Αυτόνομο όχημα ηχητικής ξενάγησης

Φύλλα εργασίας

2ο Γυμνάσιο Μοσχάτου

Σχολικό έτος 2024 - 25

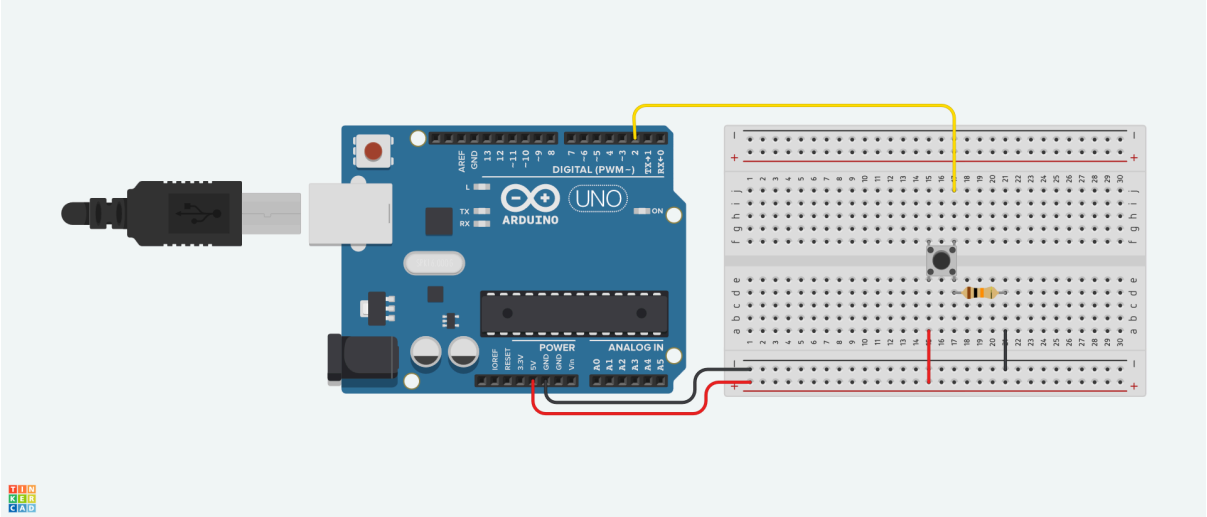
Εισαγωγή

Τα φύλλα εργασίας που περιλαμβάνονται στις παρούσες σημειώσεις εξηγούν τη λειτουργία και τη χρήση των επιμέρους στοιχείων που χρησιμοποιούμε στο αυτόνομο όχημα ηχητικής ξενάγησης. Η εκτέλεση των δραστηριοτήτων μπορεί να γίνει εξ ολοκλήρου στο ελεύθερο λογισμικό TinkerCAD και δεν απαιτείται η αγορά εξοπλισμού. Για τη δωρεάν χρήση του TinkerCAD απαιτείται η δημιουργία ενός λογαριασμού μέσω της ιστοσελίδας του προγράμματος.

Φύλλο εργασίας 1 - Ανάγνωση ψηφιακού αισθητήρα (διακόπτη) και εγγραφή ψηφιακών δεδομένων

Ένας τρόπος εισαγωγής δεδομένων και αλληλεπίδρασης με το Arduino, είναι το πάτημα ενός διακόπτη, το οποίο μπορεί να προκαλεί κάποια άλλη ενέργεια. Οι διακόπτες είναι απλά εξαρτήματα που έχουν μόνο δύο καταστάσεις: ανοιχτός ή κλειστός. Όταν ο διακόπτης είναι κλειστός τότε περνάει ρεύμα από αυτόν, ενώ όταν ο διακόπτης είναι ανοιχτός δεν περνάει ρεύμα. Μερικοί διακόπτες πίεσης κλείνουν όταν τους πιέζουμε, ενώ μερικοί ανοίγουν.
Για την εμφάνιση των δεδομένων μπορεί να χρησιμοποιηθεί το ενσωματωμένο LED στον ακροδέκτη 13 του Arduino ή, εφόσον αυτό δεν επαρκεί, να συνδεθούν εξωτερικά LED σε οποιονδήποτε ψηφιακό ακροδέκτη του.

<u>Α. Ανάγνωση ψηφιακών δεδομένων - σύνδεση pull-down</u>
Σχεδιάζουμε το παρακάτω κύκλωμα στο TINKERCAD:



Η αντίσταση του κυκλώματος έχει τιμή 10ΚΩ. Με την παραπάνω σύνδεση ο ακροδέκτης 2 του Arduino συνδέεται στη γείωση μέσω της αντίστασης και όταν πατηθεί ο διακόπτης συνδέεται στα 5V.

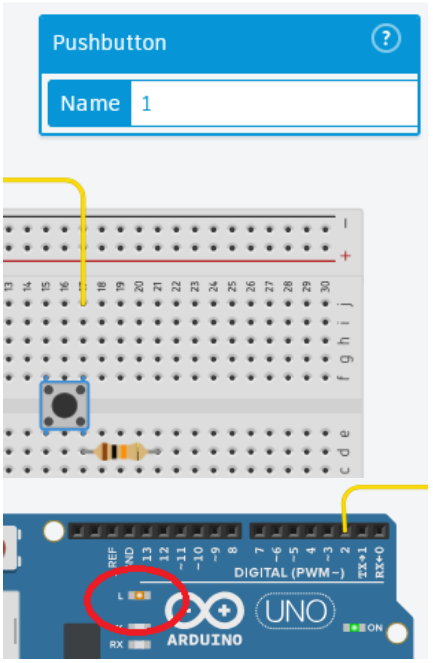
Εισάγουμε τον παρακάτω κώδικα:

```
int buttonPin = 2;
int LED = 13;
void setup() {
        pinMode(buttonPin, INPUT);
        pinMode(LED, OUTPUT);
}
void loop() {
        int buttonValue = digitalRead(buttonPin);
        if (buttonValue == HIGH) {
        digitalWrite(LED,HIGH);
        } else {
        digitalWrite(LED, LOW);
        }
}
```

Το πρόγραμμα διαβάζει την τιμή του διακόπτη πίεσης και εμφανίζει την κατάστασή του στο ενσωματωμένο LED του Arduino.

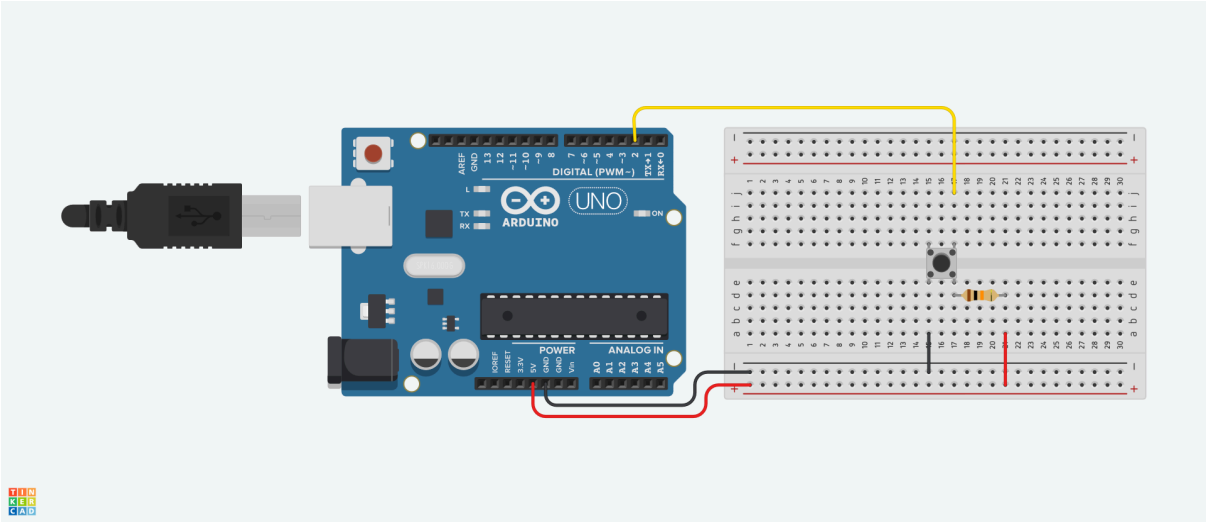
Ξεκινάμε την προσομοίωση του κυκλώματος στο TinkerCAD.

Επιλέγουμε τον διακόπτη στον χώρο σχεδίασης και πιέζουμε:



Παρατηρούμε την εναλλαγή στο ενσωματωμένο LED του Arduino. Όταν πιέζεται ο διακόπτης το ενσωματωμένο LED ανάβει.

<u>Β. Ανάγνωση ψηφιακών δεδομένων - σύνδεση pull-up</u>
Τροποποιούμε το κύκλωμα στο TINKERCAD όπως παρακάτω:



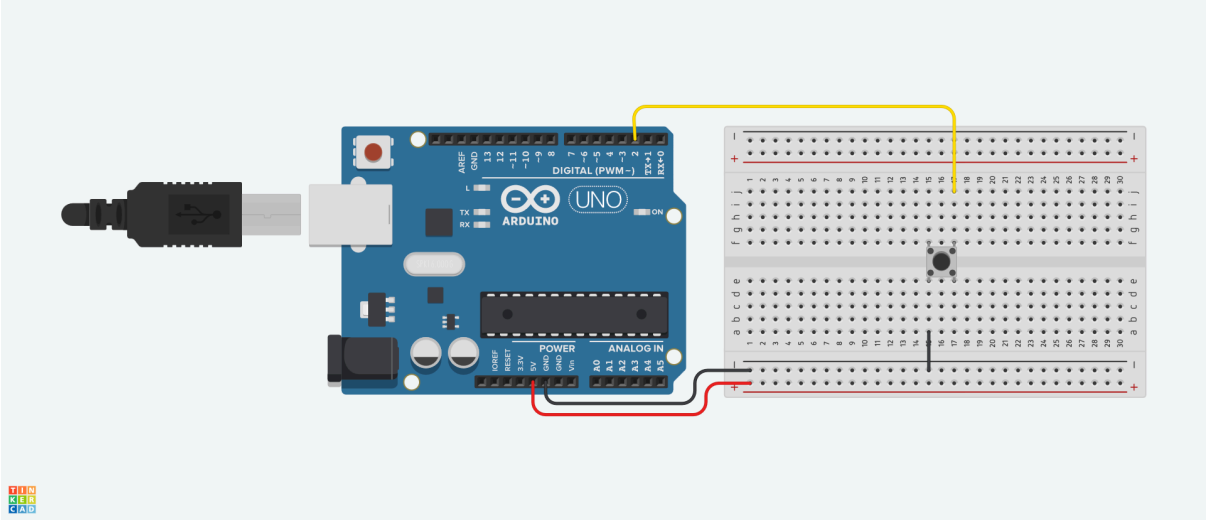
Με την παραπάνω σύνδεση ο ακροδέκτης 2 του Arduino συνδέεται στα 5V μέσω της αντίστασης και όταν πατηθεί ο διακόπτης συνδέεται στη γείωση.

Δεν κάνουμε καμία τροποποίηση στο πρόγραμμα και ξεκινάμε την προσομοίωση. Το ενσωματωμένο LED είναι αναμμένο και σβήνει όταν πατηθεί ο διακόπτης.

Γ. Ανάγνωση ψηφιακών δεδομένων - ενσωματωμένη αντίσταση pull-up

Η σύνδεση μιας αντίστασης είναι απαραίτητη για τη σωστή λειτουργία ενός διακόπτη. Ωστόσο κάθε ακροδέκτης του Arduino έχει ήδη μια αντίσταση pull-up, που μπορούμε να ενεργοποιήσουμε με μία μόνο αλλαγή στον κώδικα.

Τροποποιούμε το κύκλωμα στο TINKERCAD όπως φαίνεται παρακάτω:



Τροποποιούμε τον κώδικα, όπως φαίνεται παρακάτω:

```
int buttonPin = 2;
int LED = 13;
void setup() {
        pinMode(buttonPin, INPUT_PULLUP);
        pinMode(LED, OUTPUT);
}
void loop() {
        int buttonValue = digitalRead(buttonPin);
        if (buttonValue == HIGH) {
        digitalWrite(LED,HIGH);
        } else {
        digitalWrite(LED, LOW);
        }
}
```

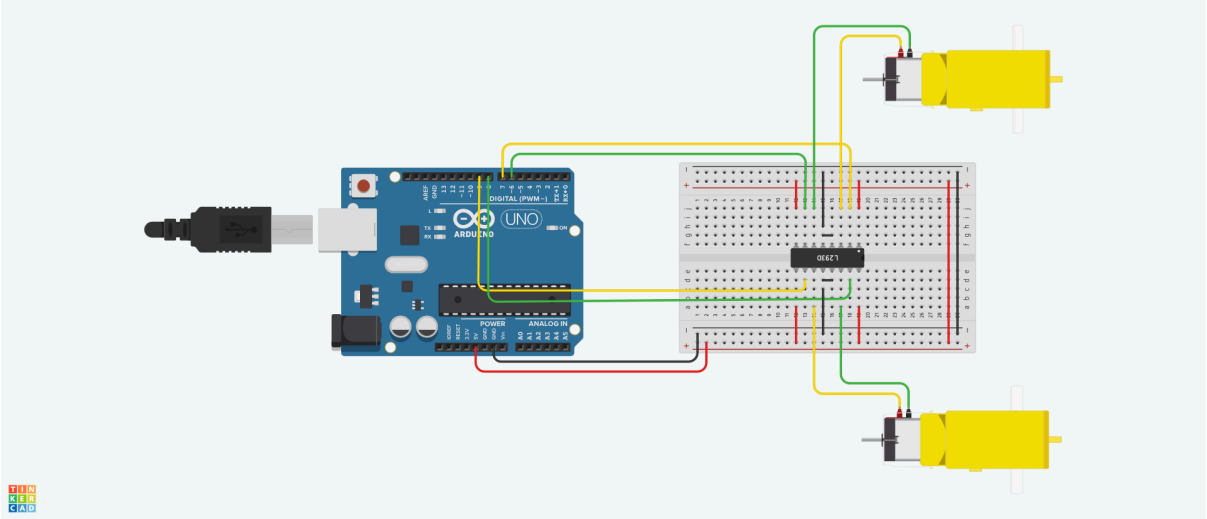
Ξεκινάμε την προσομοίωση του κυκλώματος στο TinkerCAD.

Το κύκλωμα συμπεριφέρεται όπως και στην προηγούμενη περίπτωση.

Φύλλο εργασίας 2 - Έλεγχος κινητήρων DC με το L293

Ο έλεγχος ενός κινητήρα DC είναι πολύ απλός. Όταν στους ακροδέκτες του κινητήρα εφαρμόζουμε την τάση προς μία κατεύθυνση, ο κινητήρας περιστρέφεται προς τη μία πλευρά, ενώ, αν αντιστρέψουμε την τάση, ο κινητήρας θα γυρίσει αντίθετα. Μπορούμε να ελέγξουμε έναν μικρό κινητήρα συνδέοντάς τον απευθείας με ένα ψηφιακό ακροδέκτη του Arduino. Ωστόσο, κάθε κινητήρας μεγαλύτερος από ένα νόμισμα θα έκαιγε τον ψηφιακό ακροδέκτη και πιθανότατα και το Arduino. Για να οδηγήσουμε έναν μεγαλύτερο κινητήρα, πρέπει να χρησιμοποιήσουμε ένα κύκλωμα με τρανζίστορ ή κάποιο ειδικό ολοκληρωμένο, Με το ολοκληρωμένο οδήγησης L293D, μπορούμε να ελέγξουμε δύο κινητήρες DC.

<u>Α. Έλεγχος της κατεύθυνσης δύο κινητήρων DC με το L293</u>
Σχεδιάζουμε το παρακάτω κύκλωμα στο TINKERCAD:

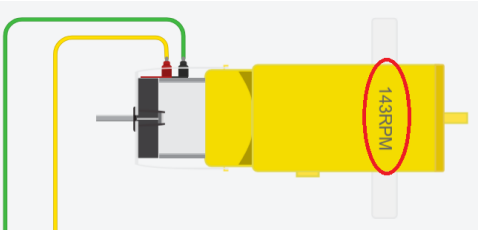


Εισάγουμε τον παρακάτω κώδικα:

```
//Motor A
const int motorPin1  = 7;
const int motorPin2  = 6;
//Motor B
const int motorPin3  = 9;
const int motorPin4  = 8;

//This will run only one time.
void setup(){
        //Set pins for motors
        pinMode(motorPin1, OUTPUT);
        pinMode(motorPin2, OUTPUT);
        pinMode(motorPin3, OUTPUT);
        pinMode(motorPin4, OUTPUT);

        //This code  will turn Motor A clockwise for 2 sec.
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, LOW);
```

```
        digitalWrite(motorPin4, LOW);
        delay(2000);

        // This code will turn Motor A & B clockwise for 2 sec.
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin4, LOW);
        delay(2000);

        //This code will turn Motor B clockwise for 2 sec.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin4, LOW);
        delay(2000);

        //This code will turn Motor B counter-clockwise for 2 sec.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, HIGH);
        delay(2000);

        //This code will turn Motor A clockwise for 1 sec.
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, LOW);
        delay(1000);

        //This code will turn Motor B clockwise for 1 sec.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin4, LOW);
        delay(1000);

        //And this code will stop motors
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, LOW);
}

void loop(){

}
```

Το πρόγραμμα περιστρέφει τον κινητήρα A ή B ή και τους δύο μαζί διαδοχικά, είτε κατά τη φορά του ρολογιού είτε αντίστροφα. Επειδή η ενεργοποίηση των κινητήρων γίνεται στην ενότητα setup(), η κάθε περιστροφή γίνεται μόνο μια φορά και στο τέλος οι κινητήρες σταματάνε.



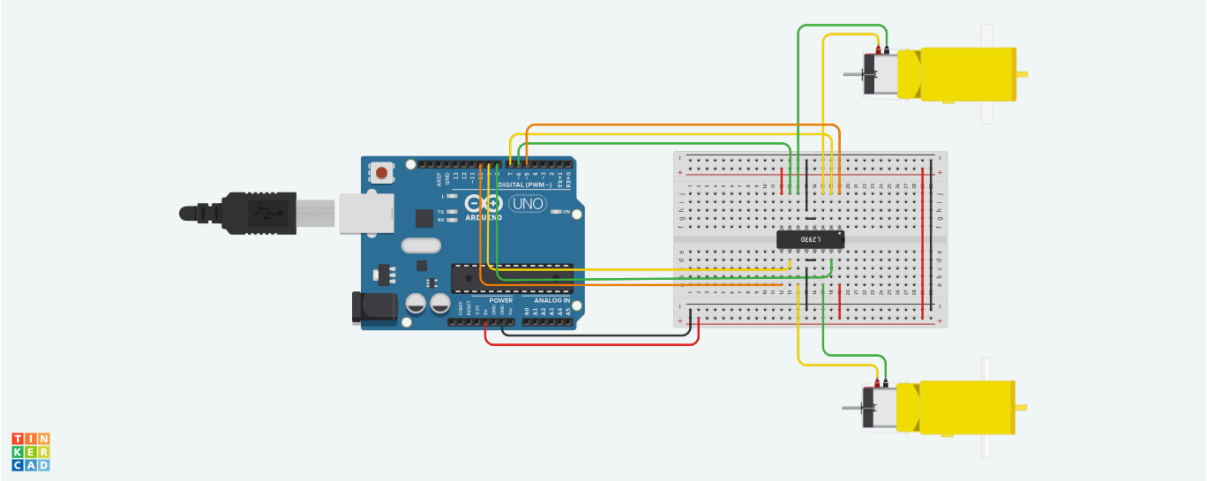
Ξεκινάμε την προσομοίωση του κυκλώματος στο TinkerCAD και παρατηρούμε την περιστροφή των κινητήρων.

Για να επαναληφθεί η ακολουθία των περιστροφών πρέπει να επανεκκινήσουμε την προσομοίωση ή να πιέσουμε το διακόπτη Reset στο Arduino.

Β. Έλεγχος της κατεύθυνσης και της ταχύτητας δύο κινητήρων DC με το L293

Μπορούμε να ελέγξουμε την ταχύτητα ενός κινητήρα χρησιμοποιώντας τη διαμόρφωση πλάτους παλμού (PWM). Χρησιμοποιώντας αυτήν τη μέθοδο, το ρεύμα εφαρμόζεται στον κινητήρα σε σύντομους παλμούς: όσο μεγαλύτερος είναι ο παλμός, τόσο μεγαλύτερη είναι η ταχύτητα,

Τροποποιούμε το κύκλωμα στο TINKERCAD όπως παρακάτω:



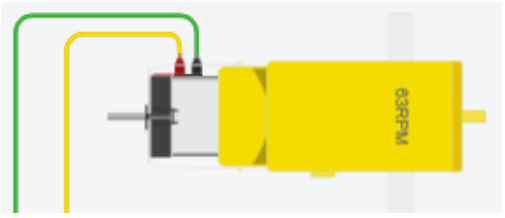
Εισάγουμε τον παρακάτω κώδικα:

```
//Motor A
const int enA = 5;
const int motorPin1  = 7;
const int motorPin2  = 6;
//Motor B
const int enB = 10;
const int motorPin3  = 9;
const int motorPin4  = 8;

//This will run only one time.
void setup(){
        //Set pins for motors
```

```
        pinMode(enA, OUTPUT);
        pinMode(enB, OUTPUT);
        pinMode(motorPin1, OUTPUT);
        pinMode(motorPin2, OUTPUT);
        pinMode(motorPin3, OUTPUT);
        pinMode(motorPin4, OUTPUT);
}

void loop(){
        // turn on motors
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin4, LOW);
        // accelerate from zero to maximum speed
        for (int i = 0; i < 256; i++){
                analogWrite(enA, i);
                analogWrite(enB, i);
                delay(20);
         }
        // decelerate from maximum speed to zero
        for (int i = 255; i >= 0; --i){
                analogWrite(enA, i);
                analogWrite(enB, i);
                delay(20);
        }
        // now turn off motors
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, LOW);
        delay(1000);
}
```

Το πρόγραμμα περιστρέφει και τους δύο κινητήρες ταυτόχρονα με αυξανόμενη ταχύτητα μέχρι το μέγιστο και στη συνέχεια ελαττώνει την ταχύτητα μέχρι το 0, όπου οι κινητήρες απενεργοποιούνται για 1 δευτερόλεπτο, πριν ο κύκλος επαναληφθεί.

Ξεκινάμε την προσομοίωση του κυκλώματος στο TinkerCAD και παρατηρούμε την περιστροφή των κινητήρων.

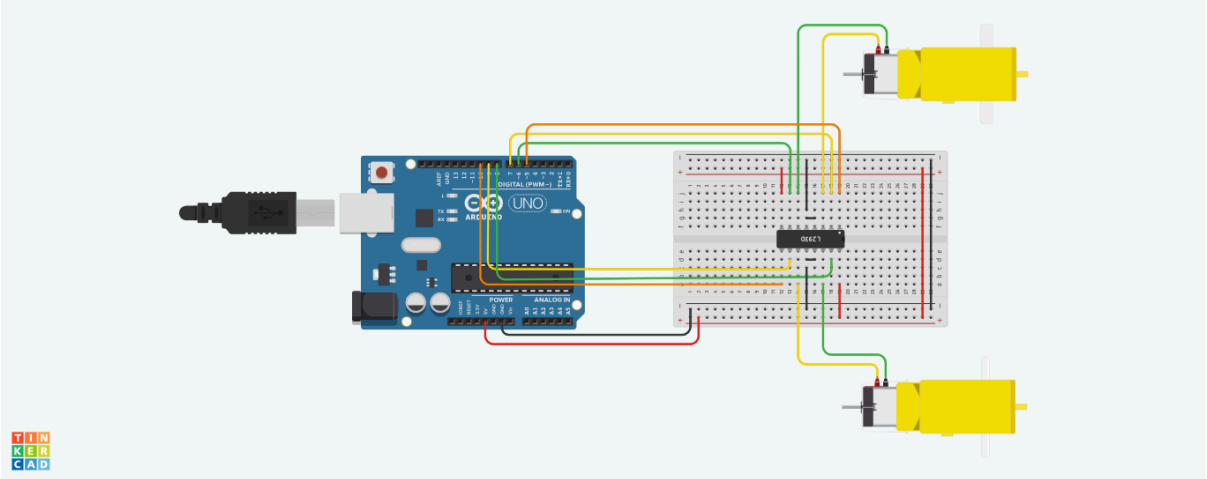


Φύλλο εργασίας 3 - Έλεγχος των κινήσεων μιας ρομποτικής πλατφόρμας με 2 κινητήρες DC και το L293

Μπορούμε να ελέγξουμε την κίνηση μιας ρομποτικής πλατφόρμας ελέγχοντας την κατεύθυνση και την ταχύτητα περιστροφής δύο κινητήρων DC, οι οποίοι είναι τοποθετημένοι στις δύο πλευρές της πλατφόρμας. Αν η πλατφόρμα έχει ερπύστριες, ο κάθε κινητήρας κινεί μια ερπύστρια, ενώ αν η πλατφόρμα έχει τροχούς, ο κάθε κινητήρας ελέγχει έναν τροχό και υπάρχει ένας ή περισσότεροι ελεύθεροι τροχοί για τη στήριξη της πλατφόρμας.

Α. Έλεγχος των κινήσεων μιας ρομποτικής πλατφόρμας

Σχεδιάζουμε το παρακάτω κύκλωμα στο TINKERCAD:



Εισάγουμε τον παρακάτω κώδικα:

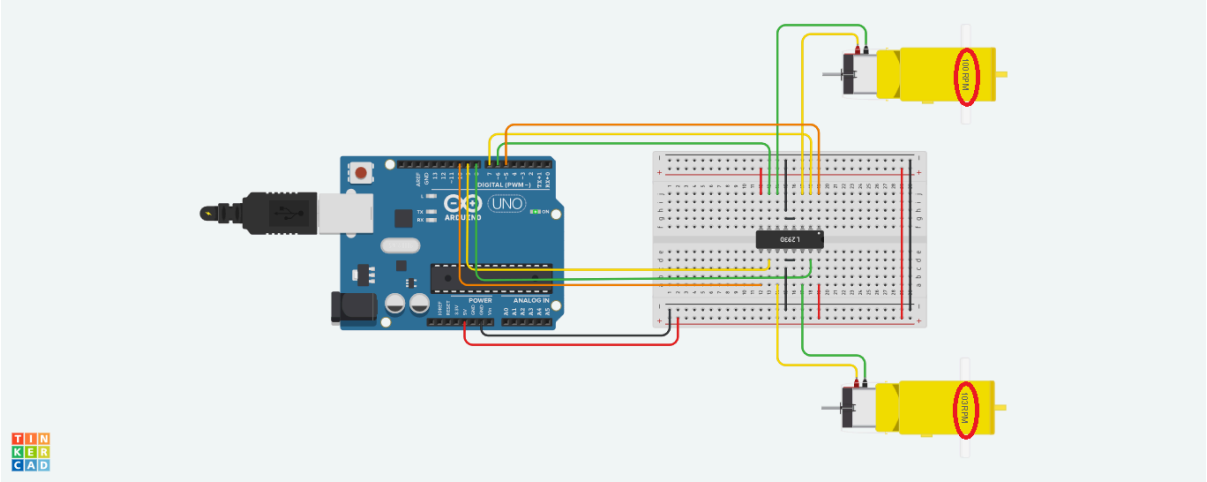
```
//Motor A
const int enA = 5;
const int motorPin1  = 7;
const int motorPin2  = 6;
//Motor B
const int enB = 10;
const int motorPin3  = 9;
const int motorPin4  = 8;

//This will run only one time.
void setup(){
        //Set pins for motors
        pinMode(enA, OUTPUT);
        pinMode(enB, OUTPUT);
        pinMode(motorPin1, OUTPUT);
        pinMode(motorPin2, OUTPUT);
        pinMode(motorPin3, OUTPUT);
        pinMode(motorPin4, OUTPUT);
        analogWrite(enA,180);
        analogWrite(enB,180);
}
```

```
void loop(){
	// This code will move platform forward for 1 sec.
	digitalWrite(motorPin1, HIGH);
	digitalWrite(motorPin2, LOW);
	digitalWrite(motorPin3, HIGH);
	digitalWrite(motorPin4, LOW);
	delay(1000);
	// This code will move platform backward  for 1 sec.
	digitalWrite(motorPin1, LOW);
	digitalWrite(motorPin2, HIGH);
	digitalWrite(motorPin3, LOW);
	digitalWrite(motorPin4, HIGH);
	delay(1000);
	//This code  will turn platform left for 0.5 sec.
	digitalWrite(motorPin1, HIGH);
	digitalWrite(motorPin2, LOW);
	digitalWrite(motorPin3, LOW);
	digitalWrite(motorPin4, LOW);
	delay(500);
	//This code  will turn platform right for 0.5 sec.
	digitalWrite(motorPin1, LOW);
	digitalWrite(motorPin2, LOW);
	digitalWrite(motorPin3, HIGH);
	digitalWrite(motorPin4, LOW);
	delay(500);
	//This code  will sharp turn platform left for 0.5 sec.
	digitalWrite(motorPin1, HIGH);
	digitalWrite(motorPin2, LOW);
	digitalWrite(motorPin3, LOW);
	digitalWrite(motorPin4, HIGH);
	delay(500);
	//This code  will sharp turn platform right for 0.5 sec.
	digitalWrite(motorPin1, LOW);
	digitalWrite(motorPin2, HIGH);
	digitalWrite(motorPin3, HIGH);
	digitalWrite(motorPin4, LOW);
	delay(500);
	//This code will stop platform for 2 sec.
	digitalWrite(motorPin1, LOW);
	digitalWrite(motorPin2, LOW);
	digitalWrite(motorPin3, LOW);
	digitalWrite(motorPin4, LOW);
	delay(2000);
}
```

Το πρόγραμμα περιστρέφει τους δύο κινητήρες με την ταχύτητα που ρυθμίζεται από τις εντολές analogWrite() και φορά περιστροφής που καθορίζεται από την πολικότητα των ακροδεκτών. Με την κάθε τετράδα εντολών digitalWrite() εκτελείται και μια κίνηση της

πλατφόρμας, η οποία συνεχίζεται για όσο χρόνο καθορίζεται με την εντολή delay(). Αφού εκτελούνται όλες οι πιθανές κινήσεις, οι κινητήρες απενεργοποιούνται για 2 δευτερόλεπτα, πριν ο κύκλος επαναληφθεί.

Ξεκινάμε την προσομοίωση του κυκλώματος στο TinkerCAD και παρατηρούμε την περιστροφή των κινητήρων.



Β. Έλεγχος των κινήσεων μιας ρομποτικής πλατφόρμας με υπορουτίνες

Χωρίς να μεταβληθεί το κύκλωμα, τροποποιούμε τον κώδικα, δημιουργώντας για κάθε διακριτή κίνηση της πλατφόρμας μια υπορουτίνα και εκτελώντας τες με την ίδια σειρά. Ο κώδικας τροποποιείται ως εξής:

```
//Motor A
const int enA = 5;
const int motorPin1  = 7;
const int motorPin2  = 6;
//Motor B
const int enB = 10;
const int motorPin3  = 9;
const int motorPin4  = 8;

//This will run only one time.
void setup(){
        //Set pins for motors
        pinMode(enA, OUTPUT);
        pinMode(enB, OUTPUT);
        pinMode(motorPin1, OUTPUT);
        pinMode(motorPin2, OUTPUT);
        pinMode(motorPin3, OUTPUT);
        pinMode(motorPin4, OUTPUT);
        analogWrite(enA,180);
        analogWrite(enB,180);
}

void loop(){
        forward();
```

```
        backward();
        turnLeft();
        turnRight();
        sharpTurnLeft();
        sharpTurnRight();
        stop();
}

void forward(){
        // This code will move platform forward for 1 sec.
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin4, LOW);
        delay(1000);
}

void backward(){
        // This code will move platform backward  for 1 sec.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, HIGH);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, HIGH);
        delay(1000);
}

void turnLeft(){
        //This code  will turn platform left for 0.5 sec.
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, LOW);
        delay(500);
}

void turnRight(){
        //This code  will turn platform right for 0.5 sec.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin4, LOW);
        delay(500);
}

void sharpTurnLeft(){
        //This code  will sharp turn platform left for 0.5 sec.
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
```

```
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, HIGH);
        delay(500);
}

void sharpTurnRight(){
        //This code  will sharp turn platform left for 0.5 sec.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, HIGH);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin4, LOW);
        delay(500);
}

void stop(){
        //This code will stop platform for 2 sec.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, LOW);
        delay(2000);
}
```

Όταν ένα τμήμα κώδικα επαναλαμβάνεται αρκετές φορές ή όταν ο κώδικας του κυρίως τμήματος γίνεται πολύπλοκος, μπορούμε να χρησιμοποιήσουμε υπορουτίνες, ώστε ο κώδικας να γίνει πιο εύκολα κατανοητός.
Εισάγουμε τις υπορουτίνες με την παρακάτω μορφή:

```
void name(){
        code
}
```

Κάθε φορά που θέλουμε να εκτελεστεί ο κώδικας που περιέχεται στην υπορουτίνα, αρκεί να γράψουμε το όνομα της υπορουτίνας:

```
        name();
```

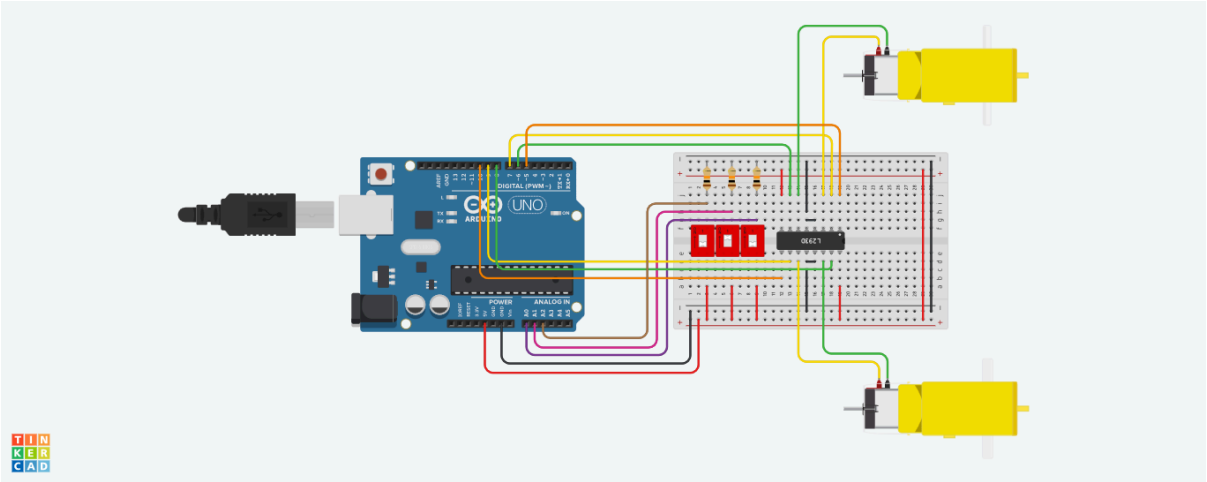
Ξεκινάμε την προσομοίωση του κυκλώματος στο TinkerCAD. Οι κινητήρες περιστρέφονται με την ίδια φορά και για τον ίδιο χρόνο, όπως και στην προηγούμενη περίπτωση.

Φύλλο εργασίας 4 - Ακόλουθος γραμμής με αισθητήρες IR και το L293

Ένας συνηθισμένος τρόπος κατεύθυνσης ενός ρομπότ είναι ο ακόλουθος γραμμής, όπου το ρομπότ ακολουθεί μια μαύρη γραμμή σε λευκό πάτωμα με τη βοήθεια αισθητήρων υπερύθρων (IR). Όταν οι υπέρυθρες ανακλώνται στο λευκό πάτωμα ο αισθητήρας δίνει το ψηφιακό 0, ενώ όταν απορροφώνται από τη μαύρη γραμμή δίνει το ψηφιακό 1. Δυστυχώς στο TinkerCAD δεν περιλαμβάνονται αισθητήρες υπερύθρων, οπότε για την προσομοίωση θα χρησιμοποιήσουμε ένα διακόπτη για κάθε αισθητήρα.

Έλεγχος των κινήσεων μιας ρομποτικής πλατφόρμας που ακολουθεί γραμμή

Σχεδιάζουμε το παρακάτω κύκλωμα στο TINKERCAD:



Για την εξομοίωση των αισθητήρων υπερύθρων χρησιμοποιούμε διακόπτες DIP με αντιστάσεις των 10KΩ σε σύνδεση pull-down.

Εισάγουμε τον παρακάτω κώδικα:

```
//Motor A
const int enA = 5;
const int motorPin1  = 7;
const int motorPin2  = 6;
//Motor B
const int enB = 10;
const int motorPin3  = 9;
const int motorPin4  = 8;
//IRsensors
const int leftIR = A2;
const int middleIR = A1;
const int rightIR = A0;
int leftID, middleID, rightID;

void setup(){
        //Set pins for motors
        pinMode(motorPin1, OUTPUT);
        pinMode(motorPin2, OUTPUT);
```

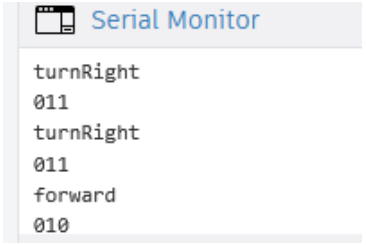
```
	pinMode(motorPin3, OUTPUT);
	pinMode(motorPin4, OUTPUT);
	pinMode(enA, OUTPUT);
	pinMode(enB, OUTPUT);
	analogWrite(enA,180);
	analogWrite(enB,180);
	//Set pins as inputs for IR sensors
	pinMode(leftIR, INPUT);
	pinMode(middleIR, INPUT);
	pinMode(rightIR, INPUT);
	// initialize serial communication at 9600 bits per second:
	Serial.begin(9600);
}

void loop(){
	leftID = digitalRead(leftIR);
	middleID = digitalRead(middleIR);
	rightID = digitalRead(rightIR);
	if (leftID==HIGH && middleID==HIGH && rightID==HIGH){
		stop();
		Serial.println("stop");
	}
	else if (leftID==LOW && middleID==HIGH && rightID==LOW){
		forward();
		Serial.println("forward");
	}
	else if (leftID==HIGH && middleID==HIGH && rightID==LOW){
		turnLeft();
		Serial.println("turnLeft");
	}
	else if (leftID==LOW && middleID==HIGH && rightID==HIGH){
		turnRight();
		Serial.println("turnRight");
	}
	else if (leftID==HIGH && middleID==LOW && rightID==LOW){
		sharpTurnLeft();
		Serial.println("sharpTurnLeft");
	}
	else if (leftID==LOW && middleID==LOW && rightID==HIGH){
		sharpTurnRight();
		Serial.println("sharpTurnRight");
	}
	else //(leftID==LOW && middleID==LOW && rightID==LOW){
		backward();
		Serial.println("backward");
	}
	Serial.print(leftID);
	Serial.print(middleID);
```

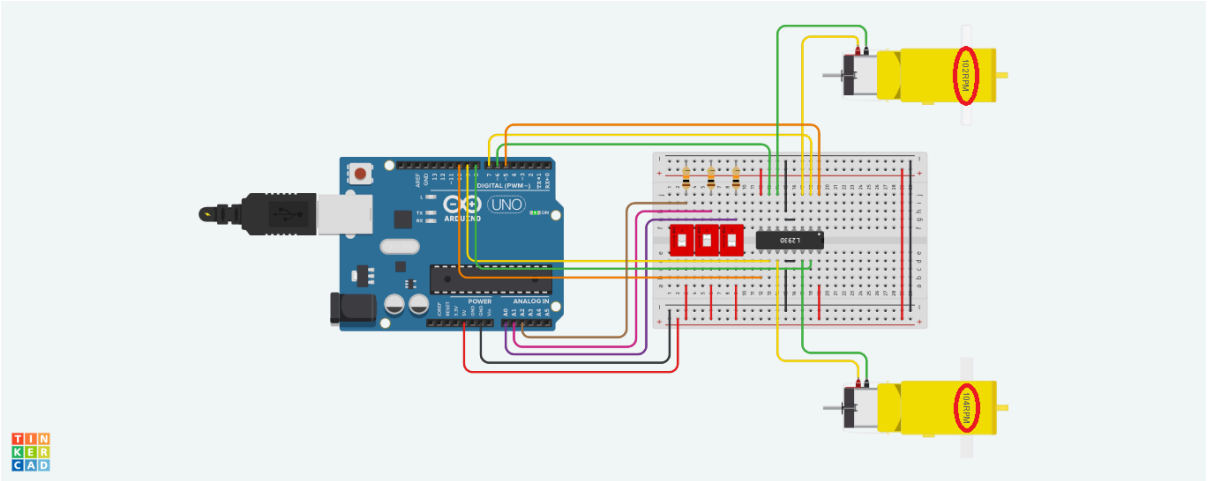
```
        Serial.println(rightID);
}

void forward(){
        // This code will move platform forward.
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin4, LOW);
}

void backward(){
        // This code will move platform backward.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, HIGH);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, HIGH);
}

void turnLeft(){
        //This code  will turn platform left.
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, LOW);
}

void turnRight(){
        //This code  will turn platform right.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin4, LOW);
}

void sharpTurnLeft(){
        //This code  will sharp turn platform left.
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, HIGH);
}

void sharpTurnRight(){
        //This code  will sharp turn platform left.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, HIGH);
        digitalWrite(motorPin3, HIGH);
```

```
        digitalWrite(motorPin4, LOW);
}

void stop(){
        //This code will stop platform.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, LOW);
}
```

Το πρόγραμμα διαβάζει τις τιμές των τριών αισθητήρων υπερύθρων και ανάλογα με τις ενδείξεις καλεί τις υπορουτίνες για τις αντίστοιχες κινήσεις. Αν η πλατφόρμα συναντήσει μια κάθετη μαύρη γραμμή, τότε και οι τρεις αισθητήρες έχουν την τιμή HIGH και η πλατφόρμα σταματά. Αν η πλατφόρμα βγει εκτός της μαύρης γραμμής, τότε και οι τρεις αισθητήρες έχουν την τιμή LOW και η πλατφόρμα οπισθοχωρεί μέχρι να συναντήσει ξανά την γραμμή. Για τον έλεγχο της λειτουργίας του προγράμματος χρησιμοποιούνται οι εντολές Serial.print() και Serial.println(), οι οποίες εμφανίζουν στη σειριακή τις τιμές των αισθητήρων και την κίνηση που αντιστοιχεί.

Ξεκινάμε την προσομοίωση του κυκλώματος στο TinkerCAD και παρατηρούμε τις ενδείξεις της σειριακής και την περιστροφή των κινητήρων.

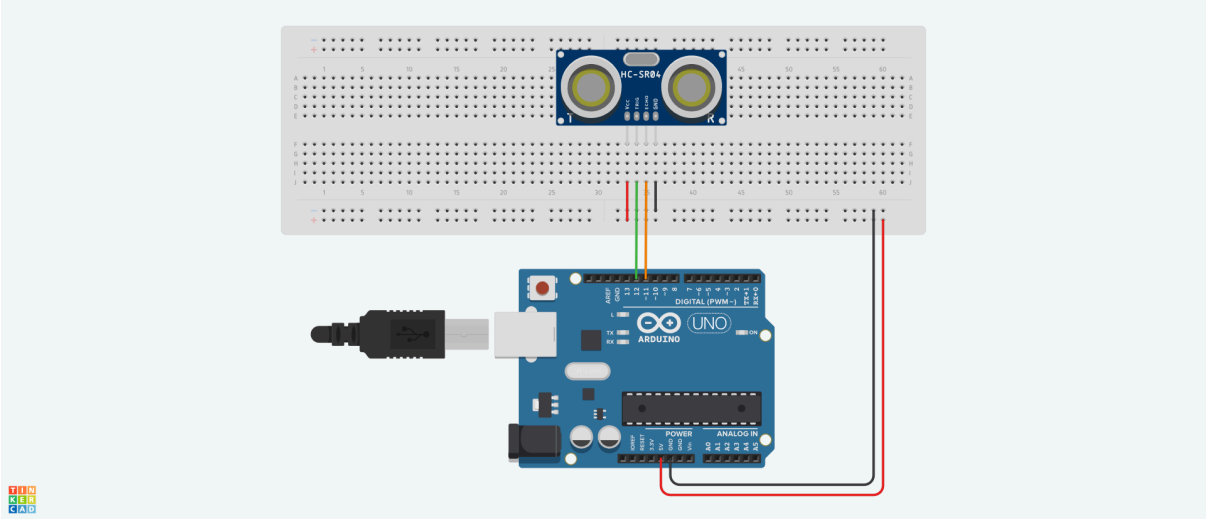




Φύλλο εργασίας 5 - Μελέτη αισθητήρα υπερήχων

Ο αισθητήρας υπερήχων χρησιμοποιεί ήχο πάνω από την μέγιστη συχνότητα ακοής για να ανιχνεύσει απόσταση. Ο αισθητήρας εκπέμπει ένα σύντομο παλμό ήχου και περιμένει να επιστρέψει. Μετρά τον χρόνο που χρειάζεται για να ταξιδέψει ο ήχος, να ανακλαστεί σε ένα αντικείμενο και μετά να επιστρέψει στον αισθητήρα. Επειδή η ταχύτητα του ήχου είναι γνωστή, ο συνολικός χρόνος που απαιτείται για την επιστροφή του ήχου εξαρτάται από την απόσταση του αντικειμένου.

<u>Α. Μέτρηση απόστασης με τον αισθητήρα υπερήχων</u>
Σχεδιάζουμε το παρακάτω κύκλωμα στο TINKERCAD:



Εισάγουμε τον παρακάτω κώδικα:

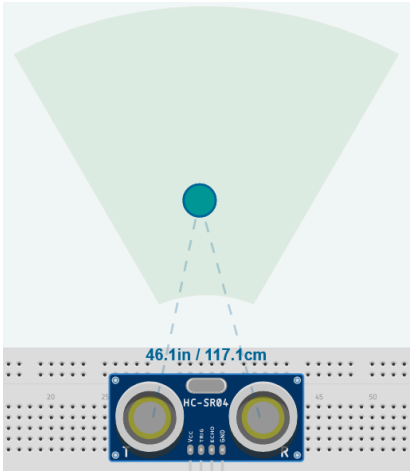
```
const int echoPin=11;
const int trigPin=12;
long duration;
long distance;

void setup() {
        Serial.begin(9600);
        pinMode(trigPin, OUTPUT);
        pinMode(echoPin, INPUT);
}

void loop() {
        digitalWrite(trigPin, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);
        duration = pulseIn(echoPin, HIGH);
        distance = (duration/2) / 29.1;

        Serial.print(distance);
```

```
        Serial.println(" cm");
        delay(1000);
}
```

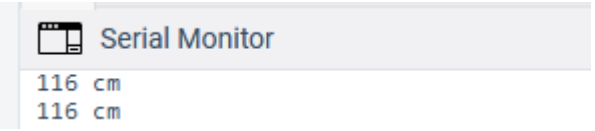
Το πρόγραμμα διαβάζει την απόσταση ενός εμποδίου από τον αισθητήρα υπερήχων και καταγράφει την ένδειξη σε εκατοστά στη σειριακή.

Ξεκινάμε την προσομοίωση του κυκλώματος στο TinkerCAD.

Επιλέγουμε τον αισθητήρα υπερήχων στον χώρο σχεδίασης και μεταβάλλουμε τη θέση του αντικειμένου:



Παρατηρούμε την ένδειξη στη σειριακή:



<u>Β. Χρήση υπορουτίνας για την μέτρηση της απόστασης</u>

Χωρίς να μεταβάλουμε το κύκλωμα, τροποποιούμε τον κώδικα ως εξής:

```
const int echoPin=11;
const int trigPin=12;
long duration;
long distance;
long cm;

void setup() {
        Serial.begin(9600);
        pinMode(trigPin, OUTPUT);
        pinMode(echoPin, INPUT);
}

void loop() {
        cm = getDistance();

        Serial.print(cm);
        Serial.println(" cm");
```

```
        delay(1000);
}

long getDistance(){
        digitalWrite(trigPin, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);
        duration = pulseIn(echoPin, HIGH);
        distance = (duration/2) / 29.1;
        return distance;
}
```

Μπορούμε να χρησιμοποιήσουμε τις υπορουτίνες για να υπολογίσουμε κάποια μεγέθη και να επιστρέψουμε μόνο τις απαραίτητες τιμές στο κυρίως πρόγραμμα.
Εισάγουμε την υπορουτίνα:

```
long getDistance(){

}
```

Η λέξη long στην αρχή σημαίνει ότι η υπορουτίνα θα επιστρέφει στο κυρίως πρόγραμμα μια τιμή τύπου long. Στο εσωτερικό της υπορουτίνας εισάγουμε τις εντολές για την ανάγνωση του αισθητήρα υπερήχων και τον υπολογισμό της απόστασης. Στο τέλος της υπορουτίνας επιστρέφουμε τη μεταβλητή distance στο κυρίως πρόγραμμα με την εντολή:

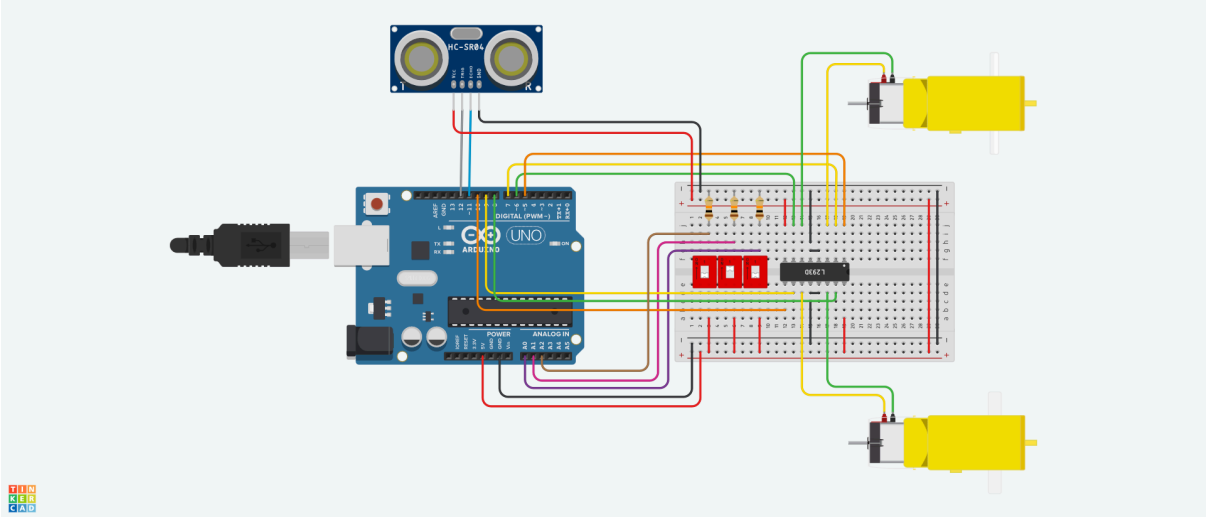
```
        return distance;
```

Στην ενότητα setup() εισάγουμε και τη μεταβλητή cm για την αποθήκευση της τιμής που επιστρέφει η υπορουτίνα στο κυρίως πρόγραμμα.
Επαναλαμβάνουμε την προσομοίωση του κυκλώματος στο TinkerCAD, όπου τα αποτελέσματα πρέπει να είναι παρόμοια με την προηγούμενη υλοποίηση.

Φύλλο εργασίας 6 - Ακόλουθος γραμμής με αποφυγή σύγκρουσης

Ενσωματώνουμε τον αισθητήρα υπερήχων στον ακόλουθο γραμμής και το ρομπότ εκτός από το να ακολουθεί μια μαύρη γραμμή σε λευκό πάτωμα μπορεί να αποφεύγει και τις συγκρούσεις με αντικείμενα που μπορεί να βρεθούν στη διαδρομή του.

<u>Έλεγχος των κινήσεων μιας ρομποτικής πλατφόρμας που ακολουθεί γραμμή και αποφεύγει τα εμπόδια</u>
Σχεδιάζουμε το παρακάτω κύκλωμα στο TINKERCAD:



Εισάγουμε τον παρακάτω κώδικα:

```
//Motor A
const int enA = 5;
const int motorPin1  = 7;
const int motorPin2  = 6;
//Motor B
const int enB = 10;
const int motorPin3  = 9;
const int motorPin4  = 8;
//Ultrasonic Sensor
const int echoPin = 11;
const int trigPin = 12;
long duration, distance, cm;
//IRsensors
const int leftIR = A2;
const int middleIR = A1;
const int rightIR = A0;
int leftID, middleID, rightID;

void setup(){
        //Set pins for motors
        pinMode(motorPin1, OUTPUT);
```

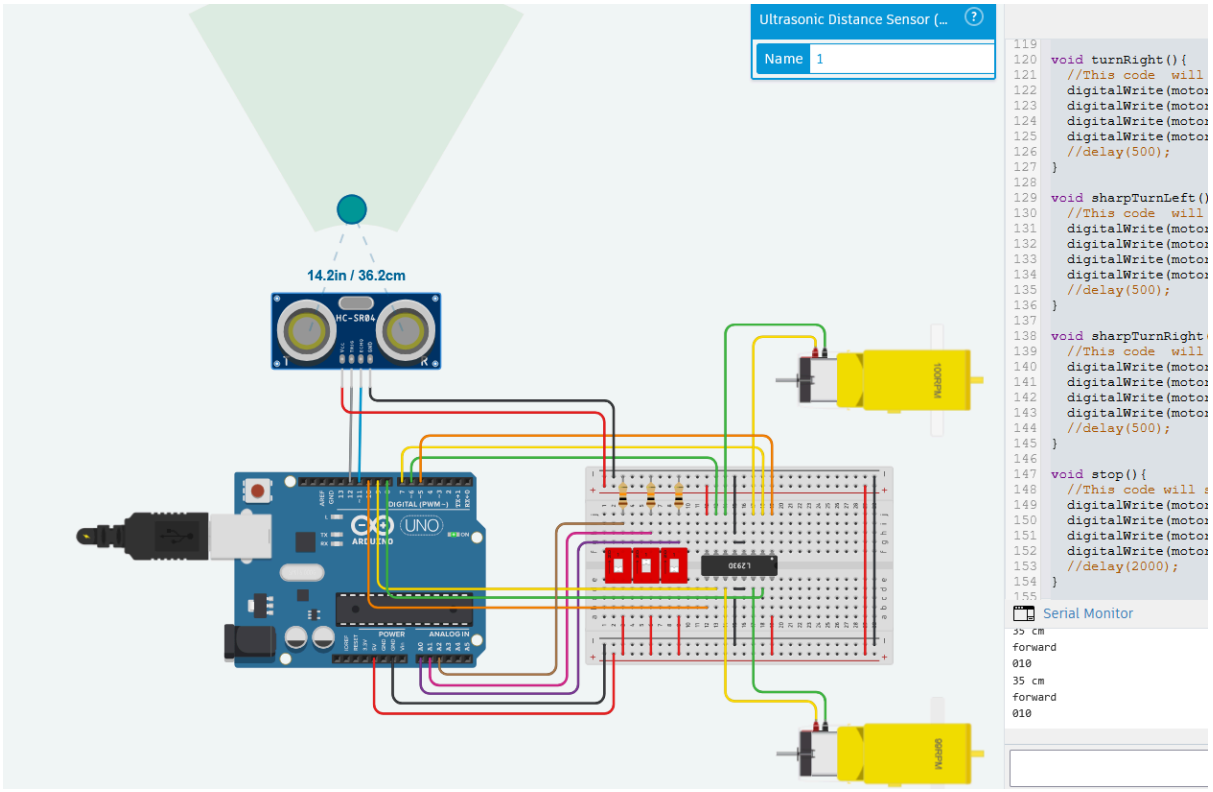
```
        pinMode(motorPin2, OUTPUT);
        pinMode(motorPin3, OUTPUT);
        pinMode(motorPin4, OUTPUT);
        pinMode(enA, OUTPUT);
        pinMode(enB, OUTPUT);
        analogWrite(enA,180);
        analogWrite(enB,180);
        //Set pins for ultrasonic sensor
        pinMode(echoPin, INPUT);
        pinMode(trigPin, OUTPUT);
        //Set pins as inputs for IR sensors
        pinMode(leftIR, INPUT);
        pinMode(middleIR, INPUT);
        pinMode(rightIR, INPUT);
        // initialize serial communication at 9600 bits per second:
        Serial.begin(9600);
}

void loop(){
        cm = getDistance();
        Serial.print (cm);
        Serial.println(" cm");
        if (cm <= 15){
                stop();
                Serial.println("stop at object");
        }
        else {
                move();
        }
}

void move(){
        leftID = digitalRead(leftIR);
        middleID = digitalRead(middleIR);
        rightID = digitalRead(rightIR);
        if (leftID==HIGH && middleID==HIGH && rightID==HIGH){
                stop();
                Serial.println("stop");
        }
        else if (leftID==LOW && middleID==HIGH && rightID==LOW){
                forward();
                Serial.println("forward");
        }
        else if (leftID==HIGH && middleID==HIGH && rightID==LOW){
                turnLeft();
                Serial.println("turnLeft");
        }
        else if (leftID==LOW && middleID==HIGH && rightID==HIGH){
```

```
                turnRight();
                Serial.println("turnRight");
        }
        else if (leftID==HIGH && middleID==LOW && rightID==LOW){
                sharpTurnLeft();
                Serial.println("sharpTurnLeft");
        }
        else if (leftID==LOW && middleID==LOW && rightID==HIGH){
                sharpTurnRight();
                Serial.println("sharpTurnRight");
        }
        else //(leftID==LOW && middleID==LOW && rightID==LOW){
                backward();
                Serial.println("backward");
        }
        Serial.print(leftID);
        Serial.print(middleID);
        Serial.println(rightID);
}

void forward(){
        // This code will move platform forward for 1 sec.
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin4, LOW);
}

void backward(){
        // This code will move platform backward  for 1 sec.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, HIGH);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, HIGH);
}

void turnLeft(){
        //This code  will turn platform left for 0.5 sec.
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, LOW);
}

void turnRight(){
        //This code  will turn platform right for 0.5 sec.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, LOW);
```

```
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin4, LOW);
}

void sharpTurnLeft(){
        //This code  will sharp turn platform left for 0.5 sec.
        digitalWrite(motorPin1, HIGH);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, HIGH);
}

void sharpTurnRight(){
        //This code  will sharp turn platform left for 0.5 sec.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, HIGH);
        digitalWrite(motorPin3, HIGH);
        digitalWrite(motorPin4, LOW);
}

void stop(){
        //This code will stop platform for 2 sec.
        digitalWrite(motorPin1, LOW);
        digitalWrite(motorPin2, LOW);
        digitalWrite(motorPin3, LOW);
        digitalWrite(motorPin4, LOW);
}

long getDistance(){
        digitalWrite(trigPin, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);
        duration = pulseIn(echoPin, HIGH);
        distance = (duration/2) / 29.1;
        return distance;
}
```

Το πρόγραμμα ελέγχει την απόσταση που έχει διαθέσιμη για να κινηθεί και αν είναι μεγαλύτερη από την καθορισμένη τιμή τότε καλεί την υπορουτίνα της κίνησης για να παρακολουθήσει τη γραμμή στο πάτωμα.

Ξεκινάμε την προσομοίωση του κυκλώματος στο TinkerCAD και παρατηρούμε τη λειτουργία του κυκλώματος μεταβάλλοντας τις τιμές των αισθητήρων καθώς και τις ενδείξεις της σειριακής.
Ένα στιγμιότυπο της προσομοίωσης φαίνεται παρακάτω:


Ultrasonic Distance Sensor (...
Name 1
14.2in / 36.2cm
HC-SR04
UNO
ARDUINO
L293D
Serial Monitor
35 cm
forward
010
35 cm
forward
010