Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση

Τόμ. 1 (2005)

3ο Πανελλήνιο Συνέδριο «Διδακτική της Πληροφορικής»

Διδασκαλία Βασικών Προγραμματιστικών Εννοιών στο Περιβάλλον Οπτικού Προγραμματισμού ROBOLAB

Σπύρος Τσοβόλας, Βασίλης Κόμης

Βιβλιογραφική αναφορά:

Τσοβόλας Σ., & Κόμης Β. (2026). Διδασκαλία Βασικών Προγραμματιστικών Εννοιών στο Περιβάλλον Οπτικού Προγραμματισμού ROBOLAB. *Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση*, *1*, 420–431. ανακτήθηκε από https://eproceedings.epublishing.ekt.gr/index.php/cetpe/article/view/8734

Διδασκαλία Βασικών Προγραμματιστικών Εννοιών στο Περιβάλλον Οπτικού Προγραμματισμού ROBOLAB

Σπύρος Τσοβόλας, Βασίλης Κόμης
ΤΕΕΑΠΗ, Πανεπιστήμιο Πατρών
stsovol@upatras.gr, komis@upatras.gr

ΠΕΡΙΛΗΨΗ

Στην εργασία αυτή παρουσιάζονται και μελετώνται διδακτικά οι βασικές προγραμματιστικές δομές όπως υλοποιούνται στο περιβάλλον οπτικού προγραμματισμού Robolab. Το περιβάλλον αυτό χρησιμοποιείται τα τελευταία χρόνια στο πλαίσιο μιας εποικοδομιστικής προσέγγισης για τη διδασκαλία των εννοιών του προγραμματισμού σε συνδυασμό με βασικές έννοιες από την τεχνολογία ελέγχου και το χειρισμό ρομποτικών συσκευών. Η εκπαιδευτική έρευνα στην εν λόγω περιοχή δεν έχει ακόμα απαντήσει σε μια σειρά από γενικότερα ερωτήματα όπως για ποιες ηλικίες είναι κατάλληλο και για το εάν μπορεί να χρησιμοποιηθεί αποτελεσματικά στην πρωτοβάθμια και τη δευτεροβάθμια εκπαίδευση για τη διδασκαλία του προγραμματισμού και μια σειρά από ειδικότερα ερωτήματα σχετικά με τη χρήση οπτικών εντολών (και γενικότερα της προσέγγισης που βασίζεται στον οπτικό προγραμματισμό) στην κατανόηση των προγραμματιστικών δομών και πως αυτή η κατανόηση υποστηρίζεται από τον κατάλληλο χειρισμό προγραμματιζόμενων συσκευών.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: *Διδασκαλία Πληροφορικής, Robolab, Οπτικός προγραμματισμός*

ΕΙΣΑΓΩΓΗ

Τα τελευταία χρόνια, η τεχνολογία ελέγχου συνδέθηκε σε μεγάλο βαθμό με τον οπτικό προγραμματισμό και έδωσε ευκαιρία στους ερευνητές και στους εκπαιδευτικούς να εμπλέξουν τους μαθητές σε διαδικασίες επίλυσης ανοιχτών προβλημάτων (Jarvinen 1998, Lavonen 2001). Στην πλειοψηφία τους δηλαδή οι προσπάθειες αυτές δεν απομόνωσαν τον οπτικό προγραμματισμό με σκοπό, δια μέσω αυτού, να διδαχθούν προγραμματιστικές δομές αλλά τον συνέδεσαν με έλεγχο συσκευών, με πλοήγηση στο επίπεδο, με εξερεύνηση του χώρου χωρίς τη διαμεσολάβηση του ανθρώπινου σώματος και των αισθήσεων κλπ. Οι διάφορες εκδοχές του οπτικού προγραμματισμού που εστίασαν στην τεχνολογία ελέγχου έχουν περιορισμένο «λεξιλόγιο», αυτό που χρειάζεται για τον έλεγχο αυτών των συσκευών: λήψη τιμών αισθητήρων (ερέθισμα) και κατάλληλη αντίδραση στα ερεθίσματα δηλαδή ρύθμιση της ισχύος των εξόδων (μοτέρ ή λαμπάκια). Συνολικά αυτός ο προγραμματισμός μιας συσκευής δίνει την επιθυμητή συμπεριφορά στη συσκευή. Στην εργασία αυτή παρουσιάζονται και μελετώνται

Πρακτικά Εργασιών 3ου Πανελλήνιου Συνεδρίου «Διδακτική της Πληροφορικής»
Α. Τζιμογιάννης (επιμ.)
Πανεπιστήμιο Πελοποννήσου
Κόρινθος, 7-9 Οκτωβρίου 2005

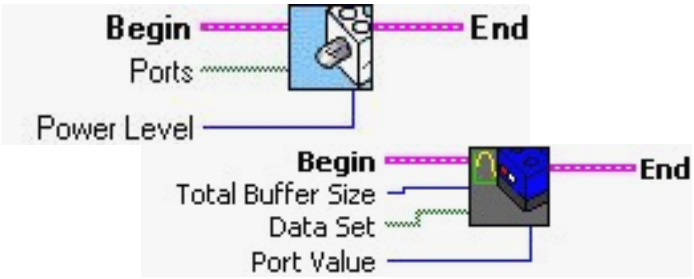
διδακτικά οι βασικές προγραμματιστικές δομές όπως υλοποιούνται στο περιβάλλον οπτικού προγραμματισμού Robolab. Το Robolab έλκει την καταγωγή του αφενός από το επιστημονικό λογισμικό LabView για την καθοδήγηση ρομπότ και αφετέρου από τη δημιουργία και τον προγραμματισμό κατασκευών τύπου Lego.

ΤΟ ΠΕΡΙΒΑΛΛΟΝ ROBOLAB

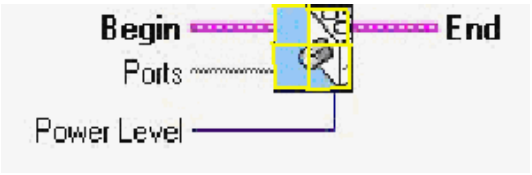
Ένα πρόγραμμα διαβάζεται από αριστερά προς τα δεξιά και είναι μια **ακολουθία εικονιδίων**.

Κάθε εικονίδιο έχει νήματα με τα οποία ενώνεται με το προηγούμενο και το επόμενο.

Μερικά εικονίδια έχουν πλέον των δύο νημάτων σύνδεσης. Στο πάνω μέρος κάθε εικονιδίου, αριστερή και δεξιά γωνία αντίστοιχα υπάρχουν οι θέσεις Begin και End που βοηθούν τη συναρμολόγηση – ένωση με προηγούμενα και επόμενα εικονίδια. Τα υπόλοιπα νήματα χρειάζονται για δηλώσεις παραμέτρων όπως θύρα σύνδεσης, ισχύς λειτουργίας, είδος δεδομένων, τιμή θύρας.



Το εικονίδιο χωρίζεται σε περιοχές ανάλογα με τα νήματα που διαθέτει. Το επόμενο εικονίδιο που περιγράφει ένα λαμπάκι έχει τέσσερα νήματα γι' αυτό χωρίζεται σε τέσσερις περιοχές. Από κάθε περιοχή ξεκινά ένα νήμα.

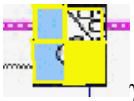


Έτσι στο παραπάνω εικονίδιο, που δηλώνει τη λειτουργία σε ένα λαμπάκι, φαίνονται χωρισμένες και γεμισμένες με κίτρινο χρώμα οι τέσσερις περιοχές που αντιστοιχούν:

Begin  για ένωση με το επόμενο εικονίδιο

End  για ένωση με το προηγούμενο εικονίδιο

Ports  για δήλωση σε ποια θύρα είναι συνδεδεμένο

Power Level  για δήλωση της ισχύος λειτουργίας

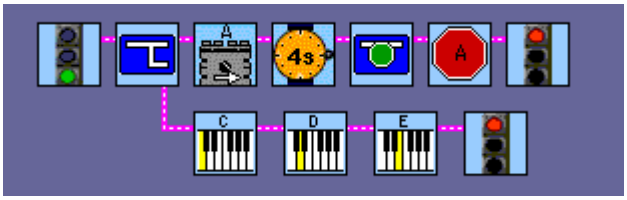
Το επόμενο εικονίδιο περιγράφει τον αισθητήρα θερμοκρασίας: είναι συνδεδεμένος στην είσοδο 3 και ελέγχει αν η θερμοκρασία του ξεπεράσει τους 30^{o} Κελσίου (δίνει τιμή 0 και 1, το 1 αντιστοιχεί σε θερμοκρασίες μεγαλύτερες των 30^{o} Κελσίου και 0 αντιστοιχεί σε τιμές μικρότερες ή ίσες των 30^{o} Κελσίου). Όπως φαίνεται, ακόμα και οι παράμετροι εντολών αντιστοιχίζονται με εικονίδια.

ΒΑΣΙΚΑ ΔΟΜΙΚΑ ΣΤΟΙΧΕΙΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΣΤΟ ROBOLAB

1. Ακολουθία: υλοποιείται με μια σειρά εικονιδίων που ενώνονται και διαβάζονται από αριστερά προς τα δεξιά. Το επόμενο πρόγραμμα σε μορφή ψευδοκώδικα: Αρχή, μοτέρ Α εμπρός, αναμονή 1S, κλείσιμο εξόδου Α, τέλος

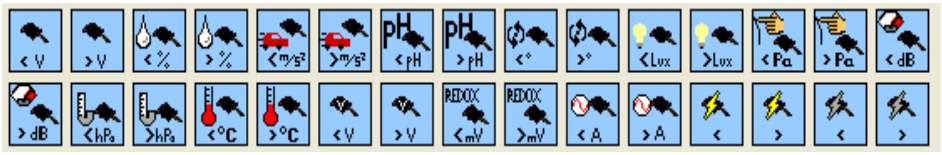
2. Δόμηση σε υπορουτίνες (μέχρι 8): Όταν η ακολουθία μεγαλώσει γίνεται δυσανάγνωστη και για αυτό υπάρχει δυνατότητα να γραφτούν υπορουτίνες (κομμάτια

κώδικα που μπορούν να λειτουργούν και αυτόνομα) που καλούνται σε κατάλληλη θέση στο τρέχον πρόγραμμα.
Στο παρακάτω πρόγραμμα ανοίγει το μοτέρ στη θέση A, λειτουργεί 4 sec, στη συνέχεια (κλήση υπορουτίνας) αναπαράγονται οι νότες C, D, E και κλείνει το μοτέρ στη θέση A.



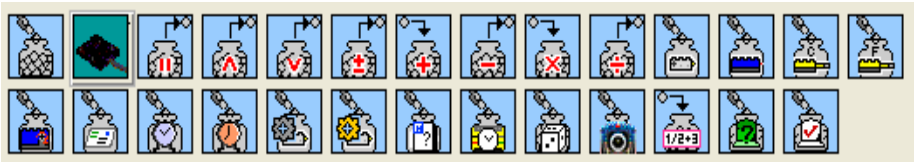
Ισοδύναμα ο κώδικας χωρίς κλήση υπορουτίνας

3. Περίμενε ώσπου … συνθήκη (wait for …): Πρόκειται για χρονική καθυστέρηση έως ότου ικανοποιηθεί κάποια συνθήκη. Η συνθήκη αυτή μπορεί να περιέχει τιμές αισθητήρων, τυχαία καθυστέρηση, καθυστέρηση για ορισμένο χρόνο ή ώσπου να ληφθούν συγκεκριμένο πλήθος τιμών των αισθητήρων.



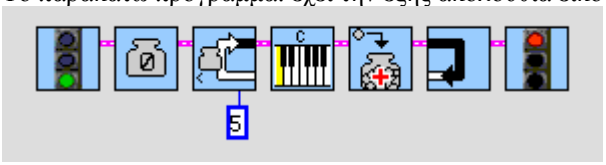
Στο παράδειγμα που ακολουθεί το μοτέρ στην έξοδο A θα κινηθεί για 1s αφού δηλώνεται η λειτουργία του, μετά αναμονή για 1s (wait for 1s), μετά διακοπή λειτουργίας της εξόδου A και τέλος προγράμματος.

4. Μεταβλητή (container): Οι μεταβλητές ονομάζονται containers και οι τρεις τυπικές μεταβλητές γενικής χρήσεως διακρίνονται με τα χρώματά τους (κόκκινη, κίτρινη, μπλε). Υπάρχουν επίσης μεταβλητές για κάθε αισθητήρα και κάθε μαθηματική πράξη που αλλάζει την τιμή της μεταβλητής αντιστοιχίζεται κατάλληλο εικονίδιο.



Έχουν επίσης προβλεφτεί και εικονίδια για άλλους αισθητήρες που δεν υπάρχουν στο κανονικό πακέτο όπως PH, ήχος, πίεση, υγρασία, τάση κλπ.

Το παρακάτω πρόγραμμα: έχει την εξής ακολουθία εικονιδίων:



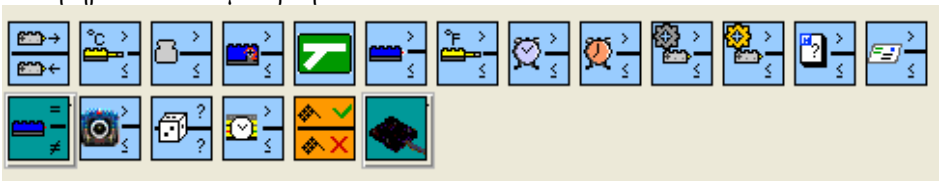
Αρχή, μηδενισμός της μεταβλητής (εννοείται η κόκκινη αφού δεν ονομάζεται), επανάλαβε εφόσον η τιμή της μεταβλητής είναι μικρότερη από 5, παίξιμο της νότας C, αύξηση κατά 1 της μεταβλητής, τέλος
Για κάθε ενέργεια διαχείρισης της μεταβλητής χρειάστηκε και το αντίστοιχο εικονίδιο:

 μηδενισμός τιμής

 σύγκριση τιμής μεταβλητής

 ανάθεση τιμής (αύξηση κατά 1)

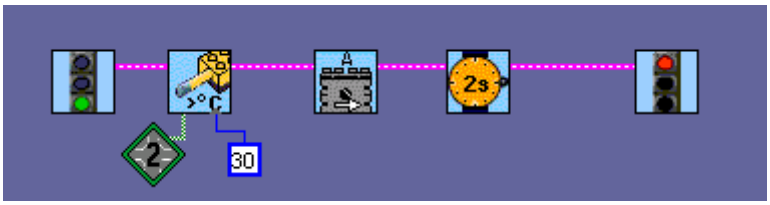
5. Λήψη απόφασης (if …then)
5.α Έλεγχος ανισότητας (μεγαλύτερο/ μικρότερο ή ίσο): Καλύπτει τις τιμές των αισθητήρων και των μεταβλητών.



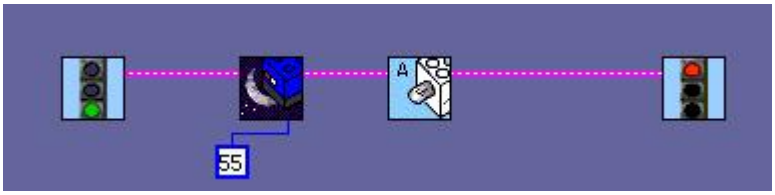
5.β Έλεγχος ισότητας (ίσο / διάφορο) που καλύπτει επίσης τιμές αισθητήρων ή μεταβλητών

Ο έλεγχος ισότητας και ανισότητας πρέπει να κλείνει με το εικονίδιο επανένωσης της διακλάδωσης

Για παράδειγμα, στο παρακάτω πρόγραμμα αν η θερμοκρασία ανεβεί πάνω από ένα ορισμένο όριο τότε λειτουργεί ο ανεμιστήρας. Εδώ ο αισθητήρας θερμοκρασίας τοποθετήθηκε στην είσοδο 2 και ο ανεμιστήρας στην έξοδο Α θα λειτουργήσει για δύο δευτερόλεπτα όταν η θερμοκρασία ξεπεράσει τους 30^{o} C. Σε πραγματικές συνθήκες, στα ψυγεία των αυτοκινήτων, αυτή η θερμοκρασία πλησιάζει τους 90^{o} C.



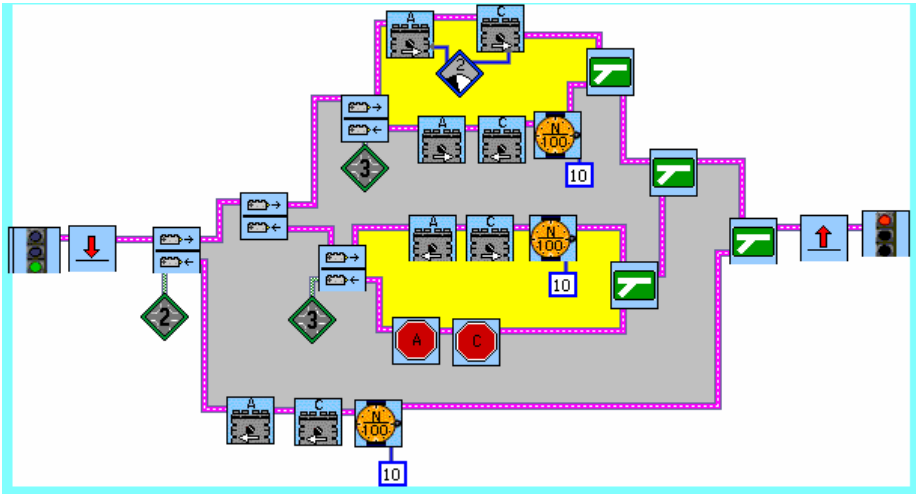
Το παρακάτω πρόγραμμα υποστηρίζει τη λειτουργία του δημόσιου φωτισμού που ανάβει αυτόματα όταν σκοτεινιάσει. Αν ο φωτισμός πέσει κάτω από ένα όριο (55) τότε θα ανάψει το λαμπάκι στην έξοδο Α.



Παρότι υποστηρίζονται εμφωλιασμένα if, το Robolab τα αποφεύγει και δεν δίνει ούτε ένα παράδειγμα προγραμματισμού με αυτό το στυλ. Αντιθέτως χρησιμοποιεί πολύ το στυλ άλματος (goto / jump) το οποίο σε εκφραστικό επίπεδο, συναρτήσει των γλωσσολογικών γνώσεων, φαίνεται ευκολότερο από τα εμφωλιασμένα if αλλά ο έλεγχος είναι πιο δύσκολος (Κόμης 2001).

Το παρακάτω παράδειγμα είναι πρόγραμμα ελέγχου αυτόνομης συσκευής (όχημα-μπουλντόζα) που σαν πιλοτήριο έχει τρεις αισθητήρες αφής:

Αν κανένας αισθητήρας δεν είναι πατημένος όχημα θα κινείται εμπρός.
Όσο είναι πατημένος ο αισθητήρας αφής 2 το όχημα θα κινείται προς τα πίσω
Όσο είναι πατημένος ο αισθητήρας αφής 1 το όχημα θα στρίβει αριστερά
Όσο είναι πατημένος ο αισθητήρας αφής 3 το όχημα θα στρίβει δεξιά
Αν κρατιούνται πατημένοι ταυτόχρονα οι αισθητήρες αφής 1 και 3 το όχημα θα ακινητοποιείται.



Στο παραπάνω παράδειγμα καθίσταται ολοφάνερη η διαφορετική δυσκολία ανάγνωσης ενός έτοιμου προγράμματος στις δύο προσεγγίσεις, την οπτική και τη λεκτική που παρατίθεται στη συνέχεια, σε περιβάλλον Microworlds EX Robotics.

```
to auto1
 loop [auto2]
end
 to auto3
 ifelse switch2 [athatway cthatway aon con wait 2]
[ifelse and switch1 switch3 [aoff coff]
  [if switch1 [athatway cthisway aon con wait 2]
   if switch3 [cthatway athisway aon con wait 2]]]
end
```

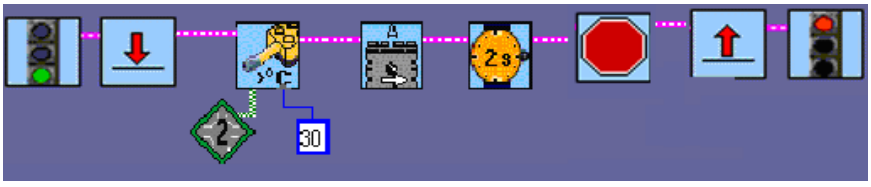
6. Επανάληψη (loop while): Και εδώ η επανάληψη ελέγχεται από τις τιμές αισθητήρων ή των τριών μεταβλητών γενικής χρήσης. Για κάθε αισθητήρα υπάρχει το αντίστοιχο εικονίδιο ελέγχου ανισότητας α) μεγαλύτερο /μικρότερο ίσο β) ίσο / διάφορο

Επίσης υπάρχει επανάληψη αριθμητική «επανάλαβε ν φορές» αρκεί το ν να μην ξεπερνά τον αριθμό 15.

Για παράδειγμα, το παρακάτω πρόγραμμα κινεί το μοτέρ στη θέση Α, αρχικά εμπρός. Στη συνέχεια και για όσο χρονικό διάστημα η τιμή του αισθητήρα θερμοκρασίας (στη θέση 1 αφού δεν δηλώνεται η θύρα) είναι μικρότερη από 30^{o} C (αφού δεν δηλώνεται η τιμή θα είναι η προεπιλεγμένη δηλαδή 30^{o} C) το μοτέρ θα εναλλάσσει μπρος–πίσω την κίνησή του κάθε 1s. Αν η θερμοκρασία ξεπεράσει τους 30^{o} C τότε το μοτέρ θα σταματήσει.

7. Δομή ελέγχου - Αλλαγή ροής (go to / jump): Η robolab υποστηρίζει τη δομή άλματος και κάθε άλμα παριστάνεται με το χρώμα του. Μπορούμε να έχουμε μέχρι έξι άλματα στον κώδικα του προγράμματος.

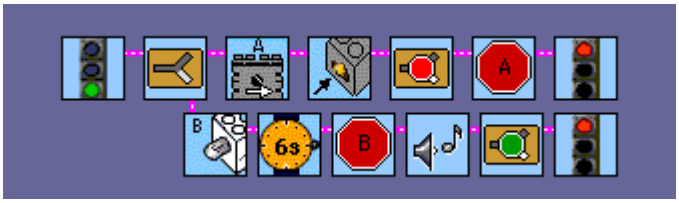
Στο παρακάτω παράδειγμα τα δύο εικονίδια red land   και red jump δημιουργούν μια ατέρμονα επανάληψη του κώδικα που περικλείουν δηλαδή όταν η θερμοκρασία ξεπερνά τους 30^{o} C το μοτέρ να λειτουργεί για 2s.



8. Παράλληλες διαδικασίες – παράλληλος προγραμματισμός (task split): Στην πραγματικότητα πρόκειται για ψευδοπαράλληλο προγραμματισμό μιας και ο ένας

επεξεργαστής ασχολείται τμηματικά με τουλάχιστον δύο διαδικασίες αλλά η μεγάλη ταχύτητά του δίνει την αίσθηση της ταυτόχρονης παράλληλης επεξεργασίας.

Στο παρακάτω παράδειγμα εκτελούνται δύο εργασίες:



α) Το μοτέρ στη θέση Α ξεκινά και εργάζεται κατά τη θετική φορά. Αν πατηθεί ο αισθητήρας αφής (στη θέση 1) τότε σταματούν τα πάντα (διαδικασίες και συσκευές). β) Το λαμπάκι στη θέση Β ανάβει για 6s, στη συνέχεια σβήνει και ακούγεται ένας ήχος. Στη συνέχεια και οι δύο διαδικασίες ξεκινούν από την αρχή (από το σημείο task split). Επειδή υπάρχει περίπτωση μια συσκευή (ένα μοτέρ) να ελέγχεται από δύο διαδικασίες, υπάρχει πρόβλεψη να οριστεί προτεραιότητα (task priority) σε κάθε διαδικασία.

ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΚΑΙ ΜΕΙΟΝΕΚΤΗΜΑΤΑ ΤΟΥ ΟΠΤΙΚΟΥ ΠΕΡΙΒΑΛΛΟΝΤΟΣ

Η αντιστοίχηση φυσικών οντοτήτων στο περιβάλλον διευκολύνει το μαθητή. Απεικονίζονται όχι μόνο οι φυσικές οντότητες αλλά και οι ιδιότητές τους και οι λειτουργίες τους. Ο μαθητής μπορεί σε σύντομο χρονικό διάστημα να γράψει κώδικα που να λειτουργεί. Δεν χρειάζεται «*να μάθει πολλά για να κάνει λίγα*» αλλά με μια απλή επίδειξη του περιβάλλοντος, μπορεί να κατασκευάσει πρόγραμμα ελέγχου ρομποτικής συσκευής όπως για παράδειγμα ένα όχημα που κινείται στο επίπεδο. Ειδικά τα εικονίδια λειτουργίας των εξόδων είναι προφανή. Στο περιβάλλον συγγραφής του κώδικα γίνεται ταυτόχρονα και συντακτικός έλεγχος, έτσι απαλλάσσεται ο προγραμματιστής (μαθητής) από συντακτικά λάθη. Φυσικά παραμένουν τα λογικά λάθη που θα πρέπει να αναζητηθούν στον έλεγχο (δοκιμή). Όσο για τα λάθη χρόνου εκτέλεσης σπανίζουν μιας και το πρόγραμμα συνήθως κατασκευάζεται για να δίνει συμπεριφορές σε ρομποτικές συσκευές, δηλαδή ανάλογα με τις τιμές των τριών αισθητήρων παρέχεται ισχύς στις τρεις εξόδους. Μια διαίρεση με το μηδέν, για παράδειγμα, είναι πολύ σπάνιο έως απίθανο γεγονός.

Ως μειονεκτήματα μπορούμε να αναφέρουμε έξι κύριες κατηγορίες που προέκυψαν από έρευνα με μαθητές πρωτοβάθμιας εκπαίδευσης.

Τα εικονίδια είναι πολλά και ίσως μερικά να είναι περιττά.

Για παράδειγμα, τα τρία εικονίδια που θέτουν σε λειτουργία ένα λαμπάκι στις εξόδους A, B, C υπάρχουν μόνο για λόγους πλήρους αντιστοίχισης συσκευών / εικονιδίων και άρα καλύτερης κατανόησης του κώδικα.

Αυτό γιατί αν αντί για το εικονίδιο λαμπάκι θέσουμε το εικονίδιο του μοτέρ το αποτέλεσμα είναι το ίδιο και το λαμπάκι θα δουλεύει κανονικά.

<u>Τα εικονίδια είναι πολλά και τίθεται θέμα διακριτότητας</u>.
Για παράδειγμα, για τον αισθητήρα θερμοκρασίας υπάρχουν πολλά διακριτά εικονίδια:

wait for – Περίμενε ώσπου η τιμή του αισθητήρα…..

loop while - Επανάλαβε όσο η τιμή του αισθητήρα …

if … then – έλεγχος ροής ανάλογα με την τιμή του αισθητήρα.

Η Robolab προτείνει το παραπάνω πρόγραμμα ως κατάλληλο για μικρά παιδιά ακόμη και του Δημοτικού. Δεν είναι εύκολο για μικρούς μαθητές να διακρίνουν όλα αυτά τα εικονίδια και τη λειτουργία τους, η χρήση τους γίνεται κυρίως με μηχανικό τρόπο.

<u>Η αντιστοιχία εικονιδίων και οντοτήτων/ λειτουργιών δεν είναι πάντα αυτονόητη</u>

Έστω το παραπάνω δείγμα κώδικα: *το μοτέρ στην έξοδο Α να κινηθεί*, στη συνέχεια *αναμονή για ένα δευτερόλεπτο* κλπ., εννοείται στην αναμονή 1s να συμβαίνουν τα προηγούμενα για 1s δηλαδή το μοτέρ Α θα κινείται, όταν τελειώσει το 1s τότε θα διαβαστεί και θα ληφθεί υπόψη η νέα οδηγία που θα ακολουθεί το εικονίδιο 1s.

Ένας μαθητής το μεταφράζει ως εξής «*να κινηθεί το μοτέρ Α και μετά να σταματήσουν τα πάντα για ένα δευτερόλεπτο*». Μεταφέρει προφανώς τη φράση της καθημερινής ζωής «περίμενε / wait» και τη μεταφράζει «*στάσου μην κάνεις τίποτα για ένα δευτερόλεπτο, μετά συνέχισε αυτό που έκανες*» Δηλαδή ο μαθητής αυτός προτείνει λειτουργία του μοτέρ συνεχή και διακοπή λειτουργίας για 1s.

Άλλος μαθητής πρότεινε πως η εντολή *περίμενε 1s* πρέπει να τοποθετηθεί μπροστά από την κίνηση του μοτέρ μιας και φαίνεται πιο λογικό να δίνουμε πρώτα χρόνο και μετά δράση δηλαδή «για ένα δευτερόλεπτο θέλω να κινηθεί το μοτέρ Α».

Η αναζήτηση του λάθους δεν είναι τόσο εύκολη υπόθεση

Στο επόμενο πρόγραμμα το όχημα κινείται μπρος μέχρι να πατηθεί ο αισθητήρας αφής. Τότε αντιστρέφει την κίνησή του για 2s και σταματάει.

Όμως οι μαθητές ξέχασαν να τοποθετήσουν το εικονίδιο που σταματά το μοτέρ με αποτέλεσμα ενώ τελειώνει το πρόγραμμα το όχημα να κινείται όπισθεν συνεχώς χωρίς να σταματάει.

Ένας μαθητής προσπαθώντας να βρει το λάθος, το εντοπίζει στο στράβωμα του νήματος που ενώνει τα εικονίδια! Η αναζήτηση του λάθους δεν είναι τόσο εύκολη υπόθεση, ειδικά όταν ο κώδικας μεγαλώσει.

Το πρόβλημα της αναζήτησης εικονιδίου

Στα συστήματα βοήθειας και υποστήριξης του προγραμματιστή υπάρχουν ταχύτατες μέθοδοι (όπως η δυαδική αναζήτηση) εύρεσης της κατάλληλης εντολής. Σε κείμενα (λέξεις) υπάρχει για παράδειγμα η ταξινόμηση κατά τύπο, κατά λειτουργία, κατά ονομασία κλπ. και βοηθείται έτσι πολύ ο προγραμματιστής στην εύρεση της επιθυμητής εντολής και των συναφών με αυτή εντολών. Στα εικονίδια δεν είναι δυνατό να υπάρχει αυτή η εξυπηρέτηση. Ο μαθητής-προγραμματιστής πρέπει να θυμάται το μοναδικό δρομολόγιο ανάκτησής του από την παλέτα εικονιδίων.

ΣΥΖΗΤΗΣΗ - ΣΥΜΠΕΡΑΣΜΑΤΑ

Στο ερώτημα ποια προγραμματιστικά μοντέλα και ποιες γλώσσες προγραμματισμού πρέπει να χρησιμοποιούμε στο σχολείο, η απάντηση δεν είναι καθόλου προφανής. Οι (Hirst et al. 2002) προτείνουν τη logo για την πρωτοβάθμια εκπαίδευση. Για τη δευτεροβάθμια εκπαίδευση προτείνουν τον οπτικό προγραμματισμό (π.χ. με Robolab) και σε μεγαλύτερες ηλικίες προτείνουν τις κλασσικές γλώσσες. Οι γλώσσες αυτές (Pascal, C, java ….) είναι κατασκευασμένες να ελευθερώνουν τις ικανότητες του προγραμματιστή.

Το περιβάλλον του Robolab, αναγνωρίζοντας τις γνωστικές δυσκολίες στις μικρές ηλικίες και στους αρχάριους προγραμματιστές, προτείνει οκτώ επίπεδα εμπλοκής του μαθητή, αυξανόμενης δυσκολίας. Τα επίπεδα αυτά δεν είναι επαρκή σε σχέση με τις ικανότητες των μαθητών της πρωτοβάθμιας εκπαίδευσης. Σύμφωνα με τη δική μας εμπειρία από μελέτες περίπτωσης στο Δημοτικό προκύπτουν τα ακόλουθα: στην Πέμπτη τάξη οι μαθητές εργαζόμενοι με το Robolab μπορούν να σχηματίζουν ακολουθία

εντολών αρκετά μεγάλη. Μπορούν επίσης να προγραμματίσουν μία έξοδο δηλαδή ένα μοτέρ είτε απλά είτε χρησιμοποιώντας τη δομή wait for…. <τιμή αισθητήρα>. Όταν δεν υπάρχει γραμμικότητα (όπως στην λήψη απόφασης ifelse) παρουσιάζεται μεγάλη δυσκολία. Οι μαθητές της Έκτης τάξης φαίνεται να μπορούν να προχωρήσουν περισσότερο σε σχέση με τους μαθητές της Πέμπτης τάξης. Μπορούν να προγραμματίζουν με διακλάδωση (ifelse) και να ελέγχουν τη δράση δύο εξόδων που συνήθως είναι μοτέρ. Το σημαντικότερο είναι πως οι μαθητές του Δημοτικού προτείνουν για τις κατασκευές τους συμπεριφορές πολύπλοκες που απαιτούν προγραμματιστικές δομές δύσκολες όπως η επανάληψη ή η προτεραιότητα διαδικασίας (task priority). Προγραμματιστικά τέτοιες συμπεριφορές ξεπερνούν το επίπεδο των μαθητών και ακόμα και αν ο διδάσκων κατασκευάσει μια τέτοια συμπεριφορά, οι μαθητές δεν είναι ικανοί να οικειοποιηθούν τον κώδικα. Είναι ένα σημείο που χρειάζεται ιδιαίτερη προσοχή, αφού απαιτείται αρκετή εξάσκηση σε ποικιλία καταστάσεων με απλούστερες συμπεριφορές και βηματική οικοδόμηση του προγράμματος. Η μέθοδος δοκιμή και πλάνη είναι η καταλληλότερη για την τεχνολογία ελέγχου γι'αυτό και απαιτείται διάθεση χρόνου στους μαθητές να κάνουν δοκιμές με τις κατασκευές τους ώστε να επιτύχουν την απαιτούμενη συμπεριφορά.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Hirst A., Johnson J., Petre M., Price B. & Richards M. (2002), *What is the best environment-language for teaching robotics using Lego MindStorms*?, Department of Telematics, The Open University, UK

Jarvinen E.-M. (1998), The Lego/Logo learning environment in technology education: An Experiment in a Finnish Context, *Journal of Technology Education*, 9(2)

Lavonen J.-M., Meisalo M. & Lattu M., (2001), Problem solving with an icon oriented programming tool: A case study in technology education, *Journal of Technology Education*, 12(2), 21-34

Cyr M. N., *Mindstorms for schools. Using ROBOLAB*, Lego System, Denmark

Στεργιοπούλου Ν. (1999), *Microworlds Pro βιβλίο μαθητή,* Ινστιτούτο Τεχνολογίας Υπολογιστών

Βακάλη κ.α, (1999), *Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον*, Αθήνα: Παιδαγωγικό Ινστιτούτο

Κόμης Β. (2001), *Διδακτική της Πληροφορικής*, Πάτρα: Ελληνικό Ανοιχτό Πανεπιστήμιο

Τζαβάρας κ.α., (1999), *Πληροφορική Γυμνασίου,* Αθήνα: Παιδαγωγικό Ινστιτούτο

Δαπόντες Ν. (1989), *Η διδασκαλία της Logo στη δευτεροβάθμια εκπαίδευση*, Αθήνα: Gutenberg

Μικρόπουλος Τ. & Λαδιάς Τ. (2000), *Η Logo στην εκπαιδευτική διαδικασία*, Ιωάννινα: Πανεπιστήμιο Ιωαννίνων

ΦΕΚ1373-18/10/2001, *ΔΕΠΠΣ*

ΦΕΚ 1374 & 1375-18/10/2001, *Προγράμματα Σπουδών Δημοτικού, Γυμνασίου, Λυκείου*