).

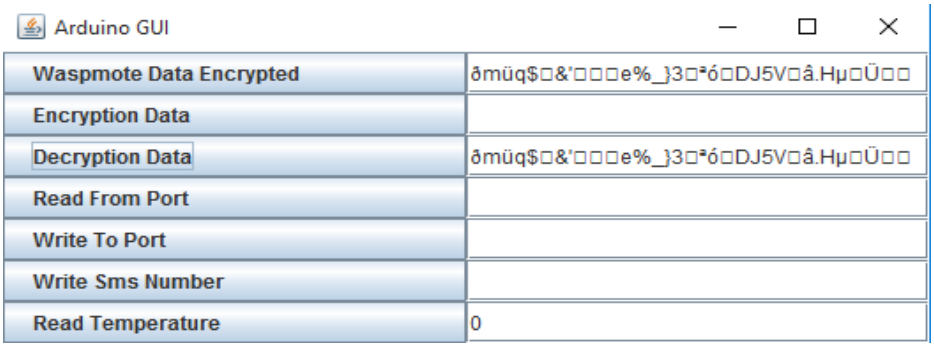
```
E#
1. Encrypt message
================================
Current ASCII Frame:
Length: 26
Frame Type:  128
frame (HEX): 3C3D3E8000233338373235373034362 36E6F64655F3031233023
frame (STR): <=>□□#387257046#node_01#0#
================================
```

Εικόνα 2.7

Στην ενσύρματη λειτουργία η μονή αλλαγή που κάναμε σε σχέση με την ασύρματη λειτουργία στον κώδικα ήταν να βάλουμε πάνω από την γραμμή 60 τον εξής κώδικα:
Frame.showframe();

Έπειτα δοκιμάσαμε την παραπάνω διαδικασία στην Java (χρησιμοποιώντας το Netbeans).

Αφού ανοίξαμε και τρέξαμε το πρόγραμμα είδαμε το ακόλουθο Serial Monitor: (Εικόνα 2.8)



Εικόνα 2.8

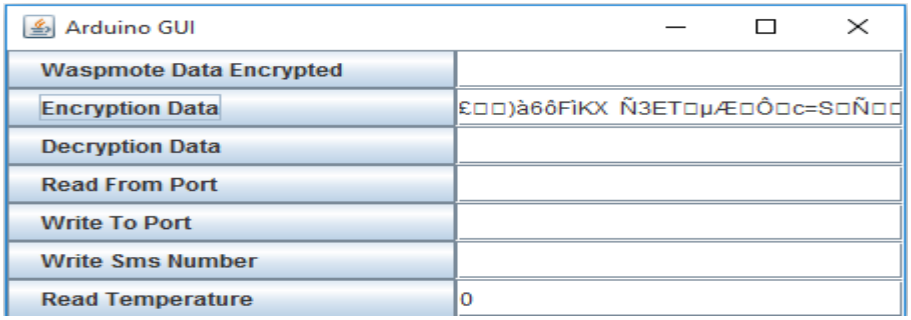
Έπειτα πατήσαμε το κουμπί Waspmote Data Encrypted ούτως ώστε να πάρουμε το κρυπτογραφημένο μήνυμα από την σειριακή θύρα του Waspmote Gateway. Αφού πήραμε το

κρυπτογραφημένο πήραμε το περιεχόμενο που υπήρχε στο κουτάκι διπλά από το Waspmote Data Encrypted και το επικολλήσαμε διπλά από το κουτάκι του κουμπιού Decrypt και το πατήσαμε. Όμως λόγω του ότι γενικά το Hardware το οποίο είχαμε υπό την κατοχή μας, ήταν ασταθές πήραμε το εξής μήνυμα (Εικόνα 2.9).

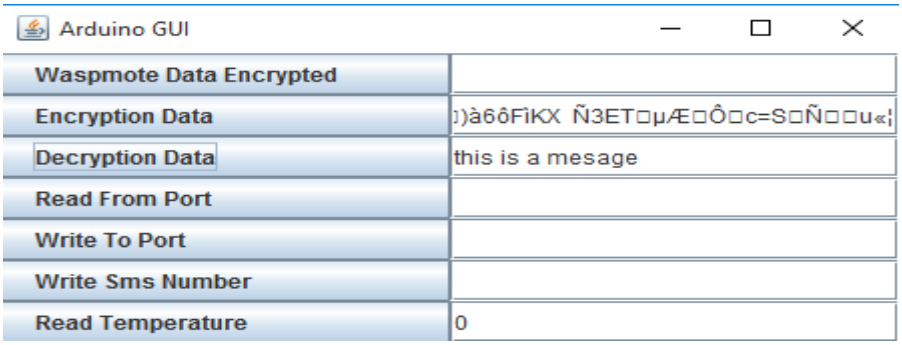
```
 SEVERE: null
]javax.crypto.IllegalBlockSizeException: Input length must be multiple of 16 when decrypting with padded cipher
        at com.sun.crypto.provider.CipherCore.doFinal(CipherCore.java:913)
        at com.sun.crypto.provider.CipherCore.doFinal(CipherCore.java:824)
        at com.sun.crypto.provider.AESCipher.engineDoFinal(AESCipher.java:436)
        at javax.crypto.Cipher.doFinal(Cipher.java:2165)
        at javaarduino04082015.EncryptDecrypt.aesDecryption(EncryptDecrypt.java:60)
        at javaarduino04082015.ArduinoGui$3.actionPerformed(ArduinoGui.java:145)
        at javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:2022)
        at javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2346)
        at javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:402)
        at javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:259)
        at javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:252)
        at java.awt.Component.processMouseEvent(Component.java:6525)
        at javax.swing.JComponent.processMouseEvent(JComponent.java:3324)
        at java.awt.Component.processEvent(Component.java:6290)
        at java.awt.Container.processEvent(Container.java:2234)
        at java.awt.Component.dispatchEventImpl(Component.java:4881)
```

Εικόνα 2.9

Το προαναφερόμενο ότι το σύστημα είναι ασταθές σημαίνει το εξής: ότι κάποιες φορές γίνεται αποτελεσματικά η αποκρυπτογράφηση. Αλλά τις περισσότερες φορές λόγω του ότι η Java δεν καταφέρνει να αποκρυπτογραφήσει σωστά τα δεδομένα (Εικόνα 2.9) παίρνουμε το παραπάνω σφάλμα. Αυτό συμβαίνει γιατί λόγω της ασύρματης μετάδοσης υπάρχει φθορά δεδομένων. Ενώ αν βάλουμε χειροκίνητα το οποιοδήποτε μήνυμα στο Encryption Data(Εικόνα 2.10) και έπειτα τα δεδομένα τα οποία εμφανίζονται, τα βάλουμε στο Decrypt, η αποκρυπτογράφηση γίνεται αποτελεσματικά(Εικόνα 2.11).



Εικόνα 2.10



Εικόνα 2.11

ΚΕΦΑΛΑΙΟ 3

Σε αυτό το κεφάλαιο θα γίνει πλήρης αναφορά για τα AT Commands. Πιο συγκεκριμένα θα αναλυθεί για το τι είναι , το ποια είναι η χρήση τους και το πώς μας βοήθησαν στο να στήσουμε το δίκτυο κινητής τηλεφωνίας και να εντοπίσουμε το σωστό σήμα GPS.

Οι AT Commands είναι εντολές οι οποίες χρησιμοποιούνται για να ρυθμίσουν ένα Modem. Το AT είναι η συντομογραφία για το ATtention. Η κάθε γραμμή εντολών ξεκινάει με "AT" η "at". Αυτός είναι ο λόγος για τον οποίο οι εντολές του modem αποκαλούνται αλλιώς και AT Commands. Οι περισσότερες από αυτές τις εντολές οι οποίες χρησιμοποιούνται για να ελέγξουν τα dial-up modems, όπως τα ATD (Dial), ATA (Answer), ATH (Hook control) και ATO (Return to online data state), έχουν επίσης την δυνατότητα υποστήριξης GSM/GPRS modems και κινητών τηλέφωνων.

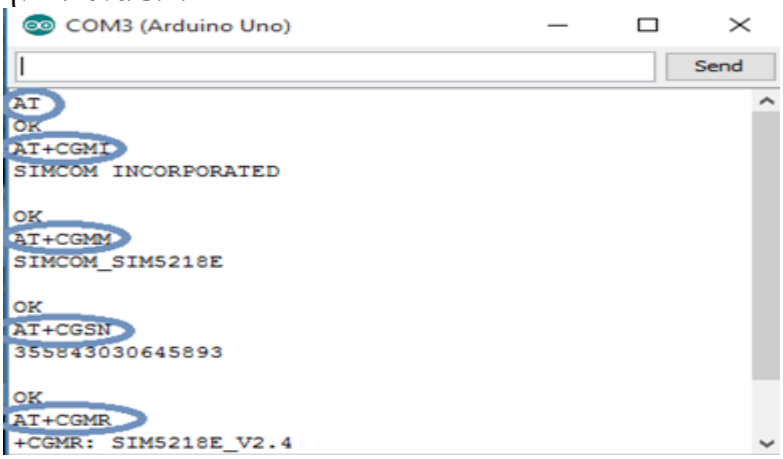
Εξάλλου εκτός από αυτό το κοινό set εντολών, τα GSM/GPRS modems και τα κινητά τηλεφωνά υποστηρίζουν ένα ειδικό σύνολο εντολών για τα GSM δίκτυα οι οποίες περιλαμβάνουν ένα σύνολο για την αποστολή και λήψη SMS μηνυμάτων όπως για παράδειγμα οι: AT+CMGS (Αποστολή SMS μηνύματος), AT+CMSS (Αποστολή SMS μηνύματος από την μνήμη, AT+CMGL (Μας εμφανίζει όλα τα μηνύματα που υπάρχουν μέσα στην Sim) και AT+CMGR (Μας εμφανίζει όλα τα διαβασμένα μηνύματα).

Να σημειώσουμε δε ότι η εντολή "AT" ενημερώνει το modem πως μια γραμμή εντολών ξεκινάει. Φυσικά το "AT" δεν αρκεί αλλά θα πρέπει να υπάρχει και κάτι άλλο μαζί του, για παράδειγμα το AT+CMGL που αναφέραμε παραπάνω.

Ας αναφέρουμε επιγραμματικά μερικές από τις εργασίες οι οποίες μπορούν να γίνουν με την χρήση των AT Commands.

/* Να πάρουμε βασικές πληροφορίες για τις κινητές μας συσκευές η το GSM/GPRS Modem*/

Για παράδειγμα: Το όνομα του κατασκευαστή(AT+CGMI), τον αριθμό του μοντέλου της συσκευής(AT+CGMM), τον αριθμό IMEI(International Mobile Equipment Identity-AT+CGSN), και την έκδοση λογισμικού της συσκευής(AT+CGMR). Όλα τα προαναφερόμενα φαίνονται και στην Εικόνα 3.1.



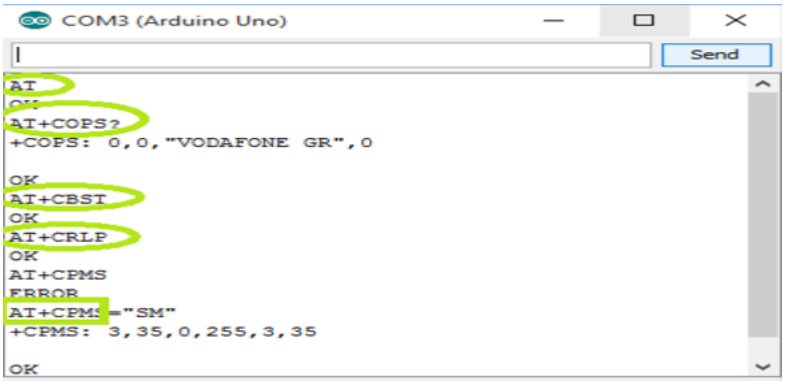
Εικόνα 3.1

Να στείλουμε (AT+CMGS, AT+CMSS), να διαβάσουμε(AT+CMGR, AT+CMGL), να γράψουμε (AT+CMGW) η να διαγράψουμε (AT+CMGD) μηνύματα SMS και να λάβουμε ειδοποίηση μόλις μας έρθει ένα νέο μήνυμα (AT+CNMI).Κάποια από τα προαναφερόμενα φαίνονται και στην Εικόνα 3.2.

```
AT
OK
AT+CMGF=1
OK
AT+CPMS="SM"
+CPMS: 4,35,0,255,4,35

OK
AT+CMGL="ALL"
+CMGL: 0,"REC READ","+306978716080","","15/07/24,12:54:18+12"
Ma ti tha ginei
+CMGL: 1,"REC READ","+306974331832","","15/07/24,13:09:31+12"
Shmera
+CMGL: 2,"REC READ","+306974331832","","15/07/24,12:54:37+12"
Shmera
+CMGL: 3,"REC UNREAD","6785105110102111","","15/08/01,13:07:41+12"
TA   PAKIA TOY MHNA! 600'  PO  CU + 600MB  IA 30MEPE  A O THN  OITHTIKH  PO  OPA   PEAN KA E MHNA X PI  NA BA EI  KAPTA, X PI  NA KANEI  TI OTA!

OK
AT+CMGD=1
OK
AT+CMGL="ALL"
+CMGL: 0,"REC READ","+306978716080","","15/07/24,12:54:18+12"
Ma ti tha ginei
+CMGL: 2,"REC READ","+306974331832","","15/07/24,12:54:37+12"
Shmera
+CMGL: 3,"REC READ","6785105110102111","","15/08/01,13:07:41+12"
TA   PAKIA TOY MHNA! 600'  PO  CU + 600MB  IA 30MEPE  A O THN  OITHTIKH  PO  OPA   PEAN KA E MHNA X PI  NA BA EI  KAPTA, X PI  NA KANEI  TI OTA!

OK
AT+CNMI
OK
```

Εικόνα 3.2

Να δούμε η να αλλάξουμε τις ρυθμίσεις των κινητών συσκευών η των GSM/GPRS modem. Για παράδειγμα, για αν δούμε τις παραμέτρους του GSM δικτύου(AT+COPS), το είδος του διακομιστή (AT+CBST), για να ορίσουμε τις παραμέτρους των πρωτοκόλλων ραδιοζεύξης (AT+CRLP) και για να επιλέξουμε ποια μνήμη θα δούμε(Sim η συσκευή- AT+CPMS). Όλα τα προαναφερόμενα φαίνονται και στην Εικόνα 3.3 , όπου στην εντολή AT+CPMS=”SM”, το SM σημαίνει Sim.



Εικόνα 3.3

Να λάβουμε το σωστό σήμα GPS. Για παράδειγμα: Για να επιλέξουμε ποιο mode GPS θα χρησιμοποιήσουμε δηλαδή το Assisted(A-GPS) η το S-GPS(GPS με την χρήση servers κλπ(AT+CGPS), ζητάει πληροφορίες από το GPS(συντεταγμένες-AT+CGPSINFO) και για να στήσουμε έναν URL Server στο GPS(AT+CGPSURL).). Όλα τα προαναφερόμενα φαίνονται και στην Εικόνα 3.4.

Εικόνα 3.4

Τώρα ας δούμε πιο αναλυτικά το πώς χρησιμοποιήσαμε τα AT Commands στην πτυχιακή και πιο συγκεκριμένα στο πως χρησιμοποιήθηκαν για την από αποστολή και λήψη των SMS, την διαγραφή τους από την μνήμη της Sim, την λήψη του σήματος GPS κα σε πολλές άλλες λειτουργίες.

Για όλες τις AT Command χρησιμοποιήθηκαν δυο τρόποι: μέσω κώδικα και χειροκίνητα. Στον πρώτο τρόπο φορτώσαμε τον ανάλογο κώδικα στο Arduino και μετρά ανοίξαμε το Serial Monitor να δούμε εάν ήρθε η απορρίφθηκε μια κλήση, εάν ήρθε η έφυγε ένα μήνυμα κλπ. Στον δεύτερο τρόπο ανοίξαμε το Serial Monitor και δώσαμε χειροκίνητα τα AT Commands(μέσω του Send που υπάρχει εκεί), αφού φυσικά φορτώσαμε έναν κενό κώδικα στο Arduino Board(Το ίδιο ισχύει και για την χρήση του Putty).

Παρακάτω θα αναλυθούν και οι δυο τρόποι. Αξίζει να αναφέρουμε πως για να στείλουμε τις εντολές χειροκίνητα θα πρέπει τα Jumpers να είναι στην θέση USB στο Shield και για τον κώδικα στην αρχή θα πρέπει το ένα Jumper να είναι στην θέση USB(για την φόρτωση του κενού κώδικα όσο και για την χρήση των AT Commands) ώστε να φορτωθεί ο κώδικας και μετρά πρέπει να είναι στην θέση Ard ώστε να ξεκινήσει η ανάλογη λειτουργία(Ανάλογα τον κώδικα).

3.1 AT Commands Και SMS Μέσω Των Σημάτων 3G/GPRS

Για την αποστολή και λήψη SMS μηνυμάτων όσο και την διαγραφή τους χρησιμοποιήσαμε δυο τρόπους

Αποστολή SMS Μέσω Των AT Commands

1ος τρόπος(Μέσω κώδικα):

Στο Arduino μέσω του Arduino Ide φορτώσαμε τον παρακάτω κώδικα: (Εικόνες 3.5, 3.6, 3.7)

```
int8_t answer;
int onModulePin= 2;
char aux_str[30];

//Change here your data
const char pin[]="5916";
const char phone_number[]="+306978716080";
const char sms_text[]="Test-Arduino-Hello World";

void setup(){

  pinMode(onModulePin, OUTPUT);
  Serial.begin(115200);

  Serial.println("Starting...");
  power_on();

  delay(3000);

  //sets the PIN code
  sprintf(aux_str, "AT+CPIN=%s", pin);
  sendATcommand(aux_str, "OK", 2000);

  delay(3000);

  Serial.println("Connecting to the network...");

  while( (sendATcommand("AT+CREG?", "+CREG: 0,1", 500) ||
    sendATcommand("AT+CREG?", "+CREG: 0,5", 500)) == 0 );

  Serial.print("Setting SMS mode...");
  sendATcommand("AT+CMGF=1", "OK", 1000);    // sets the SMS mode to text
  Serial.println("Sending SMS");

  sprintf(aux_str,"AT+CMGS=\"%s\"", phone_number);
  answer = sendATcommand(aux_str, ">", 2000);    // send the SMS number
  if (answer == 1)
  {
    Serial.println(sms_text);
    Serial.write(0x1A);
    answer = sendATcommand("", "OK", 20000);
    if (answer == 1)
    {
      Serial.print("Sent ");
    }
```

Εικόνα 3.5

```
    else
    {
      Serial.print("error ");
    }
  }
  else
  {
    Serial.print("error ");
    Serial.println(answer, DEC);
  }

}


void loop(){

}

void power_on(){

  uint8_t answer=0;

  // checks if the module is started
  answer = sendATcommand("AT", "OK", 2000);
  if (answer == 0)
  {
    // power on pulse
    digitalWrite(onModulePin,HIGH);
    delay(3000);
    digitalWrite(onModulePin,LOW);

    // waits for an answer from the module
    while(answer == 0){      // Send AT every two seconds and wait for the answer
      answer = sendATcommand("AT", "OK", 2000);
    }
  }

}
```

Εικόνα 3.6

```
int8_t sendATcommand(char* ATcommand, char* expected_answer, unsigned int timeout){

  uint8_t x=0,  answer=0;
  char response[100];
  unsigned long previous;

  memset(response, '\0', 100);    // Initialice the string

  delay(100);

  while( Serial.available() > 0) Serial.read();    // Clean the input buffer

  Serial.println(ATcommand);    // Send the AT command


    x = 0;
  previous = millis();

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

  return answer;
}
```

Εικόνα 3.7

Ας δούμε πιο λεπτομερειακά τις παραπάνω εντολές(επισημασμένες) , από τις παραπάνω εικόνες.

Στην Εικόνα 3.5 υπάρχουν επισημασμένες οι εντολές:

Const char pin[]=""; Είναι ένας πίνακας χαρακτήρων(συμβολοσειρά) στον οποίο εισάγουμε το Pin ώστε να συνδεθεί η Sim στο δίκτυο της αντίστοιχης κινητής τηλεφωνίας.

Const char phone number[]=""; Εδώ στην ουσία εισάγουμε το νούμερο του παραλήπτη στον οποίο θέλουμε να στείλουμε το μήνυμα

"AT+CPIN=%"; Μας βοηθάει να δούμε στο Serial Monitor ότι έχει εισαχθεί το Pin

Connecting to the network… Εδώ στην ουσία βλέπουμε ότι έχει ξεκινήσει η σύνδεση με το εκάστοτε δίκτυο κινητής τηλεφωνίας

"AT+CREG?" Μας παρέχει πληροφορίες σχετικά με τον αν έχουμε συνδεθεί στο δίκτυο η όχι, και επίσης μας παρέχει πληροφορίες σχετικά με την τεχνολογία πρόσβασης των κυψελών που έχουμε συνδεθεί. ΠΧ: Αν η εντολή είναι AT+CREG=1,1 σημαίνει ότι ο συνδρομητής έχει συνδεθεί στο συμβατικό δίκτυο GSM

"AT+CMGF=1" Εδώ σε αυτή την εντολή ανάλογα τον αριθμό(0 ή 1) μας εμφανίζει το μήνυμα ως κείμενο(1) η ως μια σειρά από bytes(0-PDU Mode).

"AT+CMGS=\"%\"" Στην ουσία η εντολή αυτή είναι η εντολή η οποία κάνει όλη την δουλειά, δηλαδή μας στέλνει τα SMS μηνύματα.(Είτε σε κείμενο είτε σε PDU Mode).

Στην Εικόνα 3.6 υπάρχει επισημασμένη η εντολή:

Answer=sendATcommand("AT", "OK", 2000); Στην πρώτη περίπτωση το AT ακολουθείται από ένα OK που σημαίνει ότι το σύστημα(Arduino+ 3G/GPRS Shield) έχει μπει σε λειτουργία και στην δεύτερη περίπτωση αποστέλλονται κάθε δυο seconds το AT μέχρι να έρθει κάποια απάντηση από το σύστημα.

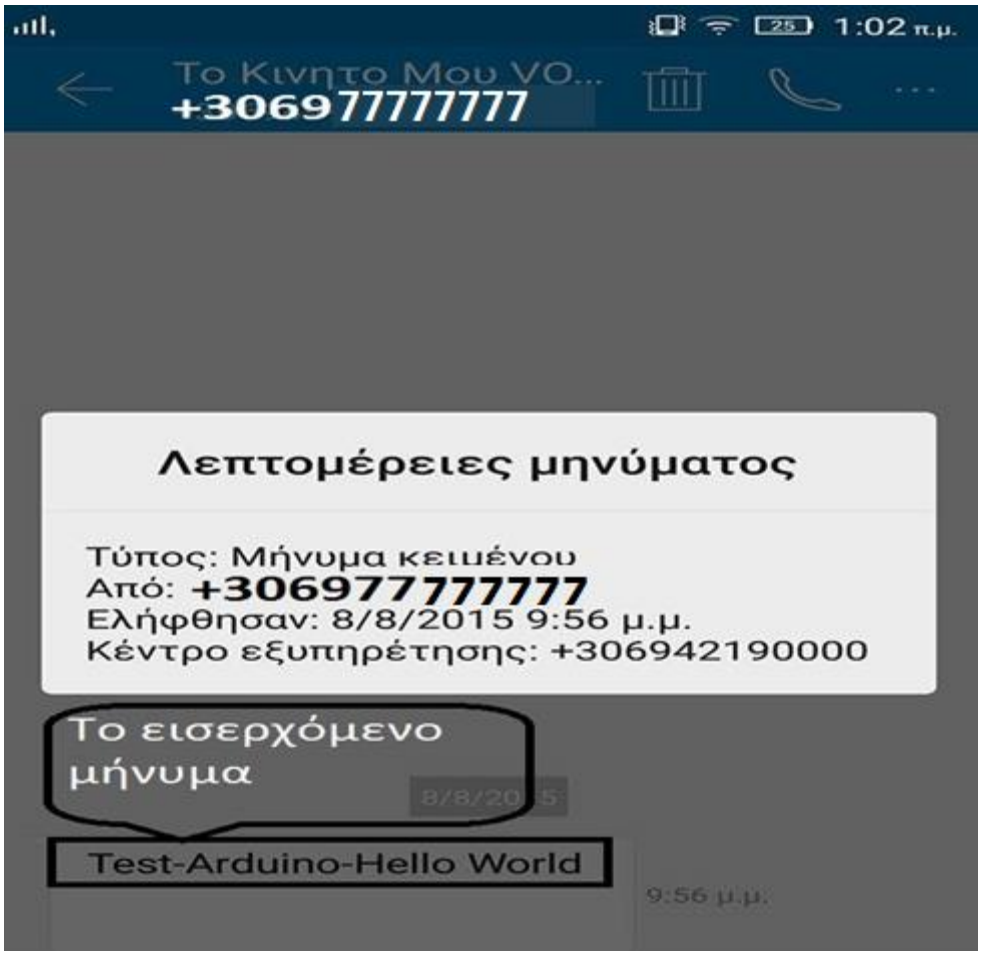
Στην Εικόνα 3.7 υπάρχει επισημασμένη η εντολή:

AT command Αυτή η εντολή μας εμφανίζει στην ουσία κάθε φορά το ποια εντολή τρέχει(από αυτές που προαναφέρθηκαν).

Το αποτέλεσμα φαίνεται στις Εικόνες 3.8 και 3.9

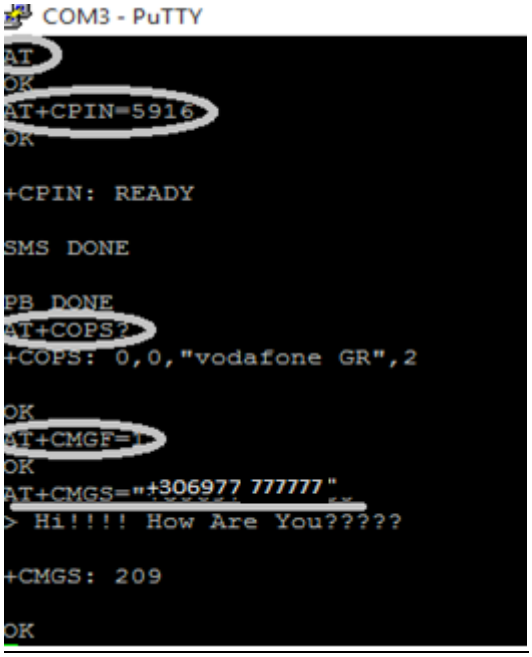
```
Starting...
Starting...
AT
AT
AT+CPIN=5916
Connecting to the network...
AT+CREG?
AT+CREG?
AT+CREG?
Setting SMS mode...AT+CMGF=1
Sending SMS
AT+CMGS="6977777777"
Test-Arduino-Hello World
□
Sent
```

Εικόνα 3.8

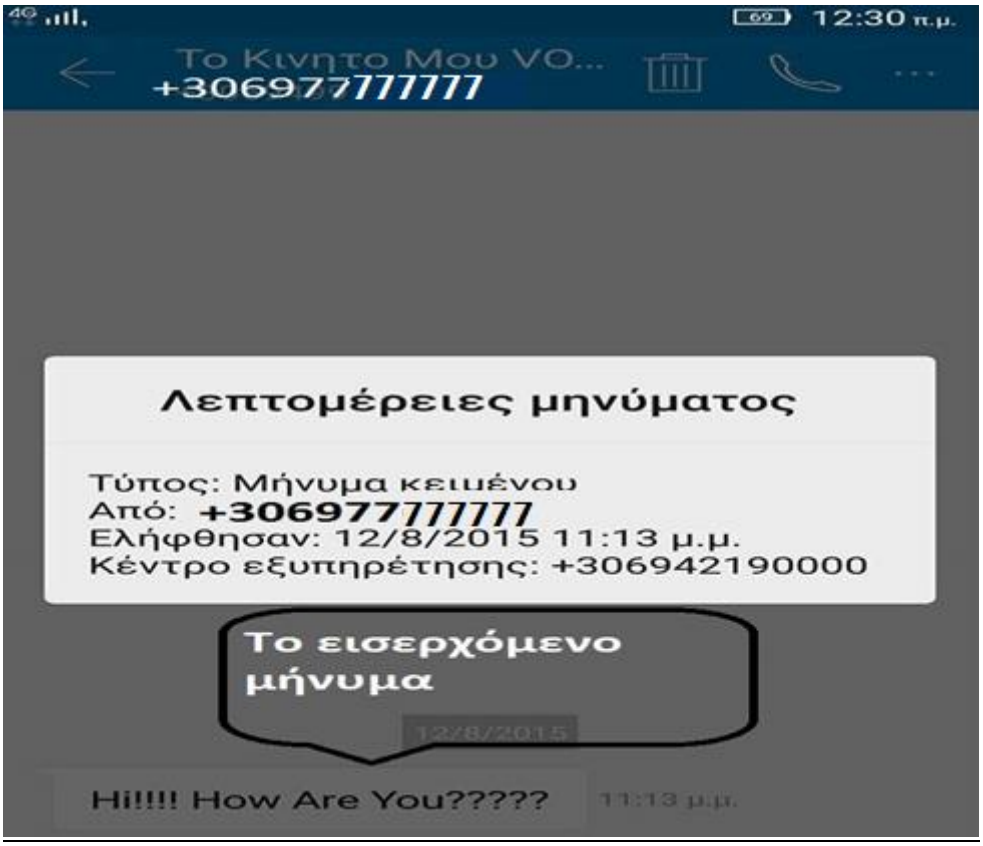


Εικόνα 3.9

2ος τρόπος(Εισαγωγή AT Commands χειροκίνητα):



Εικόνα 3.10

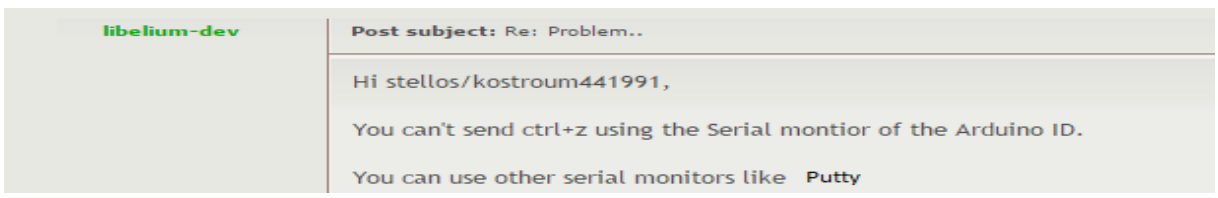


Εικόνα 3.11

Τώρα ας εξηγήσουμε το πώς στείλαμε ένα μήνυμα χειροκίνητα, οπού το αποτέλεσμα φαίνεται στις παραπάνω Εικόνες (310 και 3.11).

Αρχικά στο Putty βάλαμε την εντολή AT και πατήσαμε το Enter για να δούμε ότι το σύστημα λειτουργεί σωστά. Για να βεβαιωθούμε ότι λειτουργεί σωστά είδαμε από κάτω από το AT εάν OK. Έπειτα καταχωρήσαμε το Pin με την εντολή AT+CPIN=”Pin” και βεβαιωθήκαμε ότι συνδεθήκαμε στο δίκτυο μέσω της εντολής AT+COPS?. Επειδή θέλαμε να στείλουμε κείμενο βάλαμε την εντολή AT+CMGF=1. Εν συνέχεια βάλαμε την εντολή AT+CMGS=”+30Number”, και αφού πατήσαμε το Enter μας έβγαλε το > για να εισάγουμε το μήνυμα μας. Για να ολοκληρωθεί η σύνταξη του μηνύματος και να γίνει η αποστολή του πατήσαμε στο πληκτρολόγιο τον συνδυασμό Ctrl+Z. Και τέλος μπορούμε να δούμε ότι το μήνυμα στάλθηκε (στις παραπάνω εικόνες), και στην εντολή +CMGS:209 το 209 είναι ο αύξων αριθμός μηνυμάτων.

Να σχολιάσουμε σε αυτό το σημείο πως για να πραγματοποιηθεί η αποστολή του μηνύματος χειροκίνητα χρησιμοποιήσαμε το Putty και όχι το Arduino(Serial Monitor), και ο λόγος είναι ότι ο συνδυασμός **Ctrl+Z='\032' = '\x1A'** δεν υποστηρίζεται από το Arduino. Αυτή η διαπίστωση έγινε μέσω του Official Forum της εταιρίας του Hardware του οποίου διαθέταμε και η απάντηση τους φαίνεται στην Εικόνα 3.12.



Εικόνα 3.12

Λήψη SMS Μέσω Των AT Commands

1ος τρόπος(Μέσω κώδικα):

Στο Arduino μέσω του Arduino Ide φορτώσαμε τον παρακάτω κώδικα: (Εικόνες 3.13, 3.14, 3.15)

```
const char pin[]="5916";
int8_t answer;
int x;
int onModulePin= 2;
char SMS[200];
char aux_str[30];


void setup(){

  pinMode(onModulePin, OUTPUT);
  Serial.begin(115200);

  Serial.println("Starting...");
  power_on();

  delay(3000);

  //sets the PIN code
  sprintf(aux_str, "AT+CPIN=%s", pin);
  sendATcommand(aux_str, "OK", 2000);

  delay(3000);

  Serial.println("Setting SMS mode...");
  sendATcommand("AT+CMGF=1", "OK", 1000);    // sets the SMS mode to text
  sendATcommand("AT+CPMS=\"SM\",\"SM\",\"SM\"", "OK", 1000);    // selects the memory

  answer = sendATcommand("AT+CMGR=1", "+CMGR:", 2000);    // reads the first SMS
  if (answer == 1)
  {
    answer = 0;
    while(Serial.available() == 0);
    // this loop reads the data of the SMS
    do{
      // if there are data in the UART input buffer, reads it and checks for the answer
      if(Serial.available() > 0){
        SMS[x] = Serial.read();
        x++;
        // check if the desired answer (OK) is in the response of the module
        if (strstr(SMS, "OK") != NULL)
        {
          answer = 1;
        }
      }
    }
```

Εικόνα 3.13

```
    while(answer == 0);     // Waits for the answer with time out

    SMS[x] = '\0';

    Serial.print(SMS);

  }
  else
  {
    Serial.print("error ");
    Serial.println(answer, DEC);
  }


}

void loop(){

}

void power_on(){

  uint8_t answer=0;

  // checks if the module is started
  answer = sendATcommand("AT", "OK", 2000);
  if (answer == 0)
  {
    // power on pulse
    digitalWrite(onModulePin,HIGH);
    delay(3000);
    digitalWrite(onModulePin,LOW);

    // waits for an answer from the module
    while(answer == 0){     // Send AT every two seconds and wait for the answer
      answer = sendATcommand("AT", "OK", 2000);
    }
  }

}
```

Εικόνα 3.14

```
int8_t sendATcommand(char* ATcommand, char* expected_answer, unsigned int timeout){

  uint8_t x=0,  answer=0;
  char response[100];
  unsigned long previous;

  memset(response, '\0', 100);    // Initialice the string

  delay(100);

  while( Serial.available() > 0) Serial.read();    // Clean the input buffer

  Serial.println(ATcommand);    // Send the AT command


    x = 0;
  previous = millis();

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

  return answer;
}
```

Εικόνα 3.15

Ας δούμε πιο λεπτομερειακά τις παραπάνω εντολές(επισημασμένες) , από τις παραπάνω εικόνες.

Στην Εικόνα 3.12 υπάρχουν επισημασμένες οι εντολές:

Const char pin[]=""; Είναι ένας πίνακας χαρακτήρων(συμβολοσειρά) στον οποίο εισάγουμε το Pin ώστε να συνδεθεί η Sim στο δίκτυο της αντίστοιχης κινητής τηλεφωνίας.

"AT+CPIN=%"; Μας βοηθάει να δούμε στο Serial Monitor ότι έχει εισαχθεί το Pin

Connecting to the network… Εδώ στην ουσία βλέπουμε ότι έχει ξεκινήσει η σύνδεση με το εκάστοτε δίκτυο κινητής τηλεφωνίας

"AT+CMGF=1" Εδώ σε αυτή την εντολή ανάλογα τον αριθμό(0 ή 1) μας εμφανίζει το μήνυμα ως κείμενο(1) η ως μια σειρά από bytes(0-PDU Mode).

"AT+CPMS=\"SM\",\"SM\",\"SM\"" Σε αυτήν την γραμμή στην ουσία το σύστημα επιλεγεί από πού θα διαβάσει το εισερχόμενο μήνυμα , και συγκεκριμένα από την Sim η από την μνήμη του τηλεφώνου.

Να προσθέσουμε πως αντί για το SM μπορούσαμε να χρησιμοποιήσουμε και άλλες εντολές.

*ME

Εμφανίζει τα μηνύματα από ένα Modem η μια συσκευή τηλεφώνου. Ο αριθμός των μηνυμάτων που μπορούν να αποθηκευτούν εξαρτάται από την εσωτερική μνήμη της συσκευής.

*****MT

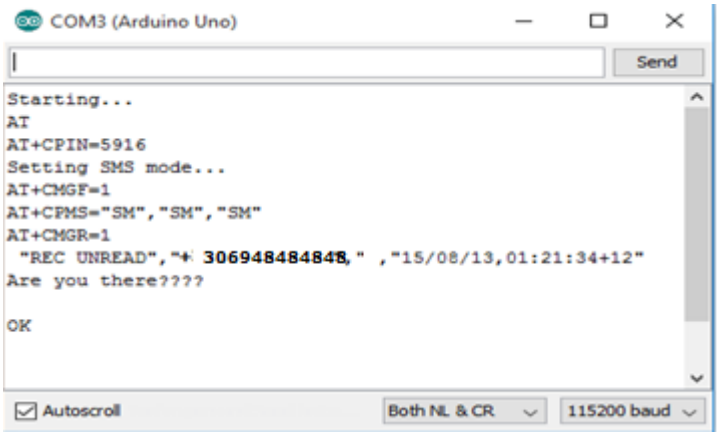
Μας εμφανίζει τα μηνύματα απ' όλες τις μνήμες της συσκευής(Sim και εσωτερική). Όταν η συσκευή υποστηρίζει και τα δυο είδη μνήμων η εντολή MT χειρίζεται τις μνήμες σαν να ήταν μια.

*BM

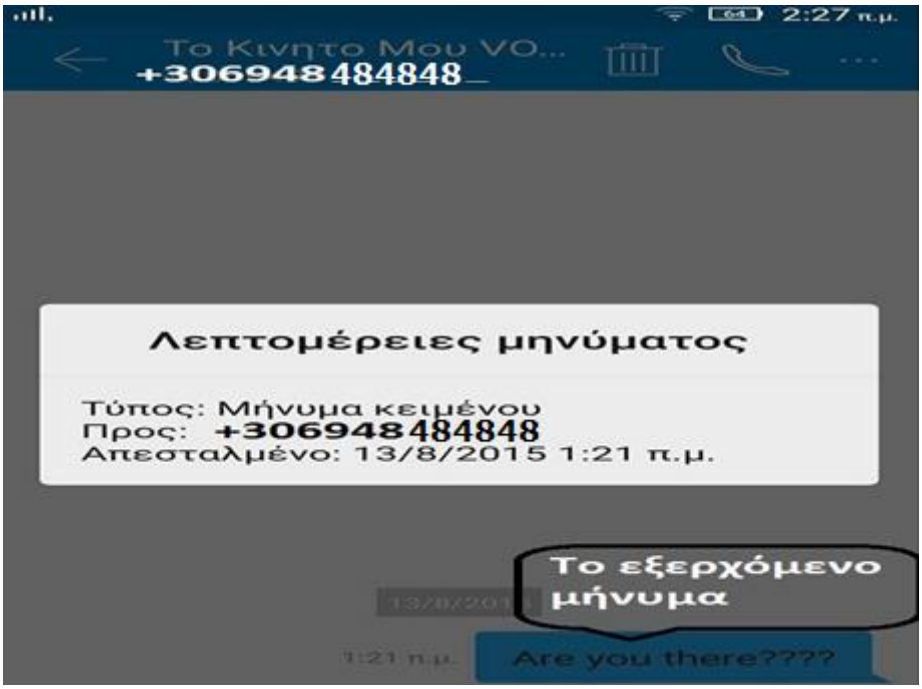
Μας εμφανίζει τα είδη αποθηκευμένα μηνύματα. Δεν χρησιμοποιείται για να αποθηκεύει SMS μηνύματα.

*SR
Αν έχουμε ορίσει να έρχεται αναφορά σχετικά με την κατάσταση του μηνύματος(Αν έχει αποσταλεί η όχι), αυτά τα μηνύματα αποθηκεύονται εκεί. Τα μηνύματα αυτά εμφανίζονται με τον ίδιο τρόπο που εμφανίζονται και τα μηνύματα SMS.
"AT+CMGR=X" Αφού έχουμε επιλέξει από πού θα διαβάσουμε το μήνυμα(Sim η μνήμη τηλεφώνου), ανάλογα το X λεμέ στο σύστημα να μας εμφανίσει και το ανάλογο μήνυμα. Πχ: Αν X=15 τότε λεμέ στο σύστημα να μας εμφανίσει το 15^{ο} μη διαβασμένο μήνυμα.
Στην Εικόνα 3.13 υπάρχει επισημασμένη η εντολή:
Answer=sendATcommand("AT", "OK", 2000); Στην πρώτη περίπτωση το AT ακολουθείται από ένα OK που σημαίνει ότι το σύστημα(Arduino+ 3G/GPRS Shield) έχει μπει σε λειτουργία και στην δεύτερη περίπτωση αποστέλλονται κάθε δυο seconds το AT μέχρι να έρθει κάποια απάντηση από το σύστημα.
Στην Εικόνα 3.14 υπάρχει επισημασμένη η εντολή:
AT command Αυτή η εντολή μας εμφανίζει στην ουσία κάθε φορά το ποια AT εντολή τρέχει.

Το αποτέλεσμα φαίνεται στις Εικόνες 3.16 και 3.17



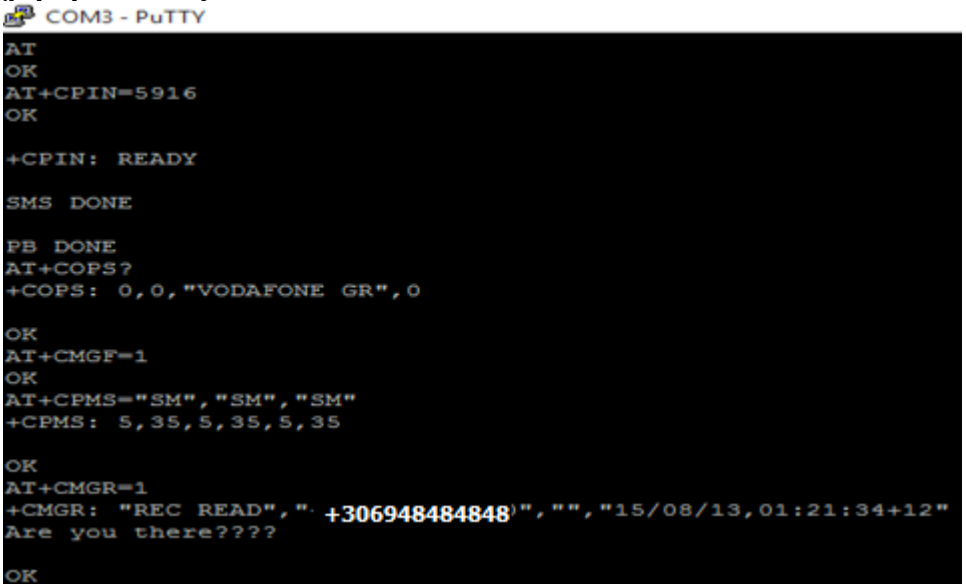
Εικονα 3.16



Εικόνα 3.17

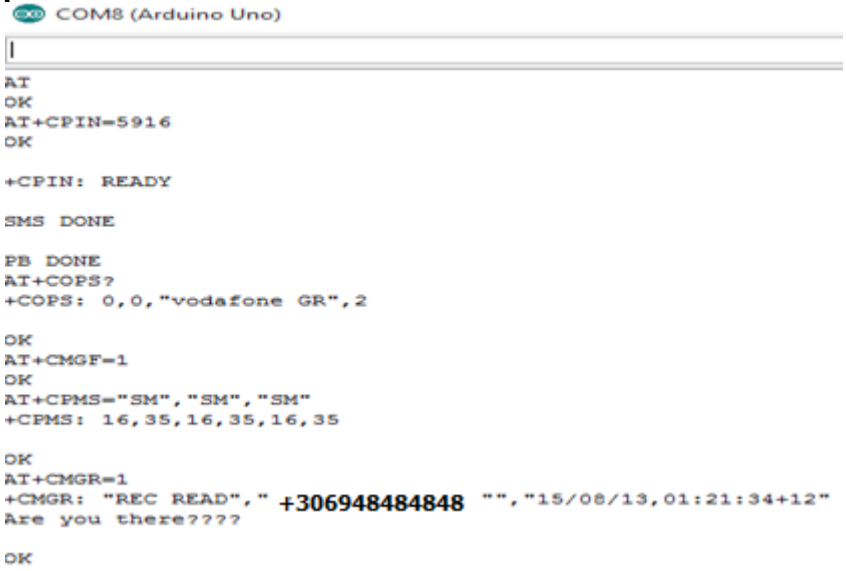
2ος τρόπος(Εισαγωγή AT Commands χειροκίνητα):

Με την χρήση του Putty



Εικόνα 3.18

Με την χρήση του Arduino Ide



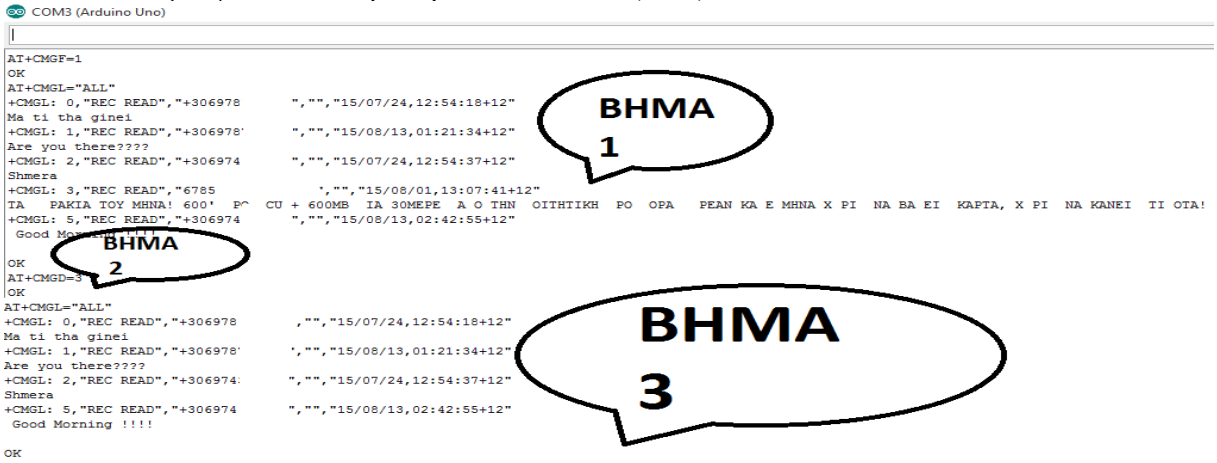
Εικόνα 3.19

Το πώς έγινε η χειροκίνητη διαδικασία(Εικόνες 3.18 και 3.19) δεν χρειάζεται να εξηγήσουμε κάτι παραπάνω αφού όλες τις εντολές τις έχουμε εξηγήσει στο απερχόμενο κομμάτι Απλά αυτό που πρέπει να τονίσουμε σε αυτό το σημείο είναι το εξής: αν παρατηρήσουμε τις Εικόνες 3.15 και 3.17 θα δούμε ότι στο αποτέλεσμα στην μια υπάρχει "REC UNREAD" και στην άλλη "REC READ".αλλά για το ίδιο μήνυμα. Αυτό συμβαίνει γιατί όταν ανοίξαμε το Serial Monitor του Arduino το μήνυμα το οποίο είχαμε στείλει δεν είχε διαβαστεί γι' αυτό υπήρχε το UNREAD(Αδιάβαστο). Όμως μετά που ανοίξαμε το Putty και ξανατρέξαμε χειροκίνητα τις εντολές το μήνυμα είχε ήδη διαβαστεί από το Arduino γι' αυτό αυτή την φορά μας το εμφάνισε με READ(Διαβασμένο).

Διαγραφή SMS Μέσω AT Commands

Μέχρι τώρα είδαμε το πώς μπορούμε να στείλουμε και να λάβουμε μηνύματα μέσω των AT Commands. Τώρα θα δούμε το πώς μπορούμε να αδειάσουμε την μνήμη συσκευής(Εσωτερική η Sim). Η διαφορά με τα προηγούμενα είναι ότι αυτό το κάνουμε μόνο χειροκίνητα , και όχι με κώδικα. Ο λόγος είναι ότι θέλουμε ο χρήστης να βλέπει ο ίδιος όλα τα μηνύματα που υπάρχουν στην συσκευή και ανάλογα με τον αύξων αριθμό του μηνύματος να επιλεγεί πιο θα διαγράψει.

Το αποτέλεσμα φαίνεται στην παρακάτω Εικόνα(3.20).



Εικόνα 3.20

Το άδειασμα της μνήμης όπως βλέπουμε από την παραπάνω εικόνα γίνεται σε 3 βήματα. Στο πρώτο βήμα χρησιμοποιούμε πρώτα απ' όλα την εντολή AT+CMGF=1 γιατί θέλουμε να δούμε όλα τα μηνύματα κειμένου που υπάρχουν στην Sim. Μετά χρησιμοποιήσαμε την εντολή AT+CMGL="ALL", οπού το L σημαίνει List και μας εμφανίζει όλα τα μηνύματα που υπάρχουν στην μνήμη. Στο δεύτερο βήμα χρησιμοποιήσαμε την εντολή AT+CMGD="X" που X ο αριθμός του μηνύματος που θέλουμε να διαγράψουμε(D=Delete). Στην παραπάνω εικόνα θέλαμε να διαγράψουμε το μήνυμα 3. Και τέλος στο βήμα 3 βλέπουμε πως το μήνυμα έχει διαγραφεί από την μνήμη.

3.2 AT Commands Και Λήψη Συντεταγμένων Μέσω Του Συστήματος GPS

1ος τρόπος(Μέσω κώδικα):

Στο Arduino μέσω του Arduino Ide φορτώσαμε τον παρακάτω κώδικα: (Εικόνες 3.21, 3.22, 3.23)

```
int8_t answer;
int onModulePin= 2;
char gps_data[100];
int counter;

void setup(){

  pinMode(onModulePin, OUTPUT);
  Serial.begin(115200);

  Serial.println("Starting...");
  power_on();

  delay(5000);

  // starts GPS session in stand alone mode
  answer = sendATcommand("AT+CGPS=1,1","OK",1000);
  if (answer == 0)
  {
    Serial.println("Error starting the GPS");
    Serial.println("The code stucks here!!");
    while(1);
  }
}
```

Εικόνα 3.21

```
void loop(){

  answer = sendATcommand("AT+CGPSINFO","+CGPSINFO:",1000);     // request info from GPS
  if (answer == 1)
  {

    counter = 0;
    do{
      while(Serial.available() == 0);
      gps_data[counter] = Serial.read();
      counter++;
    }
    while(gps_data[counter - 1] != '\r');
    gps_data[counter] = '\0';
    if(gps_data[0] == ',')
    {
      Serial.println("No GPS data available");
    }
    else
    {
      Serial.print("GPS data:");
      Serial.print(gps_data);
      Serial.println("");
    }

  }
  else
  {
    Serial.println("Error");
  }

  delay(5000);
}

void power_on(){

  uint8_t answer=0;

  // checks if the module is started
  answer = sendATcommand("AT", "OK", 2000);
  if (answer == 0)
  {
    // power on pulse
    digitalWrite(onModulePin,HIGH);
    delay(3000);
```

Εικόνα 3.22

```
digitalWrite(onModulePin,LOW);

    // waits for an answer from the module
    while(answer == 0){
      // Send AT every two seconds and wait for the answer
      answer = sendATcommand("AT", "OK", 2000);
    }
  }

}


int8_t sendATcommand(char* ATcommand, char* expected_answer1, unsigned int timeout)
{

  uint8_t x=0,  answer=0;
  char response[100];
  unsigned long previous;

  memset(response, '\0', 100);     // Initialize the string

  delay(100);

  while( Serial.available() > 0) Serial.read();     // Clean the input buffer

  Serial.println(ATcommand);     // Send the AT command
k = 0;
  previous = millis();

  // this loop waits for the answer
  do{

    if(Serial.available() != 0){
      response[x] = Serial.read();
      x++;
      // check if the desired answer is in the response of the module
      if (strstr(response, expected_answer1) != NULL)
      {
        answer = 1;
      }
    }
    // Waits for the answer with time out
  }
  while((answer == 0) && ((millis() - previous) < timeout));
return answer;
}
```

Εικόνα 3.23

Ας δούμε πιο λεπτομερειακά τις παραπάνω εντολές(επισημασμένες) , από τις παραπάνω εικόνες.

Στην Εικόνα 3.21 υπάρχει επισημασμένη η εντολή:

“AT+CGPS=1,1”: Αυτή η εντολή στην ουσία υπάρχει ώστε να ξεκινήσει η λειτουργία του GPS και επίσης ορίζουμε το ποιο GPS θα χρησιμοποιήσουμε (Stand Alone η A η S GPS). Γενικά η σύνταξη της εντολής είναι το εξής: AT+CGPS=”input,input”(Inputs= 0,0, 0,1,1,1, 1,0)

Στην Εικόνα 3.22 υπάρχουν επισημασμένες οι εντολές:

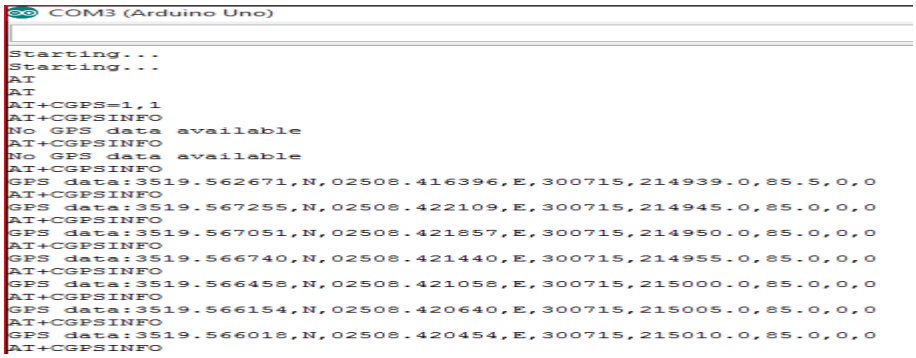
Answer=sendATcommand(“AT”, “OK”, 2000); Στην πρώτη περίπτωση το AT ακολουθείται από ένα OK που σημαίνει ότι το σύστημα(Arduino+ 3G/GPRS Shield) έχει μπει σε λειτουργία και στην δεύτερη περίπτωση αποστέλλονται κάθε δυο seconds το AT μέχρι να έρθει κάποια απάντηση από το σύστημα.

“AT+CGPSINFO”,”+CGPSINFO” Αυτή η γραμμή στην ουσία κάνει όλη την δουλειά. Δηλαδή παίρνει ανά δευτερόλεπτο τις συντεταγμένες μέσω του GPS. Οι γραμμές παρακάτω μας εμφανίζουν στην οθόνη(Serial Monitor) τα δεδομένα η όχι.

Στην Εικόνα 3.23 υπάρχει επισημασμένη η εντολή:

AT command Αυτή η εντολή μας εμφανίζει στην ουσία κάθε φορά το ποια AT εντολή τρέχει.

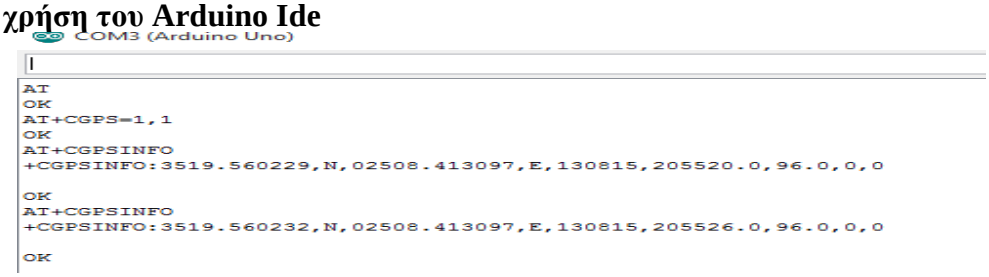
Το αποτέλεσμα φαίνεται στην Εικόνα 3.24



Εικόνα 3.24

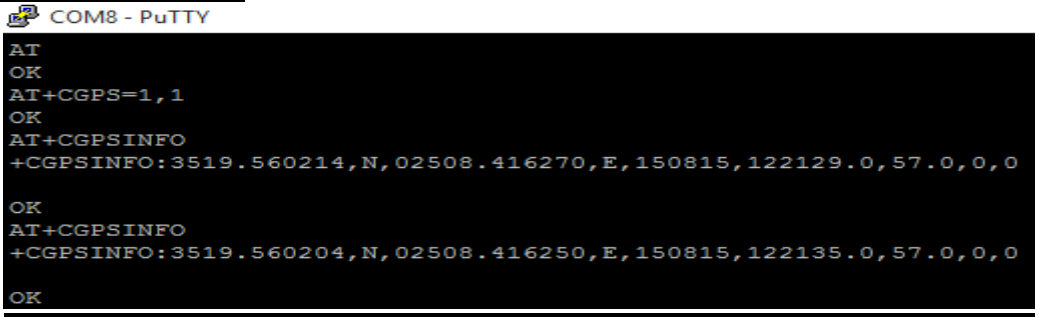
2ος τρόπος(Εισαγωγή AT Commands χειροκίνητα):

Με την χρήση του Arduino Ide



Εικόνα 3.25

Με την χρήση του Putty



Εικόνα 3.26

Το πώς έγινε η χειροκίνητη διαδικασία(Εικόνες 3.25 και 3.26) δεν χρειάζεται να εξηγήσουμε κάτι παραπάνω αφού όλες τις εντολές τις έχουμε εξηγήσει παραπάνω.

Για την πραγματοποίηση όλων των παραπάνω αξίζει να αναφέρουμε πως και μέσω κώδικα και μέσω χειροκίνητου τρόπου έγινε η ιδιά διακόσια με τα SMS.

3.3 AT Commands Και Πραγματοποίηση Κλήσεων

Λαμβάνοντας Κλήσεις Μέσω AT Commands

1ος τρόπος(Μέσω κώδικα):

Στο Arduino μέσω του Arduino Ide φορτώσαμε τον παρακάτω κώδικα: (Εικόνες 3.27, 3.28)\

Εικόνα 3.27

```
void power_on() {
uint8_t answer = 0;
// checks if the module is started
answer = sendATcommand("AT", "OK", 2000);
if (answer == 0)
{
// power on pulse
digitalWrite(onModulePin, HIGH);
delay(3000);
digitalWrite(onModulePin, LOW);

// waits for an answer from the module
while (answer == 0) { // Send AT every two seconds and wait for the answer
answer = sendATcommand("AT", "OK", 2000);
}
}
}
int8_t sendATcommand(char* ATcommand, char* expected_answer, unsigned int timeout) {
uint8_t x = 0, answer = 0;
char response[100];
unsigned long previous;

memset(response, '\0', 100); // Initialize the string

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

Εικόνα 3.28

Ας δούμε πιο λεπτομερειακά τις παραπάνω εντολές(επισημασμένες) , από τις παραπάνω εικόνες.

Στην Εικόνα 3.27 υπάρχουν επισημασμένες οι εντολές:

Cost char pin[]=""; Είναι ένας πίνακας χαρακτήρων(συμβολοσειρά) στον οποίο εισάγουμε το Pin ώστε να συνδεθεί η Sim στο δίκτυο της αντίστοιχης κινητής τηλεφωνίας.

Στην Εικόνα 3.28 υπάρχουν επισημασμένες οι εντολές:

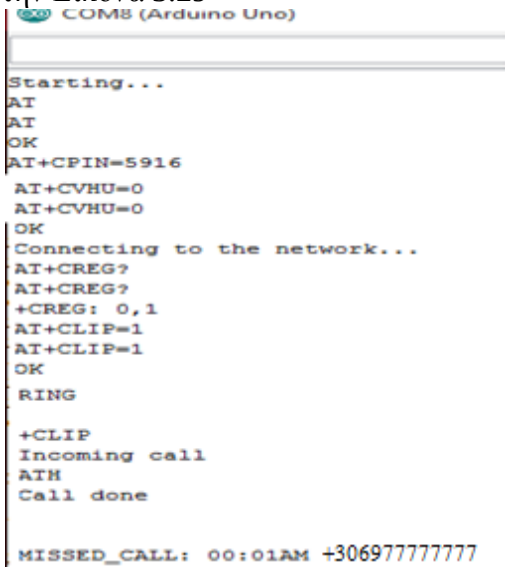
"AT+CPIN=%"; Μας βοηθάει να δούμε στο Serial Monitor ότι έχει εισαχθεί το Pin

"AT+CVHU=value" Αυτή η εντολή αποφασίζει ποτέ η εντολή ATH θα πραγματοποιήσει κλήση, ποτέ θα διακοπεί η όχι. Αν το value είναι ίσο με 0 τότε το Drop DTR αγνοείται αλλά παίρνουμε ως απάντηση ένα OK. Το ATH δεν λειτουργεί σε αυτήν την περίπτωση. Αν το value είναι ίσο με 1 τότε το Drop DTR και το ATH δεν λειτουργούν αλλά παίρνουμε ως απάντηση ένα OK. Κι αν η τιμή του value είναι ίσο με 2 τότε η συμπεριφορά του Drop DTR εξαρτάται ανάλογα με τις ρυθμίσεις του &D, εδώ όπως και στην πρώτη περίπτωση το ATH απενεργοποιείται.

"AT+CREG?" Μας παρέχει πληροφορίες σχετικά με τον αν έχουμε συνδεθεί στο δίκτυο η όχι, και επίσης μας παρέχει πληροφορίες σχετικά με την τεχνολογία πρόσβασης των κυψελών που έχουμε συνδεθεί. ΠΧ: Αν η εντολή είναι AT+CREG=1,1 σημαίνει ότι ο συνδρομητής έχει συνδεθεί στο συμβατικό δίκτυο GSM

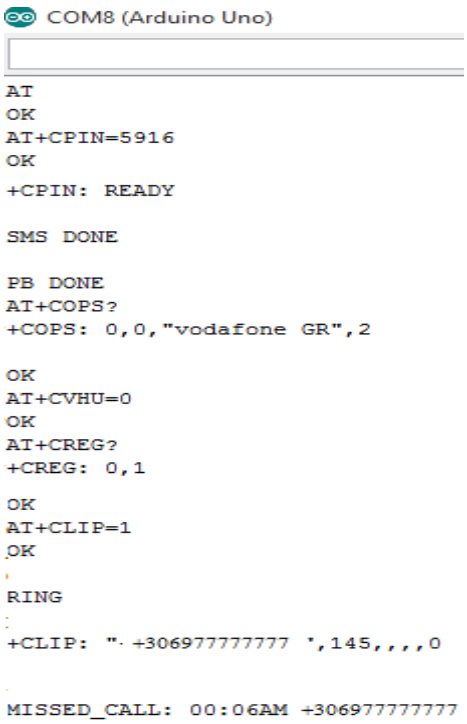
"AT+CLIP=1" Αναγνώρισή και προβολή της εισερχόμενης κλήσης

Στην Εικόνα 3.28 υπάρχουν επισημασμένες οι εντολές :
Answer=sendATcommand("AT", "OK", 2000); Στην πρώτη περίπτωση το AT ακολουθείται από ένα OK που σημαίνει ότι το σύστημα(Arduino+ 3G/GPRS Shield) έχει μπει σε λειτουργία και στην δεύτερη περίπτωση αποστέλλονται κάθε δυο seconds το AT μέχρι να έρθει κάποια απάντηση από το σύστημα.
AT command Αυτή η εντολή μας εμφανίζει στην ουσία κάθε φορά το ποια AT εντολή τρέχει.
Serial.println(response); Αυτή η γραμμή μπήκε για τον λόγο ότι ο κώδικας της Cooking Hacks δεν μας έδινε τον απαραίτητο έλεγχο που θα έπρεπε όσον αφορά στις τιμές επιστροφής των AT Commands.
Το αποτέλεσμα φαίνεται στην Εικόνα 3.29



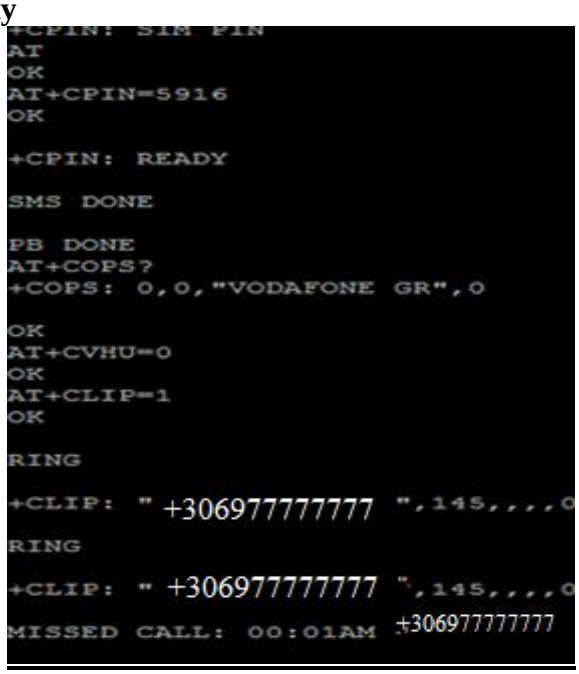
Εικόνα 3.29

2ος τρόπος(Εισαγωγή AT Commands χειροκίνητα):
Με την χρήση του Arduino Ide



Εικόνα 3.30

Με την χρήση του Putty



Εικόνα 3.31

Οι περισσότερες εντολές είναι σύμφωνα με τα προηγούμενα που -περιγράψαμε παραπάνω και για αυτό και δεν χρειάζεται να σχολιάσουμε κάτι αφού ότι έγινε στην χειροκίνητη διαδικασία(Εικόνες 3.30 και 3.31) στην ουσία έχει εξηγηθεί στην διαδικασία μέσω κώδικα. Το μόνο που αξίζει να σχολιάσουμε είναι αυτό που μας έχει εμφανίσει διπλά από το τηλεφωνικό νούμερο στην Εικόνα 3.31, το 145. Γενικά υπάρχουν δυο περιπτώσεις τις οποίες θα μπορούσε να μας εμφανίσει (129 η 145). Το 129 είναι στην επικοινωνία ISDN(Τοπική –Local Service) και το 145 είναι για διεθνής κλήσεις(International Services).

Πραγματοποιώντας Κλήσεις Μέσω AT Commands

1ος τρόπος(Μέσω κώδικα):

Στο Arduino μέσω του Arduino Ide φορτώσαμε τον παρακάτω κώδικα: (Εικόνες 3.32, 3.33,3.34)

```
int8_t answer;
int onModulePin = 2;
int button = 12;
char aux_str[30];
const char pin[] = "5916";
const char phone_number[] = "+306977696969";    // ********* is the number to call
void setup() {
  pinMode(onModulePin, OUTPUT);
  pinMode(button, INPUT);
  Serial.begin(115200);
  Serial.println("Starting...");
  power_on();
  delay(3000);
  sprintf(aux_str, "AT+CPIN=%s", pin);
  sendATcommand(aux_str, "OK", 2000);
  delay(3000);
  Serial.println("Connecting to the network...");

  sendATcommand("AT+CVHU=0", "OK", 10000);

  while ( (sendATcommand("AT+CREG?", "+CREG: 0,1", 500) ||
           sendATcommand("AT+CREG?", "+CREG: 0,5", 500)) == 0 );
  Serial.print("Calling to ");
  Serial.print(phone_number);
  Serial.println("...Press button to hang");
  sprintf(aux_str, "ATD%s;", phone_number);
  sendATcommand(aux_str, "OK", 10000);
  while (digitalRead(button) == 1);
  Serial.println("ATH");
  Serial.println("Call disconnected");
}
```

Εικόνα 3.32

```
void loop() {

}

void power_on() {

  uint8_t answer = 0;

  // checks if the module is started
  answer = sendATcommand("AT", "OK", 2000);
  if (answer == 0)
  {
    // power on pulse
    digitalWrite(onModulePin, HIGH);
    delay(3000);
    digitalWrite(onModulePin, LOW);

    // waits for an answer from the module
    while (answer == 0) {   // Send AT every two seconds and wait for the answer
      answer = sendATcommand("AT", "OK", 2000);
    }
  }

}
```

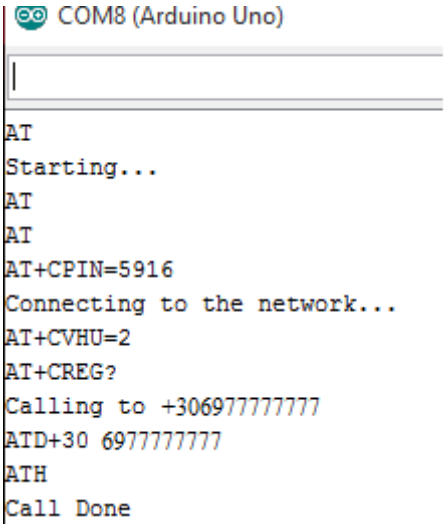
Εικόνα 3.33

```
int8_t sendATcommand(char* ATcommand, char* expected_answer, unsigned int timeout) {

  uint8_t x = 0,  answer = 0;
  char response[100];
  unsigned long previous;

  memset(response, '\0', 100);    // Initialize the string

  delay(100);

  while ( Serial.available() > 0) Serial.read();   // Clean the input buffer

  Serial.println(ATcommand);    // Send the AT command


  x = 0;
  previous = millis();

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

  return answer;
}
```

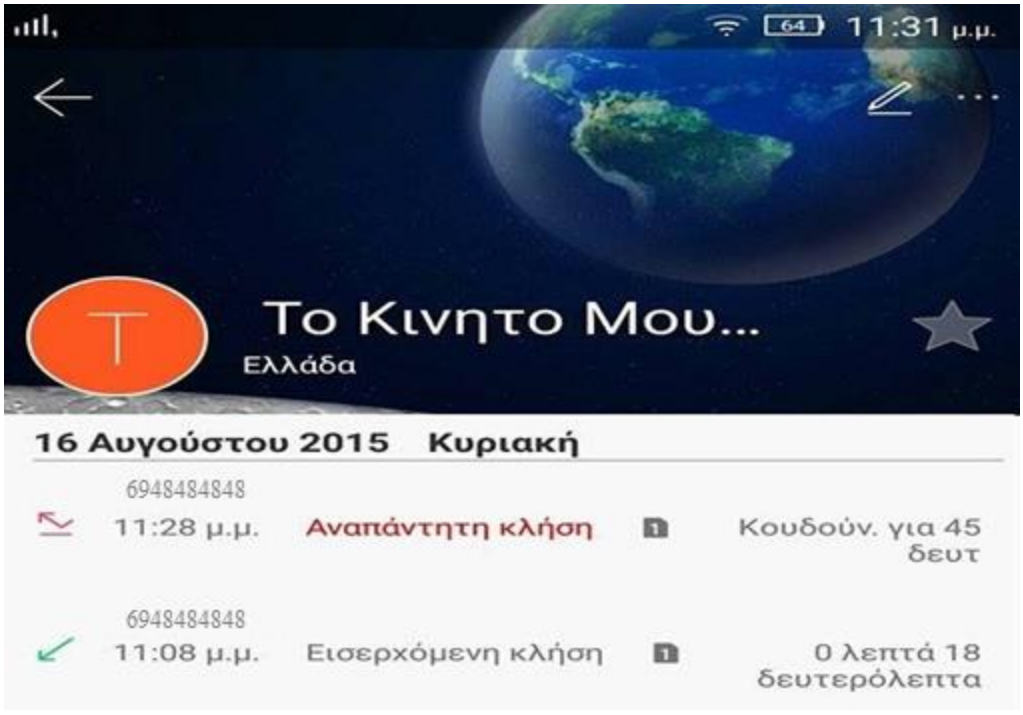
Εικόνα 3.34

Ας δούμε πιο λεπτομερειακά τις παραπάνω εντολές(επισημασμένες) , από τις παραπάνω εικόνες.

Σχετικά με τις παραπάνω εικόνες δεν χρειάζεται να σχολιάσουμε κάτι στις εντολές . Και ο λόγος είναι ότι όλες οι εντολές έχουν χρησιμοποιηθεί για την πραγματοποίηση και λήψη κλήσεων, για την αποστολή και λήψη μηνυμάτων κλπ. Η μονή εντολή που χρησιμοποιήθηκε παραπάνω σε σχέση με τους προηγουμένους κώδικες είναι η εντολή ATD. Η εντολή ATD είναι αυτή η οποία στην ουσία "πληκτρολογεί" το νούμερο προς κλήση. Αν η κλήση δεν πραγματοποιηθεί σωστά επιστρέφει τις εξής τιμές: 1 δεν υπάρχει ο συνδρομητής η απασχολημένος ο καλούμενος συνδρομητής . 2 και σημαίνει ότι υπάρχει γενικό σφάλμα ,3 και σημαίνει ότι υπάρχουν λόγοι ασφάλειας και η κλήση δεν μπορεί να πραγματοποιηθεί όπως πχ: δεν υπάρχει SIM, και 4 υπάρχει άγνωστη αίτια. Το αποτέλεσμα φαίνεται στις Εικόνες 3.35 και 3.36.



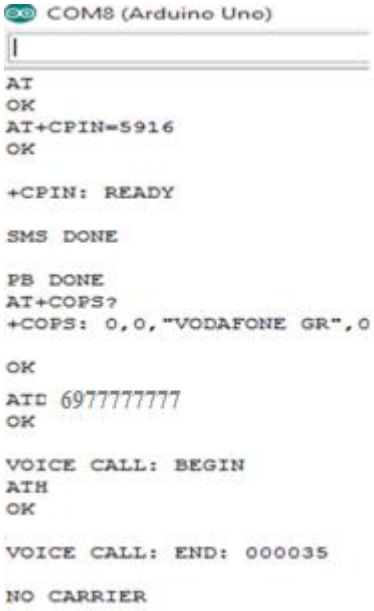
Εικόνα 3.35



Εικόνα 3.36

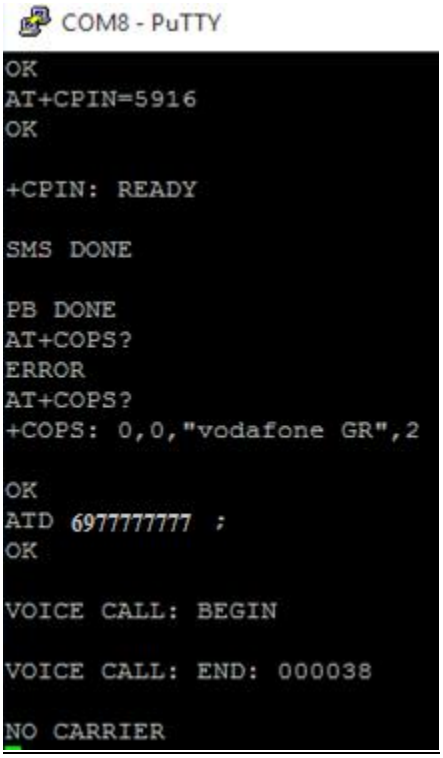
2ος τρόπος(Εισαγωγή AT Commands χειροκίνητα):

Με την χρήση του Arduino Ide



Εικόνα 3.37

Με την χρήση του Putty



Εικόνα 3.38

Στον χειροκίνητο τρόπο με το οποίον πραγματοποιήθηκαν οι κλήσεις δεν χρειάζεται να αναφέρουμε κάτι καθώς όλα έχουν αναφερθεί παραπάνω.

ΚΕΦΑΛΑΙΟ 4

Στο κεφάλαιο αυτό θα γίνει αναφορά στην σειριακή επικοινωνίας της Java.

4.1.1 Εισαγωγή

Ένα από τα μεγαλύτερα πλεονεκτήματα της JAVA είναι η δυνατότητα να τρέχει ανεξάρτητα από οποιαδήποτε πλατφόρμα. Δυστυχώς αυτή η δύναμη της JAVA γίνεται η αχίλλειος πτέρνα όσον αναφορά την χρήση της πάνω σε συστήματα σειριακής επικοινωνίας μιας και αυτό προϋποθέτει την ύπαρξη ειδικευμένων για την κάθε πλατφόρμα API , των οποίων η υλοποίηση είναι δύσκολη υπόθεση.

Δυστυχώς η Sun δεν έδωσε μεγάλη σημασία σχετικά με την σειριακή επικοινωνία της JAVA. Στην αρχή προσδιόρισε ένα API σειριακής επικοινωνίας το JavaComm, αλλά η υλοποίηση του δεν έγινε ποτέ μέρος κάποιας κύριας έκδοσης της JAVA. Συγκεκριμένα από το τέλος του 2005 η Sun σιωπηλά απέσυρε την υποστήριξη του JavaComm για τα Windows .Παρόλο που υπάρχει κάποια υποστήριξη για τις "παραμελημένες" πλατφόρμες κυρίως από τρίτους παράγοντες , από την μεριά της η Sun την είχε κρατήσει στο ελάχιστο δυνατό επίπεδο , με εξαίρεση την έκδοση του Sun Ray και αυτό λόγο πιέσεων των πελατών της. Το γεγονός ότι η Sun από την αρχή δεν έδωσε κάποια υλοποίηση του JavaComm για τις πλατφόρμες που βασίζονται σε Linux , οδήγησε στην ανάπτυξη μιας ανεξάρτητης , ανοικτού κώδικα βιβλιοθήκης, της RXTX , η όποια ήταν λειτουργική για αρκετές πλατφόρμες , και όχι μόνο για τις βασιζόμενες σε Linux . Το RXTX μπορεί να λειτουργήσει είτε σε παράλληλη λειτουργία με το JavaComm, είτε ανεξάρτητα από αυτό . Όταν το RXTX δουλεύει παράλληλα με το JavaComm τότε χρησιμοποιείται το JCL (JavaComm for Linux) το όποιο αποτελεί μέρος της διανομής του RXTX.

Η αδιαφορία της Sun όσων αφορά το JavaComm , και γενικά το προγραμματιστικό του μοντέλο έκανε ζημία στην φήμη της , και άφησε την υπόνοια ότι το JavaComm δεν έχει κάποια χρησιμότητα για το μέλλον. Εν αντιθέσει το RXTX παρέχει υποστήριξη σε αρκετά περισσότερες πλατφόρμες σε σχέση με το JavaComm , και τελευταία έχει υιοθετήσει το ίδιο interface με το JavaComm παρόλο που τα ονόματα πακέτων δεν είναι τα ιδία με αυτά της Sun. Επομένως το ερώτημα παραμένει, ποιος από τους 2 παραπάνω τρόπους είναι πιο αποτελεσματικός για την υλοποίηση κάποιας εφαρμογής. Εάν το ζητούμενο είναι (κάποια) μεγαλύτερη φορητότητα, τότε το JavaComm είναι μια καλή επιλογή. Εάν τώρα δεν υπάρχει υποστηριζόμενη υλοποίηση για συγκεκριμένη πλατφόρμα όσων αφορά το JavaComm αλλά υπάρχει η αντίστοιχη υλοποίηση από την πλευρά του RXTX, τότε το RXTX μπορεί να χρησιμοποιηθεί σαν οδηγός (driver) του JavaComm πάνω στην πλατφόρμα αυτή. Όποτε χρησιμοποιώντας κάποιος το JavaComm μπορεί να έχει την υποστήριξη για όλες τις πλατφόρμες είτε σε αυτές παρέχεται κάποια επίσημη υλοποίηση από την Sun είτε από το RXTX μαζί με το JCL . Με τους παραπάνω τρόπους δεν είναι αναγκαία κάποια αλλαγή στην εφαρμογή , και μπορεί επίσης να λειτουργήσει με το κοινό interface του JavaComm.

4.1.2 Εγκατάσταση

Γενικά και το RXTX και το JavaComm έχουν κάποιες ιδιορρυθμίες σε ότι έχει να κάνει με την εγκατάσταση τους, και για αυτό το λόγο πρέπει να γίνονται με προσοχή όλες οι διαδικασίες που σχετίζονται με την εγκατάσταση. Επίσης θέλουν κάποια προσοχή σε ότι αφορά την έκδοση των JDK καθώς επίσης και των εκδόσεων των βιβλιοθηκών τόσο του JavaComm όσο και του RXTX

Χρήση Του Java Webstart Για Την Εγκατάσταση Του JavaComm

Ένα κοινό πρόβλημα τόσο για το JavaComm όσο και για το RXTX είναι ότι και τα 2 έχουν θέματα με την χρήση του Java Webstart για την εγκατάσταση τους .
Το JavaComm είναι επίσης "διάσημο" , γιατί απαιτεί την χρήση του αρχείου javax.comm.properties το όποιο πρέπει να τοποθετηθεί μέσα στις βιβλιοθήκες του αντιστοίχου JDK, το όποιο δεν μπορεί να γίνει χρησιμοποιώντας το Java Webstart , το ποιο είναι ιδιαίτερα λυπηρό για τι η ύπαρξη αυτού του αρχείου είναι ένα από τα μεγαλύτερα σχεδιαστικά ατοπήματα από την μεριά της Sun ,και η όποια δεν το παραδέχεται λέγοντας ότι έχει μεγάλο βαθμό χρησιμότητας .Το περιεχόμενο αυτού του αρχείου όπως και οι ιδιότητες του περιγράφονται στην παρακάτω γραμμή:

```
driver=com.sun.comm.Win32Driver
```

Με τον παρακάτω "τρόπο" μπορούμε να αγνοήσουμε/παρακάμψουμε το προαναφερθέν "χρήσιμο" αρχείο , επιτρέποντας μας να εγκαταστήσουμε το JavaComm χρησιμοποιώντας το Java Webstart,και ο τρόπος είναι ο εξής:
Για αρχή απενεργοποιήσαμε τον security manager

```
System.setSecurityManager(null);
```

Και μετά όταν πάμε να αρχικοποιήσουμε το JavaComm API, πάμε και αρχικοποιούμε τον οδηγό (Driver) χειροκίνητα

```
String driverName = "com.sun.comm.Win32Driver"; // or get as a JNLP
property
CommDriver commDriver =
(CommDriver)Class.forName(driverName).newInstance();
commDriver.initialize();
```

4.1.3 RXTX

Όταν πάμε να χρησιμοποιήσουμε το RXTX πάνω σε κάποιες πλατφόρμες , μπορεί να μας ζητηθούν κάποια δικαιώματα χρήσης ή πρόσβασης σε ότι έχει να κάνει με τις σειριακές συσκευές , και αυτό είναι ένα πράγματα που δεν μπορούν να γίνουν με την χρήση του Java Webstart. Όποτε κατά την εκκίνηση του RXTX θα πρέπει να χρησιμοποιηθούν δικαιώματα υπέρ του χρήστη . Επίσης να αναφέρουμε ότι το RXTX διαθέτει έναν αλγόριθμο προτυποποίησης ο οποίος έχει να κάνει με την αναγνώριση των εγκύρων ονομάτων των σειριακών συσκευών .Ο αλγόριθμος αυτός όμως δημιουργεί πρόβλημα όταν χρησιμοποιηθεί σε "μη τυποποιημένες " συσκευές όπως για παράδειγμα είναι η μετατροπή του USB σε σειριακό. Αυτός ο μηχανισμός μπορεί παρακαμφθεί από τις ιδιότητες συστήματος.

4.1.4 Το JavaComm API

Εισαγωγή

Το επίσημο API για σειριακή επικοινωνία στην Java είναι το JavaComm API. Δυστυχώς το API αυτό δεν αποτελεί μέρος της κύριας έκδοσης της Java και έτσι η όποια υλοποίηση αυτού πρέπει να κατέβει και να εγκατασταθεί ανεξάρτητα.
Όπως και με τις βιβλιοθήκες του JavaComm έτσι και το API δεν τυγχάνει της προσοχής της Sun σε θέματα συντήρησης εδώ και πολύ καιρό. Αραιά και που διορθώνει κάποια προβλήματα, αλλά δεν έχει ασχοληθεί με κάποια μεγάλη αναμόρφωση που είναι και απολύτως απαραίτητη.

Χρήση Του JavaComm API

Τα 3 πράγματα που χρειάζεται να κάνει κάποιος για να προγραμματίσει πάνω σε σειριακές γραμμές χρησιμοποιώντας το JavaComm είναι τα εξής ακόλουθα :

1. Απαριθμούμε όλα τα σειριακά κανάλια (ταυτοποίηση καναλιών) που είναι διαθέσιμα στο JavaComm
2. Επιλογή των επιθυμητών τακτοποιημένων καναλιών από όσα έχουμε διαθέσιμα
3. Κτήση/χρήση των καναλιών μέσω του ταυτοποιητή καναλιών

Η απαρίθμηση και επιλογή των επιθυμητών καναλιών γίνεται τυπικά σε ένα βρόχο όπως φαίνεται παρακάτω:

```
import javax.comm.*;
import java.util.*;
...
//
// Platform specific port name, here= a Unix name
//
// NOTE: On at least one Unix JavaComm implementation JavaComm
//       enumerates the ports as "COM1" ... "COMx", too, and not
//       by their Unix device names "/dev/tty...".
//       Yet another good reason to not hard-code the wanted
//       port, but instead make it user configurable.
//
String wantedPortName = "/dev/ttya";

//
// Get an enumeration of all ports known to JavaComm
//
Enumeration portIdentifiers = CommPortIdentifier.getPortIdentifiers();
//
// Check each port identifier if
//   (a) it indicates a serial (not a parallel) port, and
//   (b) matches the desired name.
//
CommPortIdentifier portId = null;  // will be set if port found
while (portIdentifiers.hasMoreElements())
{
    CommPortIdentifier pid = (CommPortIdentifier)
portIdentifiers.nextElement();
    if(pid.getPortType() == CommPortIdentifier.PORT_SERIAL &&
       pid.getName().equals(wantedPortName))
    {
        portId = pid;
        break;
    }
}
if(portId == null)
{
    System.err.println("Could not find serial port " + wantedPortName);
    System.exit(1);
}
//
// Use port identifier for acquiring the port
//
...
```

Μόλις το αναγνωριστικό του καναλιού βρεθεί , τότε μπορούμε να το χρησιμοποιήσουμε

```
//
// Use port identifier for acquiring the port
//
SerialPort port = null;
try {
    port = (SerialPort) portId.open(
        "name", // Name of the application asking for the port
        10000   // Wait max. 10 sec. to acquire port
    );
} catch(PortInUseException e) {
    System.err.println("Port already in use: " + e);
    System.exit(1);
}
//
// Now we are granted exclusive access to the particular serial
// port. We can configure it and obtain input and output streams.
//
...
```

4.1.5 Αρχικοποιώντας Ένα Σειριακό Κανάλι

Η αρχικοποίηση ενός σειριακού καναλιού είναι μια ευθύς και ξεκάθαρη διαδικασία. Είτε περνάμε ξεχωριστής τις ρυθμίσεις επικοινωνίας (baud rate , data bits ,stop bits, parity), είτε τις ρυθμίζουμε όλες μαζί ταυτόχρονα χρησιμοποιώντας την πολύ εύχρηστη μέθοδο setSerialPortParams() όπως γίνεται στο παράδειγμα παρακάτω:

```
import java.io.*;
...

//
// Set all the params.
// This may need to go in a try/catch block which throws UnsupportedCommOperationException
//
port.setSerialPortParams(
    115200,
    SerialPort.DATABITS_8,
    SerialPort.STOPBITS_1,
    SerialPort.PARITY_NONE);

//
// Open the input Reader and output stream. The choice of a
// Reader and Stream are arbitrary and need to be adapted to
// the actual application. Typically one would use Streams in
// both directions, since they allow for binary data transfer,
// not only character data transfer.
//
BufferedReader is = null;  // for demo purposes only. A stream would be more typical.
PrintStream    os = null;

try {
  is = new BufferedReader(new InputStreamReader(port.getInputStream()));
} catch (IOException e) {
  System.err.println("Can't open input stream: write-only");
  is = null;
}
```

```
//
// New Linux systems rely on Unicode, so it might be necessary to
// specify the encoding scheme to be used. Typically this should
// be US-ASCII (7 bit communication), or ISO Latin 1 (8 bit
// communication), as there is likely no modem out there accepting
// Unicode for its commands. An example to specify the encoding
// would look like:
//
//     os = new PrintStream(port.getOutputStream(), true, "ISO-8859-1");
//
os = new PrintStream(port.getOutputStream(), true);


//
// Actual data communication would happen here
// performReadWriteCode();
//

//
// It is very important to close input and output streams as well
// as the port. Otherwise Java, driver and OS resources are not released.
//
if (is != null) is.close();
if (os != null) os.close();
if (port != null) port.close();
```

4.1.6 Απλή Μεταφορά Δεδομένων- Γράφοντας Δεδομένα

Το να γράψει κάποιος δεδομένα πάνω σε σειριακό κανάλι είναι απλό σαν τις λειτουργίες εισόδου εξόδου της Java . Ωστόσο πρέπει να έχουμε την προσοχή μας αν χρησιμοποιούμε AT commands (Hayes protocol) όπως:

1. Να μην χρησιμοποιούμε εντολές που εισάγουν αυτόματα το \n για διαχωριστή (πχ println) στο OutputStream μιας και το πρωτόκολλο του Hayes λειτουργεί ορίζοντας το \r\n σαν διαχωριστές ,ανεξαρτήτως λειτουργικού συστήματος
2. Αφότου γράψουμε στο OutputStream , ο buffer του InputStream θα περιέχει την τελευταία εντολή που δέχθηκε σε επανάληψη, όποτε για να μην έχουμε προβλήματα ιδιαίτερα στην περίπτωση που το modem έχει ρυθμιστεί για επανάληψη , καλό θα ήταν να καθαρίζουμε τον buffer InputStream μετά από κάθε χρήση του (ώστε να μην έχουμε λανθασμένη επανάληψη εντολών).
3. Όταν (κακώς) χρησιμοποιούμε έναν Reader/Writer καλό θα ήταν να βάζουμε την κωδικοποίηση(encoding) στο US-ASCII παρά σε ότι έχει τυπικά σαν default η κάθε πλατφόρμα.
4. Αφού ο κύριος σκοπός ενός Modem είναι η μεταφορά δεδομένων χωρίς αυτά να υποστούν κάποια αλλαγή , τότε θα τον πρέπει η επικοινωνία με το modem να χρησιμοποιεί InputStream/OutputStream και όχι Reader/Writer.

Παρακάτω ακολουθεί παράδειγμα χρήσης:

```
// Write to the output
os.print("AT");
os.print("\r\n"); // Append a carriage return with a line feed

is.readLine(); // First read will contain the echoed command you sent to
it. In this case: "AT"
is.readLine(); // Second read will remove the extra line feed that AT
generates as output
```

4.1.7 Απλό Διάβασμα Δεδομένων

Εφόσον η διαδικασία της εγγραφής ήταν επιτυχής και εύκολη υπόθεση, τότε και η διαδικασία του διαβάσματος είναι το ίδιο απλή, όπως μπορούμε να δούμε παρακάτω:

```
// Read the response
String response = is.readLine(); // if you sent "AT" then response ==
"OK"
```

4.1.8 Προβλήματα Με Την Χρήση Των Reading/Writing

Οι παραπάνω τρόποι που δείξαμε σχετικά με την εγγραφή και την ανάγνωση δεδομένων από και προς σειριακά κανάλια , έχουν και κάποια μειονεκτήματα, τα όποια έχουν να κάνουν με την παρεμπόδιση των Input/Output, και συγκεκριμένα αυτό γίνεται όταν δεν υπάρχουν διαθέσιμα δεδομένα για διάβασμα ή όταν ο buffer του output που ήταν για εγγραφή ήταν πλήρης , τότε οι μέθοδοι δεν επέστρεφαν κάτι και έτσι η εφαρμογή "πάγωνε". Και για να είμαστε ακριβείς όταν το νήμα που ήταν υπεύθυνο για τις διαδικασίες εγγραφής/ανάγνωσης παγώσει , και αυτό το νήμα τυγχάνει να είναι και το κύριο νήμα της εφαρμογής τότε και το σύνολο της εφαρμογής θα παγώσει μέχρις ότου να διευθετηθεί το κόλλημα στο τμήμα της εγγραφής/ανάγνωσης (πχ δεν είχαμε διαθέσιμα δεδομένα για ανάγνωση, η εφαρμογή κόλλησε μέχρι να έχουμε πάλι δεδομένα και με το που τα δούμε η εφαρμογή ξεκολλήσει). Φυσικά ένας απλός τρόπος για να μην έχουμε το παραπάνω πρόβλημα θα ήταν να έχουμε ένα ανεξάρτητο νήμα για τις διαδικασίες εγγραφής/ανάγνωσης ώστε να μην εξαρτώνται από το κύριο νήμα της εφαρμογής και να δημιουργούνται τα παραπάνω προβλήματα.

4.1.9 Χρήση Γεγονότων Στη Σειριακή Επικοινωνία

Το JavaComm API μας παρέχει ένα μηχανισμό ειδοποιήσεων πάνω σε γεγονότα (Event Driven Notifications) με σκοπό την αντιμετώπιση των προβλημάτων παρεμπόδισης εισόδων/εξόδων . Η Sun και αυτόν τον μηχανισμό δεν τον άφησε χωρίς προβλήματα. Κατ' αρχήν η εφαρμογή μπορεί να καταγράψει event listeners πάνω σε συγκεκριμένα σειριακά κανάλια και να κρατήσει πληροφορίες για σημαντικά γεγονότα πάνω στα κανάλια αυτά . Οι 2 πιο σημαντικοί τύποι γεγονότων σχετικά με τις διαδικασίες εγγραφής/ανάγνωσης είναι οι εξής:

`javax.comm.SerialPortEvent.DATA_AVAILABLE` και
`javax.comm.SerialPortEvent.OUTPUT_BUFFER_EMPTY`

Αλλά και εδώ προκύπτουν τα 2 εξής προβλήματα:

1ον Μόνο ένας event listener μπορεί να καταχωρηθεί για κάθε ένα σειριακό κανάλι , αυτό υποχρεώνει τον κάθε προγραμματιστή να γράψει τερατώδεις listeners για κάθε τύπο γεγονότων.

2ον το OUTPUT_BUFFER_EMPTY είναι ένας προαιρετικός τύπος event ,και μέσα στο documentation της Sun αναφέρει ότι δεν υποστηρίζεται από όλες τις υλοποιήσεις του JavaComm

Παρακάτω παρατίθενται οι βασικές αρχές για την δημιουργία, την υλοποίηση και διαχείριση ενός Serial Event Handler :

```
import javax.comm.*;

/**
 * Listener to handle all serial port events.
 *
 * NOTE: It is typical that the SerialPortEventListener is implemented
 *       in the main class that is supposed to communicate with the
 *       device. That way the listener has easy access to state information
 *       about the communication, e.g. when a particular communication
 *       protocol needs to be followed.
 *
 *       However, for demonstration purposes this example implements a
 *       separate class.
 */
class SerialListener implements SerialPortEventListener {

    /**
     * Handle serial events. Dispatches the event to event-specific
     * methods.
     * @param event The serial event
     */
    @Override
    public void serialEvent(SerialPortEvent event){

        //
        // Dispatch event to individual methods. This keeps this ugly
        // switch/case statement as short as possible.
        //
        switch(event.getEventType()) {
            case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
                outputBufferEmpty(event);
                break;

            case SerialPortEvent.DATA_AVAILABLE:
                dataAvailable(event);
                break;

/* Other events, not implemented here ->
            case SerialPortEvent.BI:
                breakInterrupt(event);
                break;
```

```
            case SerialPortEvent.CD:
                carrierDetect(event);
                break;

            case SerialPortEvent.CTS:
                clearToSend(event);
                break;

            case SerialPortEvent.DSR:
                dataSetReady(event);
                break;

            case SerialPortEvent.FE:
                framingError(event);
                break;

            case SerialPortEvent.OE:
                overrunError(event);
                break;

            case SerialPortEvent.PE:
                parityError(event);
                break;
            case SerialPortEvent.RI:
                ringIndicator(event);
                break;
<- other events, not implemented here */

        }
    }

    /**
     * Handle output buffer empty events.
     * NOTE: The reception of this event is optional and not
     *       guaranteed by the API specification.
     * @param event The output buffer empty event
     */
    protected void outputBufferEmpty(SerialPortEvent event) {
        // Implement writing more data here
    }

    /**
     * Handle data available events.
     *
     * @param event The data available event
     */
    protected void dataAvailable(SerialPortEvent event) {
        // implement reading from the serial port here
    }
}
```

Αφότου γίνει η υλοποίηση του listener , μπορεί να χρησιμοποιηθεί σε γεγονότα για συγκεκριμένο σειριακό κανάλι. Για να γίνει αυτό ένα instance του listener πρέπει να προστεθεί πάνω στο κανάλι. Και ακόμα ο υποδοχέας για κάθε τύπο γεγονότος πρέπει να ζητηθεί ξεχωριστά .

```
SerialPort port = ...;
...
//
// Configure port parameters here. Only after the port is configured it
// makes sense to enable events. The event handler might be called
immediately
// after an event is enabled.
...

//
// Typically, if the current class implements the SerialEventListener
interface
// one would call
//
//        port.addEventListener(this);
//
// but for our example a new instance of SerialListener is created:
//
port.addEventListener(new SerialListener());

//
// Enable the events we are interested in
//
port.notifyOnDataAvailable(true);
port.notifyOnOutputEmpty(true);

/* other events not used in this example ->
port.notifyOnBreakInterrupt(true);
port.notifyOnCarrierDetect(true);
port.notifyOnCTS(true);
port.notifyOnDSR(true);
port.notifyOnFramingError(true);
port.notifyOnOverrunError(true);
port.notifyOnParityError(true);
port.notifyOnRingIndicator(true);
<- other events not used in this example */
```

4.2 JSSC

4.2.1 JSSC Java Simple Serial Connector

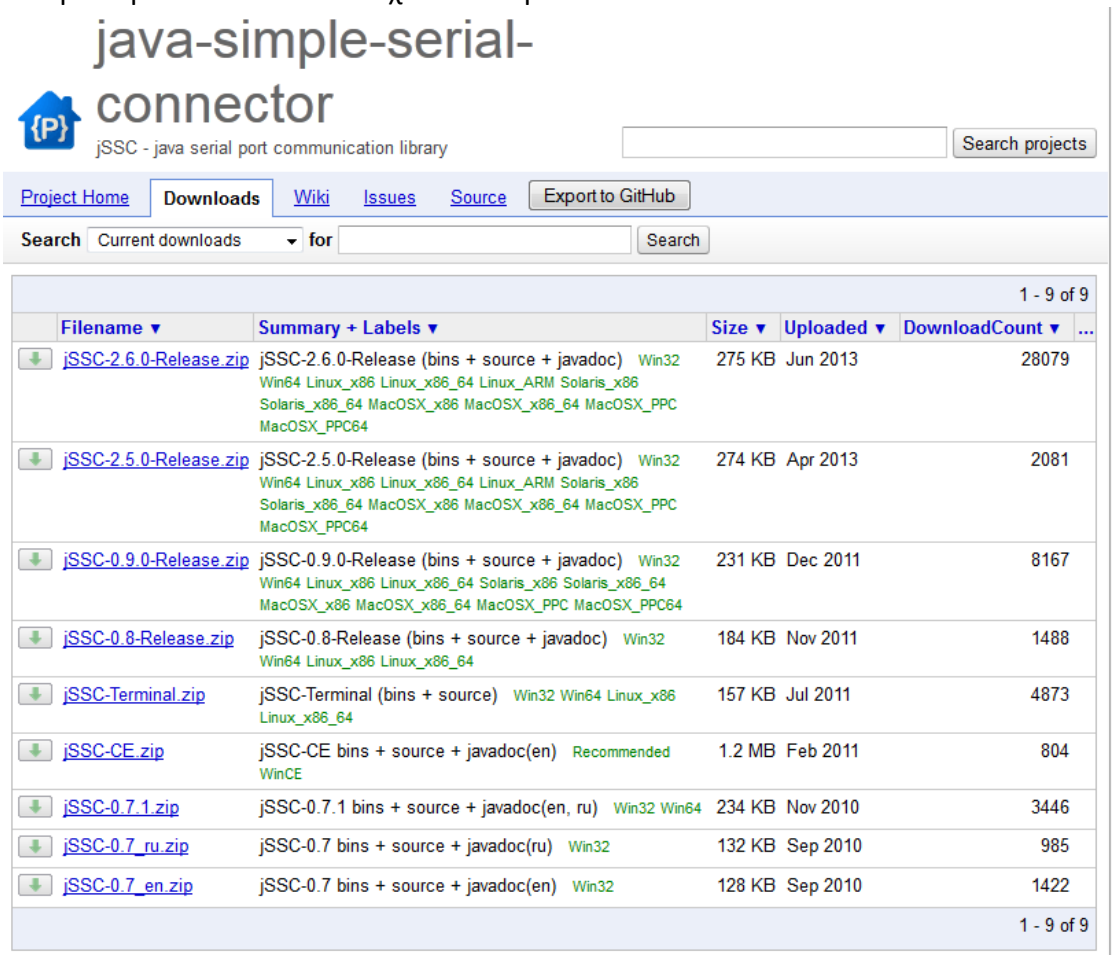
Εισαγωγή

Το Java Simple Serial Connector ή αλλιώς εν συντομία JSSC είναι άλλο ένα project ανοικτού κώδικα που μας επιτρέπει με πάρα πολύ απλό τρόπο και σε αντίθεση με το προβληματικό JavaComm να επιτύχουμε σειριακή επικοινωνία ανάμεσα σε έναν μεγάλο αριθμό από πλατφόρμες και την Java , προσφέροντας μια ξεκάθαρη διεπαφή (interface) και πάνω από όλα να υποστηρίζεται από μια πάρα πολύ ενεργή και μεγάλη κοινότητα (και εδώ σε αντίθεση με το νεκρό JavaComm). Οι πλατφόρμες που υποστηρίζει το JSSC έως τώρα είναι: Win32, Win64,

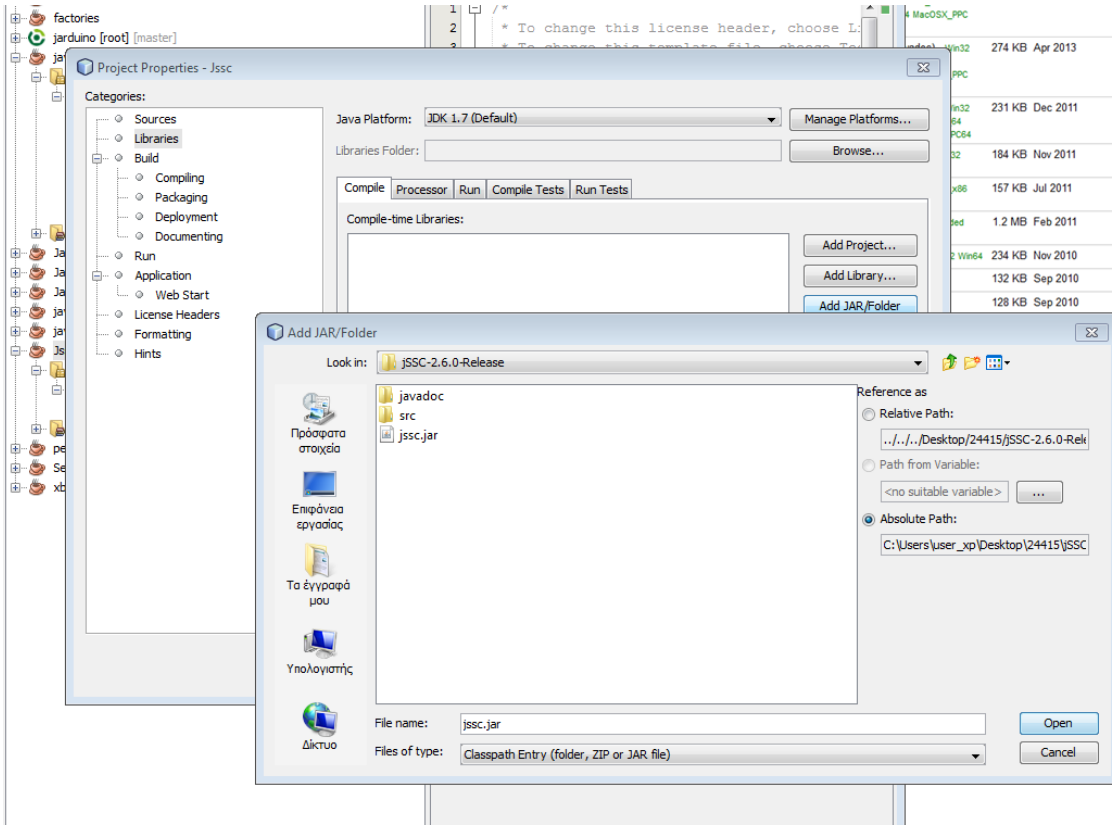
Linux x86, Linux x64, Linux ARM, Solaris x86, Solaris x64, MacOSX x86 , MacOSX x64 MacOSX PPC και MacOSX PPC 64 .Το JSSC μας παρέχει δυνατότητες όπως η εύρεση των αναγνωριστικών ονομάτων κάθε σειριακού καναλιού , το να κάνουμε εγγραφή και ανάγνωση δεδομένων πάνω από στα κανάλια αυτά, χρήση γεγονότων (event listeners), και πολλά ακόμα. Το JSSC σχεδιάστηκε για να λειτουργεί 24/7 πάνω σε multi threaded συστήματα και επί του παρόντος χρησιμοποιείται πάνω σε συστήματα αυτοματισμών , συλλογής, και καταγραφής πληροφοριών. Και φυσικά λόγο της ευκολίας χρήσης του και των προαναφερθέντων πλεονεκτημάτων του , το JSSC επιλέχτηκε και από μέρους μας για την υλοποίηση του project μας.

4.2.2 Απόκτηση Και Εγκατάσταση Του JSSC

Για να αποκτήσουμε το JSSC , πήγαμε στο αντίστοιχο κομμάτι του code.google.com και το κατεβάσαμε από τον αντίστοιχο σύνδεσμο :

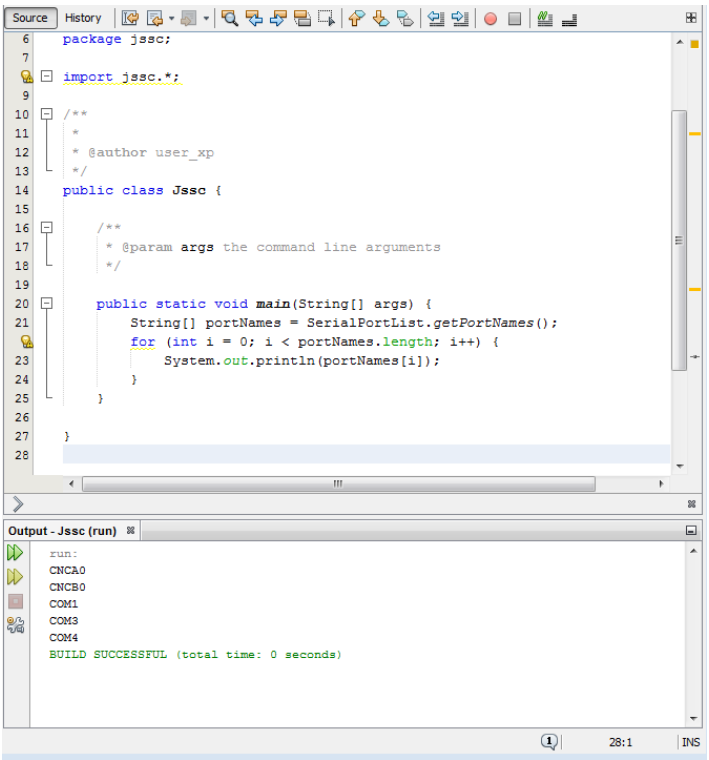


Και αφού περάσουν μερικά δευτερόλεπτα μέχρι να γίνει η λήψη , το αποσυμπιέζουμε , και μετά προσθέτουμε το jar στις βιβλιοθήκες του project μας όπως δείχνουμε παρακάτω:

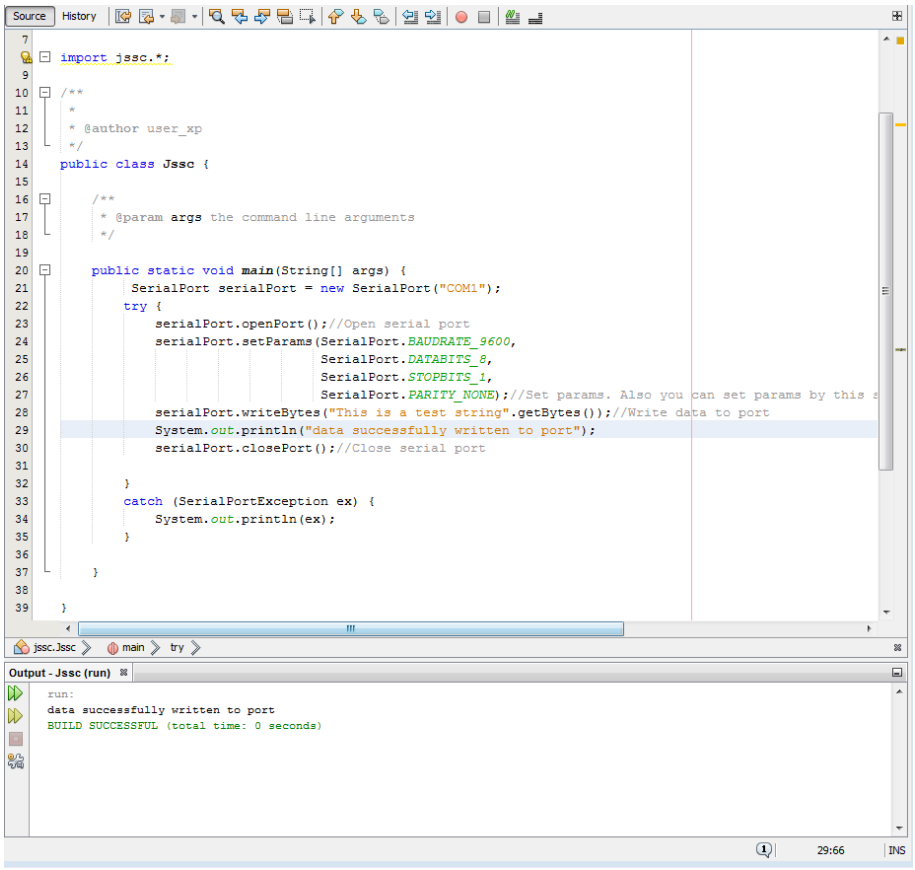


4.2.3 Παραδείγματα Χρήσης Του JSSC

Αφού εισάγουμε το jar στις βιβλιοθήκες του το μονό που χρειάζεται είναι να την κάνουμε import στον κώδικα, μετά τρέχουμε παραδείγματα για port names :



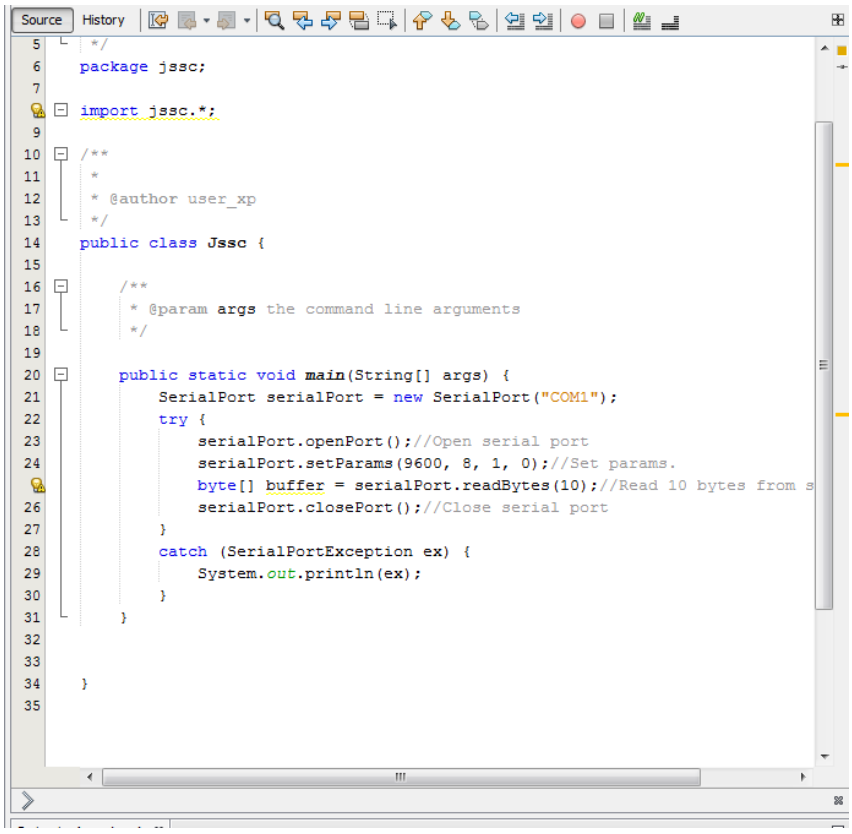
Εγγραφή Δεδομένων:



```
import jssc.*;

/**
 *
 * @author user_xp
 */
public class Jssc {

    /**
     * @param args the command line arguments
     */

    public static void main(String[] args) {
        SerialPort serialPort = new SerialPort("COM1");
        try {
            serialPort.openPort();//Open serial port
            serialPort.setParams(SerialPort.BAUDRATE_9600,
                                 SerialPort.DATABITS_8,
                                 SerialPort.STOPBITS_1,
                                 SerialPort.PARITY_NONE);//Set params. Also you can set params by this s
            serialPort.writeBytes("This is a test string".getBytes());//Write data to port
            System.out.println("data successfully written to port");
            serialPort.closePort();//Close serial port

        }
        catch (SerialPortException ex) {
            System.out.println(ex);
        }

    }

}
```

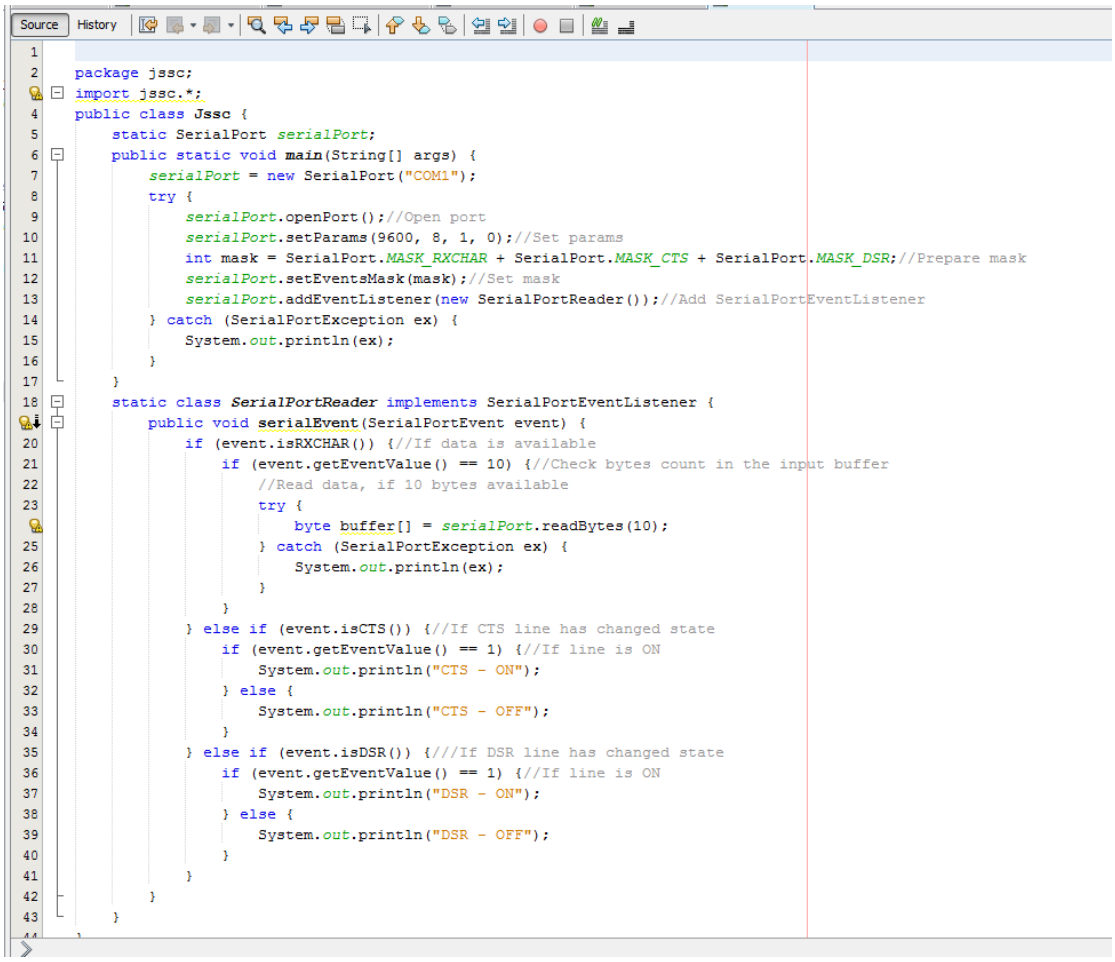
```
Output - Jssc (run)
run:
data successfully written to port
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ανάγνωση Δεδομένων:



```
*/
package jssc;

import jssc.*;

/**
 *
 * @author user_xp
 */
public class Jssc {

    /**
     * @param args the command line arguments
     */

    public static void main(String[] args) {
        SerialPort serialPort = new SerialPort("COM1");
        try {
            serialPort.openPort();//Open serial port
            serialPort.setParams(9600, 8, 1, 0);//Set params.
            byte[] buffer = serialPort.readBytes(10);//Read 10 bytes from s
            serialPort.closePort();//Close serial port
        }
        catch (SerialPortException ex) {
            System.out.println(ex);
        }
    }


}
```

Χρήση Γεγονότων (Event Listener):



```
package jssc;
import jssc.*;
public class Jssc {
    static SerialPort serialPort;
    public static void main(String[] args) {
        serialPort = new SerialPort("COM1");
        try {
            serialPort.openPort();//Open port
            serialPort.setParams(9600, 8, 1, 0);//Set params
            int mask = SerialPort.MASK_RXCHAR + SerialPort.MASK_CTS + SerialPort.MASK_DSR;//Prepare mask
            serialPort.setEventsMask(mask);//Set mask
            serialPort.addEventListener(new SerialPortReader());//Add SerialPortEventListener
        } catch (SerialPortException ex) {
            System.out.println(ex);
        }
    }
    static class SerialPortReader implements SerialPortEventListener {
        public void serialEvent(SerialPortEvent event) {
            if (event.isRXCHAR()) {//If data is available
                if (event.getEventValue() == 10) {//Check bytes count in the input buffer
                    //Read data, if 10 bytes available
                    try {
                        byte buffer[] = serialPort.readBytes(10);
                    } catch (SerialPortException ex) {
                        System.out.println(ex);
                    }
                }
            } else if (event.isCTS()) {//If CTS line has changed state
                if (event.getEventValue() == 1) {//If line is ON
                    System.out.println("CTS - ON");
                } else {
                    System.out.println("CTS - OFF");
                }
            } else if (event.isDSR()) {///If DSR line has changed state
                if (event.getEventValue() == 1) {//If line is ON
                    System.out.println("DSR - ON");
                } else {
                    System.out.println("DSR - OFF");
                }
            }
        }
    }
```

ΚΕΦΑΛΑΙΟ 5

Στο κεφάλαιο αυτό θα αναλυθεί η δυνατότητα της ασφάλειας δεδομένων (κρυπτογράφηση και αποκρυπτογράφηση)μέσω της Java.

Java Cryptography Architecture (JCA)

Εισαγωγή

Η java σαν πλατφόρμα δίνει μεγάλη σημασία πάνω στην ασφάλεια δεδομένων καθώς επίσης παρέχει μεγάλες δυνατότητες πάνω σε αυτόν τον τομέα. Οι δυνατότητες αυτές περιλαμβάνουν την ασφάλεια γλώσσας ,την κρυπτογραφία ,τις υποδομές δημοσίων κλειδιών , πιστοποιήσεις, ασφάλεια επικοινωνιών και διαχείριση πρόσβασης.

Το JCA αποτελεί ένα από τα κυριά τμήματα της πλατφόρμας και περιλαμβάνει μια αρχιτεκτονική καθώς και μια σειρά από APIs τα οποία αφορούν συστήματα όπως ψηφιακές υπογραφές , υπηρεσίες επαληθεύσεις μηνυμάτων (message digests) , πιστοποιητικά και επικυρώσεις πιστοποιητικών, κρυπτογραφία (με συμμετρικό ή μη block), κρυπτογραφία ροών, γεννήτριες κλειδιών και συστήματα διαχείρισης αυτών, καθώς επίσης ασφαλείς γεννήτριες παράγωγης τυχαίνω αριθμόν . Αυτά τα APIs δίνουν την δυνατότητα στους λοιπούς προγραμματιστές να ενσωματώνουν ευκολά τα προαναφερθέν συστήματα στον κώδικα των εφαρμογών τους . Η αρχιτεκτονική των παραπάνω APIs βασίστηκε στις ακόλουθες βασικές αρχές

- Ανεξαρτησία υλοποίησης : Οι εφαρμογές δεν χρειάζεται να υλοποιούν αλγορίθμους ασφάλειας , αντ' αυτού μπορούν απλά να ζητούν υπηρεσίες ασφάλειας από την πλατφόρμα της Java . Οι υπηρεσίες αυτές υλοποιούνται από διάφορους προμηθευτές , οι οποίες ενσωματώνονται στην Java μέσω προτυποποιημένων διεπαφών . Μια εφαρμογή μπορεί να βασίζεται σε πολλαπλούς ανεξάρτητους providers για μέγιστη λειτουργικότητα των υπηρεσιών ασφάλειας
- Δια λειτουργικότητα υλοποίησης: Οι εφαρμογές παρέχουν μεγάλο βαθμό δια λειτουργικότητας ανεξάρτητα από την υλοποίηση. Έτσι οι υλοποιήσεις είναι ανεξάρτητες από τους παρόχους και οι πάροχοι ανεξάρτητοί από τις υλοποιήσεις
- Επεκτασιμότητα αλγορίθμων: Η Java σαν πλατφόρμα περιλαμβάνει αρκετούς ενσωματωμένους παρόδους , οι οποίοι από την μεριά τους υλοποιούν κάποιες βασικές υπηρεσίες ασάφειας οι οποίες είναι ευρέως διαδεδομένες σήμερα. Παρόλα αυτά μερικές εφαρμογές μπορεί να βασίζονται σε νέα αναπτυσσόμενα πρότυπα και μπορεί να μην παρέχεται κάποια υλοποίηση για αυτές ακόμα. Και για αυτόν τον λόγο η java επιτρέπει την εγκατάσταση "ιδιότυπων" παροχών οι όποιοι μπορούν να υλοποίησίν τις παραπάνω υπηρεσίες

5.2 Αρχιτεκτονική- Παροχή Υπηρεσιών Κρυπτογραφίας

Η java.security.provider είναι η βασική κλάση για όλους τούς παρόδους ασφάλειας. Με τον ορό πάροχο ασφάλειας (ή απλά παροχή)εννοούμε το πακέτο ή τα πακέτα τα οποία υλοποιούν τις διάφορες υπηρεσίες ασφάλειας. Κάθε παροχής περιέχει ένα instance αυτής της κλάσης, και το οποίο με την σειρά του κρατάει το όνομα το προμηθευτή και μια λίστα με όλους τους αλγορίθμους και υπηρεσίες ασφαλείας όπου αυτοίς υλοποίει. Όταν κάποιός αλγόριθμός/υπηρεσίας χρειαστείς , το πλαίσιο του JCA συμβουλεύετέ τα στοιχειά του προμηθευτεί , και αν υπαρχή κάποιά συσχέτισή τοτέμ το instance δημιουργείται.
Οι προμηθευτείς περιχέουν ένα σύνολό πακέτων τα οποία παρέχουν συμπαγείς υλοποιήσεις των παρεχόμενων αλγόριθμων κρυπτογραφίας. Σε κάθε εγκατάσταση του Java JDK ,εγκαθιστάτε και ρυθμίζεται "εργοστασιακά" και ένα σύνολό από προμηθευτείς. Παρόλα αυτά περεταίρω προμηθευτές μπορούν να εγκατασταθούν από την μεριά του Client καθώς επίσης μπορούν να να ρυθμίσουν την σειρά προτίμησης σχετικά με τις υπηρεσίες που θέλει να του παρέχονται από κάθε προμηθευτή. Για να χρησιμοποιήσει το JCA κάποιαν εφαρμογή το μόνο που έχει να κάνει είναι να καλέσει ένα αντικείμενο (object) καθώς και τους αλγορίθμους ή/και υπηρεσίες που χρειάζεται, και τότε παίρνει μια υλοποίησή από κάποιον από τους εγκαταστημένους προμηθευτείς. Αλλιώς το πρόγραμμα ζητάει τις παραπάνω υπηρεσίες από καίον συγκεκριμένο προμηθευτεί απλά καλώντας το όνομα του προμηθευτεί και της υπηρεσίας πχ:

```
md = MessageDigest.getInstance("MD5");
md = MessageDigest.getInstance("MD5", "ProviderC");
```

5.3 Διαχείριση Κλειδιών

Μια βάση δεδομένων που ονομάζεται keystore μπορεί να χρησιμοποιηθεί για την αποθήκευση κλειδιών και πιστοποιητικών. Τα keystores είναι διαθέσιμα για όλες τις ανάγκες των εφαρμογών που αφορούν πιστοποιήσεις ,υπογραφές ή κρυπτογράφηση.
Οι εφαρμογές μπορούν να έχουν πρόσβαση στις βάσεις των Key Store υλοποιώντας την κλάση η οποία βρίσκεται στο πακέτο java.security. Η εργοστασιακή έκδοση της Key Store παρέχεται και υλοποιείται από την Sun Microsystems. Υλοποιεί την keystore σαν αρχείο, χρησιμοποιώντας ένα τύπο file format εν ονόματη Jks . Φυσικά υπάρχουν και άλλοι τύποι format για keystores όπως παραδείγματος χάρη το Jecks το οποίο παρέχει καλύτερες δυνατότητες κρυπτογράφησης από το Jks, καθώς επίσης και το Pkcs12 το οποίο βασίζεται στο RSA . Οι εφαρμογές μπορούν να επιλέξουν διάφορες υλοποιήσεις KeyStores από διάφορους προμηθευτείς

5.4 Κλάσεις Μηχανής (Engine Classes) Και Αλγόριθμοι

Μια engine class παρέχει την διεσπάρη για συγκεκριμένες υπηρεσίες κρυπτογράφησης ανεξάρτητες από ειδικευμένους αλγορίθμους ή παρόδους .Η κλάση μηχανής μπορεί να περιέχει :

- Λειτουργίες κρυπτογράφησης
- Γεννήτριες ή μετατροπείς κρυπτογραφικού υλικού
- Αντικείμενα (KeyStores ή πιστοποιητικά) τα οποία ενθυλακώνουν τα κρυπτογραφημένα δεδομένα ώστε να μπορούν να χρησιμοποιηθούν σε υψηλό επίπεδο αφαιρετικότητας

5.5 Η Κλάση Security

Η Security class διαχειρίζεται τους εγκαταστημένους παρόδους και όλες τις ιδιότητες ασφάλειας. Περιέχει μόνο στατικές μεθόδους και ποτέ δεν αρχικοποιείται .Οι μέθοδοι για την προσθαφαίρεση προμηθευτών και για τις ρυθμίσεις των ιδιοτήτων της Security ,μπορούν μόνο να εκτελεστούν από "έμπιστα " προγράμματα.

"Εμπιστο " θεωρείται ένα πρόγραμμα το οποίο είναι είτε:

- Μια τοπική εφαρμογή η οποία δεν τρέχει υπό την επίβλεψη ενός διαχειριστή ασφάλειας είτε
- Κάποια εφαρμογή η οποία έχει δικαίωμα να εκτελέσει τις συγκεκριμένες μεθόδους

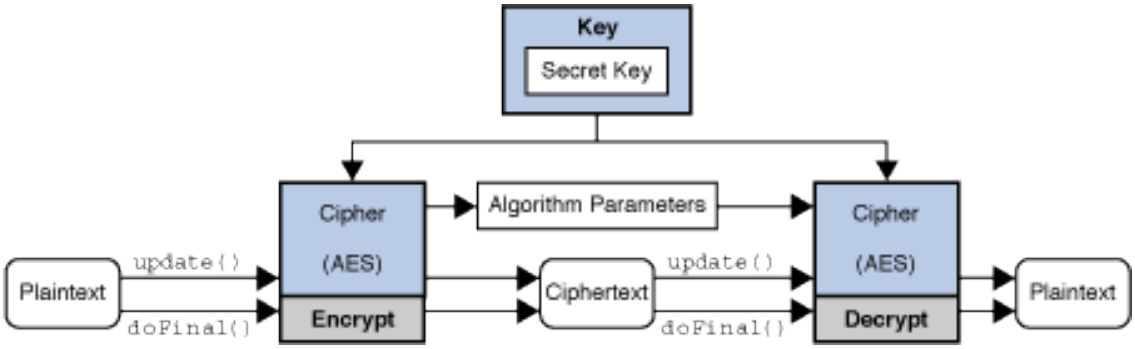
Ο ορισμός ότι ένα τμήμα κώδικα θεωρείται έμπιστο ώστε να πραγματοποιήσει κάποια ενεργεία απαιτεί ότι η εφαρμογής έχει τα καταλληλά δικαιώματα για την ενεργεία αυτή. Τα αρχεία διαχείρισης κανόνων τα οποία βρίσκονται στην εγκατάσταση του κάθε JDK ορίζουν τα δικαιώματα τα οποία έχει κάθε τμήμα κώδικα ανάλογα την προέλευση του. Η προέλευση του πηγαίου κώδικα ορίζεται από την διεύθυνση URL από όπου προήλθε ο κώδικας, αλλά και τα όποια δημοσιά κλειδιά τα οποία είχαν αντιστοίχιση σε ιδιωτικά κλειδιά τα οποία χρησιμοποιήθηκαν για να υπογράφουν τα δημοσιά κλειδιά στον πηγαίο κώδικα. Ένα παράδειγμα για το πώς δίνονται τα δικαιώματα στα αρχεία διαχείρισης κανόνων είναι το εξής:

```
grant codeBase "file:/home/sysadmin/", signedBy "sysadmin" {
   permission java.security.SecurityPermission "insertProvider.*";
   permission java.security.SecurityPermission "removeProvider.*";
   permission java.security.SecurityPermission "putProviderProperty.*";
};
```

Το παραπάνω αρχείο διαχείρισης κανόνων διευκρινίζει ότι ο κώδικας που φορτώθηκε και είναι υπογραμμένος από το JAR στην θέση /home/sysadmin/ και μπορεί να προσθέσει ή να αφαιρέσει Providers καθώς επίσης να διαχειριστεί τις ιδιότητες τους.

5.6 Η Κλάση Cipher

Η κλάση Cipher παρέχει τις λειτουργίες των αλγορίθμων κρυπτογραφίας που χρησιμοποιούνται για την κρυπτογράφηση και αποκρυπτογράφηση δεδομένων. Κρυπτογράφηση είναι η διαδικασία όπου περνούμε απλά δεδομένα (Cleartext) και κάποιο κλειδί και παράγουμε κρυπτογραφημένα δεδομένα. Αποκρυπτογράφηση είναι η αντιστροφή διαδικασία



5.7 Συμμετρική Και Ασύμμετρη Κρυπτογραφία

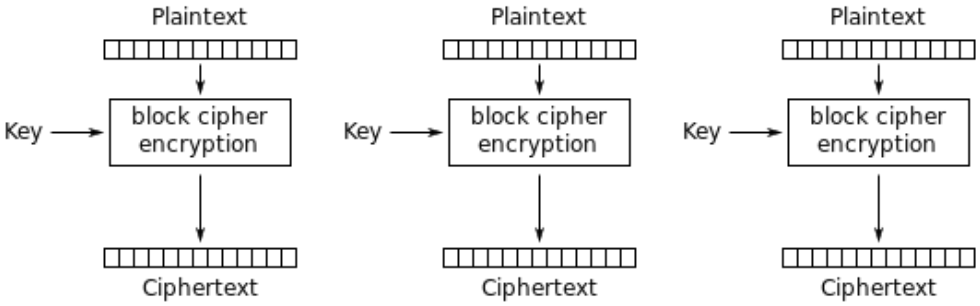
Υπάρχουν 2 είδη κρυπτογραφίας η συμμετρική , γνωστή και ως κρυπτογραφία μυστικού κλειδιού και η ασύμμετρη γνωστή και ως κρυπτογραφία δημοσίου κλειδιού. Στην συμμετρική και οι δυο πλευρές μοιράζονται το κλειδί για να ολοκληρωθούν οι διαδικασίες και είναι μέγιστης σημασίας η ασφάλεια του κλειδιού. Αντίθετα στην ασύμμετρη χρησιμοποιείται ένα ζευγάρι δημοσίου/ιδιωτικού κλειδιού για την κρυπτογράφηση και αποκρυπτογράφηση. Έτσι κάποιός κρυπτογραφεί με το δημόσιο κλειδί , το οποίο καταχωρεί σε κάποια αξιόπιστη βάση δημοσίων κλειδιών και έτσι όταν κάποιός θέλει να μπορεί να επικοινωνήσει μαζί του να χρησιμοποιήσει το δημόσιο κλειδί. Αλλά μόνο με το ιδιωτικό κλειδί μπορεί να γίνει η αποκρυπτογράφηση ώστε να ολοκληρωθεί η διαδικασία. Οι συμμετρικοί αλγόριθμοί είναι ταχύτερη από του ασύμμετρους και για αυτό το λόγο οι ασύμμετροι χρησιμοποιούνται για την διαδικασία ανταλλαγής κλειδιών ώστε να αρχικοποιήσεις η συμμετρική διαδικασία .

5.8 Κρυπτογράφηση Ροής Και Μπλοκ Δεδομένων

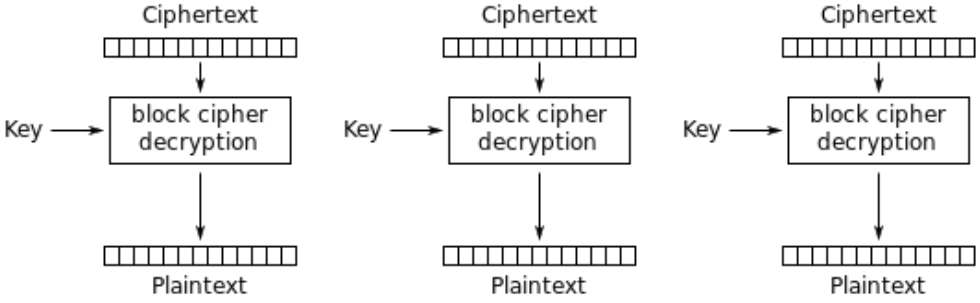
Στην κρυπτογράφηση μπλοκ τα δεδομένα χωρίζονται σε "κομμάτια" συγκεκριμένου μεγέθους (block size) και αν κάποιο μπλοκ δεν έχει αρκετά byte για να συμπληρώσει το επιθυμητό μέγεθος τότε το "γεμίζουμε" μέχρι να φτάσει σε αυτό. Γνωστοί μέθοδοι γεμίσματος είναι οι zeroBytePadding και το PKCS5Padding. Αφού ολοκληρωθεί η διαδικασία τότε μπορεί να γίνει η κρυπτογράφηση. Εν αντιθέσει στην περίπτωση της ροής τα δεδομένα τεμαχίζονται σε επίπεδο byte (η και bit ακόμα) και η κρυπτογράφηση γίνεται άμεσα και συνεχόμενα.

5.9 Μέθοδοι Λειτουργίας (CBC vs ECB)

Το ECB (Electronic Codebook) είναι η πιο απλή μέθοδος μιας και το μήνυμα χωρίζεται σε μπλοκ και κάθε ένα από αυτά κρυπτογραφείται ανεξάρτητα από τα άλλα.

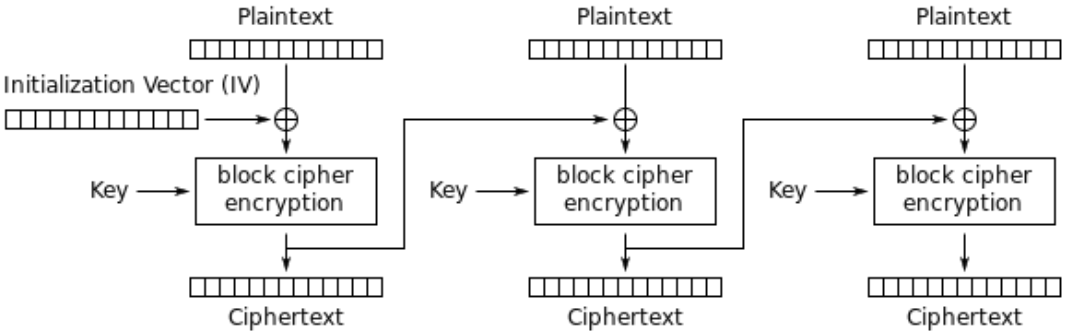


Electronic Codebook (ECB) mode encryption

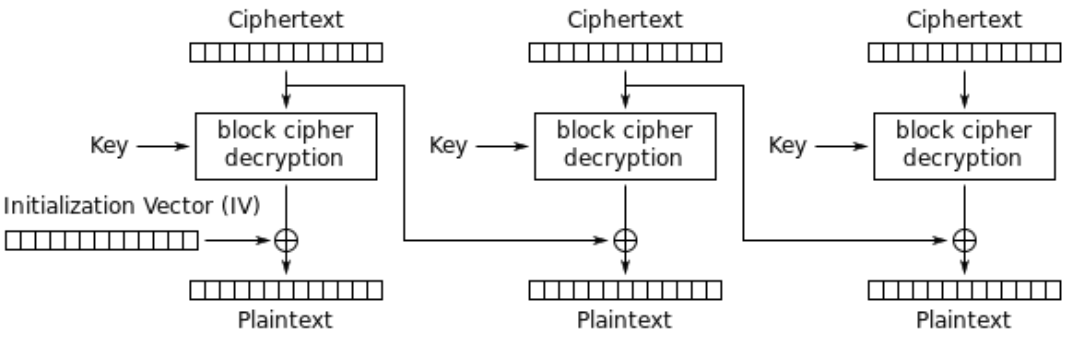


Electronic Codebook (ECB) mode decryption

Το μειονέκτημα αυτής της μεθόδου είναι ότι όμοια μπλοκ δεδομένων (plaintext) θα έχουν ως έξοδο πανομοιότυπα κρυπτογραφημένα δεδομένα (cipher text) κάνοντας έτσι πιο απλή κάθε προσπάθεια κρυπτανάλυσης.
Στο CBC (Cipher Block Chaining) κάθε μπλοκ δεδομένων γίνεται XOR με το προηγούμενο μπλοκ πριν κρυπτογραφηθεί (το πρώτο γίνετε XOR με έναν πίνακα αρχικοποιήσεις) . με αυτόν τον τρόπο κάθε κρυπτογραφημένο μπλοκ εξαρτάται από όλα τα μπλοκ που επεξεργάστηκαν μέχρι και αυτό κάνοντας την κρυπτανάλυση πολύ πιο δύσκολη σε σχέση με την ECB.



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

5.10 Η Κρυπτογράφηση Στο Σύστημα Μας

Παρακάτω παρουσιάζουμε τον τρόπο με τον οποίο γίνονται οι διαδικασίες κρυπτογράφησης και αποκρυπτογράφησης στο σύστημα μας

Για αρχή να αναφέρουμε ότι οι 2 συναρτήσεις παίρνουν 2 ορίσματα η κάθε μια , ένα από αυτά είναι τα δεδομένα και το άλλο είναι το κλειδί . Η συνάρτηση της κρυπτογράφησης παίρνει σαν δεδομένοι το καθαροί μνήμα (plaintext) ενώ αντίστοιχά η αποκρυπτογράφησή περάνει τα κρυπτογραφημένα δεδομένα (cipher text).

Η Διαδικασία Της Κρυπτογράφησης

Ένα απ' τα πρώτα πράγματα που κάνουμε και στις 2 συναρτήσεις είναι να μετατρέψουμε το κλειδί από String σε byte χρησιμοποιώντας και το σωστό encoding

```
byte[] keyRaw = kleidh.getBytes("ISO-8859-1");
```

Αμέσως μεταγ. δημιουργώντας την Cipher , καλούμε όποια μέθοδο κρυπτογράφησης θέλουμε και με τι προδιαγραφές

```
Cipher chiper = Cipher.getInstance("AES/ECB/PKCS5Padding");
```

Σειρά έχει η χρήση της SecretKeySpec με σκοπό την κατασκευή ενός AES κλειδιού

```
SecretKeySpec skeySpec = new SecretKeySpec(keyRaw, "AES");
```

Στη συνέχεια αρχικοποιούμε την διαδικασία δίνοντας την εντολή για λειτουργία κρυπτογράφησης και παρέχοντας και το κλειδί.

```
chiper.init(Cipher.ENCRYPT_MODE, skeySpec);
```

Άμεσα παρέχουμε τα byte του plaintext και παίρνουμε την έξοδο πάλι σε byte.Φυσικά και εδώ προσέχουμε την κωδικοποίηση

```
byte[] encVal = chiper.doFinal(plain.getBytes("ISO-8859-1"));
```

Τέλος μετατρέπουμε τα byte σε συμβολοσειρά κοιτώντας πάντα την κωδικοποίηση και παίρνουμε την τιμή επιστροφής της συνάρτησης

```
String str = new String(encVal, "ISO-8859-1");
return str;
```

Παρακάτω παραθέτουμε όλων τον κώδικα

```
34      public String aesEncryption(String plain, String kleidh) throws InvalidKeyException,
35              IllegalBlockSizeException, BadPaddingException,
36              UnsupportedEncodingException,
37              InvalidAlgorithmParameterException, NoSuchProviderException {
38
39          byte[] keyRaw = kleidh.getBytes("ISO-8859-1");
40
41          try {
42              Cipher chiper = Cipher.getInstance("AES/ECB/PKCS5Padding");
43              SecretKeySpec skeySpec = new SecretKeySpec(keyRaw, "AES");
44              chiper.init(Cipher.ENCRYPT_MODE, skeySpec);
45              byte[] encVal = chiper.doFinal(plain.getBytes("ISO-8859-1"));
46              String str = new String(encVal, "ISO-8859-1");
47              return str;
48
49          } catch (NoSuchAlgorithmException | NoSuchPaddingException ex) {
50              Logger.getLogger(ArduinoGui.class.getName()).log(Level.SEVERE, null, ex);
51          }
52          return null;
53
54      }
```

Η Διαδικασία Της Αποκρυπτογράφησης

Όπως και στην κρυπτογράφηση έτσι και εδώ το πρώτο που κάνουμε είναι να μετατρέψουμε το κλειδί σε byte

byte[] keyRaw = kleidh.getBytes("ISO-8859-1");

Το ίδιο κάνουμε επίσης και στα κρυπτογραφημένα δεδομένα (προσοχή πάντα στην κωδικοποίηση).

byte[] imagebyte =encryptedData.getBytes("ISO-8859-1");

Αντίστοιχα με την κρυπτογράφηση και εδώ κατασκευάζουμε το AES κλειδί

SecretKeySpec skeySpec = new SecretKeySpec(keyRaw, "AES");

Αμέσως μεταγ. δημιουργώντας την Cipher , καλούμε όποια μέθοδο θέλουμε και με τι προδιαγραφές

Cipher c = Cipher.getInstance("AES/ECB/PKCS5Padding");

Έπειτα αρχικοποιούμε την διαδικασία δίνοντας την εντολή για λειτουργία αποκρυπτογράφησης και παρέχοντας και το κλειδί

c.init(Cipher.DECRYPT_MODE, skeySpec);

Τέλος παρέχουμε τα byte των κρυπτογραφημένων δεδομένων για να ολοκληρωθεί η διαδικασία

byte[] decValue = c.doFinal(imagebyte);

Μετατρέπουμε τα byte σε συμβολοσειρά και έχουμε τη τιμή επιστροφής της συνάρτησης

String decryptedValue = new String(decValue, "ISO-8859-1");

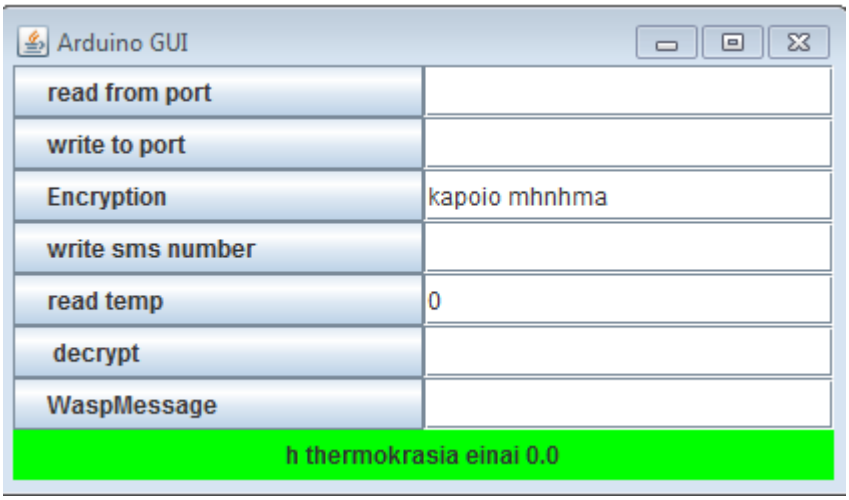
return decryptedValue;

Εν συνέχεια παραθέτουμε όλον τον κώδικα

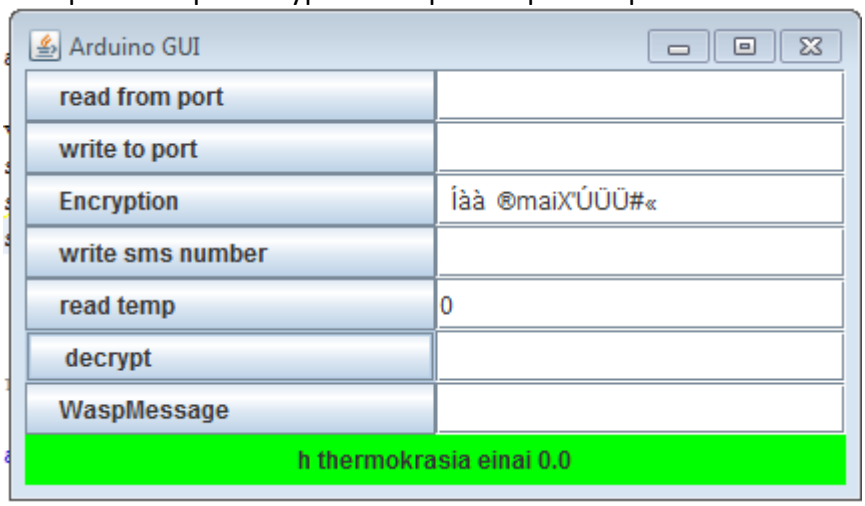
```
57      public static String aesDecryption(String encryptedData, String kleidh)
58              throws Exception {
59
60          byte[] keyRaw = kleidh.getBytes("ISO-8859-1");
61
62            byte[] imagebyte =encryptedData.getBytes("ISO-8859-1");
63
64          SecretKeySpec skeySpec = new SecretKeySpec(keyRaw, "AES");
65          Cipher c = Cipher.getInstance("AES/ECB/PKCS5Padding");
66          c.init(Cipher.DECRYPT_MODE, skeySpec);
67          byte[] decValue = c.doFinal(imagebyte);
68          String decryptedValue = new String(decValue, "ISO-8859-1");
69          return decryptedValue;
70      }
```

Παράδειγμα Χρήσης Της Εφαρμογής

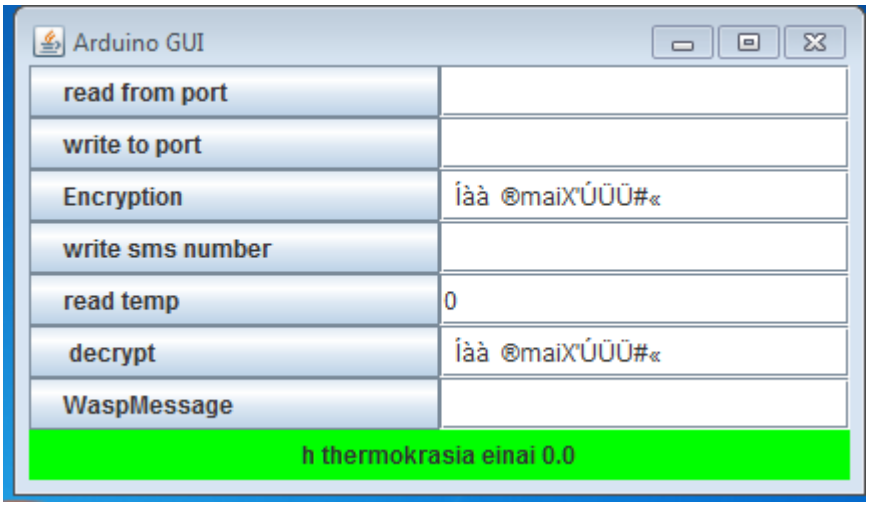
Αρχικά γράφουμε κάποιο μήνυμα στο πεδίο Encryption



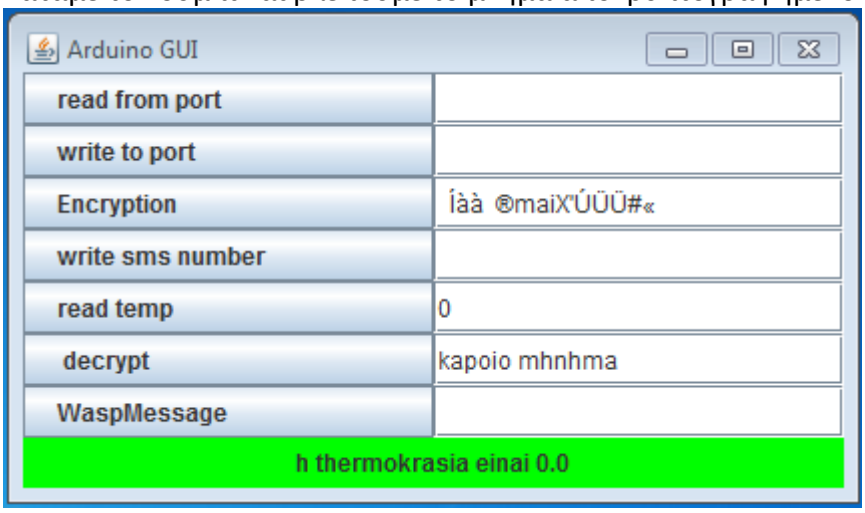
Πατάμε το κουμπί Encyption και βλέπουμε το Ciphertext



Αντιγράφουμε τα δεδομένα στο πεδίο διπλά από το decrypt button



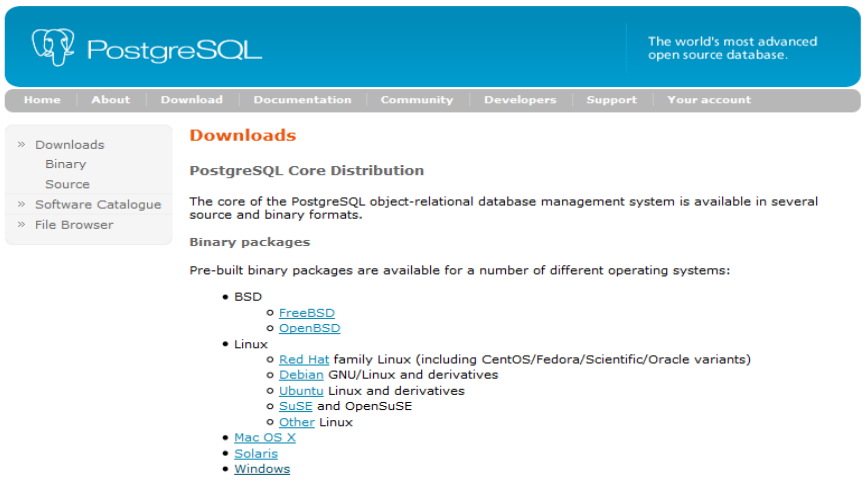
Πατάμε το κουμπί και βλέπουμε το μνήμα αποκρυπτογραφημένο



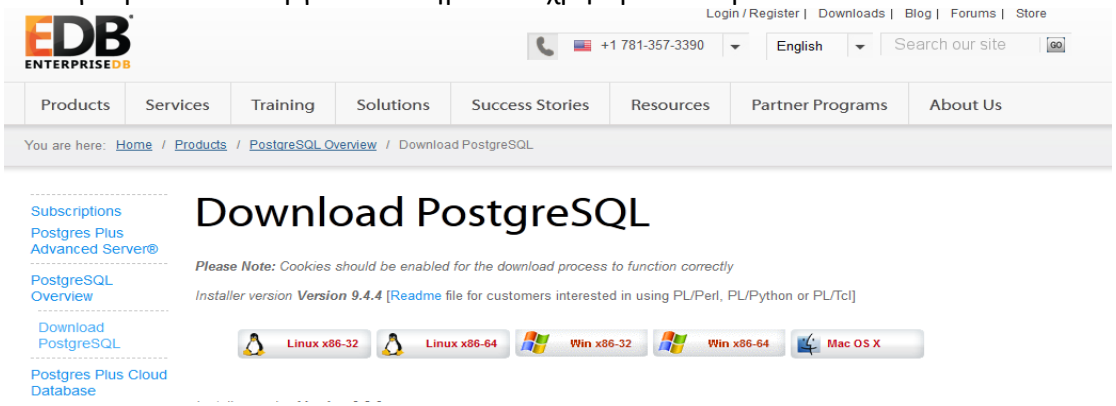
ΚΕΦΑΛΑΙΟ 6

6.1 Η Διαδικασία Λήψης και Εγκατάστασης Της PostgreSQL Ή Απλά Ως Postgres

Λήψη PostgreSQL at: http://www.postgresql.org/download/



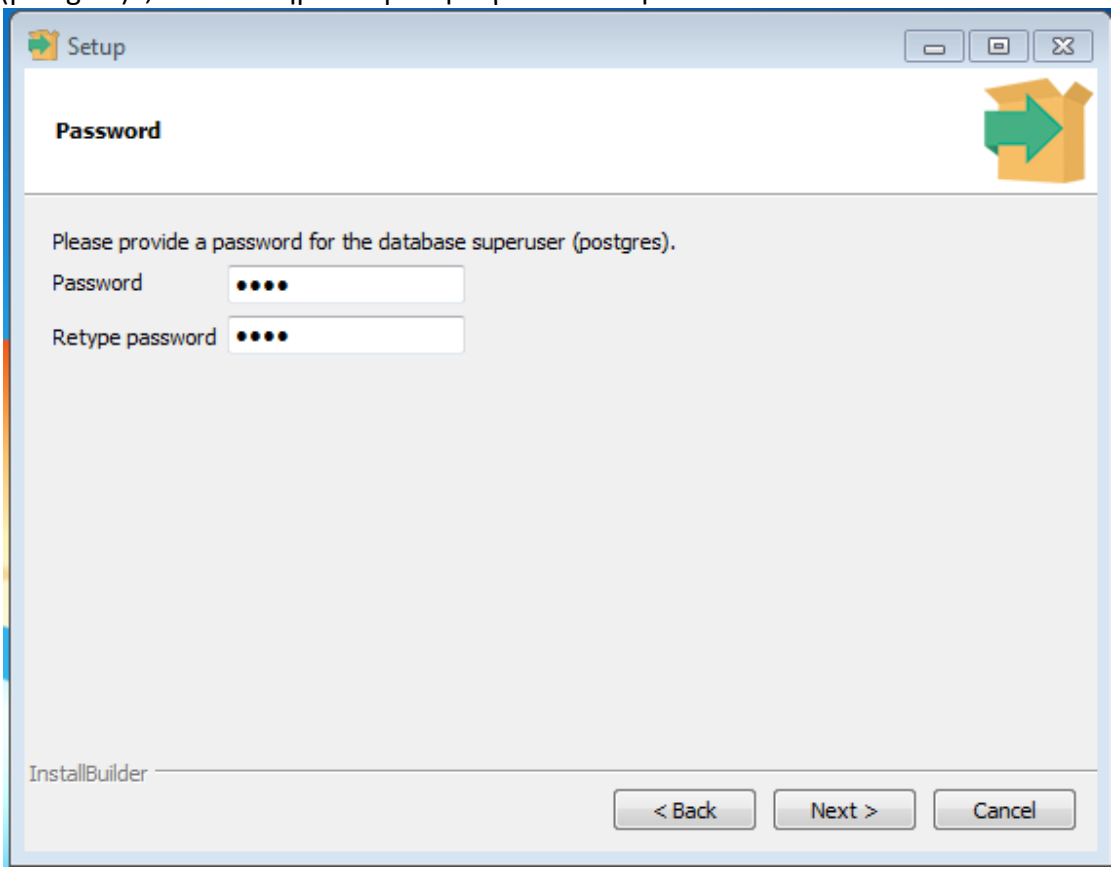
Επιλέγουμε το λειτουργικό σύστημα που χρησιμοποιούμε :



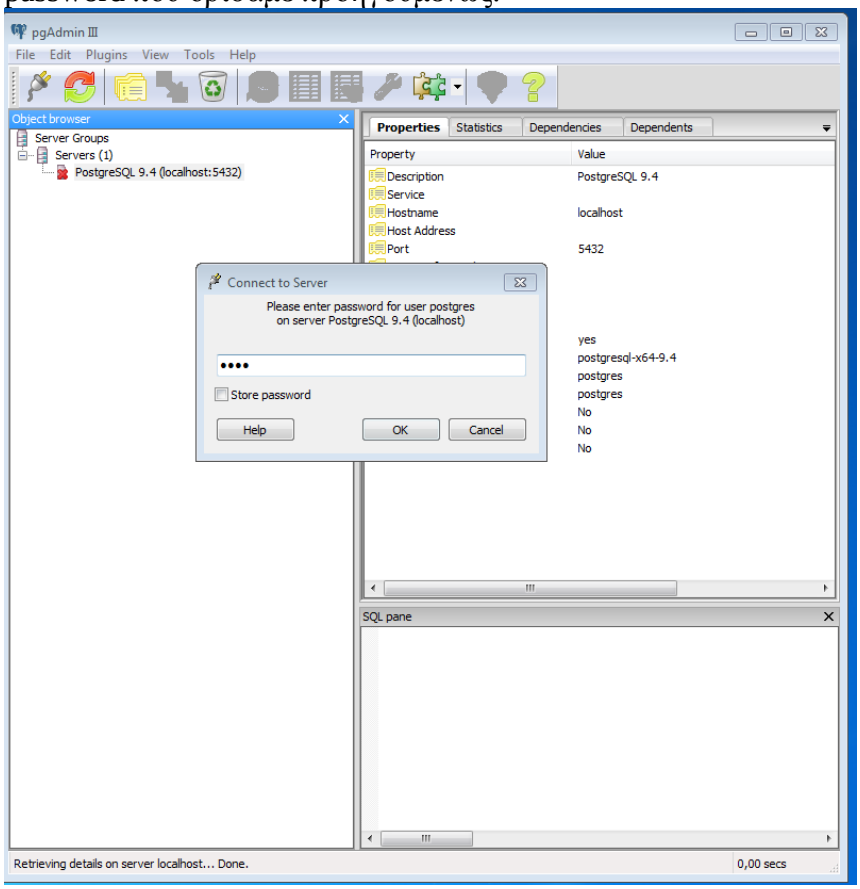
και παίρνουμε το αντίστοιχο setup, το οποίο και εκτελούμε

Θα πρέπει να δώσουμε προσοχή στο password που θα δώσουμε στην super user database

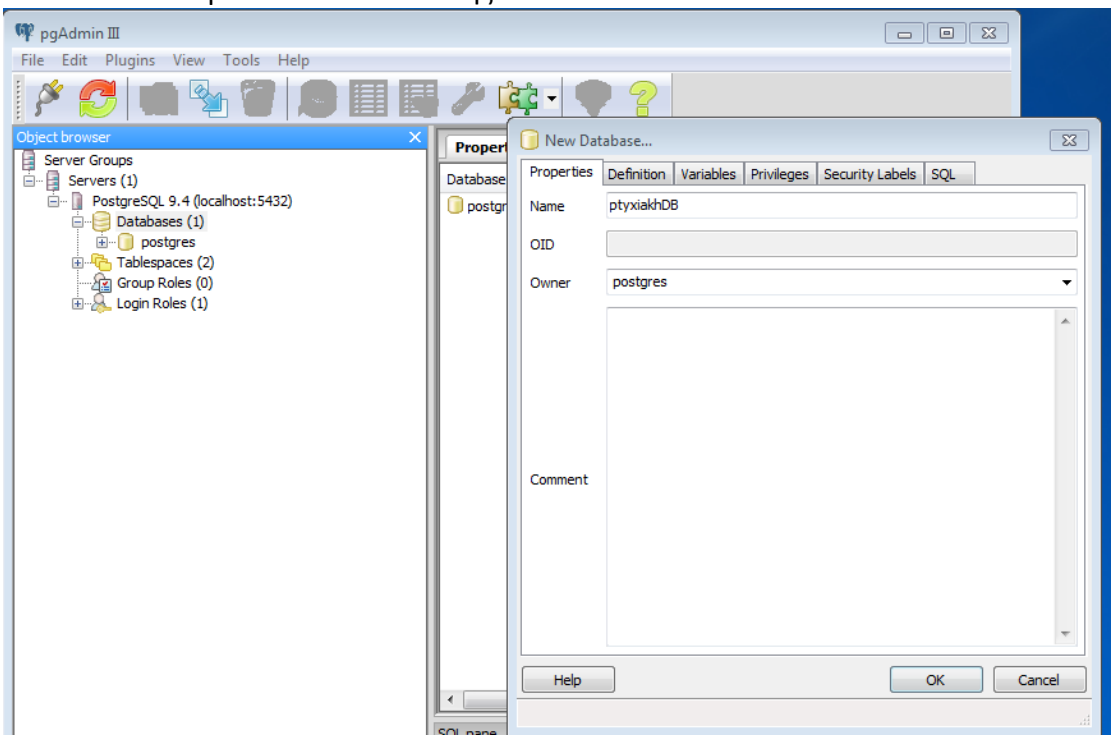
(postgres) ,και ολοκληρώνουμε την εγκατάσταση.



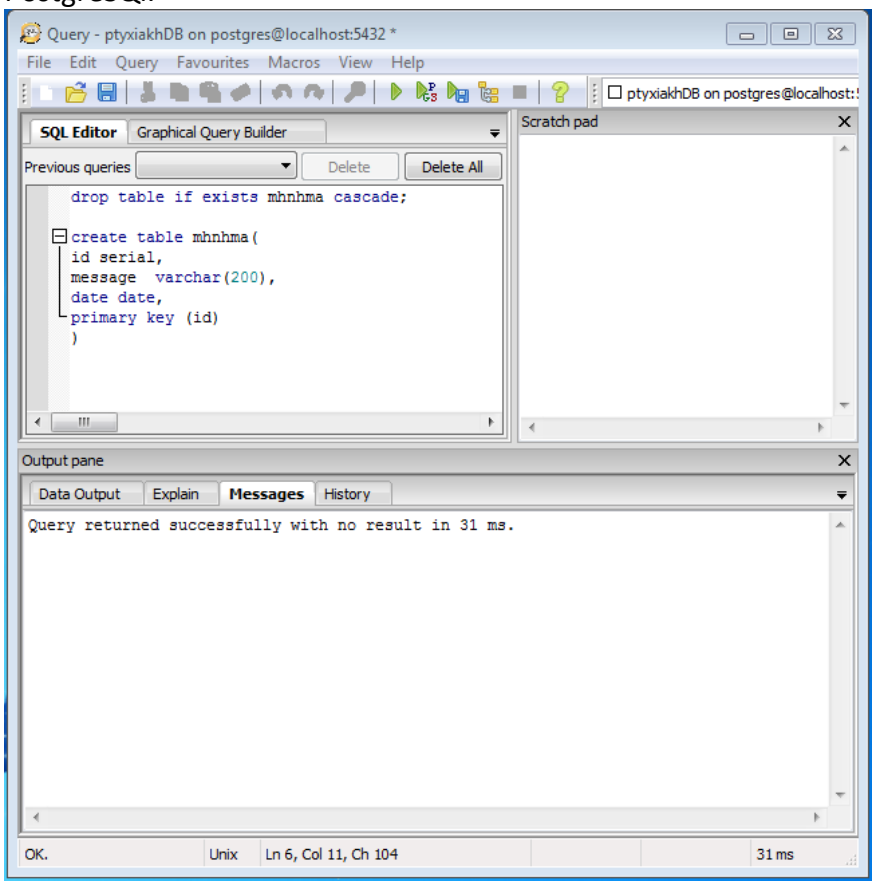
Κατά την εκκίνηση της postgres , πρέπει να συνδεθούμε με την super user Data Base με το password που ορίσαμε προηγουμένως.



Και έτσι συνδεόμαστε στον server της database



Παρακάτω έχουμε ένα παράδειγμα δημιουργίας ενός πίνακα στην βάση με την χρήση της PostgreSQl.



6.2 Διασύνδεση Της Postgres Με Την Java Στην Εφαρμογή Μας

Παρακάτω έχουμε κώδικα που κάνει αναζήτηση στην βάση (search button action Listener)

```
1   searchAll.addActionListener(new ActionListener() {
2       @Override
3       public void actionPerformed(ActionEvent ae) {
4
5
6           Connection c = null;
7           Statement stmt = null;
8           try {
9               Class.forName("org.postgresql.Driver");
10              c = DriverManager
11                  .getConnection("jdbc:postgresql://localhost:5432/ptyxiakhDB",
12                  "postgres", "1111");
13
14              stmt = c.createStatement();
15
16              ResultSet rs = stmt.executeQuery("SELECT * FROM mhnhma ;");
17
18              while (rs.next()) {
19                  //*************************************************
20              String id = rs.getString("id");
21                  String message = rs.getString("message");
22                  String date = rs.getString("date");
23                  String in_number = rs.getString("in_number");
24
25
26                  //*************************************************
27
28                  area.append(id + "; ");
29                  area.append(message + "; ");
30                  area.append(date + "; ");
31                  area.append(in_number + "; ");
32                  area.append("\n");
33              }
34              rs.close();
35              stmt.close();
36              c.close();
37          } catch (Exception e) {
38              e.printStackTrace();
39              System.err.println(e.getClass().getName() + ": " + e.getMessage());
40              System.exit(0);
41          }
42          GreatSearchPanel.repaint();
43      }
44  });
```

Στην γραμμή 6 δηλώνονται οι μεταβλητές διασύνδεσης και SQL δήλωσης
Στην γραμμή 12 καλούνται οι παράμετροι για την σύνδεση στην βάση οπού βλέπουμε address (localhost) port (5421) database name (ptyxiakhDB) server superuser (postgres) και password (1111).
Στην γραμμή 16 εκτελούμε την δήλωση στην βάση (select * from mhnhma) και στις αμέσως επόμενες γραμμές περνούμε τα εκάστοτε πεδία από τον πίνακα και τα παραθέτουμε στο java interface.

ΚΕΦΑΛΑΙΟ 7

7.1 Ανάλυση Της Διαδικασίας Αποστολής Και Λήψης Email Μέσω Της Java.

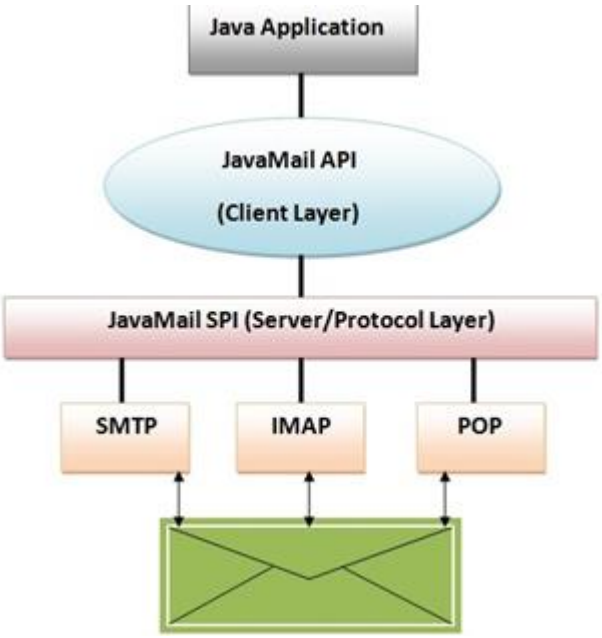
Για την αποστολή email μέσω του Interface , έγιναν πολλές προσπάθειες μέσω του Hardware του οποίου διαθέταμε. Όμως το Hardware το οποίο διαθέταμε δεν υποστήριζε mail servers οι οποίοι είχαν τα πρωτοκολλά SSL/TLS, και δυστυχώς οι περισσότεροι γνωστοί mail servers έχουν τα πρωτοκολλά αυτά. Ο λόγος είναι ότι οι περισσότεροι παροχή e-mails θα ήθελαν την ασφάλεια των δεδομένων των πελατών τους και φυσικά των servers τους. Γι' αυτό και χρησιμοποιούν τα πρωτοκολλά SSL/TLS, όπου συγκεκριμέναTLS είναι η συντομογραφία του **Secure Sockets Layer** και TLS είναι η συντομογραφία του **Transport Layer Security.**
Η επίσημη απάντηση της εταιρείας φαίνεται στην εικόνα 7.1 .

The SIM5218 allows sending receiving Email by POP3/SMTP. However, it does not support the SSL/TLS protocol. It is hardware limitation and we can not do anything about it. Simcom mentioned some firmware updates supporting SSl/TLS, but we can not ensure a certain date of this release.
We know that it is a limitation of our product but other users had this problem and they found a server that allow to send e-mails without SSL/TLS protocol.

Εικόνα 7.1

Έπειτα από λεπτομερή αναζήτηση στο internet δεν καταφέραμε να βρούμε κάποιον server ο οποίος να μην υποστηρίζει SSL/TLS. Έτσι αποφασίσαμε να στείλουμε και να λάβουμε E-mails μέσω της Java. Για να γίνει αυτό χρησιμοποιήσαμε ένα εργαλείο το οποίο προσφέρεται από την Java , το JavaMailApi.
Τώρα ας δούμε κάποια παραπάνω στοιχεία σχετικά με το JavaMailApi. Το JavaMailApi στην ουσία παρέχει μια ανεξάρτητη πλατφόρμα και ένα ανεξάρτητο πρωτόκολλο το οποίο χρησιμοποιείται για την αποστολή και την λήψη e-mailsμε την χρήση της Java, μέσω SMTP(*Simple Mail Transfer Protocol*),POP3(Post Office Protocol)και IMAP(InternetMessageAccessProtocol)κλπ.
Στην εικόνα 7.2 φαίνεται και η δομή του πως γίνεται η αποστολή email μέσω Java.

7.2 Δομή Της Διαδικασίας Αποστολής Και Λήψης Email Μέσω Java



Εικόνα 7.2

Το παραπάνω σχήμα μας επεξηγεί την πλήρη λειτουργία των επίπεδων του java mail API , και συγκεκριμένα ότι σε κάθε java εφαρμογή ,το client part του API επικοινωνεί με το server part του API ,το οποίο με την σειρά του καθορίζει τις παραμέτρους με τον εκάστοτε mail server ώστε να επιτευχθεί η σύνδεση μεταξύ client και mail server.

```
56          final String username = "ptyxiakh31523224@gmail.com";
57          final String password = "pty31523224";
58
59             send.addActionListener(new ActionListener() {
60              @Override
61              public void actionPerformed(ActionEvent ae) {
62
63                  Properties props = new Properties();
64          props.put("mail.smtp.starttls.enable", "true");
65          props.put("mail.smtp.auth", "true");
66          props.put("mail.smtp.host", "smtp.gmail.com");
67          props.put("mail.smtp.port", "587");
68                  Session session = Session.getInstance(props,
69            new javax.mail.Authenticator() {
70              protected PasswordAuthentication getPasswordAuthentication() {
71                  return new PasswordAuthentication(username, password);
72              }
73            });
74
75          try {
76
77              Message message = new MimeMessage(session);
78              message.setFrom(new InternetAddress("ptyxiakh31523224@gmail.com"));
79              message.setRecipients(Message.RecipientType.TO,
80                  InternetAddress.parse(sendToText.getText()));
81              message.setSubject(subjectText.getText());
82              message.setText(messageText.getText()
83                  + "\n\n No spam to my email, please!");
84
85              Transport.send(message);
86
87              System.out.println("Done");
88
89          } catch (MessagingException e) {
90              throw new RuntimeException(e);
91          }
92
```

Στον παραπάνω κώδικα φαίνεται η χρήση του java mail API για την αποστολή ενός Email.

Στην αρχή καθορίζονται οι παράμετροι του mail server στην περίπτωση μας google mail (γραμμές 64-67), στην συνέχεια τα στοιχεία του client (username & password γραμμές 64-67) καθώς επίσης ορίζουμε την διεύθυνση του αποστολέα(78) , του παραλήπτη (80), και τέλος συντάσσουμε το mail (subject , content γραμμές 81-82) τα οποία στοιχεία τα παίρνουμε από τα textfield της εφαρμογής μας.

Παρακάτω φαίνεται η χρήση του java mail api στην για την προβολή των εισερχομένων μηνυμάτων

.

```
362                 Properties props = new Properties();
363         props.setProperty("mail.store.protocol", "imaps");
364         try {
365             Session session = Session.getInstance(props, null);
366             Store store = session.getStore();
367             store.connect("imap.gmail.com", "ptyxiakh31523224@gmail.com", "pty31523224");
368             Folder inbox = store.getFolder("INBOX");
369             inbox.open(Folder.READ_ONLY);
370               String s0= null;
371           int i=   Integer.parseInt(mailNumber.getText());
372            Message msg = inbox.getMessage(inbox.getMessageCount()-i);
373             Address[] in = msg.getFrom();
374             for (Address address : in) {
375                 s0=address.toString();
376                 System.out.println("FROM:" + address.toString());
377             }
378             Multipart  mp = (Multipart) msg.getContent();
379            BodyPart bp = mp.getBodyPart(0);
380
382           String s3= bp.getContent().toString();
383
384             emailArea.append("FROM:" + s0+ "\n");
385              emailArea.append("SUBJECT:" + s2+ "\n");
386              emailArea.append("CONTENT:" + s3+ "\n");
387
```

Για αρχή καθορίζονται οι παράμετροι για την σύνδεση στον mail server(mail protocol, username, password) ,και ακολούθως για την διαχείριση των εισερχομένων μηνυμάτων (γραμμές 364 - 369). Στην συνέχεια επιλέγουμε τον αύξων αριθμό του (372) μηνύματος που θέλουμε και ρυθμίζουμε τις παραμέτρου (γραμμές 378-379) για να προβάλουμε τα περιεχόμενα του μηνύματος και να τα χρησιμοποιήσουμε στην εφαρμογή μας (γραμμές 384-386).

ΚΕΦΑΛΑΙΟ 8

Στο κεφάλαιο αυτό θα εξηγήσουμε τον κώδικα τον οποίο χρησιμοποιήσαμε στο Arduino για την υλοποίηση και χρήση του Interface.

8.1 Ενοποίηση Λειτουργιών Του Arduino Για Χρήση Του Δέκτη Gps Και Παράλληλα Για Την Αποστολή Και Λήψη SMS

Στο παρόν κεφάλαιο παρουσιάζουμε τον τρόπο με τον οποίο καταφέραμε και ενοποιήσαμε όλες τις βασικές λειτουργείς που χρησιμοποιεί η εφαρμογή μας ,οι όποιες περιλαμβάνουν την χρήση του gps καθώς επίσης και τις λειτουργίες ς αποστολής και λήψης SMS .
Για αρχή δηλώνουμε όλες τις μεταβλητές και σταθερές που θα χρησιμοποιηθούν από τις αντίστοιχες συναρτήσεις της εφαρμογής μας.

```
const char pin[]="5916"; // to pin gia thn sim
const char phone_number[]="+306978XXXXXX";// o ari8mos pou 8 stalei to SMS
String myMessage; // variable gia to keimeno tou SMS
char SMS[200];// char array gia thn leitoyrgeia SMS

int8_t answer;// variable gia xrhsh me ta AT commands
int x;// variable genikhs xrhshs
int onModulePin= 2;

char aux_str[30]; // char array gia xrhsh me ta AT commands

int button=0;// variable gia thn epilogh leitourgeias
int myInt;// variable gia thn epilogh toy epi8hmhtou SMS

char gps_data[100];// char array gia xrhsh me GPS
int counter;// counter gia GPS
```

Ο τρόπος λειτουργίας της εφαρμογής μας βασίζεται στην δημιουργία ξεχωριστών συναρτήσεων για κάθε βασική λειτουργιά , και με τι χρήση μιας απλής if case μπορούμε να επιλέξουμε μεταξύ των συναρτήσεων ,όπως φαίνεται παρακάτω :

```
  Serial.begin(115200);
Serial.println("epilogh leitourgias");// enarksh
while (Serial.available()==0) {  // anamonh epiloghs
  }
  button=Serial.parseInt();// diavasma epiloghs
  if(button==1){// epilogh 1 send sms
  Serial.println("grafte ena mhnhma");
  while (Serial.available()==0) {
  }
  myMessage=Serial.readString();  // diavasma mhnhatos
  send_sms();// klhsh ths send_sms
  }
  if(button==2){// epilogh 2 diavasma eiserxomenwn sms
  Serial.println("poio mhnhma thete??");
  delay(5000);
  while (Serial.available()==0) {
  }
  myInt=Serial.parseInt(); // epilogh sms pros provolh
  Serial.println("molis diavasa");
  delay(5000);
  Serial.flush();
 receiveSMS(myInt);// klhsh receiveSMS
  }
    if(button==3){// epilogh 3 gps
    Serial.println("gps.......");
 ModeGPS(); // klhsh ModeGPS
  }
```

Παρακάτω περιγράφουμε της αντίστοιχες συναρτήσεις για κάθε λειτουργία

8.2 Αποστολή SMS

Η μονή διαφορά εδώ σχετικά με την αποστολή sms που αναφέραμε στο προηγούμενο μέρος είναι η χρήση απλής συνάρτησης αντί για το κυρίως πρόγραμμα που είχαμε παραπάνω, και έτσι έχουμε:

```
void send_sms(){
Serial.println("Starting send SMS");
  power_on();
  delay(3000);
  //sets the PIN code
  sprintf(aux_str, "AT+CPIN=%s", pin);
  sendATcommand(aux_str, "OK", 2000);
  delay(3000);
  Serial.println("Connecting to the network...");
  while( (sendATcommand("AT+CREG?", "+CREG: 0,1", 500) ||
    sendATcommand("AT+CREG?", "+CREG: 0,5", 500)) == 0 );
  Serial.print("Setting SMS mode...");
  sendATcommand("AT+CMGF=1", "OK", 1000);    // sets the SMS mode to text
  Serial.println("Sending SMS");
  sprintf(aux_str,"AT+CMGS=\"%s\"", phone_number);
  answer = sendATcommand(aux_str, ">", 2000);    // send the SMS number
  if (answer == 1)
  {
    Serial.println(myMessage);
    Serial.write(0x1A);
    answer = sendATcommand("", "OK", 20000);
    if (answer == 1)
    {
      Serial.print("Sent ");
    }
    else
    {
      Serial.print("error ");
    }
  }
  else
  {
    Serial.print("error ");
    Serial.println(answer, DEC);
  }

}

```

8.3 Λήψη Και Επιλογή SMS

Όπως και στην αποστολή SMS έτσι και εδώ οι μονές διαφορές είναι η χρήση συνάρτησης αντί κυρίου προγράμματος:

```
void receiveSMS(int i){
Serial.println("Starting receive SMS");
  power_on();
  delay(3000);
  //sets the PIN code
  sprintf(aux_str, "AT+CPIN=%s", pin);
  sendATcommand(aux_str, "OK", 2000);
  delay(3000);
  Serial.println("Setting SMS mode...");
  sendATcommand("AT+CMGF=1", "OK", 1000);    // sets the SMS mode to text
  sendATcommand("AT+CPMS=\"SM\",\"SM\",\"SM\"", "OK", 1000);    // selects the memory

  String scounter= String(i);// int to string
String kati="AT+CMGR=";          //first part of AT command
String katei=kati+scounter;       //attaching both to form the "AT+CMGR=1"
char cbuff[12];
katei.toCharArray(cbuff,12);      //converting to char array

  answer = sendATcommand(cbuff, "+CMGR:", 2000);    // reads the first SMS
  if (answer == 1)
  {
    answer = 0;
    while(Serial.available() == 0);
    // this loop reads the data of the SMS
    do{
      // if there are data in the UART input buffer, reads it and checks for the answer
      if(Serial.available() > 0){
        SMS[x] = Serial.read();
        x++;
        // check if the desired answer (OK) is in the response of the module
        if (strstr(SMS, "OK") != NULL)
        {
          answer = 1;
        }
      }
    }
    while(answer == 0);    // Waits for the answer with time out

    SMS[x] = '\0';

    Serial.print(SMS);
  }
  else
  {
    Serial.print("error ");
    Serial.println(answer, DEC);
  }
}

```

8.4 Η Λειτουργία Του GPS

Και τέλος έχουμε την λειτουργία του Gps . Και εδώ οι διαφορές είναι ελάχιστες με μια μικρή διαφοροποίηση στη χρήση μιας εσωτερικής for loop αντί της void loop που είχαμε παραπάνω:

```
void ModeGPS(){
 Serial.println("Starting GPS");
  answer = sendATcommand("AT+CGPS=1,1","OK",1000);
  if (answer == 0)
  {
    Serial.println("Error starting the GPS");
    Serial.println("The code stucks here!!");
    while(1);
  }
for (int i =0;i<1000;i++){
answer = sendATcommand("AT+CGPSINFO","+CGPSINFO:",1000);    // request info from GPS
  if (answer == 1)
  {
    counter = 0;
    do{
      while(Serial.available() == 0);
      gps_data[counter] = Serial.read();
      counter++;
    }
    while(gps_data[counter - 1] != '\r');
    gps_data[counter] = '\0';
    if(gps_data[0] == ',')
    {
      Serial.println("No GPS data available");
    }
    else
    {
      Serial.print("GPS data:");
      Serial.print(gps_data);
      Serial.println("");
    }
  }
  else
  {
    Serial.println("Error");
  }
  delay(5000);
}
```

Να αναφέρουμε επίσης ότι οι κοινές και για τις 3 λειτουργιές συναρτήσεις (sendATcommand και power_on) παρέμειναν όπως είχαν και στα προηγούμενά παραδείγματα

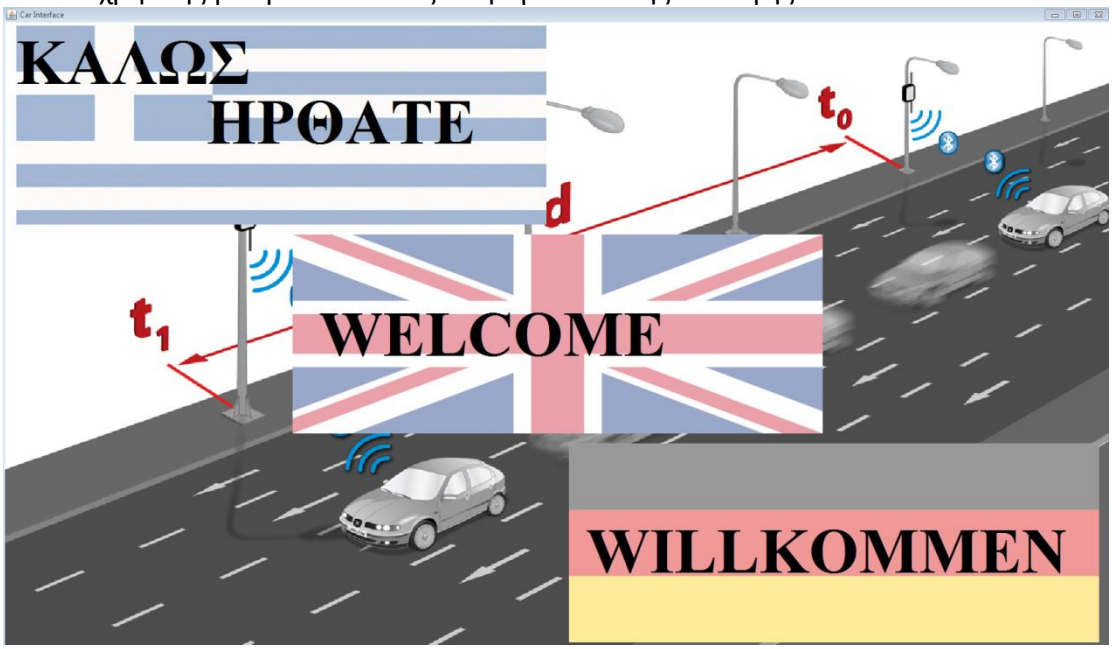
ΚΕΦΑΛΑΙΟ 9

Στο κεφάλαιο αυτό θα γίνει λεπτομερής ανάλυση της εφαρμογής .

Στο παρόν κεφάλαιο θα περιγράψουμε την java εφαρμογή που διαχωρίζει το hardware και τα δεδομένα από αυτό .

9.1 Εκκίνηση Εφαρμογής

Κατά την εκκίνηση της εφαρμογής εμφανίζετε στον χρηστή ένα μενού επιλογής γλώσσας , οπού ο χρήστης μπορεί να επιλέξει την γλωσσά της επιλογής του.



Από την προγραμματιστική πλευρά αυτό που κάναμε ήταν σε κάθε σε κάθε σημαία να βάλουμε έναν action listener και κάθε φορά που ο χρήστης πατά κάποια από αυτές να του ανοίγει την εφαρμογή στην αντίστοιχη γλωσσά. Παρακάτω δείχνουμε πως βάζουμε την εικόνα σε ένα JButton.

```
Icon warnIcon = new ImageIcon("Greece.jpg");
      JButton button = new JButton(warnIcon);
      panel.add(button);
      button.setSize(800, 300);
```

Και εδώ βλέπουμε τον action listener που καλεί την Ελληνική έκδοση της εφαρμογής (μαζί με τα απαραίτητα exceptions φυσικά).

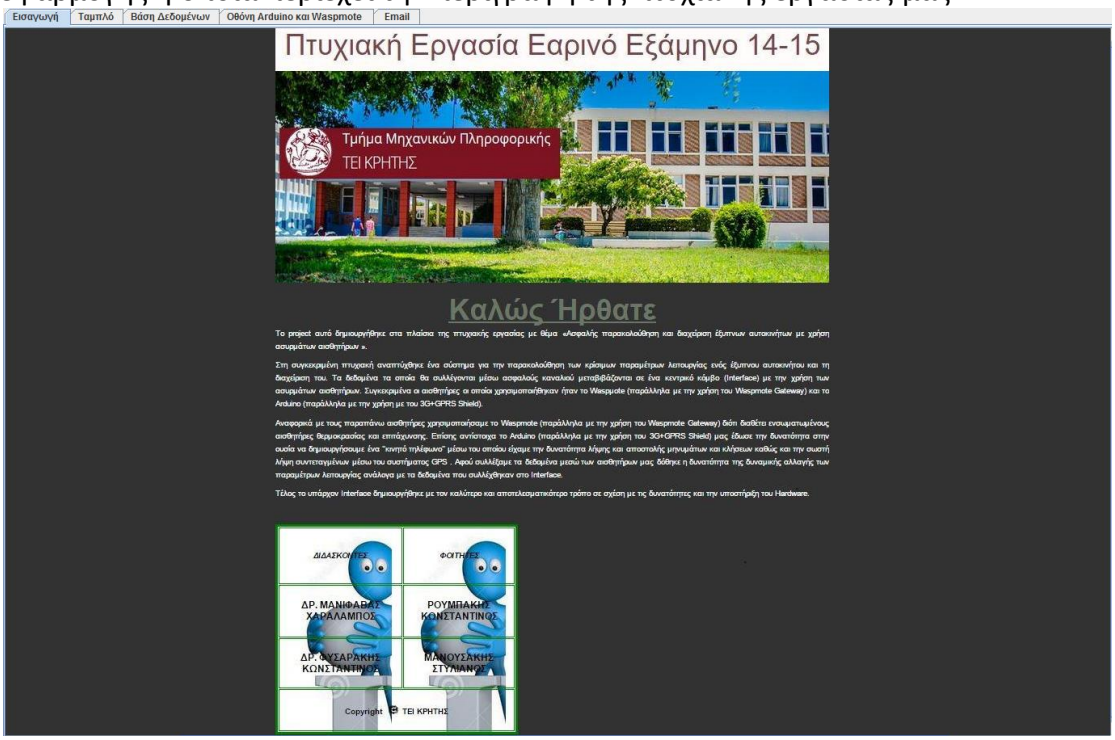
```
button.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
          try {
            JavaArduinoGRE gre = new JavaArduinoGRE();
          } catch (SerialPortException ex) {
            Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
          } catch (InterruptedException ex) {
            Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
          } catch (IOException ex) {
```

```
            Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
        } catch (JAXBException ex) {
            Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
        }
        frame.setVisible(false);
    }
});
```

Όπου JavaArduinoGRE είναι η κλάση της Ελληνικής έκδοσης. Αντίστοιχη διαδικασία γίνεται για την Αγγλική και την Γερμανική έκδοση επίσης (JavaArduinoENG και JavaArduinoGER).

9.2 Εισαγωγή

Με το μπούμε στην έκδοση της επιθυμητής γλώσσας βλέπουμε την καρτέλα εισαγωγή της εφαρμογής η όποια περιέχει την περιγραφή της πτυχιακής εργασίας μας.

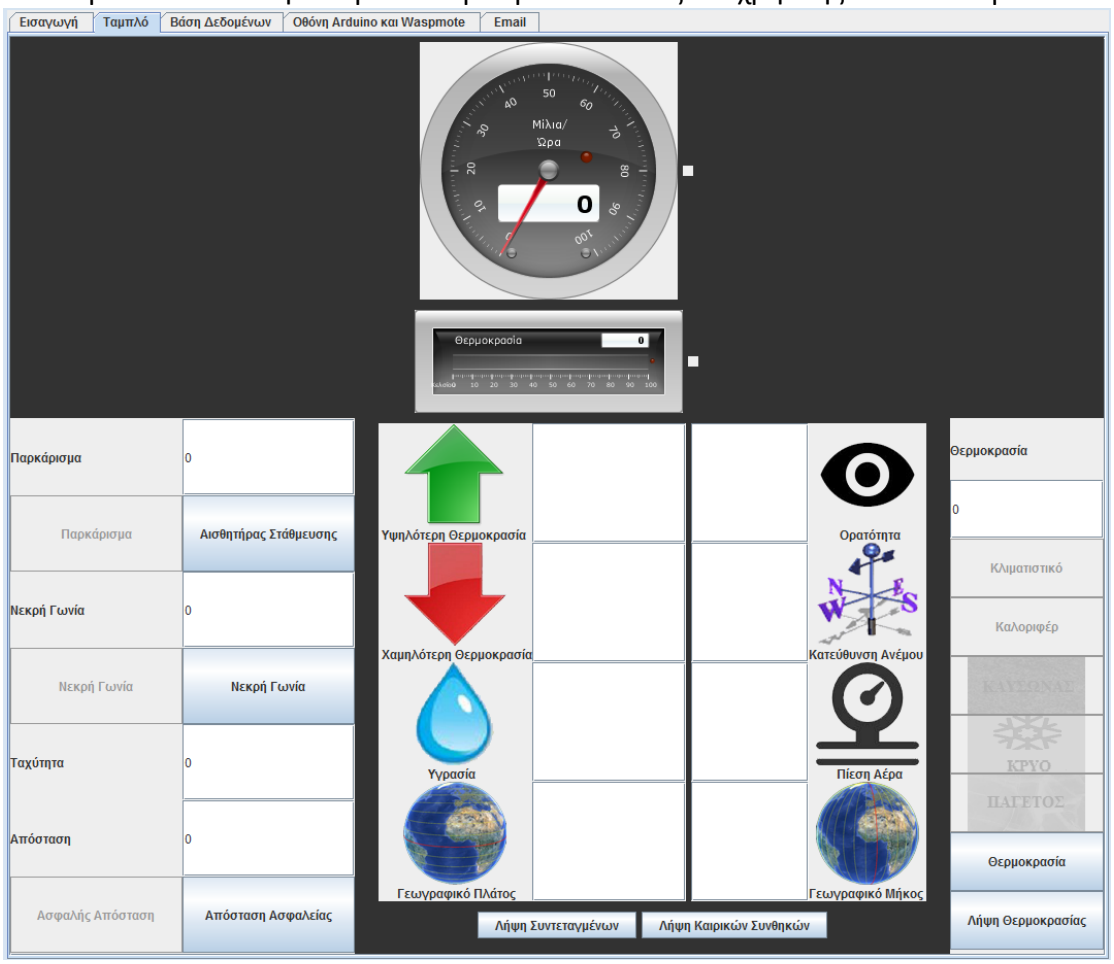


Από τεχνικής μεριάς προσπαθήσαμε να περάσουμε μια html σελίδα στην εφαρμογή αλλά από την java δεν μπορούσαμε να περάσουμε περεταίρω στιλιστικές (css) αλλαγές και έτσι αποφασίσαμε απλά να πάρουμε την html σελίδα σαν εικόνα και την βάλαμε κανονικά σε ένα Label και αυτό σε ένα container και το προσθέσαμε στο πρόγραμμα μας. Παρακάτω ο κώδικας που πράττει τα παραπάνω

```
BufferedImage image1 = null;
    JPanel superPanel = new JPanel(new BorderLayout());
    try {
        image1 = ImageIO.read(new File("imageFinal.jpg"));
    } catch (IOException ex) {
    }
    JLabel picLabel1 = new JLabel(new ImageIcon(image1));
    superPanel.add(picLabel1, BorderLayout.NORTH);
```

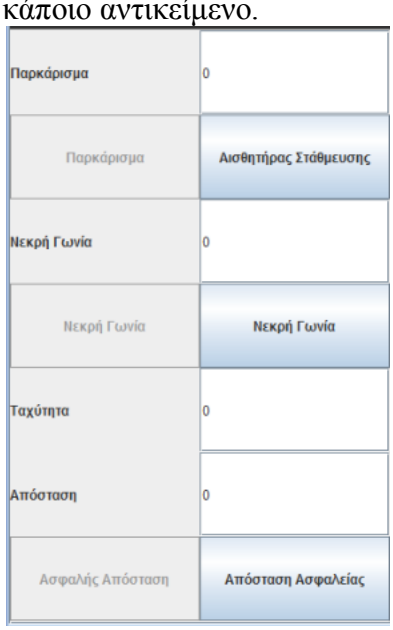
9.3 Ταμπλό

Το επόμενο πάνελ στην σειρά που μπορεί να επιλέξει ο χρήστης είναι το ταμπλό :



9.3.1 Συναρτήσεις Προσδιορισμού Ασφαλούς Απόστασης

Στο ταμπλό περιλαμβάνονται οι λειτουργίες των διαδικασιών ασφαλούς παρκαρίσματος, της νεκρής γωνίας και της απόστασης ασφάλειας από το προπορευόμενα οχήματα η γενικά από κάποιο αντικείμενο.



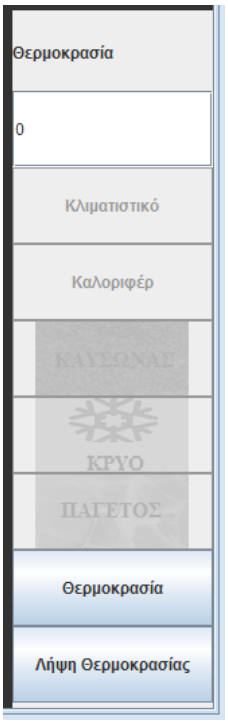
Παραδείγματος χάρη στον αισθητήρα στάθμευσης παίρνουμε (θεωρητικά) την ένδειξη από τον αισθητήρα κατά την διάρκεια της διαδικασίας της στάθμευσης και ανάλογα με την απόσταση από τα αντικείμενα και το ποσό κοντά βρίσκονται στο όχημα ενεργοποιούνται οι τρεις καταστάσεις του κουμπιού "Παρκάρισμα" . Με μια απλή χρήση if statement όσο αφορά τον κώδικα γίνετε η επιλογή του χρώματος του κουμπιού και όταν μπει στις περιπτώσεις προειδοποίησης και κίνδυνου (πορτοκάλι και κόκκινο) τότε ενεργοποιούνται και οι αντίστοιχοι ήχοι ταυτόχρονα . Οι ήχοι τρέχουν καλώντας την μια κλάση και τις μεθόδους της (AudioPlayer02) όπως η playAudio() οπού παίρνει σαν όρισμα το path στο οποίο είναι το ηχητικό αρχείο . Στην περίπτωση του κινδύνου (κόκκινο πάνελ) τότε ενεργοποιούνται και συγκεκριμένα leds στο Arduino , αρχικοποιώντας την σειριακή επικοινωνία μεταξύ java με την πλακέτα και γράφοντας τα κατάλληλα δεδομένα στο port ("1" , "2" κτλ) ενεργοποιώντας τις κατάλληλες συναρτήσεις στην μεριά του Arduino για να διαχειριστούν τα leds . Παρακάτω έχουμε τον κώδικα που πραγματοποιεί τα παραπάνω:

```
if (park <= 0.5) {
            parkingB.setBackground(Color.red);

            SerialPortParser seri = null;
            try {
               seri = new SerialPortParser();
            } catch (SerialPortException ex) {
               Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
            }
            try {
               seri.readPort2();
            } catch (SerialPortException ex) {
               Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
            } catch (UnsupportedEncodingException ex) {
               Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
            }
            try {
               Thread.sleep(1000);
               seri.writePort2("4");
            } catch (SerialPortException ex) {
               Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
            } catch (UnsupportedEncodingException ex) {
               Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
            } catch (InterruptedException ex) {
               Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
            }

            final String path = "sound file path";

            new Thread(new Runnable() {

               @Override
               public void run() {
                  try {
                     AudioPlayer02 hxoi = new AudioPlayer02();
```

```
                hxoi.playAudio(path);
                try {
                    Thread.sleep(18000);
                } catch (InterruptedException ex) {
                    Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
                }
                hxoi.stop();

            } catch (InterruptedException ex) {
                Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
            }

        }
    }).start();

}
```

Να αναφέρουμε ότι χρησιμοποιήσαμε threads με σκοπό να μπορούν να τρέξουν παράλληλα και ο χρωματισμός του κουμπιού και η χρήση των leds καθώς και η αναπαραγωγή των ήχων κίνδυνου.
Παρόμοιες διαδικασίες γίνονται και για τις συναρτήσεις της νεκρής γωνίας και της απόστασης ασφάλειας με την διάφορα στην απόσταση ασφάλειας να έγκειται στην χρήση 2 Jtextfields αντί για ένα που είχαμε στις προηγούμενες περιπτώσεις. Συγκεκριμένα περνάμε την ταχύτητα του οχήματος και ταυτόχρονα την απόσταση.

9.3.2 Συναρτήσεις Ένδειξης Θερμόμετρου



Όσον αφορά το πάνελ με τις ενδείξεις θερμοκρασίας αυτό περιέχει τα κουμπιά "θερμοκρασία" και "Λήψη Θερμοκρασίας". Το "Λήψη Θερμοκρασίας" απομονώνει τα δεδομένα θερμοκρασίας από την πλακέτα του Waspmote και μας τα επιστρέφει στο JTextField στο πάνω

μέρος του πάνελ. Με το κουμπί "Θερμοκρασία" περνούμε την τιμή από το παραπάνω πεδίο και με χρήση if statement(παρόμοιας λειτουργιάς με τις συναρτήσεις ασφαλών αποστάσεων) επιλεγεί αν θα ενεργοποιήσει το καλοριφέρ ή το κλιματιστικό, καθώς επίσης και σε ποια από τις τρεις κατάστασης θερμοκρασίας (καύσωνας ,κρύο ή παγετός) θα ενεργοποιηθεί. Όπως και στις συναρτήσεις ασφαλών αποστάσεων έτσι και εδώ στην περίπτωση που έχουμε καύσωνα ενεργοποιείται και η διαδικασία με το led και στο Arduino. Παρακάτω έχουμε τον κώδικα για το πώς καλούμε την μέθοδο για να πάρουμε δεδομένα από το Waspmote.

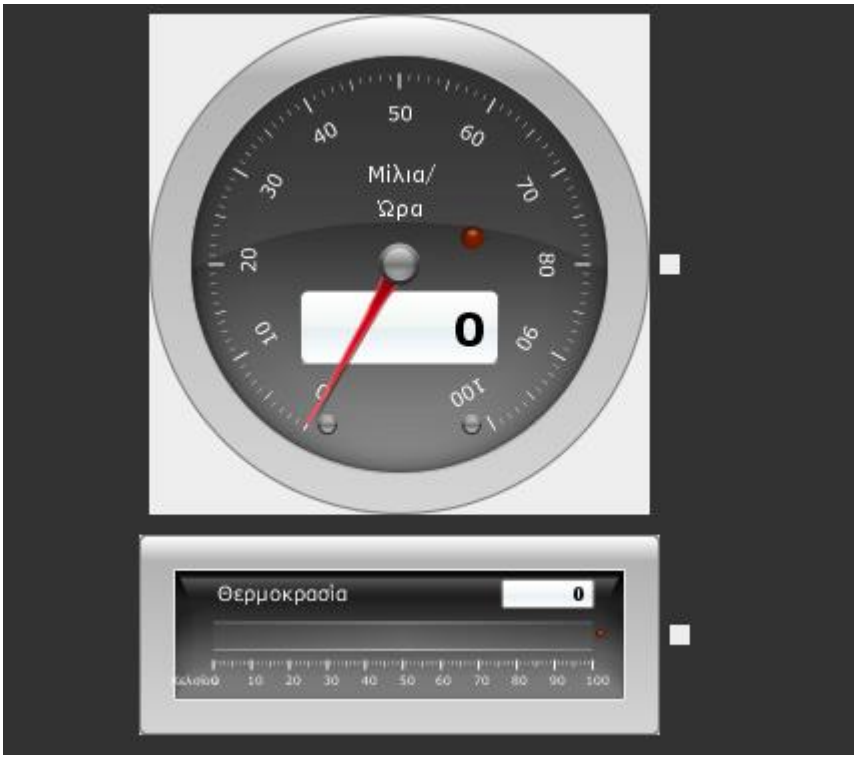
```
getTempSerial.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
          WaspmoteSerialMonitor wasp = new WaspmoteSerialMonitor();
          try {
            tempText.setText(wasp.getTemp());
          } catch (SerialPortException ex) {
            Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
          } catch (UnsupportedEncodingException ex) {
            Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
          }

        }
      });
```

Και ακολουθεί η μέθοδος getTemp() του WaspmoteSerialMonitor που αποκόπτει και επιστρέφει τα χρήσιμα δεδομένα (θερμοκρασία) από το Waspmote.

```
public String getTemp() throws SerialPortException, UnsupportedEncodingException {
      float tepm = 0;
      serialPort2.openPort();
      byte[] buffer = serialPort2.readBytes(150);
      System.out.println("read done ");
      String value = new String(buffer, "ISO-8859-7");
      serialPort2.closePort();
      try {
        Thread.sleep(1000);
      } catch (InterruptedException ex) {
        Thread.currentThread().interrupt();
      }
      int x = value.indexOf("IN_TEMP:");
      String temp0 = value.substring(x);
      int x2 = temp0.indexOf("#");
      int k = "IN_TEMP:".length();
      String temper = temp0.substring(8, x2);
      System.out.println(temper);
      return temper;
    }
```

Οι λειτουργείες για την εναλλαγή και ενεργοποίηση των πάνελ, καθώς και την χρήση των Arduino led είναι παρόμοιες με αυτή των συναρτήσεων προσδιορισμού της ασφαλής απόστασης.

9.3.3 Οι Μετρητές Ένδειξης Ταχύτητας Και Θερμοκρασίας



Για να δείξουμε τις ενδείξεις θερμοκρασίας και ταχύτητας χρησιμοποιήσαμε το eu hansolo Steelseries api (SteelSeries-3.9.3.jar στον φάκελο της εφαρμογής) ώστε να μπορέσουμε να έχουμε μια γραφική αναπαράσταση των ενδείξεων από τους αισθητήρες (πραγματικούς και θεωρητικούς) .Από την μεριά του κώδικα αυτό που κάναμε ήταν να κάνουμε import το jar και από εκεί να δουλέψουμε με τον τρόπο λειτουργιάς του api.
Για αρχή δηλώσαμε τα 2 ειδή των μετρητών που θέλαμε

```
Radial gauge = new Radial();
Linear thermo = new Linear();
```

Αμέσως μετά αρχικοποιήσαμε τις παραμέτρους (Τα "ζωγραφίσαμε") για το καθένα ,και τα τοποθετήσαμε σε ένα container για να μπουν στο πάνελ μας. Παρακάτω ακολουθεί ο κώδικας που αρχικοποιεί το θερμόμετρο.

```
public JPanel TestLinear() {
    JPanel panel = new JPanel() {
        @Override
        public Dimension getPreferredSize() {
            return new Dimension(260, 100);
        }
    };
```

```
        thermo.setTitle("Θερμοκρασία");
        thermo.setUnitString("Κελσίου");
        panel.setLayout(new BorderLayout());
        panel.add(thermo, BorderLayout.CENTER);
        GaugePanel2.add(panel);
        JPanel buttonsPanel = new JPanel();

        GaugePanel2.add(buttonsPanel, BorderLayout.NORTH);
        GaugePanel2.setBackground(new Color(50, 50, 50));
        return GaugePanel2;

    }
```

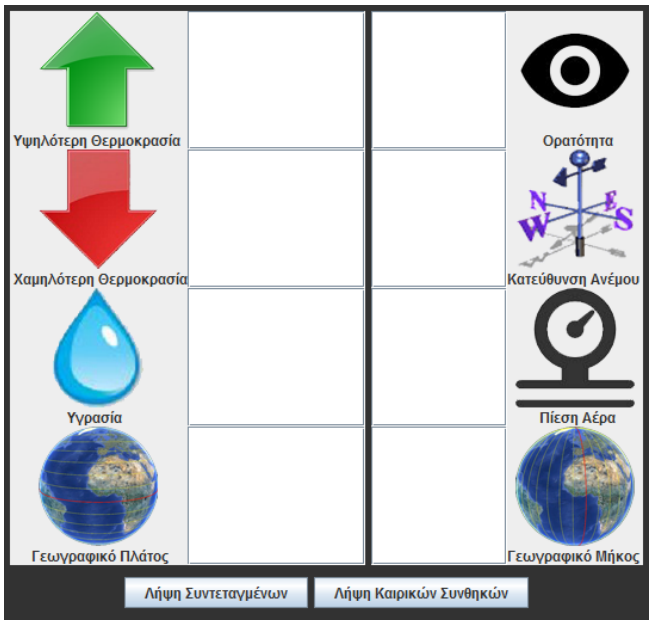
Παραπάνω φαίνεται ότι δηλώνουμε τις διαστάσεις και τα ονόματα των μονάδων μέσα στον μετρητή, καθώς επίσης φαίνεται πως τοποθετείτε στο container για να επιστραφεί στο κεντρικό ταμπλό.

Για να περάσουμε τιμές μέσα στον μετρητή το μονό που έχουμε να κάνουμε είναι να πάρουμε την τιμή και με την κατάλληλη μέθοδο του api να την χρησιμοποιήσουμε. Έτσι έχουμε για το θερμόμετρο:

```
 try {
                double value2 = Double.valueOf(tempText.getText());
                thermo.setValueAnimated(value2);
            } catch (NumberFormatException ex) {
                System.err.println("invalid input");
            }
```

Φυσικά με παρόμοιο τρόπο δουλεύει και ο μετρητής ταχύτητας

9.3.4 Πρόβλεψη Καιρικών Συνθηκών Βάση Του Δέκτη GPS



Για την λήψη των καιρικών συνθηκών χρησιμοποιήσαμε το yahoo weather api το οποίο χρησιμοποιεί συντεταγμένες με σκοπό την προβολή πρόγνωσης καιρού για κάποια συγκεκριμένη περιοχή.

9.3.5 Λήψη Συντεταγμένων GPS

Για την λήψη των συντεταγμένων χρησιμοποιήσαμε τον δεκτή Gps που βρίσκεται πάνω στο shield. Για να πάρουμε τα δεδομένα από το Gps χρησιμοποιήσαμε τις συναρτήσεις GPSlan() και GPSlon() οι όποιες σε συνεργασία με την μέθοδο από την πλευρά του Arduino μας επιστρέφουν τις συντεταγμένες γεωγραφικού πλάτους και μήκους αντίστοιχα. Παρακάτω ο κώδικας των μεθόδων :

```
public String GPSlan() throws SerialPortException, UnsupportedEncodingException {
    serialPort.openPort();
   serialPort.setParams(115200, 8, 1, 0);
    byte[] buffer = serialPort.readBytes(60);
    String value = new String(buffer, "ISO-8859-7");
    serialPort.closePort();

    String temper = util.StringCutter(value, "data:", ",N,");
    return temper;
}
```

Και

```
public String GPSlon() throws SerialPortException, UnsupportedEncodingException {
    serialPort.openPort();
  serialPort.setParams(115200, 8, 1, 0);
    byte[] buffer = serialPort.readBytes(60);
    String value = new String(buffer, "ISO-8859-7");
    serialPort.closePort();

    String temper = util.StringCutter(value, ",N,", ",E,");
    return temper;
}
```

Και οι δυο συναρτήσεις αυτό που κάνουν είναι να ανοίγουν την σειριακή επικοινωνία με την πλακέτα και να πάρουν τα δεδομένα από την συνάρτηση που δίνει τις συντεταγμένες μεσώ του Arduino . Έχοντας μελετήσει την μορφή των δεδομένων που μας δίνει το Arduino μπορούμε με ευκολία να απομονώσουμε τις συντεταγμένες με την χρήση της StringCutter() η όποια είναι η συνάρτηση που απομονώνει τα επιθήματα δεδομένα. Και εν συνεχεία τα επιστρέφουμε στο κυρίως πάνελ της εφαρμογής μας. Παρακάτω ακολουθεί ο ActionListener του κουμπιού που καλεί τις παραπάνω μεθόδους(και έχει και τα απαραίτητα exceptions).

```
getGps.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
           try {
              latT.setText(seri.GPSlan());
```

```
        } catch (SerialPortException ex) {
            Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
        } catch (UnsupportedEncodingException ex) {
            Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
        }
        try {
            lonT.setText(seri.GPSlon());;
        } catch (SerialPortException ex) {
            Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
        } catch (UnsupportedEncodingException ex) {
            Logger.getLogger(Sensors.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
});
```

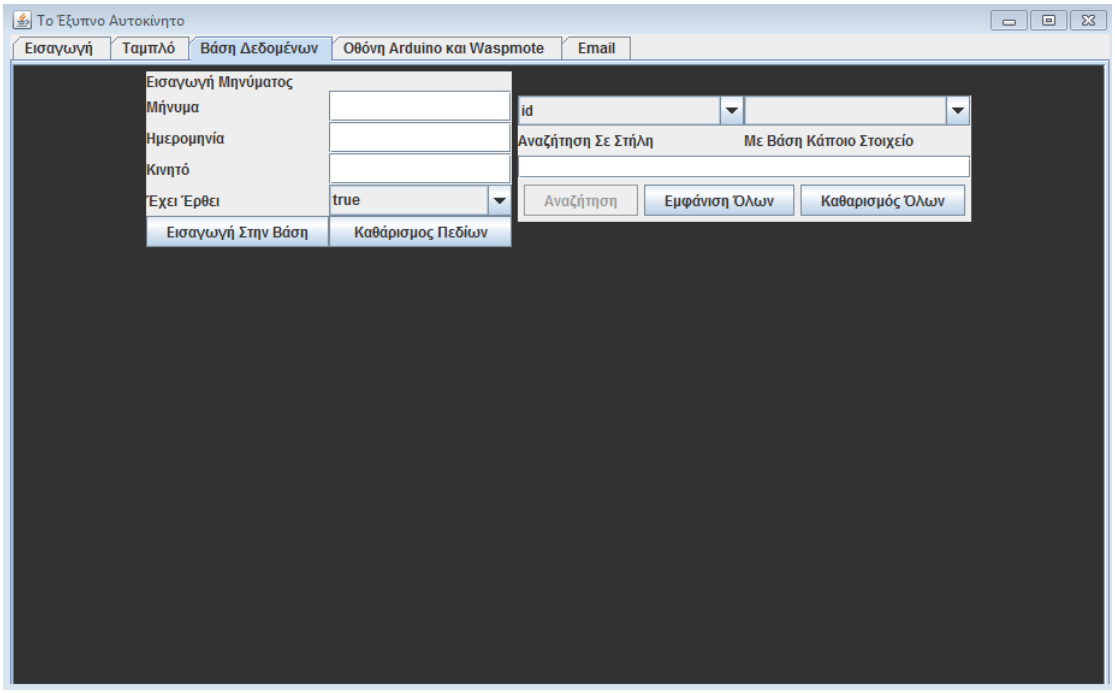
9.3.6 Προβολή Και Λήψη Καιρικών Συνθηκών

Με το που πάρουμε τα δεδομένα του Gps τότε μπορούμε να καλέσουμε το yahooWeather api βάση τον δεδομένων αυτών .Αφού πάρουμε τα δεδομένα του Gps τα μετατρέπουμε σε WOEID με το οποίο δουλεύει το yahooWeather χρησιμοποιώντας την μέθοδο calculateWOEID της κλάσης CoorrdinatesUtil. Έπειτα περνούμε τα καιρικά δεδομένα και τα περνάμε στα αντίστοιχα JTextField πάνω στο πάνελ . παρακάτω ακολουθεί η κλάση του yahooWeather.

```
public void colorWeather() throws JAXBException, IOException, Exception {

    String woeid;
    String forecast;
    String atmosphere;
    String wind0;
    YahooWeatherService service = new YahooWeatherService();
    CoorrdinatesUtil coor= new CoorrdinatesUtil();


    woeid =  coor.calculateWOEID(lonT.getText(), latT.getText());
    Channel channel = service.getForecast(woeid, DegreeUnit.CELSIUS);

    forecast = channel.getItem().getForecasts().get(0).toString();
    atmosphere = channel.getAtmosphere().toString();
    wind0 = channel.getWind().toString();

  System.out.println(forecast);
  System.out.println(atmosphere);
  System.out.println(wind0);
  tempHigh.setText(util.StringCutter(forecast, "high=", ", text") + " °C ");
  tempLow.setText(util.StringCutter(forecast, "low=", ", high") + " °C ");
  oppacity.setText(util.StringCutter(atmosphere, "visibility=", ", press") + " Meters ");
```

```
humidy.setText(util.StringCutter(atmosphere, "humidity=", ", visibi") + " % ");
wind.setText(util.StringCutter(wind0, "direction=", ", speed") + " degrees ");
pressure.setText(util.StringCutter(atmosphere, "pressure=", ", rising") + " atmospheres ");

}
```

9.4 Βάση Δεδομένων Μηνυμάτων



Το επόμενο πάνελ στη σειρά που έχει στην διάθεση του ο χρήστης είναι της βάσης δεδομένων για την χρήση πάνω στα μηνύματα (sms) που δέχεται και αποστέλλει το Arduino.

Αυτό που κάναμε εδώ ήταν να δημιουργήσουμε ένα java interface και να το συνδέσουμε με την βάση ,ώστε να μπορούμε να εκτελούμε SQL queries μέσω αυτού. Από την μεριά της εφαρμογής χρησιμοποιήσαμε την postgres έκδοση του jdbc (το postgresql-9.4-1201.jdbc4.jar στα αρχεία της εφαρμογής) για να διασυνδέουμε την java με την Postgres, και από κει και πέρα χρησιμοποιήσαμε το api του Jdbc με σκοπό να το αντιστοιχήσουμε . Όταν λοιπόν πατήσουμε το κουμπί "Εισαγωγή Στην Βάση" εκτελείται ο ακόλουθος actionListener:

```
submit.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent ae) {
        Connection c = null;
      Statement stmt = null;
      String message = String.valueOf(tField1.getText());
      String date = String.valueOf(tField2.getText());
      String in_number = String.valueOf(tField3.getText());
      String incoming= incomingBox.getSelectedItem().toString();
      try {
        Class.forName("org.postgresql.Driver");
        c = DriverManager
            .getConnection("jdbc:postgresql://localhost:5432/ptyxiakhDB",
```

```
                "postgres", "1111");
            String qry = "'" + message + "','" + date + "','" + in_number +"','" + incoming + "'";
            stmt = c.createStatement();
            String sql = "INSERT INTO mhnhma VALUES(default, " + qry + ");";
            stmt.executeUpdate(sql);
          } catch (Exception e) {
            e.printStackTrace();
            System.err.println(e.getClass().getName() + ": " + e.getMessage());
            System.exit(0);
          }
          }
    });
```

Εδώ βλέπουμε ότι αρχικοποιούμε τις μεταβλητές για την διασύνδεση
Παίρνουμε τα δεδομένα από τα JTextFields της διεπαφής
Παρακάτω πραγματοποιούμε την διασύνδεση με την βάση

```
(c = DriverManager
                .getConnection("jdbc:postgresql://localhost:5432/ptyxiakhDB",
                "postgres", "1111");
```

) περνάμε τα δεδομένα από τα Fields σε ένα sql query (String qry = "'" + message + "','" + date + "','" + in_number +"','" + incoming + "'";
)

Και εκτελούμε το query στην βάση (stmt.executeUpdate(sql);)

Στο δίπλα πάνελ εκτελούμε την αναζήτηση στην βάση εκτελώντας παρόμοια διαδικασία, έτσι όταν πατάμε το κουμπί "Αναζήτηση" εκτελείται ο ακόλουθος actionListener.

```
  search.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
          Connection c = null;
          Statement stmt = null;
          try {
            Class.forName("org.postgresql.Driver");
            c = DriverManager
                .getConnection("jdbc:postgresql://localhost:5432/ptyxiakhDB",
                "postgres", "1111");
            String arg0 = cmbox.getSelectedItem().toString();
            String arg1 = (String) cmbox2.getSelectedItem().toString();

            stmt = c.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM mhnhma WHERE " + arg0 + "=" + "'" + arg1 + "'" +
";");

            while (rs.next()) {

              String id = rs.getString("id");
              String message = rs.getString("message");
              String date = rs.getString("date");
              String in_number = rs.getString("in_number");
              String incoming = rs.getString("incoming");
              area.append(id + "; ");
              area.append(message + "; ");
```

```
                area.append(date + "; ");
                area.append(in_number + "; ");
                 area.append(incoming + "; ");
                area.append("\n");

            }
            rs.close();
            stmt.close();
            c.close();

        } catch (Exception e) {
            e.printStackTrace();
            System.err.println(e.getClass().getName() + ": " + e.getMessage());
            System.exit(0);
        }
        GreatSearchPanel.repaint();

    }
});
```

Με την διαφορά ότι παίρνουμε τα arguments από τα combo boxes (

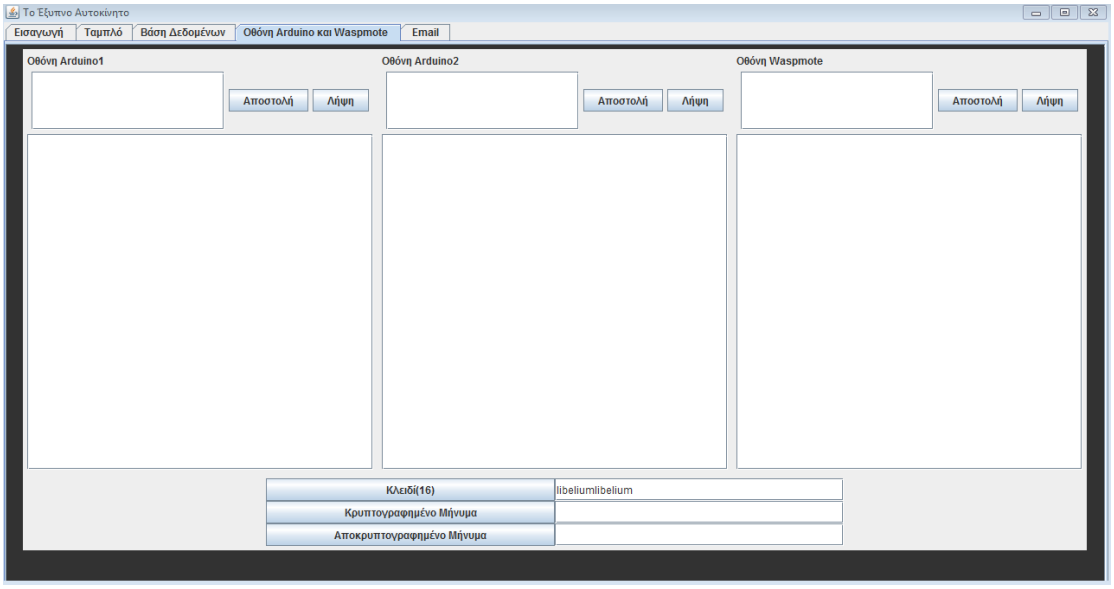
```
String arg0 = cmbox.getSelectedItem().toString();
        String arg1 = cmbox2.getSelectedItem().toString();)
```

Και τα περνάμε στο query και αργότερα περνούμε τα αποτελέσματα του και τα τοποθετούμε στο JTextArea της διεπαφής:

```
while (rs.next()) {
            String id = rs.getString("id");
            String message = rs.getString("message");
            String date = rs.getString("date");
            String in_number = rs.getString("in_number");
            String incoming = rs.getString("incoming");

            area.append(id + "; ");
            area.append(message + "; ");
            area.append(date + "; ");
            area.append(in_number + "; ");
             area.append(incoming + "; ");
            area.append("\n");

        }
```

Όσο αφορά το κουμπί "Εμφάνιση Όλων" εκτελείται παρόμοια διαδικασία με το κουμπί "Αναζήτηση" με την διαφορά ότι στο query εκτελείται η εντολή για την εμφάνιση όλων των στοιχείων του πίνακα ("SELECT * FROM mhnhma ;").

9.5 Οθόνη Arduino Και Waspmote



Το επόμενο πάνελ που βλέπει ο χρήστης είναι αυτό που περιέχει τα serial monitors των 2 Arduino και του Waspmote καθώς και το πεδίο των λειτουργιών κρυπτογράφησης και αποκρυπτογράφησης .

9.5.1 Serial Monitors

Αρχίζοντας από τα serial monitor βλέπουμε ότι χρησιμοποιούνται 2 κουμπιά για το καθένα καθώς και 2 TextArea. Με το κουμπί αποστολή γραφούμε δεδομένα στο κανάλι (serial port) και με το κουμπί λήψη περνούμε τα δεδομένα που έχουν γράψει στο κάθε κανάλι οι τρεις πλακέτες. Ο τρόπος που χρησιμοποιήσαμε για να επικοινωνήσει η εφαρμογή μας με τα serial port και επομένως με τις πλακέτες ήταν με την χρήση του JSSC api.
Έτσι όταν πατάμε το κουμπί αποστολή εκτελείται ο παρακάτω κώδικας:

```
send.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
           String value = tField.getText();
           try {
              serials.writePort(value);
           } catch (SerialPortException ex) {
              Logger.getLogger(SerialMonitors.class.getName()).log(Level.SEVERE, null, ex);
           } catch (UnsupportedEncodingException ex) {
              Logger.getLogger(SerialMonitors.class.getName()).log(Level.SEVERE, null, ex);
           }

        }
    });
```

Όπου παίρνουμε τα δεδομένα από το textArea ,και τα περνάμε στην μέθοδο writePort() της κλάσης SerialPortParser . η μέθοδος λειτουργεί ως εξής:

```
 public void writePort(String value) throws SerialPortException,
UnsupportedEncodingException {
      serialPort.openPort();
      serialPort.writeBytes(value.getBytes());
     serialPort.closePort();
  }
```

Ανοίγει το κανάλι , παίρνει τα byte των δεδομένων και τα περνάει στο κανάλι με την μέθοδο writeBytes() που ανήκει στις μεθόδους του JSSC, και εν συνεχεία κλείνει το κανάλι.

Για να λάβουμε τα δεδομένα που έχουν περαστεί στο κανάλι από της πλακέτες μας χρησιμοποιούμε την readPort() , παρακάτω φαίνεται η κλήση της:

```
 read.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
           try {

              tArea.append(serials.readPort());
           } catch (SerialPortException ex) {
              Logger.getLogger(SerialMonitors.class.getName()).log(Level.SEVERE, null, ex);
           } catch (UnsupportedEncodingException ex) {
              Logger.getLogger(SerialMonitors.class.getName()).log(Level.SEVERE, null, ex);
           }

        }
     });
```

Οπού επιστρέφει τα δεδομένα στο JTextArea. Τώρα η readPort() λειτουργεί ως εξής:

```
 public String readPort() throws SerialPortException, UnsupportedEncodingException {
     serialPort.openPort();

     byte[] buffer;
     buffer = serialPort.readBytes(50);
     String value = new String(buffer, "ISO-8859-7");
     System.out.println(value);
     serialPort.closePort();
     return value;
  }
```

Ανοίγει το κανάλι , δημιουργούμε έναν πίνακα από bytes , διαβάζουμε τα δεδομένα από το κανάλι με την μέθοδο του JSSC read Bytes(), τα μετατρέπουμε σε συμβολοσειρά με την ανάλογη κωδικοποίηση και τα επιστρέφουμε , αφότου κλείσουμε το κανάλι.

Παρόμοιες διαδικασίες γίνονται τόσο για το δεύτερο Arduino όσο και για το Waspmote.

9.5.2 Παράλληλη Χρήση Των 2 Serial Monitors Για Arduino 1 & 2

Μέχρι τώρα είδαμε την χρήση των serial monitor 1 & 2 μεμονωμένα .Τώρα θα εξηγήσουμε πως λειτουργούν και τα 2 μαζί ταυτόχρονα . Συγκεκριμένα αυτό που σκεφτήκαμε είναι ότι στο μέλλον κάθε όχημα θα διαθέτει έναν αισθητήρα αντίστοιχο με το hardware της πτυχιακής μας. Έτσι τα οχήματα θα έχουν την δυνατότητα να ανταλλάσσουν μεταξύ τους δεδομένα και πληροφορίες .

Συγκεκριμένα θα ανταλλάσσουν πληροφορίες σχετικά με τις συνθήκες περιβάλλοντος όπως φυσικά φαινόμενα που θα αποτελούσαν σοβαρό κίνδυνο για την ασφάλεια του οχήματος και των επιβαινόντων αντίστοιχα .

Περαιτέρω πληροφορίες για να επικοινωνούσαν μεταξύ τους θα μπορούσαν να είναι σχετικά με την κυκλοφοριακή συμφόρηση ή κάποιο ατύχημα ή κάποια έκτακτη διακοπή της κυκλοφορίας.

Καθώς επίσης και για να ειδοποιηθούν τα κοντινά οχήματα σχετικά με όποια βοήθεια μπορούν να προσφέρουν για οποιαδήποτε έκτακτη ανάγκη έχει ένα συγκεκριμένο όχημα .

Επιπλέον σε περίπτωση ατυχήματος θα μπορούν να ειδοποιούνται τα οχήματα και οι υπηρεσίες εκτάκτου ανάγκης για να παράσχουν τις πρώτες βοήθειες . Για αυτό τον λόγο υπάρχει μια ειδική AT COMMAND η AT+CEMNL η όποια χρησιμοποιείται για να προσθέσουμε στο σύστημα μας τα νούμερα εκτάκτου ανάγκης (πχ 911 για ΗΠΑ , 112 για ΕΕ κτλ…).

Παρόλα αυτά δεν μπορέσαμε να το υλοποιήσουμε λόγο έλλειψης κυριών εξαρτημάτων του εξοπλισμού όπως μικρόφωνο ηχείο κτλ ώστε να είναι σαν ένα πλήρες σύστημα λειτουργικού κινητού τηλέφωνου. Αλλά και να είχαμε την δυνατότητα πραγματοποίησης κλήσης με το σύστημα πλήρες δεν θα θέλαμε να πραγματοποιούσαμε κάποια κλήση προς τους αριθμούς εκτάκτου ανάγκης γιατί είναι πραγματικά για τις "**πολύ** έκτακτες ανάγκες".

9.5.3 Πεδίο Κρυπτογράφησης-Αποκρυπτογράφησης

Στο πεδίο αυτό μπορούμε να ορίσουμε το κλειδί για την διαδικασία κρυπτογράφησης-αποκρυπτογράφησης το οποίο πρέπει να είναι υποχρεωτικά 16 byte (16 char) .
Πατώντας το κουμπί "Κρυπτογραφημένο Μήνυμα" παίρνουμε τα δεδομένα από textField και τα κρυπτογραφούμε συμφώνα με το κλειδί . Ακολουθεί ο ActionListener που καλεί την διαδικασία της κρυπτογράφησης(με αρκετά exceptions):

```
encrtyptB.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
           EncryptDecrypt encdecrpt = new EncryptDecrypt();

           String key = kleidhT.getText();
```

```
            String plain = encryptT.getText();
            try {
               String girise = encdecrpt.aesEncryption(plain, key);
               encryptT.setText(girise);
            } catch (InvalidKeyException ex) {
               Logger.getLogger(SerialMonitors.class.getName()).log(Level.SEVERE, null, ex);
            } catch (IllegalBlockSizeException ex) {
               Logger.getLogger(SerialMonitors.class.getName()).log(Level.SEVERE, null, ex);
            } catch (BadPaddingException ex) {
               Logger.getLogger(SerialMonitors.class.getName()).log(Level.SEVERE, null, ex);
            } catch (UnsupportedEncodingException ex) {
               Logger.getLogger(SerialMonitors.class.getName()).log(Level.SEVERE, null, ex);
            } catch (InvalidAlgorithmParameterException ex) {
               Logger.getLogger(SerialMonitors.class.getName()).log(Level.SEVERE, null, ex);
            } catch (NoSuchProviderException ex) {
               Logger.getLogger(SerialMonitors.class.getName()).log(Level.SEVERE, null, ex);
            }

         }
      });
```

Από ότι φαίνεται παραπάνω , παίρνουμε τα δεδομένα από τα πεδία των κλειδιών και του μηνύματος που θα κρυπτογραφήσουμε τα στέλνουμε στην aesEncryption και το αποτέλεσμα που επιστρέφει αντικαθιστά το προηγούμενο κείμενο που είχαμε στο πεδίο.

Ακολουθεί η aesEncryption που έχουμε αναφέρει σε προηγούμενο κεφάλαιο

```
  public String aesEncryption(String plain, String kleidh) throws InvalidKeyException,
         IllegalBlockSizeException, BadPaddingException,
         UnsupportedEncodingException,
         InvalidAlgorithmParameterException, NoSuchProviderException {
      byte[] keyRaw = kleidh.getBytes("ISO-8859-1");
      try {
         Cipher chiper = Cipher.getInstance("AES/ECB/PKCS5Padding");
         SecretKeySpec skeySpec = new SecretKeySpec(keyRaw, "AES");
         chiper.init(Cipher.ENCRYPT_MODE, skeySpec);
         byte[] encVal = chiper.doFinal(plain.getBytes("ISO-8859-1"));
         String str = new String(encVal, "ISO-8859-1");
        String encryptedValue = new BASE64Encoder().encode(str.getBytes("ISO-8859-1"));
         return encryptedValue;

      } catch (NoSuchAlgorithmException | NoSuchPaddingException ex) {
         Logger.getLogger(ArduinoGui.class.getName()).log(Level.SEVERE, null, ex);
      }
      return null;
   }
```

Να αναφέρουμε ότι χρησιμοποιήσαμε BASE64Encoding και αυτό για να μπορέσουμε να περάσουμε το cipher text σε μηνύματα sms μιας και χωρίς αυτό είχαμε χαρακτήρες που δεν

μπορούσαν να αναγνωριστούν στα sms κάνοντας την διαδικασία αποκρυπτογράφησης αδύνατη.

Πατώντας το κουμπί "Αποκρυπτογραφημένο Μήνυμα" εκτελείτε η ανάστροφη διαδικασία Έτσι όταν πατάμε το κουμπί εκτελείται ο ακόλουθος ActionListener :

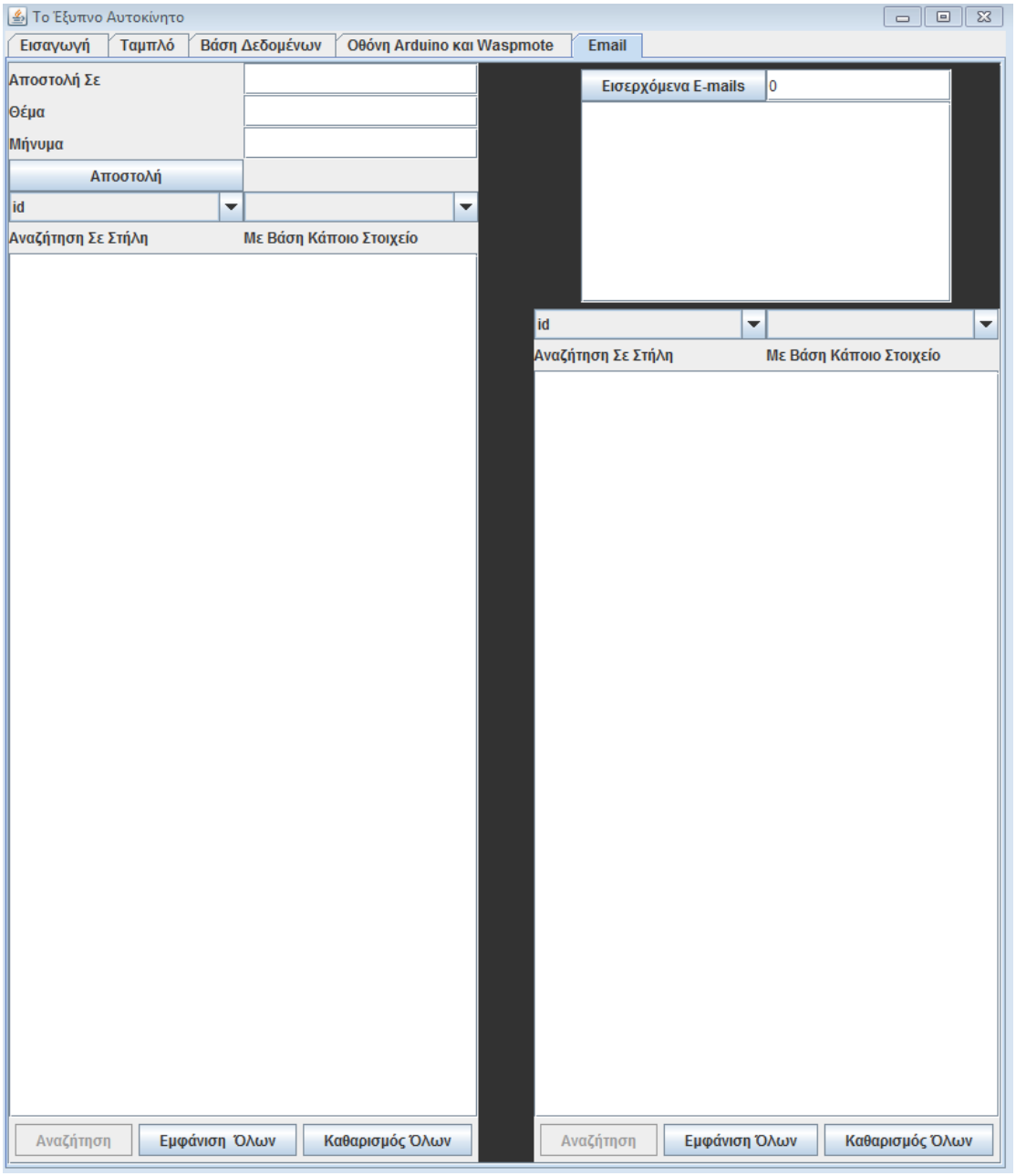
```
decryptB.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent ae) {
        EncryptDecrypt encdecrpt = new EncryptDecrypt();
        String key = kleidhT.getText();
        String cipher = decryptT.getText();
        try {
            String girise = encdecrpt.aesDecryption(cipher, key);
            decryptT.setText(girise);
        } catch (Exception ex) {
            Logger.getLogger(SerialMonitors.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
});
```

Ο οποίος παίρνει τα δεδομένα από το textField και τα περνά στην aesDecryption και αυτό που γυρίζει αντικαθιστά το κείμενο που είχαμε στο προηγούμενο πεδίο. Ακολουθεί ο κώδικας της aesDecryption που έχουμε περιγράψει σε προηγούμενο κεφάλαιο :

```
public static String aesDecryption(String encryptedData, String kleidh)
        throws Exception {
    byte[] keyRaw = kleidh.getBytes("ISO-8859-1");
    byte[] imagebyte = new BASE64Decoder().decodeBuffer(encryptedData);
    SecretKeySpec skeySpec = new SecretKeySpec(keyRaw, "AES");
    Cipher c = Cipher.getInstance("AES/ECB/PKCS5Padding");
    c.init(Cipher.DECRYPT_MODE, skeySpec);
    byte[] decValue = c.doFinal(imagebyte);
    String decryptedValue = new String(decValue, "ISO-8859-1");
    return decryptedValue;
}
```

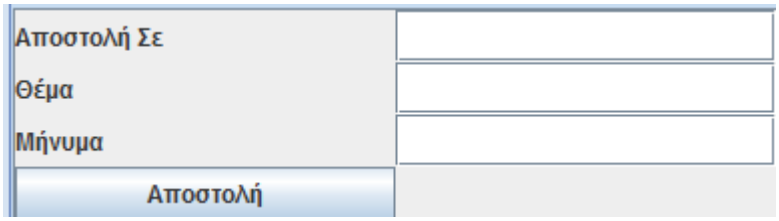
Και εδώ χρησιμοποιούμε BASE64Decoder για τον ίδιο λόγο που αναφέραμε προηγούμενος

9.6 Email



Στο τελευταίο πάνελ της εφαρμογής χρησιμοποιείται για την αποστολή και λήψη μηνυμάτων ηλεκτρονικού ταχυδρομείου καθώς και αποθήκευση τους στην βάση δεδομένων.

9.6.1 Αποστολή Email



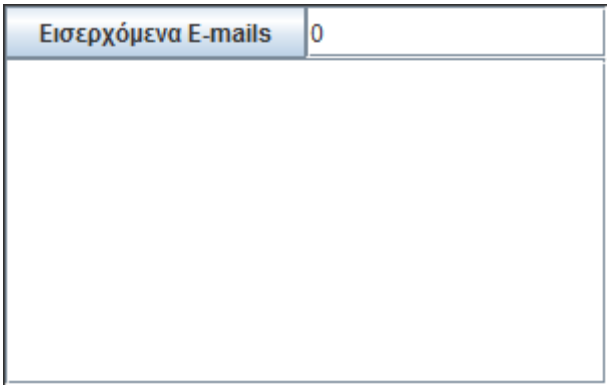
Για να στείλουμε ένα μήνυμα βάζουμε την διεύθυνση του παραλήπτη , το θέμα και το μήνυμα και πατάμε το κουμπί Αποστολή. Ταυτόχρονα τα δεδομένα αποθηκεύονται στην βάση δεδομένων. Παρακάτω ακολουθεί ο ActionListener:

```
send.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {

            Properties props = new Properties();
      props.put("mail.smtp.starttls.enable", "true");
      props.put("mail.smtp.auth", "true");
      props.put("mail.smtp.host", "smtp.gmail.com");
      props.put("mail.smtp.port", "587");
            Session session = Session.getInstance(props,
       new javax.mail.Authenticator() {
        protected PasswordAuthentication getPasswordAuthentication() {
            return new PasswordAuthentication(username, password);
        }
       });

      try {

        Message message = new MimeMessage(session);
        message.setFrom(new InternetAddress("ptyxiakh31523224@gmail.com"));
        message.setRecipients(Message.RecipientType.TO,
            InternetAddress.parse(sendToText.getText()));
        message.setSubject(subjectText.getText());
        message.setText(messageText.getText()
            + "\n\n No spam to my email, please!");

        Transport.send(message);

        System.out.println("Done");

      } catch (MessagingException e) {
        throw new RuntimeException(e);
      }


       Connection c = null;
            Statement stmt = null;
            String sendTo = String.valueOf(sendToText.getText());
            String subject = String.valueOf(subjectText.getText());
            String message = String.valueOf(messageText.getText());


            try {
               Class.forName("org.postgresql.Driver");
```

```
            c = DriverManager
                  .getConnection("jdbc:postgresql://localhost:5432/ptyxiakhDB",
                  "postgres", "1111");
            String qry = "'" + sendTo + "','" + subject + "','" + message + "'";
            stmt = c.createStatement();
            String sql = "INSERT INTO email VALUES(default, " + qry + ");";
            stmt.executeUpdate(sql);

         } catch (Exception e) {
            e.printStackTrace();
            System.err.println(e.getClass().getName() + ": " + e.getMessage());
            System.exit(0);
         }

      }
   });
```

Από ότι βλέπουμε παραπάνω παίρνουμε τα δεδομένα από τα Textfields και τα περνάμε στους παραμέτρους χρησιμοποιώντας τις μεθόδους του java mail api που έχουμε περιγράψει σε προηγούμενο κεφάλαιο , και εν συνεχεία τα περνάμε και στην βάση δεδομένων με παρόμοια διαδικασία που χρησιμοποιήσαμε και στην βάση για τα μηνύματα SMS .

9.6.2 Προβολή Εισερχόμενων Μηνυμάτων



Στην εφαρμογή μας παρατηρούμε ένα κουμπί το "Εισερχόμενα E-mails" ένα textField και ένα textArea . Πατώντας το κουμπί μας δείχνει το εισερχόμενο μήνυμα ανάλογα με τον αριθμό που έχει μέσα το textField , έτσι με το 0 περνούμε το τελευταίο , με το 1 το προ τελευταίο και ου το κάθε εξής , και μας εμφανίζει τα περιεχόμενα στο textArea από κάτω .παράλληλα τοποθετεί τα στοιχειά του κάθε μηνύματος μας στην βάση δεδομένων με παρόμοιο τρόπο που γινόταν και στα μηνύματα sms .ακλουθεί ο actionListener του κουμπιού:

```
getEmailN.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {

            Properties props = new Properties();
```

```
props.setProperty("mail.store.protocol", "imaps");
try {
   Session session = Session.getInstance(props, null);
   Store store = session.getStore();
   store.connect("imap.gmail.com", "ptyxiakh31523224@gmail.com", "pty31523224");
   Folder inbox = store.getFolder("INBOX");
   inbox.open(Folder.READ_ONLY);


     String s0= null;

 int i=   Integer.parseInt(mailNumber.getText());
  Message msg = inbox.getMessage(inbox.getMessageCount()-i);
   Address[] in = msg.getFrom();
   for (Address address : in) {
      s0=address.toString();
      System.out.println("FROM:" + address.toString());
   }
   Multipart  mp = (Multipart) msg.getContent();
  BodyPart bp = mp.getBodyPart(0);

  String s1=msg.getSentDate().toString();

  String s2= msg.getSubject().toString();

 String s3= bp.getContent().toString();

   emailArea.append("FROM:" + s0+ "\n");
    emailArea.append("SUBJECT:" + s2+ "\n");
    emailArea.append("CONTENT:" + s3+ "\n");

    Connection c = null;
      Statement stmt = null;
      try {
         Class.forName("org.postgresql.Driver");
         c = DriverManager
               .getConnection("jdbc:postgresql://localhost:5432/ptyxiakhDB",
               "postgres", "1111");
         String qry = "'" + s0 + "','" + s2 + "','" + s3 + "'";
         stmt = c.createStatement();
         String sql = "INSERT INTO emailin VALUES(default, " + qry + ");";
         stmt.executeUpdate(sql);

      } catch (Exception e) {
         e.printStackTrace();
         System.err.println(e.getClass().getName() + ": " + e.getMessage());
         System.exit(0);
```

```
            }

            contain.revalidate();
            contain.repaint();
        } catch (Exception mex) {
            mex.printStackTrace();
        }

    }

        });
```

Οι τεχνικές λεπτομέρειες του κώδικα υπάρχουν στο κεφάλαιο java mail api.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://www.cooking-hacks.com/documentation/tutorials/3g-gps-shield-arduino-raspberry-pi-tutorial/

[2] http://m2msupport.net/m2msupport/software-and-at-commands-for-m2m-modules/

[3] http://harmoniccode.blogspot.de/2011/06/steelseries-393.html

[4] https://www.cooking-hacks.com/forum/

[5] https://www.libelium.com/forum/

[6]http://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html#Introduction

[7] https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

[8]https://en.wikibooks.org/wiki/Serial_Programming/Serial_Java

[9] https://blogs.oracle.com/jtc/entry/java_serial_communications_revisited

[10]http://www.javaprogrammingforums.com/java-se-api-tutorials/5603-jssc-library-easy-work-serial-ports.html

[11] https://code.google.com/p/java-simple-serial-connector/

[12] http://www.oracle.com/technetwork/java/javamail/index.html

[13] http://www.toptechboy.com/arduino/lesson-12-simple-and-easy-way-to-read-strings-ints-and-floats-over-arduino-serial-port/

[14] http://www.codejava.net/coding/how-to-play-back-audio-in-java-with-examples

[15] http://forum.arduino.cc/index.php?topic=226727.0

[16] https://en.m.wikipedia.org/wiki/Hayes_command_set

[17] https://github.com/jantje/arduino-eclipse-plugin/issues/67

[18] https://groups.google.com/a/arduino.cc/forum/#!topic/developers/CE5LOqIX12o

[19] http://www.developershome.com/

[20] https://blogs.oracle.com/jtc/entry/java_serial_communications_revisited