

## **Source code report**

Τσούσης Παναγιώτης  
AEM: 9590  
pitsousis@ece.auth.gr

### **1. Σημειώσεις:**

Ο κώδικας έχει αναπτυχθεί με θεμέλιο τον κώδικα που δώθηκε στην εκφώνηση της εργασίας, στη σελίδα 24. Αυτό περιλαμβάνει το setup των αντικειμένων DatagramSocket και DatagramPacket, την αποστολή ενός request προς το σέρβερ Ιθάκη, την λήψη των πακέτων που αποστέλονται προς το χρήστη και το setup για την αναπαραγωγή ήχου.

Ο κώδικας δε θα αναλυθεί γραμμή προς γραμμή, αλλά από μια πιο γενική σκοπιά, ώστε να γίνει κατανοητό πως λειτουργεί, χωρίς όμως να κουράσει (όχι πολύ, τουλάχιστον) τον αναγνώστη.

### **Συνάρτηση main():**

Στις αρχικές σειρές της `main` ορίζουμε διάφορες μεταβλητές, όπως τα `ports` και τους βασικούς κωδικούς που δίνονται από το `server`, αλλά και μερικές επιπλέον ώστε να καθορίσουμε τις παραμέτρους μας για τα διάφορα ζητούμενα και να δημιουργήσουμε τους τελικούς κωδικούς που θα στείλουμε. Τέτοιες μεταβλητές είναι, για παράδειγμα, το `echo_run_time`, που καθορίζει για πόσα λεπτά θα λαμβάνουμε πακέτα `echo` από το `server`, καθώς και η `mode` που καθορίζει πόσα πακέτα ήχου θα λάβουμε.

Ο χρήστης θα πρέπει να βάλει τους σωστούς κωδικούς και `ports` όπως δίνονται από το `server`, καθώς και να αλλάξει τις παραμέτρους σε αυτά που θέλει, ώστε να λειτουργήσει σωστά η εφαρμογή.

Στις επόμενες γραμμές, καλούμε τις διάφορες συναρτήσεις που θέλουμε να τρέξουν. Τα ονόματά τους είναι αρκετά περιγραφικά, ώστε να μη χρειάζεται περαιτέρω εξήγηση το τι ακριβώς κάνει η κάθε μια, αν και η εξήγηση θα δοθεί παρακάτω. Το μόνο που θα σημειωθεί είναι ότι οι συναρτήσεις `start_main_graphics()` και `end_main_graphics()` δεν κάνουν τίποτα πέρα από την εκτύπωση μερικών `string` στην κονσόλα στην αρχή και στο τέλος της `main`, με σκοπό να κάνουν την εργασία πιο εμφανίσιμη, αλλά και να βοηθήσουν με ένα πιο διαισθητικό τρόπο τον προγραμματιστή να διαπιστώσει τη σωστή λειτουργία του προγράμματός του.

### **Συνάρτηση `stringToFile(String name, String input)`:**

Σκοπός της συνάρτησης αυτής είναι η διευκόλυνση του `output` αρχείων `txt`.

Ο χρήστης ορίζει το όνομα που θέλει να δώσει στα αρχεία καθώς και τη μεταβλητή `String` που περιέχει τα δεδομένα που θέλει να κάνει `output` και η συνάρτηση αυτή φροντίζει να εκτελέσει τη διαδικασία της εξαγωγής των δεδομένων σε αρχείο `txt`.

### **Συνάρτηση `bytesToInt(byte h, byte l)`:**

Η συνάρτηση αυτή απλά παίρνει το ‘high’ (msb) byte και το ‘low’ (lsb) ενός δεδομένου που έχουμε ‘σπάσει’ σε 2 bytes και ανασυνθέτει τον ακέραιο αριθμό που είχαμε ‘σπάσει’.

### **Συνάρτηση get\_echo(String requestCode,int sPort,int cPort,int time\_mins):**

Σκοπός της συνάρτησης, είναι να πάρει λάβει τα πακέτα echo από το server, να κάνει την επεξεργασία που επιθυμούμε και να βγάλει τα αντίστοιχα output αρχεία.

Παίρνει ως ορίσματα το request code για τα echo (with/without delay), το server listening port, το client listening port και το χρόνο για τον οποίο θα λαμβάνουμε πακέτα.

Αρχικά (σειρά 62) , ξεκινάει τα αντικείμενα DatagramSocket και DatagramPacket με τις σωστές παραμέτρους και στη συνέχεια ορίζει τις μεταβλητές που θα χρειαστούν για να τρέξει η συνάρτηση σωστά. Οι μεταβλητές δε θα σχολιαστούν, υπάρχουν, όμως, σχόλια στον κώδικα που τις εξηγούν σχετικά σύντομα.

Έπειτα (σειρά 111), τρέχει το loop στο οποίο γίνεται η λήψη πακέτων, ελέγχοντας σε κάθε loop το χρόνο που έχει περάσει, ώστε να τρέξει για το χρόνο που ζητήσαμε. Εδώ, είναι απαραίτητο να σχολιαστεί πως τρέχουμε για όσα λεπτά μας έδωσε ο χρήστης, αλλά προσθέτουμε και τα timeouts που έχουμε. Αυτό γίνεται για να λαμβάνει ο χρήστης αριθμό πακέτων ανάλογα με την ταχύτητα του server και όχι ανάλογα με το πόσο καλή είναι η σύνδεσή του, καθώς τα UDP πακέτα τείνουν να ‘χάνονται’ αρκετά συχνά στο ‘δρόμο’, μετά από παρατηρήσεις. Ταυτόχρονα, μέσα στο loop γίνεται και η αποθήκευση του χρόνου απόκρισης του server για κάθε πακέτο σε ένα arraylist.

Μετά τη λήψη (σειρά 137), ακολουθούν οι υπολογισμοί για τη ρυθμαπόδοση τα τελευταία 32 δευτερόλεπτα (για κάθε δευτερόλεπτο μετά τα 32sec λήψης). Αυτό γίνεται με βάση τα αποθηκευμένα response time με μια μέθοδο κινητών δεικτών πάνω στο arraylist.

Μετά (σειρά 179), υπολογίζουμε την πιθανότητα εμφάνισης κάθε τιμής response time, βάσει των δεδομένων που είχαμε συλλέξει. Αυτό γίνεται κάνοντας sort το αρχικό μας arraylist και μετρώντας πόσες φορές υπάρχει το κάθε στοιχείο.

Ακολουθούν (σειρά 202) οι υπολογισμοί των σηναρτήσεων SRTT, RTO και SRTTVAR, με μια απλή προγραμματιστική υλοποίηση αυτών που δίνονται από τις σημειώσεις του μαθήματος στη σελίδα 60, αλλά και από τη σελίδα <https://tools.ietf.org/html/rfc2988> .

Έπειτα (σειρά 235), γίνεται ο υπολογισμός της μέσης τιμής και της διασποράς των response time των πακέτων που λάβαμε.

Τέλος (σειρά 264), μέσω της συνάρτησης stringToTextFile(), γίνεται output αρχείων txt με όλα τα δεδομένα που λάβαμε και υπολογίσαμε. Η διάκριση μεταξύ των κωδικών EXXXX και E0000 γίνεται ώστε να δώσουμε διαφορετικό όνομα στα αρχεία, αν πρόκειται για πακέτα με καθυστέρηση απο το server ή χωρίς.

### **Συνάρτηση get\_Temp(String requestCode,int sPort,int cPort):**

Η συνάρτηση αυτή έχει ως σκοπό να πάρει τη θερμοκρασία απο το σταθμό T00 (καθώς οι άλλοι δε λειτουργούν) και να βγάλει ως output το αρχείο με τη θερμοκρασία του.

Παίρνει ως ορίσματα το request code για τα θερμοκρασία, το server listening port και το client listening port.

Αρχικά (σειρά 292) , ξεκινάει τα αντικείμενα DatagramSocket και DatagramPacket με τις σωστές παραμέτρους.

Έπειτα (σειρές 306-307), γίνεται η αποστολή του request και η λήψη του.

Τέλος (σειρά 310), γίνεται το output των δεδομένων σε αρχείο txt.

### **Αντικείμενο MusicSamples:**

Το αντικείμενο αυτό έχει ως μοναδικό σκοπό τη διευκόλυνση της επιστροφής των πινάκων των δειγμάτων και διαφορών ήχου και της μέσης τιμής και βήματος από τις συναρτήσεις DPCM\_decoder και AQ\_DPCM\_decoder στις συναρτήσεις get\_dpcm\_audio και get\_aq\_dpcm\_audio. Με άλλα λόγια, ο σκοπός του είναι μόνο οργανωτικός και δεν έχει κάποια άλλη πρακτική αξία.

### **Συνάρτηση get\_dpcm\_audio(String requestCode,int sPort,int cPort,int sound\_packets):**

Η συνάρτηση αυτή έχει ως σκοπό τη λήψη των πακέτων, το output των δειγμάτων και των διαφορών ήχου, καθώς και την αναπαραγωγή του ήχου που λάβαμε.

Παίρνει ως ορίσματα το request code για το DPCM audio, το server listening port, το client listening port και τον αριθμό των πακέτων που θέλουμε να λάβουμε.

Αρχικά (σειρά 324) , ξεκινάει τα αντικείμενα DatagramSocket και DatagramPacket με τις σωστές παραμέτρους και στη συνέχεια ορίζει τις μεταβλητές που θα χρειαστούν για να τρέξει η συνάρτηση σωστά.

Μετά (σειρά 350), στέλνουμε το αίτημα του DPCM audio στο server και ξεκινάμε το loop για τη λήψη των πακέτων. Κάθε πακέτο χρειάζεται αποκωδικοποίηση, το οποίο γίνεται μέσω της συνάρτησης DPCM\_decoder, η οποία επιστρέφει ένα αντικείμενο MusicSamples, που μας επιτρέπει να έχουμε πρόσβαση στους αποκωδικοποιημένους πίνακες. Τα δείγματα μπαίνουν στη σωστή θέση στον audio buffer και τροποποιούνται οι μεταβλητές που θα χρειαστεί να κάνουμε output.

Στη συνέχεια (σειρές 376 – 377), γίνεται το output των αρχείων txt που περιέχουν τα δεδομένα μας.

Τέλος (σειρές 385-391), γίνεται η αναπαραγωγή του ήχου που λάβαμε.

### **Συνάρτηση DPCM\_decoder(byte[] dpcm\_bytes, int mean, int step):**

Η συνάρτηση αυτή έχει ως σκοπό την αποκωδικοποίηση ενός πακέτου που είναι κωδικοποιημένο κατά DPCM και την επιστροφή της αποκωδικοποίησης του μέσω του αντικειμένου MusicSamples.

Αρχικά (σειρά 396), ορίζουμε τις μεταβλητές που θα χρειασούμε:

- `int mean` → μέσος όρος τιμών δειγμάτων
- `int step` → βήμα
- `byte[] buffer_out` → πίνακας που θα περιέχει τις τελικές τιμές των δειγμάτων μετά την αποκωδικοποίηση
- `int[] deltas` → πίνακας που θα περιέχει τις τιμές των διαφορών
- `int MSnib` → Most significant Nibble ενός byte
- `int LSnib` → Least significant Nibble ενός byte
- `int current_byte` → το byte που επεξεργαζόμαστε στο loop
- `int index` → δείκτης που δείχνει που να βάλουμε το κάθε στοιχείο της διαφοράς

Έπειτα (σειρά 408), στο loop γίνεται η επεξεργασία του κάθε byte ώστε να γεμίσουμε τον πίνακα των διαφορών με τις διαφορές των δειγμάτων. Η επεξεργασία γίνεται με bitwise operators και σύμφωνα με τη μαθηματική περιγραφή της εκφώνησης.

Μετά (σειρά 422), εφόσον γνωρίζουμε ότι ο μέσος όρος των δειγμάτων είναι 0, με μαθηματικά μπορούμε να υπολογίζουμε και το πρώτο δείγμα. *(Βέβαια, η προσέγγιση είναι θεωρητική και δεν βρέθηκε σε βιβλίο, εκφώνηση ή στο διαδίκτυο. Η εξήγηση είναι περιττή για κάτι τόσο μικρό και κάπως μακροσκελής για να μπει στην εργασία, αλλά σε περίπτωση που απαιτηθεί, θα δωθεί.)*

Στη συνέχεια (σειρά 433), με ένα loop κάνουμε αποκωδικοποίηση των διαφορών σε δείγματα και τα βάζουμε στον πίνακα `buffer_out`.

Τέλος (σειρά 437), δημιουργούμε το αντικείμενο MusicSamples με την κατάλληλη αρχικοποίηση, ώστε να μπορέσει να ‘μεταφέρει’ τις διαφορές και τα δείγματα εκτός της συνάρτησης και το επιστρέφουμε σαν αποτέλεσμα της συνάρτησης.

### **Συνάρτηση `get_aq_dpcm_audio(String requestCode,int sPort,int cPort,int sound_packets)`:**

Η συνάρτηση αυτή έχει ως σκοπό τη λήψη των πακέτων, το output των δειγμάτων και των διαφορών ήχου, καθώς και την αναπαραγωγή του ήχου που λάβαμε.

Παίρνει ως ορίσματα το request code για το AQ\_DPCM audio, το server listening port, το client listening port και τον αριθμό των πακέτων που θέλουμε να λάβουμε.

Αρχικά (σειρά 445) , ξεκινάει τα αντικείμενα DatagramSocket και DatagramPacket με τις σωστές παραμέτρους και στη συνέχεια ορίζει τις μεταβλητές που θα χρειαστούν για να τρέξει η συνάρτηση σωστά.

Μετά (σειρά 475), στέλνουμε το αίτημα του AQ-DPCM audio στο server και ξεκινάμε το loop για τη λήψη των πακέτων. Κάθε πακέτο χρειάζεται αποκωδικοποίηση, το οποίο γίνεται μέσω της συνάρτησης AQ\_DPCM\_decoder, η οποία επιστρέφει ένα αντικείμενο MusicSamples, που μας επιτρέπει να έχουμε πρόσβαση στους αποκωδικοποιημένους πίνακες και μεταβλητές. Τα δείγματα μπαίνουν στη σωστή θέση στον audio buffer και τροποποιούνται οι μεταβλητές που θα χρειαστεί να κάνουμε output.

Στη συνέχεια (σειρες 506-509), γίνεται το output των αρχείων txt που περιέχουν τα δεδομένα μας.

Τέλος (σειρές 517-523), γίνεται η αναπαραγωγή του ήχου που λάβαμε.

### **Συνάρτηση `AQ_DPCM_decoder(byte[] dpcm_bytes, int mean, int step)`:**

Η συνάρτηση αυτή έχει ως σκοπό την αποκωδικοποίηση ενός πακέτου που είναι κωδικοποιημένο κατά DPCM και την επιστροφή της αποκωδικοποίησης του μέσω του αντικειμένου MusicSamples.

Αρχικά (σειρά 528), ορίζουμε τις μεταβλητές που θα χρειασούμε:

- `int mean` → μέσος όρος τιμών δειγμάτων
- `int step` → βήμα

- `byte[] sound_bytes` → περιέχει τα bytes από το πακέτο που αφορούν αποκλειστικά τις διαφορές των δειγμάτων
- `byte[] buffer_out` → πίνακας που θα περιέχει τις τελικές τιμές των δειγμάτων μετά την αποκωδικοποίηση
- `byte[] delta` → πίνακας που θα περιέχει τις τιμές των διαφορών
- `int MSnib` → Most significant Nibble ενός byte
- `int LSnib` → Least significant Nibble ενός byte
- `int current_byte` → το byte που επεξεργαζόμαστε στο loop

Έπειτα (σειρά 544), στο loop γίνεται η επεξεργασία του κάθε byte ώστε να γεμίσουμε τον πίνακα των διαφορών με τις διαφορές των δειγμάτων. Η επεξεργασία γίνεται με bitwise operators και σύμφωνα με τη μαθηματική περιγραφή της εκφώνησης. Σημειώνεται ότι πλέον η κάθε διαφορά που προκύπτει είναι μεγαλύτερη από ένα byte, για αυτό και χρειάζεται 2 διαδοχικές θέσεις για να αποθηκευτεί σε έναν πίνακα τύπου byte.

Στη συνέχεια (σειρά 561), με ένα loop κάνουμε αποκωδικοποίηση των διαφορών σε δείγματα, τα οποία επίσης έχουν μέγεθος 2 byte και έτσι πρέπει να τα ‘σπάσουμε’ και να τα βάλουμε σε 2 διαδοχικές θέσεις στον πίνακα `buffer_out`.

Τέλος (σειρά 571), δημιουργούμε το αντικείμενο `MusicSamples` με την κατάλληλη αρχικοποίηση, ώστε να μπορέσει να ‘μεταφέρει’ τις διαφορές, τα δείγματα, τη μέση τιμή και το βήμα εκτός της συνάρτησης και το επιστρέφουμε σαν αποτέλεσμα της συνάρτησης.

### **Συνάρτηση `get_image(String requestCode,int sPort,int cPort):\`**

Σκοπός της συνάρτησης είναι να κάνει output μια εικόνα jpeg από την κάμερα με τον αντίστοιχο κωδικό.

Παίρνει ως ορίσματα το request code για τα θερμοκρασία, το server listening port και το client listening port.

Αρχικά (σειρά 583) , ξεκινάει τα αντικείμενα `DatagramSocket` και `DatagramPacket` με τις σωστές παραμέτρους.



Στη συνέχεια (σειρά 587), ορίζει ένα αντικείμενο OutputStream για την εξαγωγή της εικόνας.

Μετά (σειρά 608), στέλνει το request για την εικόνα που θέλουμε (το οποίο διαφοροποιείται ανάλογα με την κάμερα από την οποία θέλουμε να κάνουμε λήψη) και στη συνέχεια (σειρά 615), με ένα while loop ‘ταίζει’ στο OutputStream τα δεδομένα που μας έρχονται, μέχρι να πάψουν να έρχονται δεδομένα. Έτσι γίνεται η εξαγωγή της εικόνας.

### **Συνάρτηση get\_ithakicopter\_telemetry():**

Η συνάρτηση αυτή έχει ως σκοπό τη λήψη των πακέτων από την εφαρμογή του Ithakicopter.

Επειδή το client listening port είναι fixed και δε χρειάζεται να στείλουμε κωδικό, δε χρειάζονται ορίσματα.

Αρχικά (σειρά 637), ξεκινάει τα αντικείμενα DatagramSocket και DatagramPacket με τις σωστές παραμέτρους.

Στη συνέχεια, ορίζει τις μεταβλητές που θα χρειαστούμε για να κάνουμε λήψη και επεξεργασία των πακέτων.

Έπειτα (σειρά 649), έχουμε το loop λήψης των πακέτων για 200 iterations, μέσα στο οποίο γίνεται και η επεξεργασία τους για το output.

Τέλος (σειρά 672), γίνεται το output του αρχείου txt με τα δεδομένα που πήραμε.

### **Συνάρτηση get\_OBD(String requestCode,int sPort,int cPort,int time mins):**

Η συνάρτηση αυτή έχει ως σκοπό τη λήψη και το output δεδομένων σχετικά με τη λειτουργία του οχήματος OBD.

Παίρνει ως ορίσματα το request code για το OBD-II, το server listening port, το client listening port και το χρόνο για τον οποίο θα λαμβάνουμε πακέτα.

Αρχικά (σειρά 677) , ξεκινάει τα αντικείμενα DatagramSocket και DatagramPacket με τις σωστές παραμέτρους και στη συνέχεια ορίζει τις μεταβλητές που θα χρειαστούν για να τρέξει η συνάρτηση σωστά. Οι μεταβλητές δε θα σχολιαστούν, διότι τα ονόματά τους είναι αρκετά επεξηγηματικά από μόνα τους.

Έπειτα (σειρά 708), τρέχει το loop μέσα στο οποίο γίνεται η αποστολή του κάθε ένα κωδικού που χρειάζεται για τη λήψη της κάθε μεταβλητής. Μετά τη λήψη μίας παραμέτρου, γίνεται η σωστή επεξεργασία για να λάβουμε τη χρήσιμη, για εμάς πληροφορία, σύμφωνα με τους τύπους που δώθηκαν στην εκφώνηση.

Τέλος (σειρά 789), γίνεται το output του αρχείου txt με τα δεδομένα μας.

### **Συνάρτηση sleep(int ms):**

Η συνάρτηση αυτή κάνει ‘pause’ το πρόγραμμά μας για τη χρονική διάρκεια που ορίσαμε. Κύριος σκοπός της ήταν να βοηθήσει στο debugging του προγράμματος.

### **Συναρτήσεις start/end xxxx\_graphics():**

Βρίσκονται στο τέλος του προγράμματος και δεν έχουν λειτουργικό σκοπό, είναι μόνο για να γίνει ομορφότερο το πρόγραμμα.