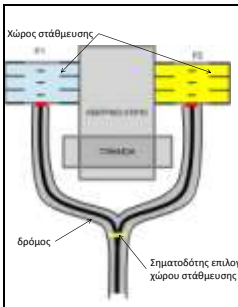
Αυτόματη στάθμευση

Αυτόματη στάθμευση φορτηγών σε βιομηχανικό χώρο



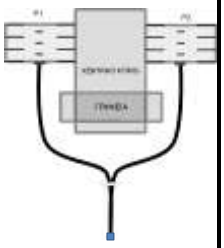
Το πρόβλημα:

ρομποτικό σύστημα **αυτόματης** οδήγησης και στάθμευσης με **επιλογή** χώρου και θέσης

Ένα εργοστάσιο διαθέτει 2 χώρους στάθμευσης των 12 οχημάτων που έχει για τη διανομή των προϊόντων που παράγει. Στο σχέδιο φαίνεται η χωροθέτηση του κτιρίου και των χώρων αυτών.
Προκειμένου να διευκολυνθεί η διαδικασία στάθμευσης των οχημάτων, μας ζητήθηκε να σχεδιάσουμε και να υλοποιήσουμε ένα ρομποτικό σύστημα που θα τοποθετηθεί στα οχήματα ώστε να σταθμεύουν χωρίς οδηγό.

Ανάλυση

- **Παρακολούθηση** μαύρης γραμμής με **επιλογή** χώρου στάθμευσης Ρ1 ή Ρ2
- **Άφιξη** στο χώρο στάθμευσης
- **Αναζήτηση** κενής θέσης
- **Στάθμευση**



Περιορισμοί:

- Ο δρόμος προς τους χώρους στάθμευσης είναι οι μαύρες γραμμές
- Εφόσον τα αυτοκίνητα της εταιρίας είναι 12 και οι θέσεις στους χώρους στάθμευσης είναι επίσης 12, όταν ένα αυτοκίνητο φθάσει στο εργοστάσιο, θα υπάρχει πάντα μια τουλάχιστον θέση για στάθμευση.
- Αν υπάρχουν θέσεις στάθμευσης και στους 2 χώρους (Ρ1 και Ρ2) τότε το όχημα θα κατευθυνθεί στο χώρο Π1. Αν ο χώρος Ρ1 είναι κατειλημμένος, δηλ. δεν υπάρχει καμία κενή θέση σε αυτόν, τότε θα υπάρχει ειδική σήμανση πριν τη διακλάδωση του δρόμου ώστε το όχημα να κατευθυνθεί προς το χώρο Ρ2 (κάτι σαν φανάρι)

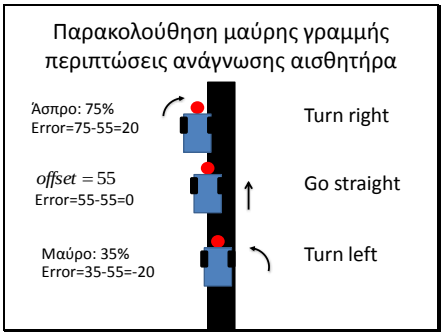
Τώρα θα αναλύσουμε τα βήματα που πρέπει να εκτελέσει το ρομποτικό σύστημα προκειμένου να παρκάρει το φορτηγό:

Αρχικά, ακολουθεί τη μαύρη γραμμή, δηλαδή το δρόμο προς το εργοστάσιο και ακολούθως επιλέγει αν θα κατευθυνθεί στο $1^{ο}$ ή το $2^{ο}$ παρκινγκ. Έπειτα, πάντα ακολουθώντας τη γραμμή, φτάνει στο χώρο στάθμευσης. Στη συνέχεια, ψάχνει και εντοπίζει την κενή θέση. Τέλος, σταθμεύει σε αυτή και το πρόγραμμα τελειώνει μόλις ολοκληρωθεί η διαδικασία στάθμευσης.

Για την παρακολούθηση της μαύρης γραμμής, το ρομπότ χρησιμοποιεί έναν αισθητήρα φωτός (light sensor). Ο αισθητήρας αυτός διαβάζει το πόσο φωτεινό ή σκοτεινό είναι ένα σώμα κατατάσσοντάς το σε μαύρο ή άσπρο.

Το άσπρο αντιστοιχεί σε 100% φωτεινότητα ενώ το μαύρο σε 0% φωτεινότητα. Ωστόσο, ανάλογα με τις συνθήκες που επικρατούν σε ένα χώρο, ο αισθητήρας μπορεί να αντιλαμβάνεται το άσπρο και το μαύρο σε διαφορετικά ποσοστά (π.χ. άσπρο 80%, μαύρο 15%). Για αυτό το λόγο, υπάρχει δυνατότητα ρύθμισης, ώστε από μία τιμή και πάνω το σώμα να θεωρείται άσπρο και από την ίδια τιμή και κάτω μαύρο. Έτσι, με βάση τις συνθήκες που επικρατούν στο δικό μας χώρο, ορίσαμε να διαβάζεται ως άσπρο οποιαδήποτε τιμή μεγαλύτερη από 50, ενώ από 50 και κάτω να διαβάζεται ως μαύρο.

Ουσιαστικά, το ρομπότ μας ακολουθεί το όριο μεταξύ της μαύρης γραμμής και της άσπρης επιφάνειας, για τις οποίες διαβάζει 35% και 75% αντίστοιχα.
Οπότε, προκειμένου να ακολουθήσει τη σωστή διαδρομή πρέπει να προχωράει πάνω στο όριο της γραμμής. Πάνω από αυτό, ο light sensor έχει τιμή ίση με τον μέσο όρο (offset) των τιμών του μαύρου και του άσπρου.



Στο σχήμα φαίνεται η περιοχή από την οποία ο light sensor διαβάζει το χρώμα: κόκκινο κυκλάκι

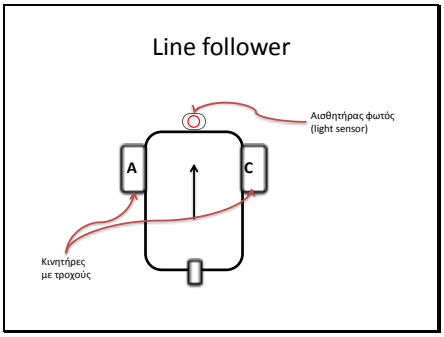
Έτσι, καθώς προχωράει, το ρομπότ μπορεί να βρεθεί σε διαφορετικές θέσεις, άρα υπάρχουν διαφορετικές τιμές που μπορεί να διαβάσει ο αισθητήρας: από πλήρες μαύρο (35) μέχρι πλήρες λευκό (75). Δηλαδή, το εύρος των τιμών που διαβάζει ο light sensor είναι [35..70].

Όταν ο αισθητήρας βρίσκεται ακριβώς πάνω από το όριο της γραμμής τότε διαβάζει το μέσο όρο δηλ. (75+35)/2=55 (offset). Στη θέση αυτή, το ρομπότ πρέπει να προχωρήσει ευθεία.

Κατά τη διάρκεια της κίνησης, όσο απομακρύνεται από την θέση αυτή, η τιμή που διαβάζει ο αισθητήρας αυξάνεται (αν απομακρύνεται προς το λευκό) ή μειώνεται (αν απομακρύνεται προς το μαύρο). Η διαφορά της τιμής που διαβάζει ο αισθητήρας από την επιθυμητή τιμή (offset), το σφάλμα δηλαδή, πρέπει να διορθωθεί με εντολή αντίστοιχης στροφής, δεξιάς ή αριστερής ανάλογα.

Έτσι, όσο πιο πολύ απομακρύνεται το ρομπότ από το όριο της γραμμής, τόσο μεγαλύτερο είναι το σφάλμα (error). Κάθε φορά, το σύστημα αφαιρεί το offset από την τιμή του light sensor , έτσι ώστε να βρει πόσο είναι το σφάλμα, να καταλάβει που βρίσκεται και να διορθώσει την πορεία

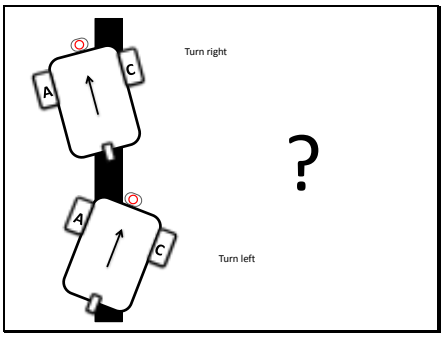
του. Έτσι, στην πρώτη περίπτωση, όπου βλέπει άσπρο (75%), το error είναι θετικός αριθμός. Σε αυτήν την περίπτωση, θα στρίψει δεξιά, για να διορθώσει την απόκλιση από το όριο της γραμμής. Στη δεύτερη περίπτωση, όπου βλέπει 55, το error είναι 0, άρα βρίσκεται στη σωστή πορεία και θα συνεχίσει ευθεία. Τέλος, στην τρίτη περίπτωση, όπου βλέπει μαύρο (35%), το error είναι αρνητικός αριθμός. Επομένως, θα στρίψει αριστερά, για να βρεθεί και πάλι στο όριο της γραμμής.



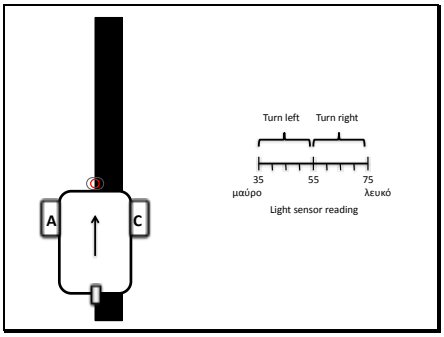
Ας δούμε πιο αναλυτικά τις παραπάνω σκέψεις…

Ένα όχημα – ρομπότ πρέπει να κινείται ώστε να ακολουθεί μια μαύρη γραμμή

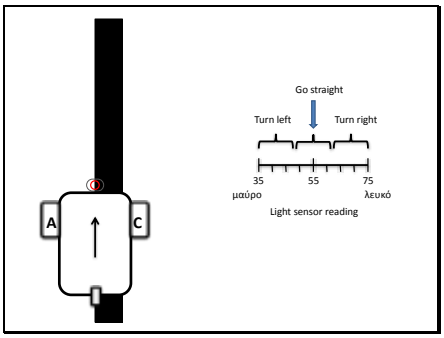
Το όχημα διαθέτει **ένα** αισθητήρα φωτός (light sensor) και **δύο** κινητήρες ανεξάρτητους (A,C)



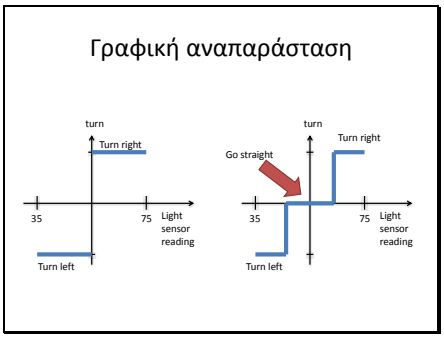
Όταν ο αισθητήρας φωτός βρίσκεται πάνω από τη μαύρη γραμμή τότε είναι λογικό το ρομπότ να κινείται ευθεία προς τα μπρος, δηλαδή και οι δυο κινητήρες να λειτουργούν με την ίδια ισχύ (power).
Αν όμως για κάποιο λόγο ο αισθητήρας βρεθεί πάνω από το λευκό δάπεδο, δηλαδή αν το ρομπότ έχει ξεφύγει από την πορεία του, πρέπει να διορθωθεί με στροφή προς τη μαύρη γραμμή. Επειδή δεν μπορεί να ανιχνευθεί αν το ρομπότ έχει ξεφύγει προς τα αριστερά ή προς τα δεξιά, δεν είναι δυνατόν να γνωρίζουμε προς τα πού πρέπει να στρίψει…



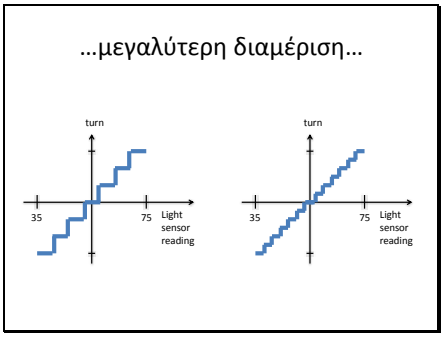
Η πιο κλασική προσέγγιση στο πρόβλημα αυτό είναι να προγραμματίσουμε το ρομπότ ώστε όταν ό αισθητήρας βλέπει μαύρο να στρίβει προς τα αριστερά (left) – οπότε θα είμαστε σίγουροι ότι κατευθύνεται προς το άσπρο δάπεδο από αριστερά της μαύρης γραμμής – και όταν βλέπει άσπρο, να στρίβει προς τα δεξιά ώστε να επανέλθει.
Έτσι, θα έχουμε μια λύση στο πρόβλημα μας, αλλά με αυτή τη λύση θα ακολουθεί τελικά το όριο της μαύρης γραμμής και θα το προσεγγίζει με μια πριονωτή γραμμή (ζικ-ζακ), χωρίς να πηγαίνει ποτέ ευθεία.
Με αυτόν τον τρόπο το ρομπότ κινείται ακολουθώντας προσεγγιστικά το όριο της γραμμής αλλά πάρα πολύ αργά.



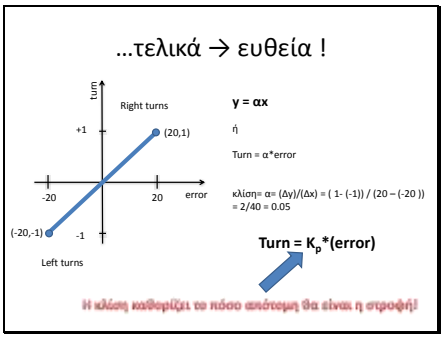
Μια βελτιωμένη προσέγγιση είναι η εξής:
Όταν ο αισθητήρας διαβάζει μια τιμή κοντά στο μέσο όρο της τιμής του μαύρου και του λευκού [(35+75)/2=55], που σημαίνει ότι βρίσκεται κοντά στο όριο τότε προγραμματίζουμε το ρομπότ να κινείται ευθεία, ενώ όταν είναι πιο μακριά από αυτό, δηλ ο αισθητήρας διαβάζει τιμές κοντά στο 35 ή το 75 να κάνει στροφή προς τα αριστερά ή δεξιά. Αυτό μας δίνει μια πιο ομαλή κίνηση ιδιαίτερα αν η γραμμή μας είναι ευθεία ή καμπύλη αλλά όχι με απότομες στροφές.



Μπορούμε να αναπαραστήσουμε γραφικά τις σκέψεις μας:
Στον οριζόντιο άξονα (χ) είναι οι τιμές του αισθητήρα φωτός (light sensor) και στον κατακόρυφο (ψ), ας πούμε η περιστροφή.
Στο αριστερό διάγραμμα είναι η πρώτη περίπτωση (ζικ-ζακ) ενώ στο δεξί είναι η βελτιωμένη στην οποία έχουμε χωρίσει τις τιμές που διαβάζει ο αισθητήρας σε 3 διαστήματα.



Γιατί να μη χωρίσουμε τις τιμές που διαβάζει ο αισθητήρας σε περισσότερα διαστήματα έτσι ώστε η «ταλάντωση» γύρω από το όριο της γραμμής να είναι πιο «ομαλή»;
Επεκτείνοντας τον προηγούμενο συλλογισμό μας αναρωτιόμαστε αν χωρίζουμε σε πάρα πολύ μικρά διαστήματα τελικά που θα καταλήξουμε;



...μα τελικά σε μια **ευθεία** (πιο σωστά σε ένα ευθύγραμμο τμήμα)!

Στο διάγραμμα, στον κατακόρυφο άξονα βάλαμε (αυθαίρετα) μέγιστη και ελάχιστη τιμή +1 και -1 που αντιστοιχούν στις μέγιστες τιμές της δεξιάς και αριστερής στροφής.
Επίσης, έχουμε μετατρέψει τις τιμές στον οριζόντιο άξονα, τις τιμές που διαβάζει ο αισθητήρας στο μέγεθος «σφάλμα» (error). Απλά έχουμε κάνει μια μετατόπιση της αρχής των αξόνων στο μέσο όρο των ακραίων τιμών του αισθητήρα.
Θυμάστε ήταν 35 για το μαύρο και 75 για το άσπρο. Δηλ. (35+75)/2=55.
Έτσι, 35-55=-20 και 75-55=20.

- Όταν το σφάλμα είναι 0 σημαίνει ότι το ρομπότ κινείται ακριβώς πάνω στο όριο της μαύρης γραμμής, οπότε δεν χρειάζεται καμία διόρθωση στην πορεία του. Προχωράει ευθεία.
- Αν το σφάλμα είναι αρνητικό ο αισθητήρας «διαβάζει»

περισσότερο μαύρο, άρα το ρομπότ πρέπει να στρίψει αριστερά και μάλιστα όσο περισσότερο η τιμή προσεγγίζει προς το -20 τόσο πιο «απότομη» πρέπει να είναι η στροφή έτσι ώστε να διορθωθεί το σφάλμα.

•Αν το σφάλμα είναι θετικό ο αισθητήρας «διαβάζει» περισσότερο άσπρο, άρα το ρομπότ πρέπει να στρίψει δεξιά και μάλιστα όσο περισσότερο η τιμή προσεγγίζει προς το +20 τόσο πιο «απότομη» πρέπει να είναι η στροφή έτσι ώστε να διορθωθεί το σφάλμα.

Η εξίσωση της ευθείας είναι:

y = αx

ή (με την ορολογία μας)

Turn = α*error

Η κλίση είναι: α= (Δy)/(Δx) = (1- (-1)) / (20 – (-20)) = 2/40 = 0.05

Η κλίση καθορίζει το πόσο απότομη θα είναι η στροφή!

Για λόγους ορολογίας θα γράψουμε την εξίσωση:

Turn = K_p*(error)

Δεδομένα και υπολογισμοί

offset=[lightvalue(white)-lightvalue(black)]/2

error=lightvalue-offset

turn=k_p·error

powerA=T_p+turn

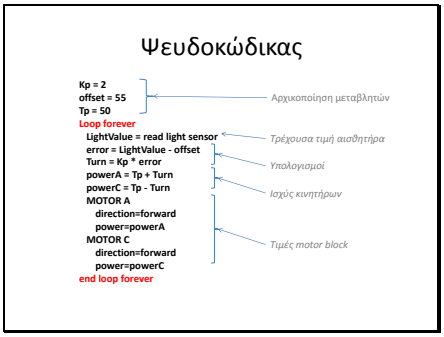
powerC=T_p-turn

Συνοψίζοντας τα προηγούμενα μπορούμε να πούμε:

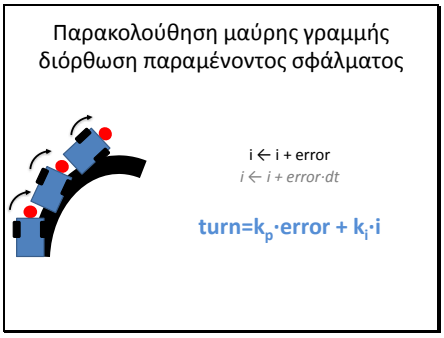
- Αρχικά πρέπει να μετρήσουμε τις τιμές του αισθητήρα φωτός (light sensor) για το μαύρο και το άσπρο χρώμα. (στο προηγούμενο ήταν 35 και 55).
- Από αυτές υπολογίζουμε το μέσο όρο που τον ονομάζουμε **offset**, δηλαδή την τιμή που πρέπει να αφαιρέσουμε από την τιμή που διαβάζει ο αισθητήρας ώστε να τη μετατρέψουμε σε «σφάλμα»
- Χρειαζόμαστε επίσης μια μεταβλητή για την k_p. Η σταθερή αυτή (κλίση της ευθείας) ουσιαστικά μας δείχνει πόσο «απότομη» είναι η ευθεία, δηλ. πόσο έντονη (απότομη) είναι η στροφή για τη διόρθωση των σφαλμάτων. Όσο πιο μεγάλη είναι αυτή, τόσο πιο απότομη είναι η στροφή, πράγμα που σημαίνει ότι το ρομπότ «προσπαθεί» να προσεγγίσει το όριο της γραμμής, δηλ. να διορθώσει το σφάλμα, πολύ γρήγορα. Όμως, αν αυτή είναι πολύ μεγάλη, αυτό έχει σαν αποτέλεσμα να πλησιάζει το όριο της γραμμής γρήγορα μεν, με μεγάλη ταχύτητα δε, οπότε να περνάει προς το μαύρο αρκετά (overshoot). Η βέλτιστη τιμή της σταθεράς αυτής προκύπτει με τη μέθοδο της δοκιμής και λάθους (trial and error).
- Η περιστροφή του ρομπότ γίνεται με παρέμβαση στην ισχύ των κινητήρων: Αν ας πούμε ο αριστερός κινητήρας έχει μεγαλύτερη ισχύ από τον άλλο, τότε κινείται με μεγαλύτερη ταχύτητα, οπότε το ρομπότ περιστρέφεται προς τα δεξιά. Έτσι, θέτουμε: **powerA**=T_p+turn και **powerC**=T_p-turn. Αφού η μεταβλητή turn μπορεί να είναι είτε θετική είτε αρνητική, το ρομπότ μπορεί να περιστρέφεται είτε αριστερά είτε δεξιά με αυτές τις εξισώσεις.

- Επανερχόμαστε στη σταθερά K_p. Με ποια τιμή να ξεκινήσουμε τον πειραματισμό μας; Αρχικά μπορούμε να «πειραματιστούμε» με μια αρχική τιμή που προκύπτει από ένα σχετικά απλό συλλογισμό: Θα θέσουμε Tp (target power) 50, δηλ. όταν το error=0 και οι 2 κινητήρες θα λειτουργούν με ισχύ 50. Το error μεταβάλλεται από -20 μέχρι +20. Αν θέλουμε η ισχύς να μεταβάλλεται από 50 σε 90 δηλ. Κατά 40 μονάδες ή από 50 σε 10 (επίσης κατά 40 μονάδες) όταν το error μεταβάλλεται από 0 σε +20 ή από 0 σε -20 αντίστοιχα, υπολογίζουμε (θυμηθείτε την κλίση):

$$Kp = (0 - 40)/(-20 - 0) = 2.$$



Να ο ψευδοκώδικας για τα παραπάνω!!!



Η διόρθωση του σφάλματος, όπως περιγράφηκε προηγουμένως, αρκεί ώστε το ρομπότ να ακολουθεί τη σωστή πορεία μόνο όταν η γραμμή που πρέπει να ακολουθήσει είναι ευθεία ή μία στροφή είναι ομαλή. Αυτό συμβαίνει επειδή σε αυτές τις περιπτώσεις το σφάλμα είναι διαδοχικά θετικό και αρνητικό. Έτσι, κάθε φορά στρίβει διαδοχικά δεξιά και αριστερά αντίστοιχα. Με αυτόν τον τρόπο, μένει πάντα κοντά στο όριο της γραμμής.
Όμως, στην περίπτωση που υπάρχει μια απότομη στροφή, το ρομπότ λογικά θα βρεθεί σε μια πιο μακρινή απόσταση από ότι στις προηγούμενες περιπτώσεις. Γι' αυτό, η διόρθωση σφάλματός που θα πραγματοποιηθεί από τον παράγοντα k_p*error δεν θα είναι αρκετή για να το φέρει ξανά στο όριο της γραμμής, καθώς το σφάλμα θα είναι ξανά θετικό με τη μέγιστη τιμή, ίδια με την προηγούμενη, αλλά επειδή η στροφή είναι απότομη, η διόρθωση (turn) που προκαλεί δεν είναι τόσο μεγάλη ώστε να μπορέσει να επαναφέρει το ρομπότ στη γραμμή. Αυτό έχει σαν αποτέλεσμα, το ρομπότ να απομακρύνεται από τη μαύρη γραμμή και να παίρνει τη στροφή πολύ «ανοικτά». Για να αντιμετωπίσουμε αυτό το πρόβλημα χρειάζεται, όταν η στροφή είναι απότομη, να έχουμε μεγαλύτερη διόρθωση από αυτή που παράγεται από την παράγοντα k_p* error, αλλά μόνο σε αυτή την περίπτωση. Γι' αυτό«συμπληρώνουμε» την εξίσωση turn με τον 2ο παράγοντα ki* I. Εδώ το i είναι το άθροισμα των προηγούμενων σφαλμάτων και αυτό είναι που λειτουργεί

με τον τρόπο που περιγράφεται στην περίπτωση της απότομης στροφής.

Έτσι, πλέον η διόρθωση θα προσδιορίζεται από την εξίσωση $turn=k_p \cdot error + k_i \cdot i$ και όχι μόνο από την $turn=k_p * error$. Με αυτόν τον τρόπο, το ρομπότ θα επανέρχεται στην κανονική του πορεία.

Βέβαια, όταν υπάρχει ευθεία ή ομαλή στροφή, το i δεν μεγαλώνει αφού το σφάλμα είναι διαδοχικά θετικό-αρνητικό, άρα η διόρθωση καθορίζεται από τον πρώτο όρο της εξίσωσης ($k_p * error$). Αντίθετα, παίζει καθοριστικό ρόλο στην προσέγγιση μιας απότομης στροφής, αφού το ρομπότ στρίβει πιο απότομα και δεν ξεφεύγει από την πορεία του.
Επομένως, ουσιαστικά το i είναι ένα είδος «μνήμης» για τα παραμένοντα σφάλματα και βοηθάει το ρομπότ να «θυμάται» το σφάλμα και να πραγματοποιεί την απαραίτητη διόρθωση.

Από την άλλη πλευρά, για να είμαστε πιο ακριβείς πρέπει να λάβουμε υπόψη μας το χρόνο (dt) μεταξύ δύο διαδοχικών μετρήσεων του αισθητήρα (στο δικό μας πρόγραμμα το dt=0,004s). Αν λάβουμε υπόψη μας αυτήν την παράμετρο, στην πραγματικότητα το i υπολογίζεται έτσι: $i = i + error*(dT)$
Όμως, αν ξεκινήσουμε με i=0 τότε το dt βγαίνει κοινός παράγοντας. Γι' αυτό, χρησιμοποιούμε μια παράμετρο **Ki**, μέσα στην οποία «κρύβεται» αυτός ο χρόνος. Η **Ki** προκύπτει με τη μέθοδο δοκιμής και λάθους, δηλαδή τη δοκιμή διαφορετικών τιμών, ώστε μέσα από τα λάθη να βρεθεί η σωστή τιμή και να πετύχουμε ομαλή εξακολούθηση της γραμμής (επειδή ο χρόνος dT είναι μικρός πρέπει στις δοκιμές μας να ξεκινήσουμε με μικρή τιμή). Έτσι, οδηγούμαστε τελικά στην εξίσωση $turn=k_p \cdot error + k_i \cdot i$.

Τέλος, πρέπει να επισημάνουμε μία ακόμη λεπτομέρεια: Σε μία απότομη στροφή, όπως είναι φυσικό, το i θα πάρει μια μεγάλη τιμή, αφού θα μείνει για αρκετό χρόνο στο άσπρο. Συνεπώς, όταν το ρομπότ προσεγγίσει τη μαύρη γραμμή, προκειμένου να επανέλθει κοντά στο 0, θα πρέπει να παραμείνει για αντίστοιχο χρόνο στο μαύρο, ώστε να εξισορροπηθεί η κατάσταση. Όμως, αυτή η παραμονή στο μαύρο μπορεί να προκαλέσει προσπέραση του ορίου της μαύρης γραμμής (overshoot), και πιθανόν και προσπέραση ολόκληρης της γραμμής. Στην πρώτη περίπτωση το ρομπότ θα καθυστερήσει, ενώ στη δεύτερη θα βγει εντελώς εκτός πορείας, αφού θα ξαναδεί άσπρο από την άλλη μεριά της γραμμής.

Για να μην συμβεί αυτό, κάθε φορά που αλλάζει το πρόσημο του error ή το $i*error<0$, μηδενίζουμε το i. Με αυτόν τον τρόπο σταματάει να επηρεάζει σε τόσο μεγάλο βαθμό την εξίσωση που μας δίνει την τιμή του turn, έως ότου ξαναχρειαστεί να την επηρεάσει σημαντικά.

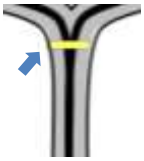
Παρακολούθηση μαύρης γραμμής ψευδοκώδικας

```
Kp ← 1
Ki ← 0,01
offset ← 55                          ! Αρχικοποίηση μεταβλητών
Tp ← 50
i ← 0

Loop forever
  LightValue ←read light sensor      ! Ανάγνωση αισθητήρα
  error ← LightValue - offset        ! Υπολογισμός σφάλματος
  i ← i+ error                       ! Υπολογισμός παραμένοντος σφάλματος
  Turn ← Kp*error + Ki*I             ! Υπολογισμος διόρθωσης
  powerA ← Tp + Turn                 ! Υπολογισμός τιμών ισχύος κινητήρων
  powerC ← Tp - Turn
  motorA  direction=forward, power=powerA
  motorC  direction=forward, power=powerC
end loop forever
```

Ουσιαστικά, για να παρακολουθήσει το ρομπότ τη μαύρη γραμμή, μετατρέπουμε αυτό που διαβάζει ο αισθητήρας σε σφάλμα και στη συνέχεια το σφάλμα σε διόρθωση στην ισχύ των κινητήρων. Αυτή η μετατροπή φαίνεται στον ψευδοκώδικα.

Επιλογή χώρου στάθμευσης

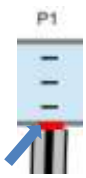


```
Kp ← 1
Ki ← 0,01
offset ← 55
Tp ← 50
i ← 0

Loop forever
  LightValue ←read light sensor
  error ← LightValue - offset
  i ← i+ error
  Turn ← Kp*error + Ki*I
  powerA ← Tp + Turn
  powerC ← Tp - Turn
  If colorsensorValue=Yellow then
    move forward 1rotation
    motorA direction=forward, power=35
  endif
    motorA  direction=forward, power=powerA
    motorC  direction=forward, power=powerC
end loop forever
```

Όταν το ρομπότ φτάσει στη διακλάδωση, πρέπει να διαλέξει προς ποιο από τα δύο παρκινγκ θα κατευθυνθεί, με βάση το ποιο έχει κενή θέση. Η επιλογή του θα καθορίζεται από την ύπαρξη ή απουσία μιας ειδικής σήμανσης. Αυτή την ύπαρξη ή απουσία της σήμανσης την αντιλαμβάνεται ο color sensor (αισθητήρας που διαβάζει τα χρώματα). Συγκεκριμένα, θα κατευθύνεται στο παρκινγκ 1 (αριστερά) όταν δεν υπάρχει η σήμανση, ενώ στο παρκινγκ 2 (δεξιά) θα πηγαίνει όταν υπάρχει η σήμανση. Στην πρώτη περίπτωση, η απουσία της σήμανσης δηλώνει ότι υπάρχει κενή θέση στο $1^{ο}$ παρκινγκ. Αντίστοιχα, η παρουσία της στη δεύτερη περίπτωση δηλώνει ότι υπάρχει κενή θέση στο $2^{ο}$ παρκινγκ, ενώ ταυτόχρονα δεν υπάρχει κενή θέση στο $1^{ο}$. Επομένως, κατευθύνεται στο $2^{ο}$. Τα κόκκινα γράμματα αντιστοιχούν στο κομμάτι του προγράμματος που σχετίζεται με την επιλογή του χώρου στάθμευσης.

Άφιξη στο χώρο στάθμευσης




```
Kp ← 1
Ki ← 0,01
offset ← 55
Tp ← 50
i ← 0

Loop until colorsensorValue=Red
  LightValue ←read light sensor
  error ← LightValue - offset
  i ← i+ error
  Turn ← Kp*error + Ki*I
  powerA ← Tp + Turn
  powerC ← Tp - Turn
  If colorsensorValue=Yellow then
    move forward 1rotation
    motorA direction=forward, power=35
  endif
  motorA  direction=forward, power=powerA
  motorC  direction=forward, power=powerC
end loop
```

Όταν το ρομπότ φτάσει μπροστά σε ένα από τα δύο παρκινγκ, πρέπει με κάποιο τρόπο να το αντιληφθεί, ώστε στη συνέχεια να αρχίσει τη διαδικασία της αναζήτησης θέσης. Προκειμένου να γίνει αυτό στην είσοδο κάθε παρκινγκ έχουμε τοποθετήσει μια κόκκινη σήμανση(κόκκινη ταινία). Την ακολουθία της γραμμής την επαναλαμβάνει μέχρι ο color sensor να δει την κόκκινη σήμανση. Μόλις γίνει αυτό, το ρομπότ σταματάει και το μέρος του προγράμματος που σχετίζεται με την παρακολούθηση της μαύρης γραμμής. Τα κόκκινα γράμματα σχετίζονται με την διάρκεια της επανάληψης αυτού του μέρους του προγράμματος.

Παρή Κυριακίδη
Βασίλης Σακελλαρίου

Ευχαριστούμε τους συμμαθητές μας:
Σιμόν Φιλ
Φοίβο Γκούμα
για τη βοήθειά τους στη δημιουργία του προγράμματος