Ενσωματωμένα Συστήματα Μικροεπεξεργαστών

MILESTONE 3: AUTONOMOUS MAP EXPLORING CAR

Καλογεράκης Στέφανος | Ζερβάκης Αρης

Εισαγωγή

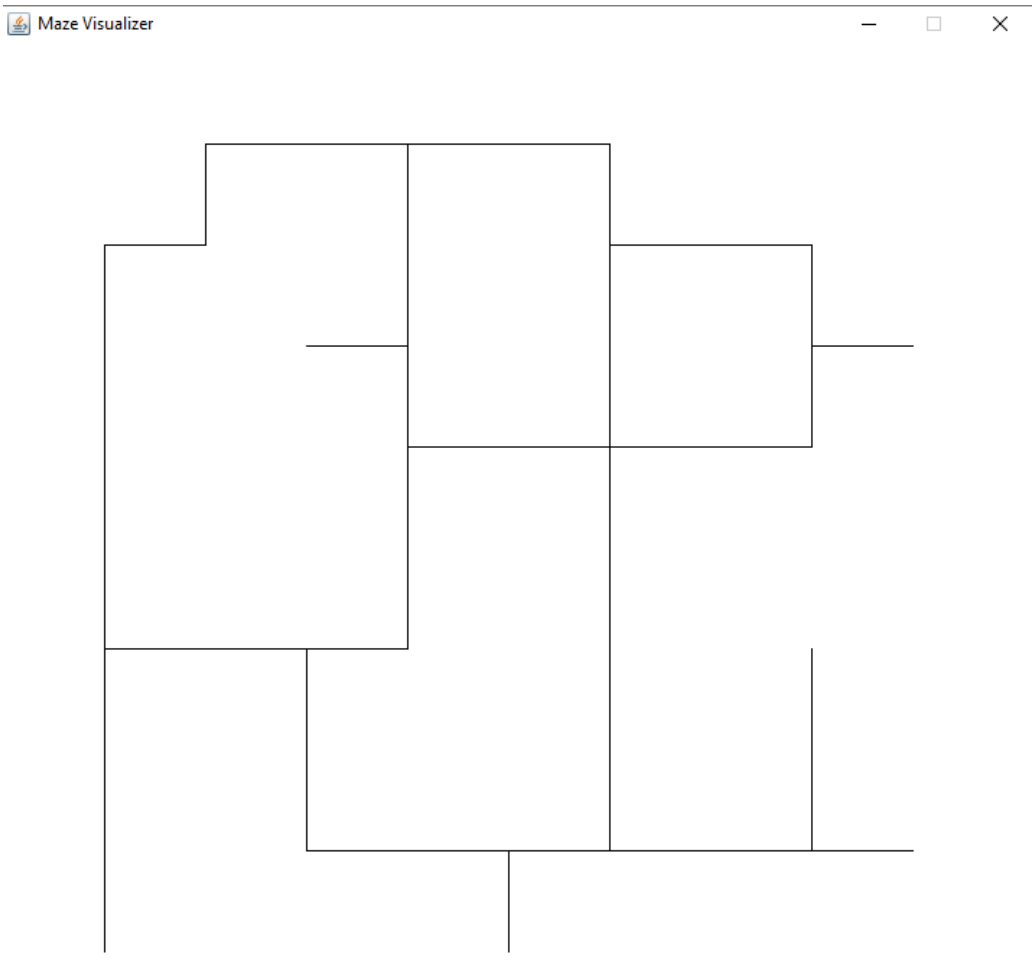
Σε συνέχεια της υλοποίησης του πρότζεκτ εξαμήνου Autonomous Map Exploring Car με την χρήση της τεχνολογίας Lego Mindstorms στο τρίτο και τελευταίο milestone είχαμε να ολοκληρώσουμε τις λειτουργίες του αυτοκινήτου που έχουν καθοριστεί. Πιο συγκεκριμένα, στο τρίτοο milestone υλοποιήθηκε ο τελικός αλγόριθμος Depth First Search (DFS) στο περιβάλλον RobotC. Ακόμα, υλοποιήθηκε και μια εφαρμογή σε Java που θα χρησιμοποιηθεί για την τελική απεικόνιση του λαβύρινθου που διέτρεξε το αυτοκίνητό μας.

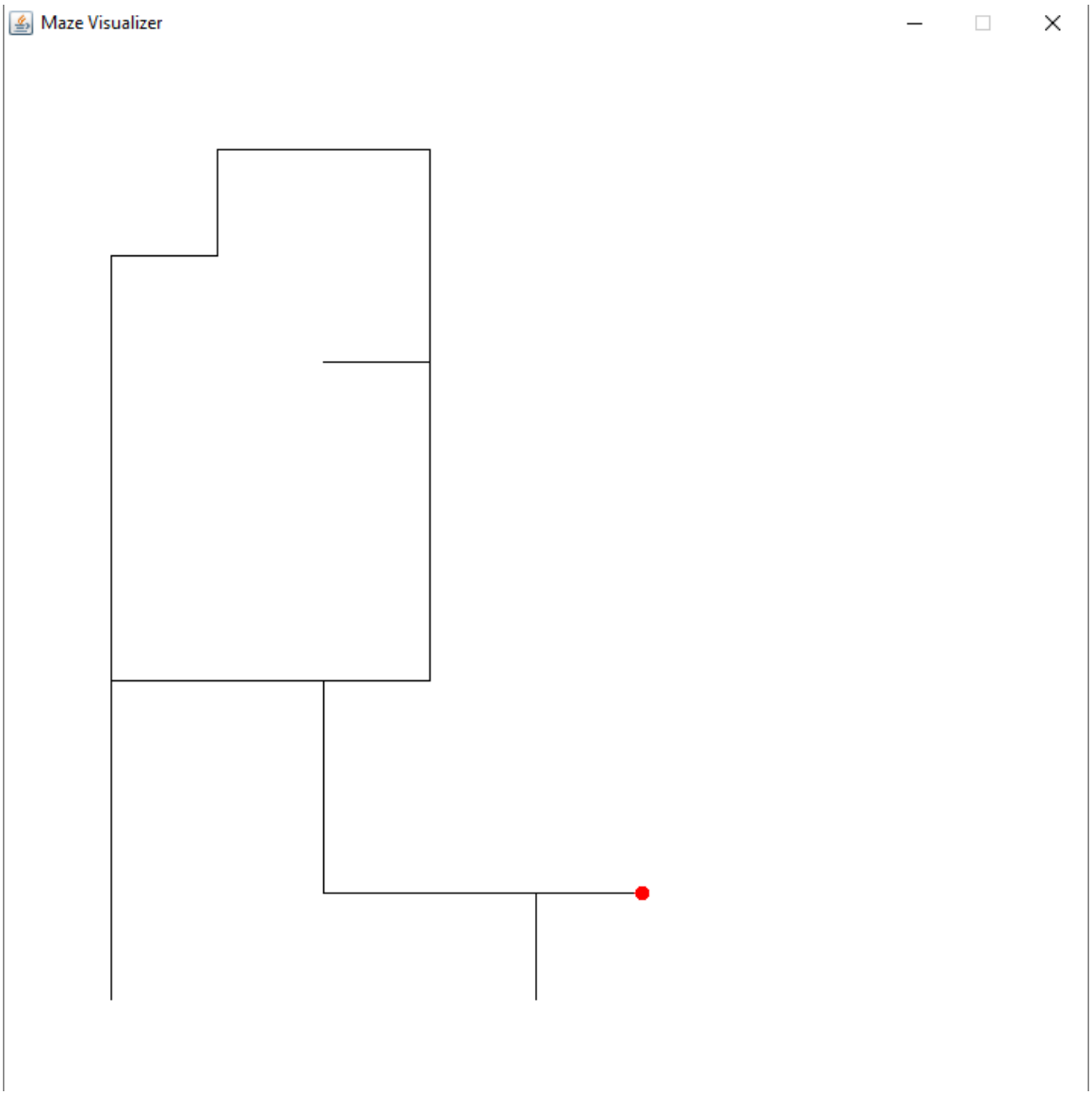
Περιγραφή

Παρακάτω περιγράφονται αναλυτικά οι υλοποίησεις που περιεγράφηκαν παραπάνω

Εξαντλητικός αλγόριθμος ανάλυσης λαβυρίνθου σε C

Σε ένα πρώτο στάδιο, οφείλαμε να βρούμε ποιόν αλγόριθμο θα χρησιμοποιήσουμε





Αλλαγές σε java αρχείο MazeVisualizer

Από το προηγούμενο milestone είχαμε προχωρήσει στην δημιουργία ενός εκτελέσιμου java jar αρχείου με όνομα MazeVisualizer με σκοπό την γραφική απεικόνιση του λαβύρινθου που πραγματοποιεί το αυτόνομο αυτοκίνητο. Η εφαρμογή αυτή παίρνει ως όρισμα το παραγόμενο αρχείο του αυτοκινήτου που περιέχει πληροφορίες σχετικές με την διαδρομή που ακολούθησε.

Όπως αναφέρθηκε όμως και στο προηγούμενο milestone θα προχωρούσαμε σε μικρές αλλαγές που οφείλονται στο fomat του αρχείου. Έτσι, μετά από τις δοκιμές μας στον λαβύρινθο και εξαγωγή όλων των απαραίτητων δεδομένων, προχωρήσαμε σε μια σειρά από αλλαγές για να παράξει η ήδη υπάρχουσα εφαρμογή το ζητούμενο αποτέλεσμα.

Αρχικά, επισημαίνουμε ότι το αρχείο εισόδου που χρησιμοποιείται είναι .csv αρχείο. Το συγκεκριμένο χαρακτηριστικό δεν επηρέασε κάπου την υλοποίηση μας απλώς επισημαίνεται σαν κομμάτι των προδιαγραφών.

	A	B	C	D	E	F
1	255	82				
2	27	0				
3	27	1				
4	27	1				
5	27	1				
6	27	1				
7	27	1				
8	27	0				
9	27	0				
10	27	0				
11	27	1				
12	27	1				
13	27	0				
14	27	1				
15	27	0				
16	27	1				
17	27	1				
18	27	1				
19	27	0				
20	27	1				
21	27	0				
22	27	1				
23	27	1				
24	27	0				
25	27	1				
26	27	0				
27	27	1				
28	27	1				
29	27	1				
30	27	0				

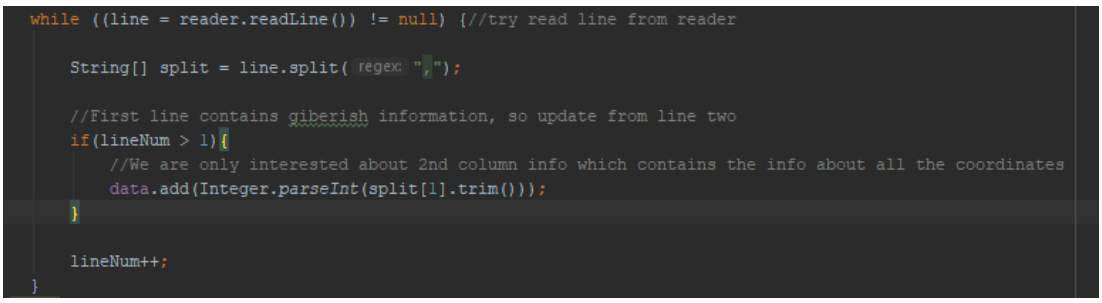
Η μορφή του αρχείου είναι όπως βλέπουμε στην παραπάνω εικόνα με τα εξής χαρακτηριστικά

- Την πρώτη γραμμή και την πρώτη στήλη δεν τις λαμβάνουμε υπόψιν στους υπολογισμούς μας καθώς παράγονται αυτόματα από το lego nxt χωρίς να μας προσφέρουν χρήσιμη πληροφορία για τους υπολογισμούς μας
- Όλα τα δεδομένα μας βρίσκονται στην στήλη B. Ο τελικός μας λαβύρινθος είναι μεγέθους όπως προαναφέραμε είναι 9 * 9 ενώ τα δεδομένα έχουν περαστεί row-based. Αυτό σημαίνει ότι οι πρώτες 9 τιμές απεικονίζουν την πρώτη γραμμή του λαβυρίνθου μας, οι επόμενες 9 την δεύτερη γραμμή κ.λ.π.
- Αναφορικά με την σημασιολογική απεικόνιση των τιμών δεν συναντάται κάποια αλλαγή σε σχέση με το προηγούμενο milestone με την τιμή 1 να δηλώνει ότι το αυτοκίνητο πέρασε από την συγκεκριμένη διαδρομή και ζωγραφίζεται ενώ η τιμή 0 δηλώνει ότι ένας κόμβος δεν επισκέφτηκε και δεν ζωγραφίζονται. Τέλος, η τιμή 2 υποδηλώνει τερματικό σημείο και συμβολίζεται με κόκκινη κουκίδα.
- Σε κάθε περίπτωση το nxt εξάγει δεδομένα αυστηρά για 9 * 9 πίνακα ακόμα και για την περίπτωση που βρεθεί τερματικός κόμβος και η τελική διαδρομή είναι πιο μικρή.

- Ο κώδικας δύναται να εκτελεστεί αυστηρά για συμμετρικούς πίνακες 9 * 9 ενώ σε περίπτωση που ο χρήστης επιθυμεί να αλλάξει το συγκεκριμένο γεγονός μπορεί να ακολουθήσει τις οδηγίες που αναγράφονται στον πηγαίο κώδικα

Τα υπόλοιπα χαρακτηριστικά και προδιαγραφές που δεν επισημαίνονται είναι ίδια με το προηγούμενο milestone.

Ο κώδικας μάλιστα ήταν γραμμένος με τέτοιο τρόπο με τις αλλαγές που επισημάνθηκαν να απαιτούν διόρθωση μερικών γραμμών που φαίνονται στην παρακάτω εικόνα με την μορφή του τελικού αρχείου να μην διαφοροποιείται καθόλου



Στην συνέχεια επισημαίνονται οι οδηγίες για την εκτέλεση του αρχείου όπως είχαν διατυπωθεί και στο προηγούμενο milestone

Ο κώδικας δοκιμάζεται με την εκτέλεση ενός .jar αρχείου με όνομα MazeVisualizer.jar. Προκειμένου να δοκιμαστεί το πρόγραμμα πρέπει να είναι εγκατεστημένη η java στον υπολογιστή του χρήστη σε έκδοση πιο καινούργια από αυτή που έγινε το build του jar αρχείου. Στο αρχείο **requirements.txt** υπάρχουν λεπτομέρειες αναφορικά με την έκδοση java που πρέπει να είναι εγκατεστημένη στον εκάστοτε υπολογιστή.

Εκτός από την εντολή java –jar που απαιτείται για την εκτέλεση τέτοιου τύπου αρχείων απαιτείται και ένα όρισμα το οποίο θα απευθύνεται στο **full path** που βρίσκεται το αρχείο εισόδου. Ακολουθεί παράδειγμα εκτέλεσης του προγράμματος από την γραμμή εντολών των Windows. Ο κώδικας εκτελείται με αντίστοιχο τρόπο και σε εκδόσεις των Linux

java -jar MazeVisualizer.jar
D:\TUC_PROJECT\TUC_Autonomous_Map_Exploring_Car\MazeVisualizer\DATA0001.csv

Συνοψίζοντας σε μια γενική μορφή θα πρέπει να ισχύει

java -jar <JAR NAME> <FULL PATH INPUT FILE>

Κώδικας RobotC για περιφορά αυτοκινήτου σε τυχαίο λαβύρινθο

Όπως και στο δέυτερο Milestone, για την περιφορά στο λαβύρινθο είχαμε στη διάθεσή μας δεδομένα τα οποία λαμβάναμε από 2 Light Sensors και ενα RGB Sensor.

Συναρτήσεις κίνησης

Για ευκολότερη κίνηση στο λαβύρινθο ξεκινήσαμε υλοποιώντας 4 διαφορετικές συναρτήσεις, μία για την κάθε ξεχωριστή κίνηση που μπορεί να πραγματοποιηθεί.

- Void forward();

  Η συνάρτηση forward αφοραά την κίνηση προς την οποία ειναι στραμένο το αυτοκίνητο τη συγκεκριμένη χρονική στιγμή. Όπως και στο Milestone 2 κύρια ιδέα της συνάρτησης είναι να γίνει contain η μαύρη γραμμη ανάμεσα στους 2 light sensors που έχουμε στη διάθεση μας. Για να μπορέσουμε να μετρήσουμε την κίνηση μας στο λαβύρινθο χρησιμοποιήμε τις συναρτήσεις nMotorEncoderTarget(MOTOR). Έτσι σε περίπτωση forward κίνησης κάθε κελί ενημερώνεται μόλις το αυτοκίνητο πραγματοποιήσει απόσταση ίση με: (distance*360)/(2*π*radius), όπου distance το μέγεθος των νοητικών κελιών του 10x10 λαβυρίνθου, και radius η ακτίνα των τροχών του αυτοκινήτου.

- Void right_turn(); και Void left_turn();

  Οι συναρτήσεις τόσο για τη δεξιά όσο και για την αριστερή στροφή είχαν υλοποιηθεί κατα το milestone 2.

- Void full_turn();

  Η συνάρτηση full_turn είναι απαραίτητη όταν συναντάμε κάποιο αδιέξοδο στο λαβύρινθο και είναι απαραίτητη η στροφη κατα 180 μοίρες. Μόλις πραγματοποιηθεί χρησιμοποιώντας και πάλι τη συνάρτηση nMotorEncoderTarget(MOTOR) πραγματοποιούμε κίνηση προς τα πίσω κατα 4 εκατοστά ώστε να επιστρέψουμε στη θέση που βρισκόμασταν πριν το full turn αλλα με την αντίθετη πια κατεύθυνση. Η απόσταση των τεσσάρων εκατοστών προέκυψε έπειτα απο πειραματισμό και δοκιμές.

Στην περίπτωση των συναρτήσεων right_turn, left_turn και full_turn εκτός απο τη βασική λειτουργίας τους, πραγματοποιήται και ενημέρωση μιας μεταβλητής η οποία μας δηλώνει την κατεύθυνση κατα την οποία κινούμαστε στο χάρτη, ως προς την αρχική μας θέση.

Μοντελοποίηση λαβυρίνθου

Λόγω τυχαιότητας της αρχικής μας θέσης πάνω στο λαβύρινθο θεώρουμε πάντα πίνακα με τις διπλάσιες διαστάσεις (δηλ. 20X20) και αρχική μας θέση το κελί (10,10). Έτσι κατά την ολοκλήρωση του αλγορίθμου μεταφέρουμε τα δεδομένα του 20x20 πίνακα σε ένα τελικό πια, διαστάσεων 10x10 τον οποίο εξάγουμε σε μορφή csv με τη λειτουργία datalogging του NXT. Αν ο πίνακας δεν έχει ενημερωθεί ολόκληρος λόγω του ότι συναντήσαμε κάποιο σημείο τερματισμού μεταφέρουμε στον 10x10 πίνακα μόνο τα δεδομένα που αποκτήσαμε μέχρι εκείνη τη στιγμή.

Αλγόριθμος Depth First Search (DFS)

Στην υλοποίηση μας η προτεραιότητα κόμβων προς επίσκεψη κατά σειρά είναι ΠΑΝΩ, ΑΡΙΣΤΕΡΑ, ΚΑΤΩ, ΔΕΞΙΑ. Το πρόγραμμα επομένως εξετάζει τους γειτονικούς κόμβους και εφόσον κάποιος είναι επισκέψιμος (αν κάποιος κόμβος έχει ήδη επισκεφτεί δεν θεωρείται επισκέψιμος) ανάλογα με την παραπάνω αλληλουχία προτεραιοτήτων αποφασίζει ποιον κόμβο θα επισκεφτεί στην συνέχεια. Ξεκινάμε έτσι τον αλγόριθμο πάντα με την κίνηση forward και θεωρούμε την κατευθυνσή μας ΠΑΝΩ. Κατά τη διαδικασία, ενημερώνονται τόσο το κελί το οποίο επισκεπτόμαστε όσο και το αριστερό και το δεξιά αυτού με τη βοήθεια των μετρήσεων από τους 2 light sensors. Κατα την ενημέρωση τον κελιών του χάρτη ενημερώνουμε επίσης και στον visit_matrix, ποια κελιά έχουν επισκεφτεί.

Βιβλιογραφία

Forum με καθοδήγηση για προβλήματα που συναντήσαμε σε ROBOTC:
http://www.robotc.net/forums/viewforum.php?f=1&sid=6decc57fb332bd1c7039d8a0840df4bb

DFS: https://en.wikipedia.org/wiki/Depth-first_search

DFS maze search: https://www.algosome.com/articles/maze-generation-depth-first.html?fbclid=IwAR0t53lVX8ygkHFH_KYkIxzBFZfNnL0po1IYU6OQIy-0GnZrLu4y6wVPa2k