Ρομποτικό όχημα και εφαρμογές STEM κατασκευή και προγραμματισμός
26-27 Ιουνίου 2023

Μέρος 2ο - Προγραμματισμός του οχήματος

Έχοντας ολοκληρώσει προηγουμένως το 1ο μέρος που αφορά την κατασκευή του ρομποτικού οχήματος, μπορούμε να προχωρήσουμε στον προγραμματισμό των αισθητήρων και των μηχανισμών.

Αρχικά συνδέουμε το Arduino στον υπολογιστή μας χρησιμοποιώντας το καλώδιο USB. Το περιβάλλον που θα χρησιμοποιήσουμε για τον προγραμματισμό είναι το Arduino IDE.

Στην Επιφάνεια εργασίας του υπολογιστή μας, στο εικονίδιο του Arduino IDE, κάνουμε **δεξί κλικ → Run as a program**. Την πρώτη φορά θα πρέπει να δηλώσουμε ποια πλακέτα Arduino διαθέτουμε, και σε ποια θύρα είναι συνδεδεμένη, επιλέγοντας από το μενού **Tools → Board → Arduino AVR Boards → Arduino Uno**, και στη συνέχεια **Tools → Port → /dev/ttyACM0** (συνήθως σε αυτή τη θύρα εντοπίζεται το Arduino στο Linux). Στη συνέχεια, κάθε φορά που φτιάχνουμε ένα πρόγραμμα (ή κάνουμε αλλαγές σε αυτό), θα το φορτώνουμε στο Arduino πατώντας το κουμπάκι Upload.

Στις δραστηριότητες που ακολουθούν, θα μας ζητείται να υλοποιήσουμε έναν αλγόριθμο, με βάση κάποιες οδηγίες που θα μας δίνονται. Κάτω από τις οδηγίες υπάρχουν κάποιες εντολές μέσα σε μαύρο πλαίσιο. Οι εντολές αυτές δίνονται για βοήθεια, όμως στις περισσότερες περιπτώσεις θα πρέπει να τις τροποποιήσουμε για να υλοποιήσουμε τον αλγόριθμο. Σε κάποιες περιπτώσεις βέβαια τις χρησιμοποιούμε όπως είναι.

Οι δραστηριότητες που περιγράφονται στα επόμενα βήματα είναι ενδεικτικές και προφανώς δεν αποτελούν μοναδική πρόταση για τον προγραμματισμό του οχήματος

1: Προγραμματισμός αισθητήρα Ultrasonic

2: Προγραμματισμός servo motors μηχανισμού pan/tilt

3: Προγραμματισμός αισθητήρα line follower και DC motors

4: Παραλλαγές, Βελτιώσεις, Προτάσεις

Δραστηριότητα 1: Προγραμματισμός αισθητήρα Ultrasonic

1.0 Έχουμε συνδέσει τον αισθητήρα Ultrasonic στο pin 4 (trigger) και στο pin A0 (echo). Θα προγραμματίσουμε το Arduino ώστε όταν δεν υπάρχει εμπόδιο μπροστά του, να ανάβει το λαμπάκι led που υπάρχει ενσωματωμένο στην πλακέτα του (pin 13). Κι όταν υπάρχει εμπόδιο μπροστά του, το λαμπάκι να σβήνει.

1.1 Πριν από τη συνάρτηση `void setup()` θα κάνουμε `define` τα pin που χρησιμοποιούμε, ώστε να τα καλούμε με κάποιο κατανοητό όνομα. Τα pin που θέλουμε ορίσουμε εμείς είναι: `trigPin` → `4`, `echoPin` → `A0`, `ledPin` → `13`.

```
#define myPin 1
```

* Η εντολή δίνεται ενδεικτικά και χρειάζεται να τροποποιηθεί.

1.2 Θα χρειαστούμε επίσης μια καθολική μεταβλητή ακέραιου τύπου για να μετράμε την απόσταση που μετράει ο αισθητήρας.

```
int distance;
```

1.3 Μέσα στη συνάρτηση `void setup()` θα ενεργοποιήσουμε το serial monitor στη θύρα `9600` για να παρακολουθούμε τις μετρήσεις,

```
Serial.begin(128000);
```

1.4 και θα δηλώσουμε ότι το `echoPin` αποτελεί **`INPUT`** ενώ το `trigPin` και `ledPin` το αποτελουν **`OUTPUT`**.

```
pinMode(myPin, INPUT);
pinMode(myPin, OUTPUT);
```

1.5 Μέσα στη συνάρτηση `void loop()` θα μετρήσουμε την τιμή του αισθητήρα, που για την ώρα ας υποθέσουμε ότι το κάνουμε ήδη με μια συνάρτηση με όνομα

```
check_Ultrasonic_Sensor();
```

1.6 και στη συνέχεια ελέγχουμε αν η απόσταση που μέτρησε είναι μεγαλύτερη από 60cm. Αν ναι, τότε ανάβουμε το `ledPin`. Αλλιώς το σβήνουμε.

```
if (distance > 25) { digitalWrite(myPin, HIGH); }
else { digitalWrite(myPin, LOW); }
```

1.7 Για επαλήθευση, γράφουμε και στο serial monitor την απόσταση που μετρήσαμε.

```
Serial.print("Distance: ");
Serial.print(distance);
Serial.println(" cm");
```

1.8 Ολοκληρώνουμε την επανάληψη βάζοντας μια μικρή καθυστέρηση πριν πάρουμε την επόμενη μέτρηση.

```
delay(200);
```

1.9 Μένει να υλοποιήσουμε τη συνάρτηση `check_Ultrasonic_Sensor()` που θα μετρήσει την απόσταση. Μέσα στη συνάρτηση αυτή θα ορίσουμε μια τοπική μεταβλητή για τον χρόνο που διαρκεί ο παλμός του αισθητήρα.

```
long duration;
```

1.10 Σβήνουμε το pin `trigPin`, περιμένουμε 2μs, το ανάβουμε, περιμένουμε 10μs και μετά το ξανασβήνουμε.

```
digitalWrite(myPin, LOW);
delayMicroseconds(2);
digitalWrite(myPin, HIGH);
delayMicroseconds(10);
digitalWrite(myPin, LOW);
```

1.11 Μετράμε τη διάρκεια διαβάζοντας τον παλμό στο pin `echoPin`

```
duration = pulseIn(myPin, HIGH);
```

1.12 και υπολογίζουμε την απόσταση με βάση τη διάρκεια, σύμφωνα με τον μετασχηματισμό που μας δίνεται από τον κατασκευαστή.

```
distance = duration * 0.034 / 2;
```

Κάνουμε upload το πρόγραμμα, και ελέγχουμε αν λειτουργεί σωστά.

Δραστηριότητα 2: Προγραμματισμός servo motors μηχανισμού pan/tilt

2.0 Έχουμε συνδέσει τα δύο servo μοτέρ στα pin 2 (pan) και pin 3 (tilt), και θέλουμε να δώσουμε κίνηση στα μοτέρ ώστε να σαρώνουν τον χώρο που βρίσκεται στην εμβέλειά τους. Το μοτέρ pan θα σαρώνει αριστερά/δεξιά από γωνία > 0° έως γωνία < 180°.

2.1 Αρχικά θα κάνουμε include τη βιβλιοθήκη του servo.

```
#include <Servo.h>
```

2.2 Κάνουμε `define` τα pin που χρησιμοποιούμε. Θα ορίσουμε το `panServoPin` → 2 και `tiltServoPin` → 3.

```
// βλέπε 1.1
```

2.3 Ορίζουμε δύο μεταβλητές τύπου `Servo` μία για κάθε μοτέρ. Μία δηλαδή για το `tilt_servo` και μία για το `pan_servo`.

```
Servo my_servo;
```

2.4 Ορίζουμε και κάποιες καθολικές μεταβλητές ακέραιου τύπου, για να κρατάμε τις γωνίες και τα όρια των μηχανισμών. `pan_angle → 0`, `tilt_angle → 0`, `pan_angle_from → 0`, `pan_angle_to → 180`, `tilt_angle_from → 0`, `tilt_angle_to → 50`, `i1 → pan_angle_from`, `i2 → tilt_angle_from`.

```
// βλέπε 1.2
```

* ανάλογα με τις προδιαγραφές του servo motor, μπορεί οι τιμές αυτές να διαφέρουν.

2.5 Μέσα στη συνάρτηση `void setup()` θα ενεργοποιήσουμε το serial monitor στη θύρα `9600` για να παρακολουθούμε τις μετρήσεις,

```
// βλέπε 1.3
```

2.6 και αντιστοιχίζουμε τις συσκευές servo στις μεταβλητές που έχουμε δηλώσει.

```
pan_servo.attach(panServoPin);
tilt_servo.attach(tiltServoPin);
```

2.7 Μέσα στη συνάρτηση `void loop()` θα καλέσουμε τις συναρτήσεις που ορίζουν την κίνηση του κάθε servo, και τις οποίες θα υλοποιήσουμε αμέσως μετά

```
pan_scan();
tilt_scan();
```

2.8 Βάζουμε και μια μικρή καθυστέρηση `70` πριν από την επόμενη επανάληψη.

```
// βλέπε 1.8
```

2.9 Τώρα θα υλοποιήσουμε τη συνάρτηση `pan_scan()`, μέσα στην οποία θα ζητάμε από το μοτέρ να ξεκινάει από την αρχική γωνία που είχαμε ορίσει σε 0 (`pan_angle_from → 0`) και να σαρώνει σιγά σιγά από τα δεξιά προς τα αριστερά, αυξάνοντας σε κάθε βήμα τη γωνία του. Για να το πετύχουμε αυτό:

- αυξάνουμε τον μετρητή `i1` κατά `1`
- ελέγχουμε `αν` ο μετρητής `i1` είναι **μικρότερος ή ίσος** από το όριο `pan_angle_to`
  - τότε ορίζουμε τη γωνία `pan_angle → i1`
- `αλλιώς`
  - ορίζουμε τη γωνία `pan_angle → 2* pan_angle_to - i1`

```
// προσπαθήστε το λίγο μόνοι σας
```

2.10 Τα όρια περιστροφής των servo δεν εκφράζονται σε μοίρες °. Για να αντιστοιχίσουμε τις γωνίες που μετράμε στις τιμές που δέχεται η συσκευή, θα κάνουμε ένα `mapping` και θα το εφαρμόσουμε στο μοτέρ με την εντολή `write`.

```
pan_servo.write( map(pan_angle, 0, 180, 60, 120) );
```

2.11 Στη συνέχεια ελέγχουμε **εάν** ο μετρητής `i1` είναι **μεγαλύτερος** από το άθροισμα `pan_angle_from+(pan_angle_to-pan_angle_from)*2`
- τότε δίνουμε στον μετρητή `i1` την τιμή `pan_angle_from`

```
// προσπαθήστε το λίγο μόνοι σας
```

οπότε στην επόμενη επανάληψη το μοτέρ θα γυρίσει στην αρχική του θέση.

2.12 Παρόμοια και για τη συνάρτηση `tilt_scan()`, μέσα στην οποία θα ζητάμε από το άλλο μοτέρ να ξεκινάει από την αρχική γωνία που είχαμε ορίσει σε 70 (`tilt_angle_from → 70`) και να σαρώνει σιγά σιγά από κάτω προς τα πάνω, αυξάνοντας σε κάθε βήμα τη γωνία του. Για να το πετύχουμε αυτό:
- αυξάνουμε τον μετρητή `i2` κατά `2`
- ελέγχουμε **αν** ο μετρητής `i2` είναι **μικρότερος ή ίσος** από το όριο `tilt_angle_to`
  - τότε ορίζουμε τη γωνία `tilt_angle → i2`
- **αλλιώς**
  - ορίζουμε τη γωνία `tilt_angle → 2* tilt_angle_to - i2`

```
// προσπαθήστε το λίγο μόνοι σας
```

2.13 Για να αντιστοιχίσουμε τις γωνίες που μετράμε στις τιμές που δέχεται η συσκευή, θα κάνουμε ένα `mapping` και θα το εφαρμόσουμε στο μοτέρ με την εντολή `write`.

```
tilt_servo.write( map(tilt_angle, 0, 50, 60, 120) );
```

2.14 Στη συνέχεια ελέγχουμε **εάν** ο μετρητής `i2` είναι **μεγαλύτερος** από το άθροισμα `tilt_angle_from+(tilt_angle_to-tilt_angle_from)*2`
- τότε δίνουμε στον μετρητή `i2` την τιμή `tilt_angle_from`

```
// προσπαθήστε το λίγο μόνοι σας
```

οπότε στην επόμενη επανάληψη το μοτέρ θα γυρίσει στην αρχική του θέση.

Κάνουμε upload το πρόγραμμα, και ελέγχουμε αν λειτουργεί σωστά.

Δραστηριότητα 3: Προγραμματισμός αισθητήρα line follower και DC motors

3.0 Ο αισθητήρας line follower αποτελείται από 5 ξεχωριστούς αισθητήρες χρώματος, τοποθετημένους ο ένας δίπλα στον άλλο, και ο συνδυασμός των μετρήσεων τους μας επιτρέπει να υπολογίσουμε αν το όχημα κινείται πχ κατά μήκος μιας μαύρης γραμμής, και να το επαναφέρουμε στην πορεία του όταν αυτό αποκλίνει.

3.1 Αρχικά δηλώνουμε τις μεταβλητές για τις τιμές των αισθητήρων, καθώς και κάποιους σταθερούς ακεραίους για λόγους ευκολίας και ομοιομορφίας. Συγκεκριμένα θα χρειαστούμε:
- έναν πίνακα const ακεραίων για τα pin που είναι συνδεδεμένοι οι αισθητήρες

```
const int sensorPins[5] = {A1, A2, A3, A4, A5};
```

- έναν μονοδιάστατο πίνακα ακεραίων για τις τιμές που μετράνε οι αισθητήρες

```
int sensorValues[5];
```

- έναν ακέραιο που εκφράζει το κατώφλι της φωτεινότητας για να ξεχωρίζουμε πότε ο μεσαίος αισθητήρας βρίσκεται πάνω από τη μαύρη γραμμή

```
int trigValue=600;
```

* ανάλογα με τις συνθήκες φωτισμού μπορεί να χρειαστεί αλλαγή στο όριο αυτό.

- έναν σταθερό ακέραιο για κάθε pin των μοτέρ με τα οποία συνεργάζεται ο αισθητήρας: `enA` → `5`, `in1` → `7`, `in2` → `8`, `enB` → `6`, `in3` → `9`, `in4` → `10`.

```
const int enA = 5;
```

- έναν ακέραιο για την ταχύτητα με την οποία θα κινείται το όχημα

```
int baseSpeed = 40;
```

3.2 Μέσα στη συνάρτηση `void setup()` θα ενεργοποιήσουμε το serial monitor στη θύρα `9600` για να παρακολουθούμε τις μετρήσεις,

```
// βλέπε 1.3
```

3.3 Δηλώνουμε για τα pin που αφορούν τα μοτέρ ότι είναι όλα `OUTPUT` (`enA`, `enB`, `in1`, `in2`, `in3`, `in4`)

```
pinMode(enA, OUTPUT);
```

3.4 Κάνουμε reset όλα τα motors. Είπαμε ότι τα pin `enA` και το `enB` που εκφράζουν την ταχύτητα των μοτέρ παίρνουν τιμές αναλογικές (από `0` έως `255`), ενώ τα pin `in1`, `in2`, `in3` και `in4` παίρνουν ψηφιακές τιμές (`0`/`1` ή αλλιώς `LOW`/`HIGH`)

```
analogWrite(enB, 0);
digitalWrite(in1, LOW);
```

3.5 Μέσα στη συνάρτηση `void loop()` θα διαβάσουμε τις τιμές των 5 αισθητήρων, χρησιμοποιώντας μια επανάληψη, κατά την οποία

- `για καθέναν` από τους `5` αισθητήρες
  - `τιμήΑισθητήρα` → `διάβασεΑναλογικήΤιμή` από `pinΑισθητήρα`
  - `τύπωσε` στο `σειριακό` monitor την `τιμήΑισθητήρα`

```
sensorValues[i] = analogRead(sensorPins[i]);
Serial.println(sensorValues[i]);
```

* οι πίνακες ξεκινάνε από το στοιχείο [0]... να το θυμάμαι αυτό στην επανάληψη!

3.6 Ανάλογα με τις τιμές που έχουν επιστρέψει οι αισθητήρες, αποφασίζουμε αν θα κινηθούμε ευθεία ή θα στρίψουμε αριστερά ή δεξιά. Συγκεκριμένα:

- Εάν η τιμήΑισθητήρα 0 είναι μεγαλύτερη από το όριο
  ΚΑΙ η τιμήΑισθητήρα 1 είναι μεγαλύτερη από το όριο
  ΚΑΙ η τιμήΑισθητήρα 2 είναι μικρότερη από το όριο
  ΚΑΙ η τιμήΑισθητήρα 3 είναι μεγαλύτερη από το όριο
  ΚΑΙ η τιμήΑισθητήρα 4 είναι μεγαλύτερη από το όριο
  - τότε κινήσουΕυθεία
- αλλιώς εάν … τιμήΑισθητήρα 0 μικρότερη Ή
  1 μικρότερη ΚΑΙ 2, 3, 4 μεγαλύτερη
  - τότε στρίψεΑριστερά
- αλλιώς εάν … τιμήΑισθητήρα 0, 1, 2 μεγαλύτερη ΚΑΙ 3 μικρότερη
  Ή 4 μικρότερη
  - τότε στρίψεΔεξιά
- αλλιώς
  - κινήσουΕυθεία

```
if (sensorValues[0]<trigValue || sensorValues[1]<trigValue && . . .
  moveForward();  // θα υλοποιηθεί παρακάτω
. . .
  turnLeft();     // θα υλοποιηθεί παρακάτω
  turnRight();    // θα υλοποιηθεί παρακάτω
```

3.7 Εκτυπώνουμε μια διαχωριστική γραμμή "==========" στο σειριακό monitor για να ξεχωρίζουμε τις επαναλήψεις

```
// βλέπε 1.7
```

και βάζουμε και μια μικρή καθυστέρηση 350 πριν την επόμενη επανάληψη.

```
// βλέπε 1.8
```

3.8 Ορίζουμε τη συνάρτηση moveForward() στην οποία ενεργοποιούμε και τα 2 μοτέρ, θέτοντας enA, enB → baseSpeed, in1, in3 → HIGH, in2, in4 → LOW.

```
analogWrite(enA, baseSpeed);
digitalWrite(in3, HIGH);
```

με τον ίδιο τρόπο ορίζουμε και τη συνάρτηση turnLeft(), όπου θέτουμε
enA, enB → baseSpeed, in2, in3 → HIGH, in1, in4 → LOW
και τη συνάρτηση turnRight() όπου θέτουμε
enA, enB → baseSpeed, in1, in4 → HIGH, in2, in3 → LOW.

Κάνουμε upload το πρόγραμμα, και ελέγχουμε αν λειτουργεί σωστά.

Δραστηριότητα 4: Παραλλαγές, Βελτιώσεις, Προτάσεις
4.1 Στο σημείο αυτό έχουμε καταφέρει να προγραμματίσουμε τον κάθε αισθητήρα ή τον κάθε μηχανισμό ξεχωριστά. Προφανώς όμως, για να είναι το ρομποτικό όχημα πλήρως λειτουργικό και αυτόνομο, θα πρέπει το πρόγραμμα που θα κάνουμε upload στο Arduino, να περιλαμβάνει όλο τον κώδικα που γράψαμε παραπάνω. Στον χρόνο που απομένει, και μέχρι την ολοκλήρωση του σεμιναρίου, ας προσπαθήσουμε να συνδυάσουμε όλα προγράμματα σε ένα, το οποίο όμως να εκτελεί όλες τις λειτουργίες. Αυτό που θα πρέπει να κάνουμε είναι: ☐ να πάρουμε όλες τις δηλώσεις των καθολικών μεταβλητών και βιβλιοθηκών, ☐ να συμπεριλάβουμε τις συναρτήσεις που υλοποιήσαμε, ☐ να μαζέψουμε τα περιεχόμενα όλων των συναρτήσεων setup() σε μία (δεν παίζει ρόλο η σειρά με την οποία θα μπουν) ☐ να μαζέψουμε τα περιεχόμενα όλων των συναρτήσεων loop() σε μία (εδώ παίζει ρόλο η σειρά με την οποία θα μπουν τα κομμάτια του κώδικα, καθώς μπορεί να επηρεάζεται η επανάληψη)
4.2 Επιπλέον, όπως είπαμε και στην αρχή του φύλλου εργασίας, τα παραδείγματα που υλοποιήσαμε δεν είναι η μοναδική πρόταση για τον προγραμματισμό του οχήματος. Θα μπορούσαμε να δώσουμε άλλες εντολές, ώστε το όχημά μας να λειτουργεί με διαφορετικό τρόπο. Με τα μέλη της ομάδας μας, ή σε συνεργασία με τις υπόλοιπες ομάδες, μπορούμε να συζητήσουμε διάφορες προτάσεις για το πώς αλλιώς θα μπορούσε να συμπεριφέρεται το ρομποτικό μας όχημα, ή να βελτιώσουμε ίσως κάποιες λειτουργίες του.
Καλή επιτυχία στην υλοποίηση των προτάσεων!!