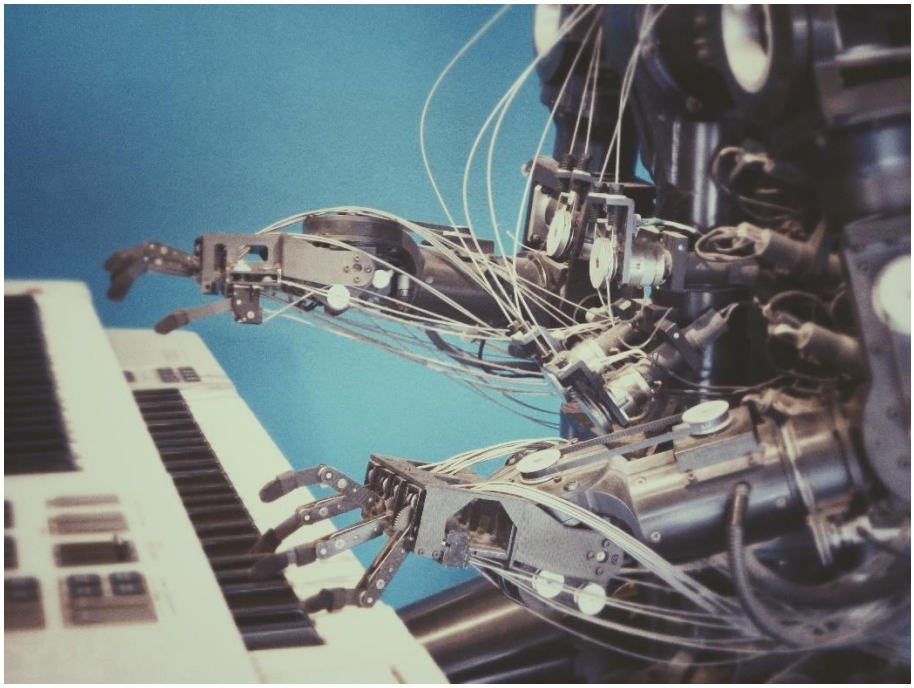
Πανεπιστήμιο Πειραιώς
ΠΜΣ Πληροφορική 2018 - 2019

Ζιώβας Κώστας
Τσατταλιός Άρης

Τελική Εργασία

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ – ΕΜΠΕΙΡΑ ΣΥΣΤΗΜΑΤΑ



Πειραιάς 2019

I. Σκοπός

Σκοπός της συγκεκριμένης εργασίας είναι η υλοποίηση (κατασκευή και προγραμματισμός) ενός ρομπότ, το οποίο θα επιλύει το «πρόβλημα του Λαβυρίνθου» (maze solving), δηλαδή της εύρεσης της βέλτιστης διαδρομής ώστε να «βγει» από τον λαβύρινθο (optimal path), ακολουθώντας μια γραμμή (line following) η οποία έχει σημεία με διασταυρώσεις ή και αδιέξοδα.

Παρακάτω θα παρουσιαστεί το θεωρητικό υπόβαθρο πάνω στο οποίο βασίζεται η παρούσα εργασία, πώς τελικά υλοποιήθηκε, τι προβλήματα παρουσιάστηκαν κατά την προσπάθεια εφαρμογής του θεωρητικού μοντέλου, και πως αυτά αντιμετωπίστηκαν.

II. Θεωρητική Εισαγωγή

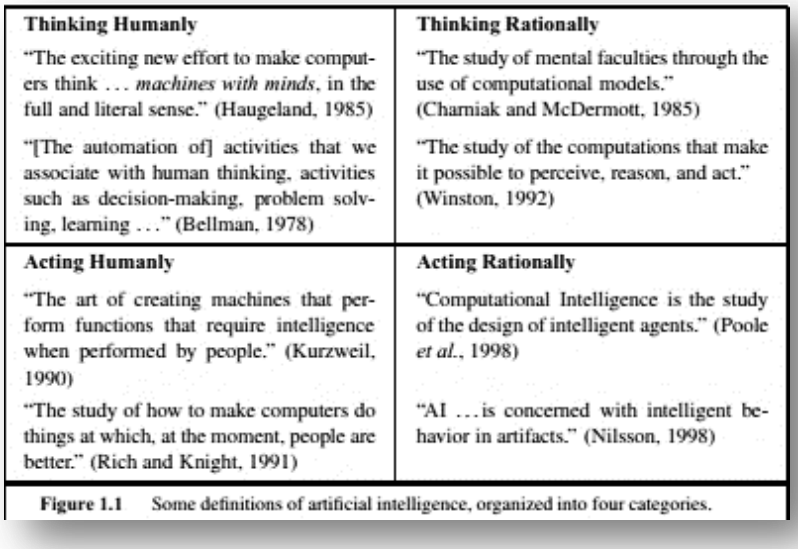
• Τεχνητή Νοημοσύνη

Σύμφωνα με την εγκυκλοπαίδεια Britannica **(Copeland, 2019)**, η τεχνητή νοημοσύνη ορίζεται ως:

«…η ικανότητα ενός Η/Υ ή ενός ρομπότ το οποίο ελέγχεται από έναν Η/Υ, να πραγματοποιήσει εργασίες οι οποίες σχετίζονται με ευφυή όντα.»

Ο Stuart Russel στο βιβλίο του *«Artificial Intelligence: A Modern Approach»* **(Russell & Norvig, 2016)**, κατά την παρουσίαση του ορισμού της *τεχνητής νοημοσύνης* αναφέρει τις (4) διαφορετικές προσεγγίσεις[1] ώστε να επιτευχθεί ένας σαφής - αυστηρός ορισμός. Επίσης εισάγει τις έννοιες του "**Machine Learning**" και του "**Agent**", οι οποίες σχετίζονται με την παρούσα εργασία.

[1] a) Οι 4 προσεγγίσεις είναι: The Turing Test approach, b) The cognitive modeling approach, c) The "laws of thought" approach, d) The rational agent approach



Thinking Humanly	Thinking Rationally
"The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)	"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al.*, 1998) "AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

Figure 1.1 Some definitions of artificial intelligence, organized into four categories.

Εικόνα 1: Οι 4 διαφορετικές προσεγγίσεις για τον ορισμό της Τεχνητής Νοημοσύνης **(Russell & Norvig, 2016)**

- # Μηχανική Μάθηση (Machine Learning)

Ορισμός

«Η δημιουργία μοντέλων[2] ή προτύπων από ένα σύνολο δεδομένων, από ένα υπολογιστικό σύστημα, ονομάζεται ***μηχανική μάθηση.»*** (Βλαχάβας κ.ά.)

Είδη

- ### Μάθηση με επίβλεψη (supervised learning)

  Εδώ το σύστημα (στην περίπτωσή μας το ρομπότ), καλείται να «μάθει» μια έννοια ή συνάρτηση από ένα συνολο δεδομένων, η οποία αποτελεί περιγραφή ενός μοντέλου. (Βλαχάβας & κ.ά.)

- ### Μάθηση χωρίς επίβλεψη (unsupervised learning)

  Το σύστημα σε αυτή την περίπτωση, πρέπει μόνο του να ανακαλύψει συσχετίσεις ή ομάδες σε ένα σύνολο δεδομένων, δημιουργώντας

[2] **Μοντέλο,** μια αφαιρετική εκδοχή του περιβάλλοντος, που δημιουργεί ο άνθρωπος με σκοπό να το κατανοήσει. (Βλαχάβας κ.ά.)

πρότυπα, χωρίς να είναι γνωστό αν υπάρχουν, πόσα και ποια είναι. (Βλαχάβας & κ.ά.)

- ## Ευφυείς Πράκτορες (Intelligent Agents)

Γενικά, **πράκτορας** ονομάζεται οτιδήποτε το οποίο παρατηρεί το περιβάλλον του μέσω αισθητήρων (sensors) και (αντι)δρά με αυτό μέσω μηχανισμών ενεργοποίησης (actuators) (πχ. Ο άνθρωπος, το ρομπότ ή ακόμα και ένα λογισμικό). Διέπεται από 3 βασικά χαρακτηριστικά: α) την ***αντιληπτική ακολουθία*** *(percept sequence)*, δηλαδή το πλήρες ιστορικό του τι έχει αντιληφθεί, β) την ***συνάρτηση πράκτορα*** (agent function), που αντιστοιχεί κάθε «στοιχείο» που έχει αντιληφθεί ο πράκτορας σε μια δράση, και γ) το ***πρόγραμμα του πράκτορα*** (agent program), το οποίο αντιστοιχεί στην απτή υλοποίηση (σε κώδικα) της αφαιρετικής μαθηματικής έννοιας "*agent function*". (Russell & Norvig, 2016)

Ειδικότερα ο ευφυής πράκτορας, ή σύμφωνα με τους (Russell & Norvig, 2016) ο «λογικός πράκτορας» (rational agent), είναι αυτός που όχι μόνο (αντι)δρά σε σχέση με το περιβάλλον του, αλλά και σωστά, πράγμα το οποίο καθορίζεται από το **ποσοστό απόδοσης** (performance measure) το οποίο εκτιμά κάθε κομμάτι της ***αντιληπτικής ακολουθίας*** *(percept sequence)*. (Russell & Norvig, 2016)

«Για κάθε πιθανή αντιληπτική ακολουθία, ένας λογικός πράκτορας πρέπει να επιλέξει μια δράση η οποία αναμένεται να μεγιστοποιήσει το ποσοστό απόδοσής του, δεδομένης της αντιληπτικής ακολουθίας και της όποιας γνώσης έχει ήδη στοιχειοθετήσει ο πράκτορας.» (Russell & Norvig, 2016)

Με απλά λόγια, ένας ευφυής πράκτορας είναι μια αυτόνομη οντότητα η οποία δρα μέσα σε ένα περιβάλλον, με σκοπό να πετύχει έναν στόχο, χρησιμοποιώντας σένσορες και μηχανισμούς ενεργοποίησης – κίνησης, αλλά και τυχόν προηγούμενη γνώση την οποία έχει «μάθει».

- ## Το πρόβλημα της επίλυσης του Λαβυρίνθου (maze solving)

Στον χώρο των κινούμενων ρομπότ, το πρόβλημα του Λαβυρίνθου είναι ένα από τα πιο κοινά προβλήματα. Οι λαβύρινθοι μπορεί να είναι διαφόρων ειδών, με ή χωρίς επαναληπτικές διαδρομές (loops) ή διαγραμμίσεις. Επίσης υπάρχουν αρκετοί διαφορετικοί αλγόριθμοι επίλυσης τέτοιων λαβυρίνθων, σχεδιασμένοι ώστε να είτε να χρησιμοποιηθούν μέσα στον λαβύρινθο από το ρομπότ, χωρίς προηγούμενη γνώση του λαβυρίνθου (πχ. Wall Follower, Random Mouse), είτε εκτός αυτού, από ρομπότ ή λογισμικό το οποίο μπορεί να δει αμέσως όλο τον λαβύρινθο (πχ. Dead-End filling, Shortest Path). (Shadman & et.al) (wikipedia, n.d.)

- **Ο αλγόριθμος Wall Follower ή "Left/Right Hand Rule"**

  Ο αλγόριθμος "Wall Follower" χρησιμοποιείται σε απλά συνδεδεμένους[3] λαβυρίνθους, ως εξής: κρατώντας με το ένα χέρι επαφή με έναν τοίχο του λαβυρίνθου, ο λύτης (το ρομπότ) εξασφαλίζει το ότι δεν θα χαθεί και θα φτάσει στην έξοδο, αν αυτή υπάρχει, ή στην είσοδο του λαβυρίνθου, έχοντας διανύσει κάθε διαδρομή τουλάχιστον μία φορά. (wikipedia, n.d.)
  Το «κλειδί» στον αλγόριθμο "Wall Follower", είναι η αναγνώριση των διασταυρώσεων, πράγμα που ορίζει και την απόφαση – ενέργεια που θα επιλεχθεί. Πιο συγκεκριμένα υπάρχουν 8 διαφορετικά είδη διασταυρώσεων και αντίστοιχες καταστάσεις στις οποίες πρέπει να ληφθεί μια απόφαση:

  1. **Σχήματος Σταυρού (Cross)** = Προχωράω αριστερά, δεξιά ή ευθεία
  2. **Διασταύρωση σχήματος «Τ»** = Προχωράω αριστερά ή δεξια
  3. **«Δεξιά Μόνο»** = Προχωράω μόνο δεξιά
  4. **«Αριστερά Μόνο»** = Προχωράω μόνο αριστερά
  5. **«Δεξιά ή ευθεία»** = Προχωράω δεξιά ή ευθεία
  6. **«Αριστερά ή ευθεία»** = Προχωράω αριστερά ή ευθεία
  7. **«Αδιέξοδο»** = Γυρίζω πίσω
  8. **Τέλος του Λαβυρίνθου** = Σταματώ

  Όμως με βάση τον κανόνα του συγκεκριμένου αλγορίθμου, προτεραιότητα έχει **πρώτα** η αριστερή στροφή, **έπειτα** η

[3] **Απλά συνδεδεμένος ή τέλειος λαβύρινθος**, αυτός που όλοι οι τοίχοι του είναι συνδεδεμένοι είτε μεταξύ τους, είτε στο εξωτερικό περίβλημα. Αντιστοιχεί στο γράφημα δένδρου(tree graph).

κίνηση ευθεία και **τέλος** η δεξιά στροφή. Έτσι οι αποφάσεις που θα ληφθούν στις παραπάνω πιθανές περιπτώσεις διασταυρώσεων είναι οι εξής:

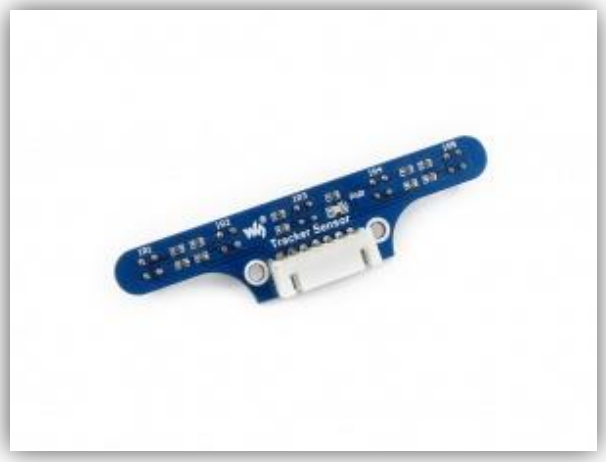
1. **Σχήματος Σταυρού (Cross) =** προχωράω αριστερά
2. **Διασταύρωση σχήματος «Τ» =** Προχωράω αριστερά
3. **«Δεξιά Μόνο» =** Προχωράω δεξιά
4. **«Αριστερά Μόνο» =** Προχωράω αριστερά
5. **«Δεξιά ή ευθεία» =** Προχωράω ευθεία
6. **«Αριστερά ή ευθεία» =** Προχωράω αριστερά
7. **«Αδιέξοδο» =** Γυρίζω πίσω
8. **Τέλος του Λαβυρίνθου =** Σταματώ

- **«Προχώρα μερικά εκατοστά…!»**

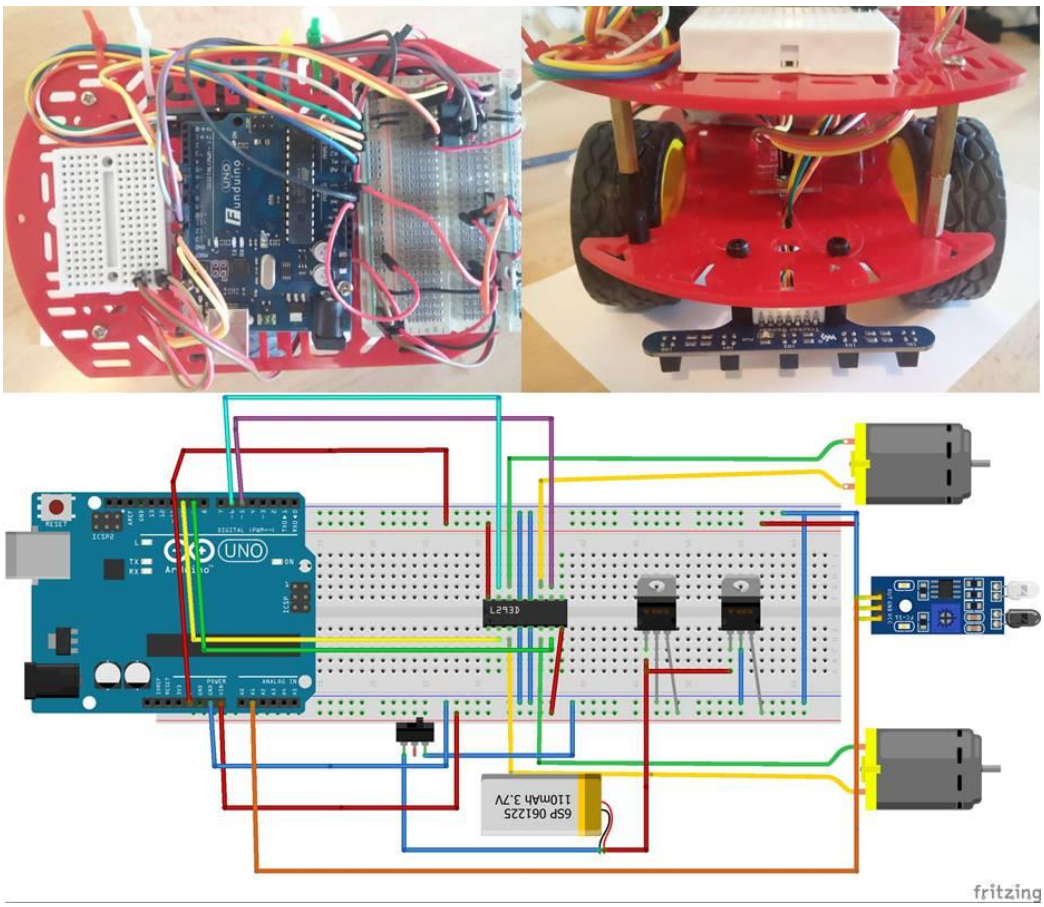
  Όταν το ρομπότ συναντά αδιέξοδο ή το τέλος του λαβυρίνθου, δεν δυσκολεύεται να τα αναγνωρίσει. Το πρόβλημα υπάρχει όταν το συναντήσει μια κάθετη γραμμή, που μπορεί να είναι διασταυρώσεις σχήματος «σταυρού», «Τ» ή στροφή (άρα οι περιπτώσεις 3 ή 4 ή 5 ή 6). Εκεί, για να μπορέσει το ρομπότ να πάρει την κατάλληλη απόφαση, πρέπει να προχωρήσει μερικά ακόμα εκατοστά, έτσι ώστε να δει τι επακολουθεί.

III. Πρακτικό Μέρος

Τμήματα του Ρομπότ	
1x	li-ion battery (11.1V)
1x	Arduino Uno
1x	Infrared Line sensors (5x) – Waveshare Track Sensor
2x	6-12-volt DC Gearmotor
2x	LM7809CT Regulators 1A (9V Constant voltage)
1x	ON/OFF Switch
2x	Breadboard



Εικόνα 2: Waveshare Tracking Sensor



Εικόνα 3: Φωτογραφίες ρομπότ και διάγραμμα εξαρτημάτων και συνδέσεων.

i. Τρόπος Λειτουργίας Αισθητήρων

Κατά την διαδικασία του “Line Following”, χρησιμοποιούνται 5 αισθητήρες. Οι 3 από αυτούς για τον έλεγχο του PID[4] και οι 2 ακριανοί για την αναγνώριση των Αριστερών (LEFT) και δεξιών (RIGHT) διασταυρώσεων. Έχοντας 5 σένσορες μας δίνεται η δυνατότητα να έχουμε την «μεταβλητή λάθους» (error variable), η οποία μας βοηθάει στο να ξέρουμε αν το ρομπότ ακολουθεί την γραμμή, βρίσκεται σε διασταύρωση ή τελείως εκτός γραμμής.

Ο σένσορας υπέρυθρων αποτελείται από μια φωτοδίοδο και ένα LED λαμπάκι. Όταν αυτός βρίσκεται πάνω στην (μαύρη) γραμμή, το φως από το λαμπάκι δεν ανακλάται πίσω στην δίοδο (απορροφάται από το μαύρο χρώμα)(χαμηλή τιμή εξόδου < 50mV), ενώ όταν έχουμε απόκλιση από την γραμμή, άρα ο σένσορας βρίσκεται σε λευκή περιοχή, έχουμε υψηλή τάση στην έξοδο της διόδου, ανάλογη με την ένταση της ανάκλασης, που αντιστοιχεί σε ένδειξη > 50mV. Όπως θα αναλυθεί και παρακάτω, οι σένσορες που χρησιμοποιήθηκαν δίνουν αναλογικές τιμές εξόδου, διαμορφωμένες σε κλίμακα από 0 έως 100. Έτσι μας δίνεται η δυνατότητα πλήρους ελέγχου της κίνησης του ρομπότ, μέσω του PID controller.

ii. Τρόπος Λειτουργίας PID controller

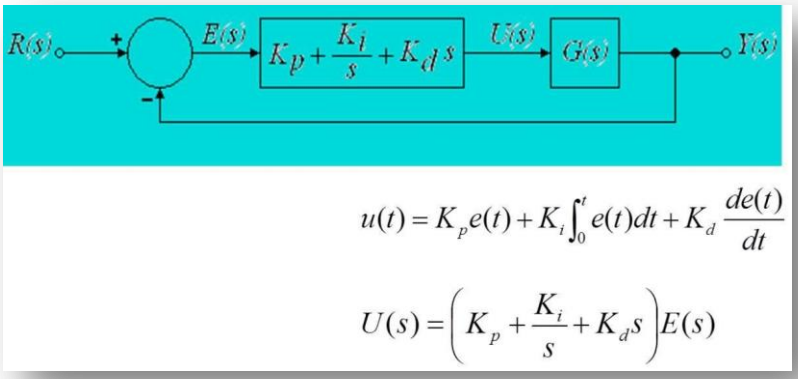
Το PID controller, όπως είπαμε χρησιμοποιείται για τον καλύτερο έλεγχο της κίνησης του ρομπότ.

<u>Αρχή Λειτουργίας</u>

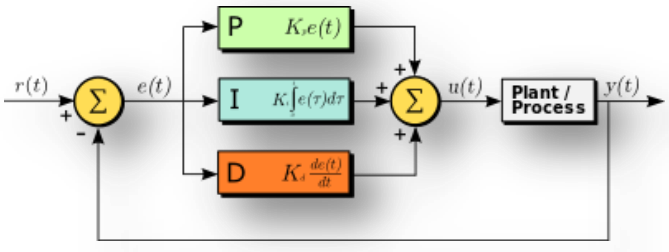
Ένας ελεγκτής PID υπολογίζει συνεχώς μια τιμή σφάλματος e(t) ως τη διαφορά μεταξύ μιας επιθυμητής τιμής και μιας μεταβλητής μετρούμενων διαδικασιών και εφαρμόζει μια διόρθωση βασιζόμενη σε αναλογικούς, ολοκληρωτικούς και διαφορικούς όρους (P, I και D) που δίνουν το όνομα τους στον τύπο του ελεγκτή. Ο ελεγκτής επιχειρεί να ελαχιστοποιήσει το σφάλμα με την πάροδο του χρόνου ρυθμίζοντας μια μεταβλητή ελέγχου u(t).

---

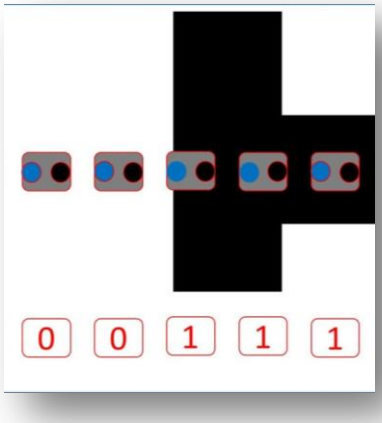
[4] **PID** (Proportional-Integral-Derivative) **Controller,** είναι ένας ελεγκτής ο οποίος διορθώνει αυτόματα την πορεία του ρομπότ, έτσι ώστε να βρίσκεται πάντα πάνω στην γραμμή (line following), και με την κατάλληλη ρύθμιση, να την ακολουθεί με όσο τον δυνατόν πιο ομαλή κίνηση. Αυτό επιτυγχάνεται μέσω της συνεχούς συσχέτισης (μέσω αναλογίας, ολοκλήρωσης ή παραγώγου) προκαθορισμένων μεταβλητών (error value, setpoint value, process variable).



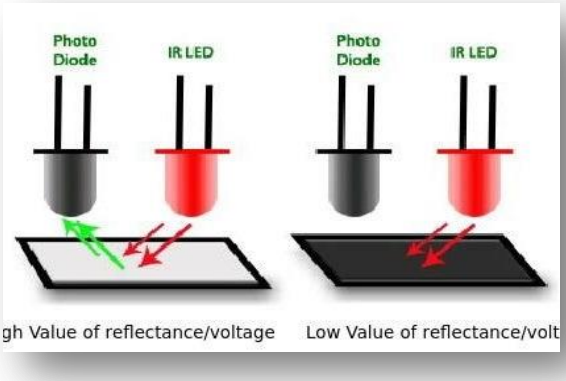
Τα Kp, Ki, Kd είναι όλα μη αρνητικά και υποδηλώνουν τους συντελεστές για τους αναλογικούς, ολοκληρωτικούς και διαφορικούς όρους, αντίστοιχα. Σε αυτό το μοντέλο ο όρος P λογίζεται για τις παρούσες τιμές του σφάλματος. Για παράδειγμα, αν το σφάλμα είναι μεγάλο και θετικό, η έξοδος ελέγχου θα είναι επίσης μεγάλη και θετική. Ο όρος I λογίζεται για προηγούμενες τιμές του σφάλματος. Για παράδειγμα, εάν η τρέχουσα έξοδος δεν είναι επαρκώς ισχυρή, το ολοκλήρωμα του σφάλματος θα συσσωρευτεί με την πάροδο του χρόνου και ο ελεγκτής θα ανταποκριθεί εφαρμόζοντας μια ισχυρότερη ενέργεια. Ο όρος D λογίζεται για πιθανές μελλοντικές τάσεις του σφάλματος, με βάση τον τρέχοντα ρυθμό μεταβολής. Η συνδεσμολογία του PID ελέγχου παρουσιάζεται στην παρακάτω εικόνα (Βεζήρης, 2017):



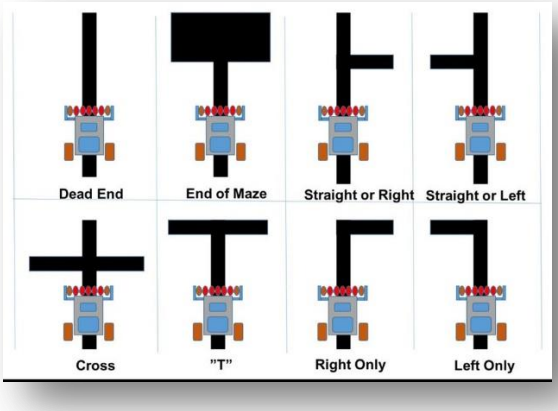
Εικόνα 4: Διάγραμμα ενός ελεγκτή PID σε ένα βρόγχο ανάδρασης. R(t) είναι η επιθυμητή τιμή διεργασίας ή setpoint και y(t) είναι η τιμή της μετρούμενης διεργασίας



Εικόνα 5: Σχεδιάγραμμα σήματος εξόδου αισθητήρων, ανάλογα με το αν ο αισθητήρας «βλέπει» την γραμμή ή όχι.



Εικόνα 6: Εικονική αναπαράσταση της λειτουργίας των φωτοδιόδων, ανάλογα με το αν «βλέπουν» την γραμμή ή όχι.



Εικόνα 7: Παραδείγματα πιθανών διασταυρώσεων του λαβυρίνθου.

iii. Υλοποίηση του κώδικα

Παρακάτω παρατίθενται ενδεικτικά τμήματα από τον κώδικα που υλοποιήθηκε.

```
void loop()
{

  while(MoveStatus!='E'){
    readIR();
     Serial.print("After read: ");
    checkPath();
    Serial.println(PathStatus);
  }
  while(MoveStatus=='E'){
    stayStill();
  }

}
```

Εικόνα 8: Η συνάρτηση για το "loop", που ελέγχει αν το ρομπότ έφτασε σε κατάσταση τερματισμού (end of maze), αν όχι διαβάζει την ένδειξη από τους σένσορες και καλή την συνάρτηση πλοήγησης.

```
void checkPath(){


   //End of maze
  if( IRs[4]<=activeGate  && IRs[3]<=blackGate  && IRs[2]<=blackGate && IRs[1]<=blackGate  && IRs[0]<=activeGate && ((millis()-timer2)>duration2)){
   stayStill();
   timer3=millis();
   oldMoveStatus=MoveStatus;
   while(millis()-timer3<duration3){
     Serial.println(millis()-timer3);
     readIR();
     if(IRs[4]>blackGate  || IRs[3]>blackGate  || IRs[2]>blackGate || IRs[1]>blackGate  || IRs[0]>blackGate){
       stayStill();
       MoveStatus='T';
       updatePathStatus('T');
       Serial.print(" TE ");
       turnLeft();
       timer2=millis();
       break;
     }
     else{

         MoveStatus='E';

     }
   }
   if(MoveStatus=='E'){
      stayStill();
      updatePathStatus('E');
      Serial.print(" End of maze ");
      timer2=millis();
   }
  }

   //Dead End
 else if( IRs[4]>=whiteGate  && IRs[3]>=whiteGate  && IRs[2]>=whiteGate && IRs[1]>=whiteGate  && IRs[0]>=whiteGate && ((millis()-timer2)>duration2)){
   stayStill();
   oldMoveStatus=MoveStatus;
   Serial.println(oldMoveStatus);
   MoveStatus='D';
   Serial.println(MoveStatus);
   updatePathStatus('D');
   turnAround();
   Serial.print(" Dead end ");
   timer2=millis();

  }
```

```
//Right 90o  turn
  else if(IRs[4]<=blackGate && IRs[3]<=blackGate && IRs[2]<=activeGate && IRs[0]>activeGate &&  ((millis()-timer2)>duration2) && MoveStatus!='E'){
   timer1=millis();
   oldMoveStatus=MoveStatus;
   while(millis()-timer1<duration1){
     Serial.println(millis()-timer1);
     readIR();
     if(IRs[0]<=blackGate){
       stayStill();
       MoveStatus='T';
       updatePathStatus('T');
       Serial.print(" TR ");
       turnLeft();
       timer2=millis();
       break;
    }
     else{

         MoveStatus='R';

     }
   }
   if(MoveStatus=='R'){
       //turnRight();
       updatePathStatus('R');
       timer2=millis();
       Serial.print(" Turn Right! ");
   }
  }

//Left 90o  turn
  else if(IRs[0]<=blackGate && IRs[1]<=blackGate && IRs[2]<=activeGate && IRs[4]>activeGate && ((millis()-timer2)>duration2) && MoveStatus!='E'){
   timer1=millis();
   oldMoveStatus=MoveStatus;
   while(millis()-timer1<duration1){
     Serial.println(millis()-timer1);
     readIR();
     if(IRs[4]<=blackGate){
       stayStill();
       MoveStatus='T';
       updatePathStatus('T');
       Serial.print(" TL ");
       turnLeft();
       timer2=millis();
       break;
    }
     else{
         MoveStatus='L';
```

```
     }
   }
   if(MoveStatus=='L'){
     turnLeft();
     updatePathStatus('L');
     timer2=millis();
     Serial.print(" Turn Left! ");
   }
  }


  else if ( MoveStatus!='E'){
   oldMoveStatus=MoveStatus;
   MoveStatus='S';
   updatePathStatus('S');
   movePID();
   Serial.print(" Move! ");
  }

}
```

Εικόνες 9,10,11: Το τμήμα του κώδικα που υλοποιεί τον έλεγχο πλοήγησης, για εύρεση στροφών, αδιεξόδων κλπ.

```
void movePID(){
    ErrorP=IRs[3]+(1.25*IRs[4])-IRs[1]-(1.25*IRs[0]);

    if(j<=count1){
     ErrorI=ErrorI+ErrorP;
     j++;
    }
    else{
     ErrorI=ErrorP/2;
     j=0;
    }

    ErrorD=ErrorP-ErrorPast;
    adjustSpeed=Kp*ErrorP+Ki*ErrorI+Kd*ErrorD;
    leftSpeed=frwSpeed-adjustSpeed;
    rightSpeed=frwSpeed+adjustSpeed;
    ErrorPast=ErrorP;
    if(leftSpeed<=minSpeed){leftSpeed=minSpeed;}
    if(rightSpeed<=minSpeed){rightSpeed=minSpeed;}

    if(leftSpeed>=maxSpeed){leftSpeed=maxSpeed;}
    if(rightSpeed>=maxSpeed){rightSpeed=maxSpeed;}

    analogWrite(leftForward , leftSpeed);
    analogWrite(leftBackward , LOW);
    analogWrite(rightForward , rightSpeed+rightWheelOffset);
    analogWrite(rightBackward , LOW);
}
```

Εικόνα 12: Το τμήμα του κώδικα που υλοποιεί τον έλεγχο του PID.

Όπως φαίνεται στα παραπάνω τμήματα κώδικα, χρησιμοποιήθηκαν πλήθος βοηθητικών συναρτήσεων, που λόγω του όγκου τους, δεν παρατέθηκαν στην παρούσα εργασία, αλλά βρίσκονται σε επισυναπτόμενο αρχείο.

IV. Συμπέρασμα

Συμπερασματικά λοιπόν, θα μπορούσαμε να πούμε ότι, σε γενικές γραμμές, η θεωρία επιβεβαιώθηκε κατά την υλοποίηση του πρακτικού μέρους. Παρόλα αυτά παρουσιάστηκαν κάποια τεχνικά προβλήματα τόσο κατά την κατασκευή του ρομπότ όσο και κατά την ανάπτυξη του λογισμικού, τα οποία ενδεικτικά παρατίθενται παρακάτω:

1. **Τροφοδοσία των συστημάτων του ρομπότ**

   Επειδή ακριβώς το ρομπότ περιλαμβάνει μια σειρά ετερόκλητων εξαρτημάτων, με διαφορετικές «ανάγκες» σε τάση τροφοδοσίας, απαιτείτο πολύ προσεκτικός σχεδιασμός του κυκλώματος. Ταυτόχρονα χρησιμοποιήθηκε ειδική επαναφορτιζόμενη μπαταρία (11.1V), ώστε να τροφοδοτεί με επαρκή τάση τα voltage regulators, τα οποία με την σειρά τους τροφοδοτούν τα αντίστοιχα κυκλώματα του ρομπότ.

2. **Απαιτητική διαδικασία ρύθμισης του ελεγκτή PID.**

   Για να επιτευχθεί η ιδανική ρύθμιση για την ομαλή κίνηση του ρομπότ κατά το “line following”, απαιτήθηκε πολύς χρόνος πειραματισμού και δοκιμών για την εύρεση των βέλτιστων τιμών του *αναλογικού*, *ολοκληρωτικού,* και *παραγωγικού* συντελεστή του ελεγκτή.

3. **Χρονοβόρα η διαδικασία βελτιστοποίησης του χρονοπρογραμματισμού των μεθόδων πλοήγησης.**

   Επειδή έχουμε σειριακή εκτέλεση των εντολών, υπήρξε η ανάγκη για τον ακριβή έλεγχο του χρόνου εκτέλεσης των διαφόρων λειτουργιών του ρομπότ. Για παράδειγμα, κατά την διάρκεια εκτέλεσης μιας στροφής, απαιτείται ακρίβεια στον χρόνο εκτέλεσης των διαφόρων τμημάτων της στροφής, ώστε να επιτευχθεί ομαλή κίνηση. Επίσης, μετά το πέρας της στροφής, και για συγκεκριμένο χρονικό διάστημα, το ρομπότ έμπαινε σε κατάσταση σταθεροποίησης της πορείας του, ώστε να αποφευχθεί τυχόν αστάθεια στην κίνηση.

Βιβλιογραφία

Copeland, B. (2019, 05 09). *Artificial intelligence*. Ανάκτηση από Encyclopædia Britannica: https://www.britannica.com/technology/artificial-intelligence

Russell, S. J., & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach* (3rd ed.). Harlow, Essex, England: Pearson.

Shadman, S., & et.al. (n.d.). Maze solving Algorithm for line following robot and derivation of linear path distance from nonlinear path. *16th International Conference on Computer and Information Technology*, (p. 6). Khulna, Bangladesh.

wikipedia. (n.d.). *Maze solving algorithm*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Maze_solving_algorithm#Wall_follower

Βεζήρης, Κ. (2017). Αναλογικά Συστήματα Ελέγχου.

Βλαχάβας, Ι., & κ.ά. (χ.χ.). Τεχνητή Νοημοσύνη.