Εαρινό
Εξάμηνο

ΕΡΓΑΣΙΑ ΣΤΑ ΠΛΑΙΣΙΑ ΤΟΥ ΜΑΘΗΜΑΤΟΣ:
ΑΣΥΡΜΑΤΑ ΔΙΚΤΥΑ ΑΙΣΘΗΤΗΡΩΝ

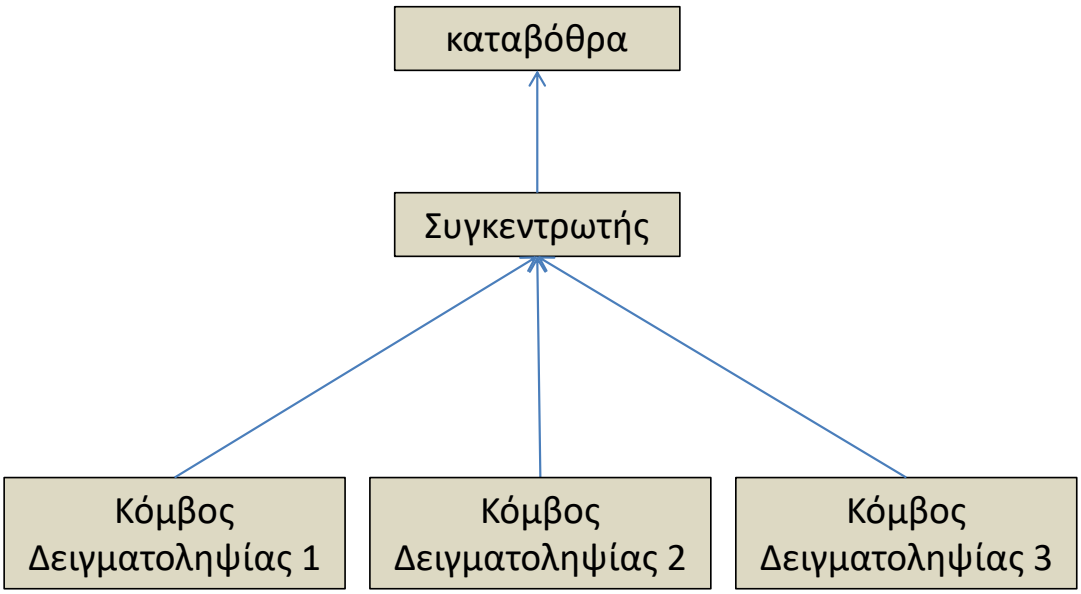
1115201300177 – ΤΟΥΜΑΣΗΣ ΑΓΓΕΛΟΣ

1837 2017 ΧΡΟΝΙΑ
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

Περιεχόμενα

ΕΙΣΑΓΩΓΗ

Στη συγκεκριμένη εργασία υλοποιήθηκε η παρακάτω ζητούμενη αρχιτεκτονική, η οποία αναπτύχθηκε στον προσομοιωτή Solarium.



Το παραδοτέο αποτελείται από 3 projects και το paradoteo.xml το οποίο περιέχει το virtual configuration για τον emulator. Τα projects είναι :

Το **Sensors** που περιέχει το MIDlet που θα γίνει deploy στους 3 κόμβους δειγματοληψίας στον προσομοιωτή μας.

Το **Collector** που περιέχει ένα MIDlet με δύο νήματα για τη συλλογή και αποστολή πληροφοριών.

Και τέλος το **Sink** με το MIDlet που εμφανίζει μήνυμα για την ύπαρξη ή όχι πυρκαγιάς ανάλογα την πληροφορία που έλαβε.

Στα επόμενα κεφάλαια αναλύονται η υλοποίηση και οι τεχνικές λεπτομέρειες των παραπάνω.

PROJECTS

Sensors

Το συγκεκριμένο project περιέχει τρία αρχεία, το MySensors.java, το NormalRandom.java και το Sensors.java.

Το αρχείο Sensors είναι το MIDlet μας το οποίο κάνει χρήση των άλλων δύο αρχείων. Αρχικά το NormalRandom.java δημιουργήθηκε για χρήση της συνάρτησης nextGaussian η οποία όμως δεν παρέχεται για αυτή την έκδοση οπότε υλοποιήθηκε περαιτέρω όπως περιγραφόταν και στο μάθημα η κανονική κατανομή στη συνάρτηση gaussian, που επιστρέφει ένα πίνακα με τις δύο τιμές (θερμοκρασίας και υγρασίας).

Στη κλάση mySensors υλοποιείται η δομή του κόμβου για την συγκεκριμένη εργασία, δηλαδή το νευρωνικό δίκτυο, τα βάρη και ο έλεγχος για ειδοποίηση φωτιάς.

Όσον αφορά τα βάρη δοκιμάστηκε να δίνονται από αρχείο αλλά τελικά η συγκεκριμένη συνάρτηση σχολιάστηκε λόγω προβλήματος αναγνώρισης του αρχείου input.txt (το οποίο παρέμεινε στο project), έτσι δίνεται στατικά πίνακας από double ο οποίος μετατρέπεται μέσω της doubleToDoubleArray σε Double (δεν γινόταν δεκτό κατευθείαν σαν Double).

Ο έλεγχος για ειδοποίηση φωτιάς γίνεται μέσω της fireAlert η οποία ανάλογα τα δοσμένα από την εκφώνηση διαστήματα αποφασίζει αν υπάρχει κίνδυνος και στην περίπτωση ασαφής τιμής την τροφοδοτεί στο νευρωνικό δίκτυο για να βγάλει αποτέλεσμα.

Το νευρωνικό δίκτυο αποτελείται από 2 νευρώνες στο επίπεδο εισόδου, 4 νευρώνες κρυφού επιπέδου και 1 νευρώνα εξόδου. Η δημιουργία τους γίνεται με την αντίστοιχη σειρά που προσδιορίζεται από την υλοποίηση του νευρωνικού που δόθηκε.

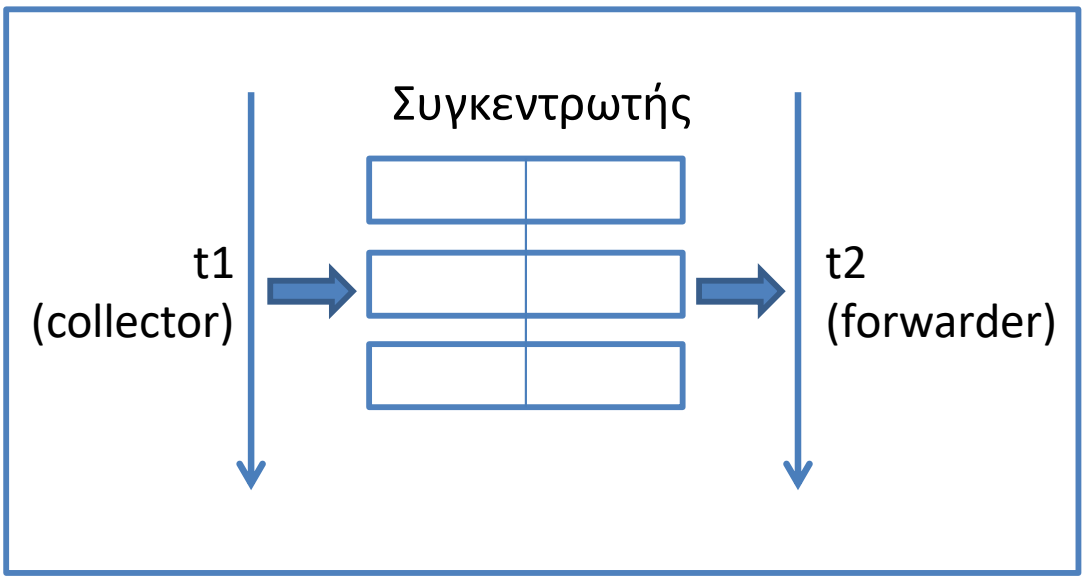
Επίσης, για τη σωστή χρήση της loadWeightsThresholds, (η οποία παρέχεται), τα βάρη που δίνονται στατικά έχουν το σωστό πλήθος έχοντας στο μυαλό τον νευρώνα πόλωσης.

Να αναφερθεί ότι τα βάρη που δίνονται στατικά επιλέχθηκαν αφού υλοποιήθηκε εξωτερικά σε άλλο project training για το νευρωνικό δίκτυο μας, ωστέ να είναι πιο σωστά τα αποτελέσματα που λαμβάνουμε.

Στη συνέχεια στην Sensors.java στέλνεται η τιμή 0 αν υπάρχει κίνδυνος και 1 αν όχι, μέσω επικοινωνίας με τον Συγκεντρωτή στη θύρα 68.

Collector

Ο Συγκεντρωτής αποτελείται από δύο συγχρονισμένα νήματα τον Collector και τον Forwardor (Ιαπωνική προφορά).



Στο νήμα Collector γίνεται η λήψη των δεδομένων από τους αισθητήρες με χρήση της θύρας 68. Αποθηκεύονται οι τριάδες των στοιχείων που φτάνουν από τους αισθητήρες σε έναν πίνακα Vectors και με χρήση συγχρονισμού οδεύουν στον Forwarder μας αφού γίνει notify η κάθε τιμή.

Τώρα στο νήμα Forwardor γίνεται αποστολή του αποτελέσματος των τιμών αυτών έπειτα από την χρήση της συνάρτησης majority που εκτελεί την “ψηφοφορία” και προσδιορίζει το πλειοψηφικό αποτέλεσμα για την καταβόθρα, οπότε στέλνει Boolean μεταβλητή ανάλογα το αποτέλεσμα.

Sink

Τέλος, το Sink αποτελείται από το Sink.java και είναι η υλοποίηση της καταβόθρας. Μέσω της θύρας επικοινωνίας 69 λαμβάνει την Boolean μεταβλητή από τον Collector και ανάλογα αν είναι αληθής η τιμή της πυροδοτεί μήνυμα συναγερμού για φωτιά, διαφορετικά εκτυπώνεται μήνυμα ότι δεν υπάρχει κάποιο πρόβλημα στο δίκτυο.

ΕΠΙΛΟΓΟΣ

Συμπερασματικά, υλοποιήθηκαν τα ζητούμενα της εργασίας και τηρήθηκαν οι συχνότητες που ορίστηκαν για κάθε επιμέρους στοιχείο με χρήση της Utils.sleep για περίοδο ίση με 10000 milliseconds.

Όλα τα προηγούμενα εκτελέστηκαν και ελέγχθηκαν στο περιβάλλον προσομοίωσης.

Αφού κάνουμε “deploy MIDlet” το jar που παράγεται από το build του Sensors στους 3 αισθητήρες, κάνουμε αντίστοιχα “deploy” το jar του Collector και του Sink, έτσι ο προσομοιωτής Solarium αποκτά την παρακάτω μορφή: